N'oluyo · Geliştirici Dokümanları · Son güncelleme: 11 Kasım 2026 · https://noluyo.net/gelistirici

N'oluyo Bot ve Geliştirici Dokümanları

N'oluyo botları, resmi kütüphaneleri, OAuth2, Gateway ve REST API için eksiksiz tek dosya doküman.

İçindekiler

- Nereden başlamalı
- Yapay zekaya yazdır
- Kurulum
- Başlangıç: ilk botun
- Tarifler
- Botu açık tut
- Temel kavramlar
- Kimlik doğrulama ve genel kurallar
- Uygulamalar ve botlar
- İzinler
- Hız sınırları
- JavaScript kütüphanesi (`noluyo-bot`)
- Python kütüphanesi (`noluyo-bot`)
- Örnekler
- Embed'ler ve bileşenler
- Bileşenler
- Komutlar
- Komut izinleri
- Otomatik tamamlama
- Formlar
- Etkileşimler
- Özel mesajlar
- HTTP ile teslim
- N'oluyo ile giriş
- Gateway
- Ses
- Intent'ler
- Shard'lar
- Doğrulama
- Rehbere girmek
- Sık hatalar
- Sözlük
- Başka platformdan gelenler
- Güvenlik
- Hatalar
- Sık sorulan sorular
- Değişiklikler
- API'ye genel bakış
- Kullanıcılar
- Sunucular
- Roller
- Üyeler
- Yasaklar
- Kanallar
- Mesajlar
- Sabitlenen mesajlar
- Tepkiler
- Uygulamalar
- Gateway
- Etkileşimler
- Komutlar
- OAuth2 (N'oluyo ile giriş)
- Keşfet (Bot rehberi)
- Gateway olayları, opcode'lar, intent'ler ve kapanış kodları

Nereden başlamalı

N'oluyo üzerinde kendi botunu çalıştırmak ya da sitene "N'oluyo ile giriş yap" düğmesi koymak için yazılım uzmanı olman gerekmez. Sana en uygun yolu seç, yalnızca o yolun sayfalarını takip et.

Bot nedir ve nerede çalışır

Bot, N'oluyo'da kendi kullanıcı adı, resmi ve **BOT** etiketi olan özel bir hesaptır; arkada onu yöneten küçük bir bilgisayar programı çalışır. Yeni gelen üyeyi karşılamak, kuralları çiğneyen mesajları silmek, anket açmak ya da ses kanalında müzik çalmak bu programın işidir.

Bilmen gereken en önemli kural şudur: **N'oluyo bot programını barındırmaz.** Portalda uygulama açtığında N'oluyo sana bot hesabını ve şifresi olan **token** değerini verir; programı ise kendi bilgisayarında ya da kiraladığın küçük bir sunucuda (VPS) sen çalıştırırsın. Bilgisayarını kapattığında orada çalışan bot da çevrimdışı olur; gün boyu açık kalmasını istediğinde onu bir sunucuya taşırsın.

Sana uygun yolu seç

Deneyimine göre üç yoldan birini seçebilirsin:

1. Kod bilmiyorum (yapay zeka ile yazdır)

Daha önce hiç kod yazmadıysan bu yoldan başla. Geliştirici portalından botunu açar, N'oluyo'nun hazır doküman dosyasını ve kopyalanmaya hazır istemi kullandığın yapay zeka asistanına verirsin; o sana çalışan dosyaları yazar, sen de bilgisayarında çalıştırırsın.

- İlk adım: Yapay zekaya yazdır
- Bilgisayar hazırlığı: Kurulum
- Hazır çalışan örnekler: Tarifler

2. Biraz kod biliyorum (resmi kütüphane ile)

JavaScript ya da Python okuyabiliyorsan resmi `noluyo-bot` kütüphanesi en rahat yoldur. Gateway bağlantısını, kopunca kaldığı yerden devam etmeyi (`resume`), hız sınırı kuyruğunu ve çoklu bağlantı (`shard`) yönetimini senin yerine halleder; sen yalnızca olay ve komut işleyicilerini yazarsın.

- Bilgisayar hazırlığı: Kurulum
- Node.js 22+ için: JavaScript kütüphanesi
- Python 3.11+ için: Python kütüphanesi
- Kopyalayıp çalıştıracağın 9 örnek: Tarifler

3. API'yi doğrudan kullanacağım

Kendi dilinde ya da hiçbir kütüphane kurmadan doğrudan HTTP ve WebSocket ile çalışmak istiyorsan protokol sade ve açıktır: düz JSON çerçeveleri, standart `fetch` ve `WebSocket`, Ed25519 imza doğrulaması ve açık hız sınırı başlıkları.

- Adım adım ilk istek ve bağlantı: Hızlı başlangıç

- Canlı olay bağlantısı: Gateway
- Kartlar, butonlar, etkileşimler, komutlar, formlar ve ses: Embed'ler, Bileşenler, Etkileşimler, Komutlar, Formlar, Ses
- Tüm uçlar ve olay şemaları: REST referansı ve Olaylar

> **İpucu:** Amacın bir sunucuya bot eklemek değil de kendi web sitene ya da uygulamana kullanıcıların N'oluyo hesabıyla giriş yapmasını sağlamaksa bot kısımlarını atlayıp doğrudan N'oluyo ile giriş yap sayfasına geç.

Zaman ve maliyet beklentisi

İş	Ne kadar sürer	Maliyet
Portalda uygulama açıp ilk mesajı göndermek	5–10 dakika	Ücretsiz
Yapay zeka ya da kütüphane ile çalışan ilk bot	15–30 dakika	Kendi bilgisayarında ücretsiz
Botu 7/24 açık tutmak için küçük Linux VPS	Kurulumu 20 dakika	Kiraladığın sunucunun aylık ücreti (N'oluyo ücret almaz)
Botu doğrulayıp Botlar rehberinde listelemek	Başvuru formu 10 dakika	Ücretsiz

N'oluyo API kullanımı, bot oluşturma, doğrulama başvurusu ve rehberde listelenme için hiçbir ücret istemez.

Dokümanı indirmek ve yapay zekaya vermek

Dokümandaki her sayfa hem insan okuması hem de yapay zeka araçları için ayrı biçimlerde sunulur:

- **Tek sayfa ham Markdown (`.md`):** Her rehber sayfasının üstündeki **Markdown olarak aç** ya da **Kopyala** düğmesiyle o sayfanın düz `.md` halini alabilirsin (örneğin `/gelistirici/nereden-baslamali.md`).
- **Tarayıcıdan `.html` veya `.pdf` kaydetmek:** Herhangi bir rehberi ya da tek sayfalık dokümanı tarayıcının **Farklı kaydet** (`.html`) veya **Yazdır → PDF olarak kaydet** (`.pdf`) seçeneğiyle bilgisayarına indirebilirsin.
- **Küçük bağlamlı yapay zeka özeti (`llms-kisa.txt`):** 30 KB altındaki bu dosya, resmi `noluyo-bot` kütüphanesiyle bot yazdırmak için gereken temel kuralları ve örnekleri tek metinde toplar; sohbet penceresi dar olan araçlara doğrudan yapıştırabilirsin (`/gelistirici/llms-kisa.txt`).
- **Tam referans metni (`llms-full.txt`) ve dizin (`llms.txt`):** Tüm rehberleri, bütün REST rotalarını, olay şemalarını, hata ve kapanış kodlarını tek dosyada taşır (`/gelistirici/llms-full.txt` ve `/gelistirici/llms.txt`). Dosya yükleme destekleyen yapay zeka araçlarına bu dosyayı ver.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zeka asistanına kod yazdıracaksan yalnızca bu giriş sayfasını değil, `/gelistirici/llms-kisa.txt` (veya `/gelistirici/llms-full.txt`) dosyasını ve Yapay zekaya yazdır sayfasındaki hazır istemi ver. Bot token'ını asla sohbete yapıştırma; kodun `process.env.BOT_TOKEN` ya da `os.environ["BOT_TOKEN"]` okumasını iste.

Sırada ne var

- Yapay zekaya yazdır: kod bilmeden adım adım bot yazdırma rehberi ve hazır istem
- Kurulum: Windows, macOS ve Linux'ta Node.js veya Python hazırlığı
- Temel kavramlar: uygulama, bot, token, sunucu, rol, izin ve intent sözcüklerinin kısa özeti

Yapay zekaya yazdır

Hiç kod bilmesen bile bir yapay zeka asistanına N'oluyo'nun dokümanını verip çalışan bir bot yazdırabilirsin. İşin sırrı yapay zekaya doğru referans dosyasını vermek, ne istediğini net tarif etmek ve botun şifresi olan token'ı gizli tutmaktır.

1. Portaldan uygulamanı aç ve token'ı `.env` dosyasına koy

1. Geliştirici portalına git, N'oluyo hesabınla giriş yap ve **Yeni uygulama** düğmesine bas.
2. Botuna bir görünen ad (örneğin `Karşılama Botu`) ve küçük harfli bir kullanıcı adı (örneğin `karsilama_botu`) verip **Oluştur**'a bas.
3. Ekranda yalnızca bir kez gösterilen **token** değerini **Kopyala** ile al.
4. Bilgisayarında botun için boş bir klasör aç (örneğin `benim-botum`), içinde `.env` adında düz metin bir dosya oluştur ve token'ı yalnızca oraya yaz:

```
BOT_TOKEN=buraya_kopyaladigin_token_gelecek
```

> **Uyarı:** Token'ı **asla yapay zeka sohbetine yapıştırma** ve **asla kod dosyalarının içine düz metin olarak yazdırma.** Sohbete yapıştırdığın an şifreni üçüncü taraf bir servise vermiş olursun. Yanlışlıkla yapıştırırsan hemen portala dönüp **Token'ı yenile** düğmesine bas.

Ardından uygulama sayfasının **Ekleme bağlantısı** bölümünden botun ihtiyaç duyacağı izinleri seç (başlangıç için **Kanalları görüntüle**, **Mesaj gönder**, **Mesaj geçmişini oku** yeterlidir), **Bağlantıyı aç**'a bas ve botu deneme sunucuna ekle.

Botun kanaldaki normal mesajların metnini okuyacaksa (örneğin küfür süzgeci ya da `!selam` gibi ünlemli komutlar için) portaldaki **Ayrıcalıklı intent'ler** bölümünden **Mesaj içeriği**'ni açmayı unutma. Yalnızca `/` eğik çizgi komutu ve buton kullanacaksan buna gerek yoktur.

2. Yapay zekaya hangi doküman dosyasını vereceksin

Yapay zeka araçları N'oluyo API'sini ezbere bilmez; dokümanı vermezsen başka sohbet platformlarının adreslerini ve kütüphanelerini uydurur. Kullandığın araca göre şu seçeneklerden birini indirip sohbete ekle:

Seçenek	Adres / Nasıl alınır	Ne zaman seçmeli
`llms-kisa.txt` (önerilen başlangıç)	`/gelistirici/llms-kisa.txt` adresinden indir ya da kopyala	30 KB altındadır. Ücretsiz ya da dar bağlamlı sohbet pencerelerine doğrudan yapıştırmak ve resmi `noluyo-bot` kütüphanesiyle hızlıca bot yazdırmak için en iyi seçimdir.
`llms-full.txt` (tam referans)	`/gelistirici/llms-full.txt` adresinden indir	Dosya yükleme destekleyen araçlara ver. Tüm rehberleri, bütün REST rotalarını, olay şemalarını, ses kurallarını ve OAuth2 akışını tek dosyada içerir.
Tek sayfa `.md`	Her rehberin üstündeki **Markdown olarak aç** / **Kopyala** düğmesi	Yalnızca tek bir özelliği (örneğin Ses ya da Formlar) ekletmek istediğinde ilgili `.md` dosyasını ver.
`.html` veya `.pdf` kaydı	Tarayıcıdan **Farklı kaydet** (`.html`) ya da **Yazdır → PDF olarak kaydet**	Yalnızca PDF veya HTML yüklemesi kabul eden kurumsal yapay zeka araçlarında kullan.

3. Kopyalanmaya hazır istem

Yukarıdaki doküman dosyasını (`llms-kisa.txt` veya `llms-full.txt`) sohbete ekledikten sonra aşağıdaki kutuyu kopyalayıp yapıştır, en altındaki `[BURAYA BOTUN NE YAPACAĞINI YAZ]` satırını kendi istediğin işle değiştir:

```
Sana ekte N'oluyo geliştirici dokümanını (llms-kisa.txt veya llms-full.txt) veriyorum. Yalnızca bu dokümandaki kurallara, rotalara ve olay adlarına uyarak N'oluyo için çalışan bir bot yaz.

Kesin kurallar:
1. Başka hiçbir sohbet platformunun kütüphanesini, adresini veya olay yapısını kullanma. Resmi `noluyo-bot` paketini (Node.js 22+ ESM) kullan; gerekirse yalnızca dokümanın açıkça söylediği standart API adresini (`https://noluyo.net/api/v1`) çağır.
2. Bot token değerini benden isteme ve koda düz metin olarak asla yazma. Kod token'ı ortam değişkeninden (`.env` dosyasından `node --env-file=.env bot.mjs` ile `process.env.BOT_TOKEN`) okusun.
3. Bana oluşturmam gereken tüm dosyaları (`package.json`, `.gitignore`, `.env.example`, `bot.mjs`) tam içerikleriyle, hiçbir yeri `// ...` diye kısaltmadan ver; `.gitignore` içinde `.env` ve `node_modules` mutlaka bulunsun.
4. Bu bot için geliştirici portalında açmam gereken ayrıcalıklı intent'leri (`GUILD_MEMBERS`, `GUILD_PRESENCES`, `MESSAGE_CONTENT`) ve sunucuda bota vermem gereken izinleri adım adım söyle.
5. Etkileşimlere (`INTERACTION_CREATE`) 3 saniye içinde yanıt ver; embed görseli gerekiyorsa dış URL yerine yalnızca `attachment://` kullan; `4004` ve `4014` gibi yeniden bağlanılmaması gereken kapanış kodlarında sonsuz döngüye girme.
6. Sonunda botu Windows, macOS ve Linux terminalinde nasıl kurup çalıştıracağımı tek tek komutlarla anlat.

İstediğim bot şu işi yapacak:
[BURAYA BOTUN NE YAPACAĞINI YAZ]
```

Python tercih ediyorsan istemdeki 1. ve 2. maddede `noluyo-bot` Python paketini (`noluyo_bot`, Python 3.11+) ve `os.environ["BOT_TOKEN"]` istediğini belirtmen yeterlidir.

4. Botunu nasıl tarif etmelisin

Yapay zeka belirsiz cümlelerde tahmin yürütür; tetikleyiciyi (ne olunca?), aksiyonu (ne yapacak?) ve sınırı (kim kullanabilir, hangi kanalda?) tek paragrafta açıkça yaz.

İyi tarif örnekleri (5 örnek):

1. **Karşılama mesajı:** "Sunucuya yeni bir üye katıldığında (`GUILD_MEMBER_ADD`), `HOSGELDIN_KANAL_ID` ortam değişkenindeki kanala yeni üyeyi etiketleyerek 'Aramıza hoş geldin!' yazan ve sunucu kurallarını özetleyen mavi renkli bir embed kartı atan bot."
2. **Butonla rol seçici:** "`/rol-paneli` eğik çizgi komutunu yalnızca `ManageRoles` izni olan yöneticiler çalıştırabilsin. Komut kanala 'Duyuru bildirimlerini almak için butona bas' mesajı ve bir buton göndersin; üye butona bastığında `DUYURU_ROL_ID` rolü onda yoksa versin, varsa geri alsın ve sonucu yalnızca basan kişinin gördüğü (`ephemeral`) bir mesajla bildirsin."
3. **Sözcük süzgeci:** "Kanalda yazılan mesajlarda `.env` içindeki `YASAKLI_KELIMELER` listesinde geçen bir sözcük varsa mesajı silsin, aynı kanala yazarı etiketleyip 'Bu sözcük yasak.' diyen bir uyarı bıraksın ve 10 saniye sonra kendi uyarı mesajını da silsin."
4. **Butonlu anket:** "`/anket` komutu `soru`, `secenek1` ve `secenek2` adında üç zorunlu metin seçeneği alsın. Kanala iki butonlu bir kart göndersin; her üye yalnızca bir kez oy verebilsin, oy değiştirirse eski oyu düşsün ve kartın üstündeki oy sayıları anında güncellensin."
5. **Zaman aşımı komutu:** "`/sustur` komutu bir `uye` (kullanıcı), `dakika` (1 ile 60 arası tam sayı) ve isteğe bağlı `sebep` alsın; yalnızca `ModerateMembers` izni olanlar kullanabilsin. Hedef üyeyi o süre kadar zaman aşımına alsın ve `DENETIM_KANAL_ID` kanalına kimin kimi neden susturduğunu yazan bir kayıt mesajı atsın."

Kaçınman gereken belirsiz tarifler (3 kötü örnek):

- *"Bana her şeyi yapan güzel bir moderasyon botu yaz."* — Hangi komutların olacağı, kimin kullanacağı ve ne ceza verileceği belirsizdir; yapay zeka yarım kalan onlarca komut üretir.
- *"Sunucudaki küfürleri engellesin ve insanlara özelden uyarı atsın."* — N'oluyo'da botlar kimseye ilk DM'yi atamaz (`403 bot_forbidden`); kurala aykırı istek botun sürekli hata almasına yol açar.
- *"Ses kanalındaki konuşmaları dinleyip yazıya döken bot yap."* — N'oluyo'da botlar seste yalnızca yayın yapar, hiçbir koşulda ses alamaz ve kimseyi duyamaz.

5. Yapay zekanın sık yanılgıları ve nasıl düzeltilir

Yapay zeka bazen kendi eski alışkanlıklarına döner. Kodda şunlardan birini görürsen kendisine hatırlatıp düzelttir:

Yanılgı	Kodda nasıl anlaşılır	Yapay zekaya ne söylemeli
Başka platformun kütüphanesini eklemek	`package.json` veya `import` satırında `noluyo-bot` dışında yabancı bir sohbet kütüphanesi görünür	"Başka platform kütüphanesi çalışmaz; onu kaldır ve dokümandaki `noluyo-bot` paketini kullan."
API adresini uydurmak	`https://api.noluyo.net` ya da `/api/v10` gibi yanlış adresler yazar	"Taban adres tam olarak `https://noluyo.net/api/v1` olmalı; dokümandaki rotaları birebir kullan."
`4014` kapanış kodunu görmezden gelmek	Bağlantı kopunca her kodda sonsuz `setTimeout` ile yeniden bağlanır	"`4014` ayrıcalıklı intent kapalı demektir, `4004` geçersiz token demektir; bunlarda yeniden bağlanma, net hata yazıp dur."
Token'ı koda gömmek	`const token = "nly_..."` gibi sabit metin yazar	"Token'ı koddan sil; yalnızca `process.env.BOT_TOKEN` oku."
Embed resmine dış URL koymak	`image: { url: "https://..." }` gönderir ve `400 validation_failed` alır	"N'oluyo'da embed görselleri dış URL kabul etmez; resmi dosya olarak yükleyip `attachment://dosya.png` kullan ya da görsel alanını kaldır."

6. Çalıştırmak, hatayı yapıştırmak ve botu açık tutmak

Bilgisayarında Node.js ya da Python kurulu değilse önce Kurulum sayfasındaki adımları tamamla. Sonra klasöründe terminali açıp yapay zekanın verdiği kurulum ve çalıştırma komutunu uygula:

```
npm install
node --env-file=.env bot.mjs
```

Hata çıkarsa ne yapmalı? Kırmızı bir hata metni görürsen panikleme; ilk seferde küçük bir izin ya da ayar eksiği çıkması çok doğaldır. Terminaldeki hata satırlarını kopyalayıp yapay zekaya yapıştır (**`.env` dosyasının içeriğini ve token'ı asla yapıştırma!**). En sık karşılaşılan hata mesajlarının anlamını kendin de Sık hatalar sayfasından bir dakikada kontrol edebilirsin.

Bittiğini nasıl anlarsın?

- Terminalde botunun bağlandığını söyleyen satırı (`... bağlandı` / `READY`) görüyorsun,
- Sunucunun üye listesinde botun çevrimiçi görünüyor,
- İstediğin komutu veya olayı denediğinde bot 1–2 saniye içinde beklediğin yanıtı veriyor ve terminalde kırmızı hata satırı çıkmıyor.

Botun kendi bilgisayarında sorunsuz çalıştığında, bilgisayarını kapattığında da açık kalmasını istiyorsan Botu açık tut rehberine geç.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu sayfanın kendisini değil, 3. adımdaki `istem` kutusunu ve `/gelistirici/llms-kisa.txt` ya da `/gelistirici/llms-full.txt` dosyasını yapay zekaya ver. Token'ın yalnızca `.env` dosyasında durduğundan her adımda emin ol.

Sırada ne var

- Kurulum: Node.js, Python, terminal ve `.env` kurulumunun ekran çıktılarıyla anlatımı
- Sık hatalar: `4014`, boş `content`, `403` ve "etkileşim başarısız" çözümleri
- Botu açık tut: botunu bir Linux sunucuda 7/24 kesintisiz çalıştırmak

Kurulum

Botunu kendi bilgisayarında çalıştırmak için iki şeye ihtiyacın var: bir programlama çalışma ortamı (**Node.js 22+** veya **Python 3.11+**) ve komut yazacağın bir **terminal** penceresi. İkisini de kurmak yalnızca birkaç dakika sürer; yalnız birini seçmen yeterlidir.

1. Çalışma ortamını kur

JavaScript ile yazacaksan **Node.js 22 veya üstünü**, Python ile yazacaksan **Python 3.11 veya üstünü** kur. Kararsızsan Node.js ile başla: Hızlı başlangıç ve Yapay zekaya yazdır örnekleri varsayılan olarak onu kullanır.

Windows

- **Node.js:** `nodejs.org` adresinden güncel LTS (22 ya da üstü) `.msi` kurulum paketini indirip ileri diyerek kur; ya da PowerShell açıp `winget install OpenJS.NodeJS.LTS` komutunu çalıştır.
- **Python:** `python.org/downloads` adresinden Python 3.11+ indirirken ilk ekrandaki **"Add python.exe to PATH"** kutusunu mutlaka işaretle; ya da `winget install Python.Python.3.12` çalıştır.

Kurulum bitince açık terminal penceren varsa kapatıp yeniden aç ki yeni komutlar tanınsın.

macOS

- **Node.js:** `nodejs.org` adresinden `.pkg` kurulum dosyasını indirip kur; Homebrew kullanıyorsan `brew install node@22` çalıştır.
- **Python:** `python.org/downloads` adresinden macOS paketini kur ya da `brew install python@3.12` çalıştır.

Linux (Ubuntu / Debian ve türevleri)

Dağıtım depolarındaki eski Node sürümleri (`node 18` altı) yerleşik `WebSocket` taşımadığı için güncel Node 22+ veya Python 3.11+ kurduğundan emin ol:

```
# Python 3.11+ ve sanal ortam aracı
sudo apt update && sudo apt install -y python3 python3-venv python3-pip curl
```

Node.js için `nodejs.org/en/download` sayfasındaki resmi ikili paketi ya da sürüm yöneticisini (`fnm` / `nvm`) kullanarak `22` sürümünü kur.

2. Terminali aç ve sürümü doğrula

- **Windows:** Başlat menüsüne `PowerShell` veya `Terminal` yazıp aç.
- **macOS:** `Cmd + Space` tuşlarına bas, `Terminal` yazıp aç.
- **Linux:** Uygulama menüsünden `Terminal` aç (`Ctrl + Alt + T`).

Kurduğun aracın sürümünü kontrol et:

```
node --version
npm --version
```

```
python3 --version
```

Beklenen çıktı (sürüm numaralarının son haneleri daha yüksek olabilir):

```
v22.14.0
10.9.2
```

ya da Python için (`Windows`'ta `python --version` da yazabilirsin):

```
Python 3.12.8
```

`command not found` veya `'node' is not recognized` görürsen terminali kapatıp yeniden aç; Python'da Windows kullanıyorsan kurulumda PATH kutusunun işaretli olduğundan emin ol.

3. Proje klasörünü oluştur

Masaüstünde ya da belgelerinde botun için temiz bir klasör aç ve içine gir:

```
mkdir ilk-botum
cd ilk-botum
```

4. `.env` ve `.gitignore` dosyalarını hazırla

Geliştirici portalından aldığın bot token'ını kodun içine değil, aynı klasördeki `.env` dosyasına koy. Yanına da bu dosyanın yanlışlıkla Git'e yüklenmesini engelleyen `.gitignore` dosyasını ekle.

```
BOT_TOKEN=buraya_portaldan_kopyaladigin_token_gelecek
```

```
.env
node_modules/
.venv/
__pycache__/
```

> **Uyarı:** Windows Not Defteri kullanıyorsan dosyayı kaydederken **Kayıt türü: Tüm Dosyalar** seç ki dosyanın adı `.env.txt` olmasın. Terminalde `ls -a` (PowerShell'de `Get-ChildItem -Force`) yazdığında tam olarak `.env` adını görmelisin.

5. `noluyo-bot` kütüphanesini kur

Resmi `noluyo-bot` kütüphanesi Node.js tarafında sıfır çalışma zamanı bağımlılığıyla, Python tarafında ise yalnızca `aiohttp` ile gelir:

```
npm init -y
npm pkg set type=module
npm install noluyo-bot
```

```
python3 -m venv .venv
# macOS / Linux:
source .venv/bin/activate
# Windows PowerShell:
# .\.venv\Scripts\Activate.ps1

pip install noluyo-bot
```

6. İlk çalıştırma ve beklenen çıktı

Klasörün içine küçük bir bağlantı testi dosyası kaydet (`bot.mjs` veya `bot.py`). Bu kod hiçbir ayrıcalıklı intent istemeden bağlanır ve hazır olduğunda botun kullanıcı adını terminale yazar:

```
// bot.mjs
import { Client, Intents } from "noluyo-bot";

const bot = new Client({
  token: process.env.BOT_TOKEN,
  intents: Intents.GUILDS,
});

bot.on("READY", (d) => {
  console.log(`Hazır! ${d.user.username} (${d.user.id}) olarak bağlandı.`);
  console.log(`Bulunduğu sunucu sayısı: ${d.guilds.length}`);
});

await bot.login();
```

```
# bot.py
import os
import noluyo_bot

bot = noluyo_bot.Client(
    token=os.environ["BOT_TOKEN"],
    intents=noluyo_bot.Intents.GUILDS,
)

@bot.on("READY")
async def on_ready(d):
    print(f"Hazır! {d['user']['username']} ({d['user']['id']}) olarak bağlandı.")
    print(f"Bulunduğu sunucu sayısı: {len(d['guilds'])}")

bot.run()
```

Şimdi terminalden çalıştır:

```
node --env-file=.env bot.mjs
```

```
# macOS / Linux:
env $(cat .env | xargs) python3 bot.py
# Windows PowerShell:
# Get-Content .env | ForEach-Object { $k,$v = $_ -split '=',2; Set-Item "Env:$k" $v }; python bot.py
```

Her şey doğruysa birkaç saniye içinde terminalde şu çıktıyı görürsün ve imleç alt satırda beklemeye devam eder:

```
Hazır! karsilama_botu (21879171579904005) olarak bağlandı.
Bulunduğu sunucu sayısı: 1
```

İmlecin beklemeye devam etmesi programın donduğu anlamına gelmez: bot şu an açık WebSocket bağlantısından olayları dinliyor ve sunucunda çevrimiçi görünüyor. Durdurmak istediğinde terminalde `Ctrl + C` tuşlarına bas.

`4004` veya `401` hatası görürsen `.env` içindeki `BOT_TOKEN` değerinin başında ve sonunda boşluk ya da tırnak kalmadığını kontrol et.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekadan kurulum adımı istediğinde Node.js 22+ için yerleşik `--env-file=.env` bayrağını (`dotenv` paketi kurmadan), Python 3.11+ için `.venv` sanal ortamını kullanmasını ve `.gitignore` içine `.env` eklemesini iste.

Sırada ne var

- Tarifler: karşılama, otomatik rol, anket ve moderasyon için kopyalanıp çalıştırılan 9 hazır bot
- JavaScript kütüphanesi veya Python kütüphanesi: resmi SDK'ların tüm özellikleri
- Botu açık tut: bilgisayarını kapattığında da botun çalışmaya devam etmesi

Başlangıç: ilk botun

Birkaç dakika sonra sunucunda kendi botun olacak ve ilk mesajını yazmış olacak. Ardından `!ping` yazılınca yanıt veren küçük bir program yazacağız.

Bunun için bir N'oluyo hesabı, yönettiğin bir sunucu, bir terminal ve `curl` yeterli. Son adımdaki program için Node.js 22 ya da üstü de gerekir; sürümünü `node --version` ile görebilirsin. Sözcükler yabancı geliyorsa önce Temel kavramlar sayfasına göz at.

> **Not:** Uygulama oluşturmak için e-posta adresinin doğrulanmış olması gerekir. Doğrulamadıysan portal bunu söyler; gelen kutundaki bağlantıya tıklaman yeter.

1. Portalı aç

Geliştirici portalına git ve her zamanki gibi N'oluyo hesabınla giriş yap.

2. Uygulamanı oluştur

Yeni uygulama düğmesine bas ve iki şey yaz:

- **Uygulama adı:** Botunun sunucularda görünen adı, en çok 32 karakter.
- **Bot kullanıcı adı:** 4 ile 32 karakter arası; küçük harf, rakam, `_` ve `.` olabilir. Sonradan değiştirilemez.

Sonra **Oluştur**'a bas.

3. Token'ı kopyala

Karşına botun token'ı çıkar. **Kopyala** ile alıp güvenli bir yere kaydet, çünkü **bir daha gösterilmez**. Kaybedersen uygulama sayfasındaki **Token'ı yenile** ile yenisini alırsın.

> **Uyarı:** Token'ı bilen herkes botunu kullanabilir. Kimseyle paylaşma, koda yazıp depoya gönderme. Ayrıntılar Güvenlik sayfasında.

Kaydettikten sonra **Kopyaladım, devam et**'e bas.

4. Botu sunucuna ekle

Uygulama sayfasındaki **Ekleme bağlantısı** bölümünde bota vereceğin izinleri seç. Bu rehber için üçü yeterli:

- Kanalları görüntüle
- Mesaj gönder
- Mesaj geçmişini oku

Bağlantı `izin=68608` ile biter. **Bağlantıyı aç**'a bas, açılan sayfada **Sunucu** listesinden sunucunu seç ve **Sunucuya ekle**'ye bas.

Bot artık sunucunda. Kendi adını taşıyan bir rol de aldı; seçtiğin izinler bu rolde durur.

5. İlk isteğini gönder

Şimdi botu **REST** ile konuşturacaksın. REST, programının belirli bir adrese HTTP isteği gönderip yanıt aldığı yoldur; bir sayfa açmaya benzer, ama yanıt JSON olarak gelir. Buradan sonra tek ihtiyacın token. Terminalde tanımla:

```
export BOT_TOKEN='kopyaladigin_token'
```

Önce token'ın çalıştığını deneyelim: bu istek botun kendi bilgilerini döndürür. Başlıktaki `Bot` sözcüğüne dikkat et:

```
curl "https://noluyo.net/api/v1/users/@me" \
  -H "Authorization: Bot $BOT_TOKEN"
```

Botun bulunduğu sunucuları listele ve sunucunun `id` değerini al. API'de sunucuya `guild` denir.

```
curl "https://noluyo.net/api/v1/users/@me/guilds" \
  -H "Authorization: Bot $BOT_TOKEN"

export GUILD_ID='sunucu id degeri'
```

Kanalları listele ve `"type": 0` olan (metin) bir kanalın `id` değerini al:

```
curl "https://noluyo.net/api/v1/guilds/$GUILD_ID/channels" \
  -H "Authorization: Bot $BOT_TOKEN"

export CHANNEL_ID='kanal id degeri'
```

Şimdi mesajı gönder:

```
curl -i -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"content":"Merhaba, ben yeni botum!"}'
```

```
const { BOT_TOKEN, CHANNEL_ID } = process.env;

const res = await fetch(`https://noluyo.net/api/v1/channels/${CHANNEL_ID}/messages`, {
  method: "POST",
  headers: {
    Authorization: `Bot ${BOT_TOKEN}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({ content: "Merhaba, ben yeni botum!" }),
});
console.log(res.status, await res.json());
```

Mesaj kanalda görünür. Yanıt `200` ile mesaj nesnesi olarak gelir. `-i` sayesinde `X-RateLimit-*` başlıklarını da görürsün; bunları Hız sınırları sayfasında açıklıyoruz.

6. Biraz daha dene

Önceki yanıttaki `id` değerini al:

```
export MESSAGE_ID='yanittaki id degeri'
```

Emoji adreste URL kodlanmış olmalıdır. `%F0%9F%91%8B` , 👋 emojisidir. Portaldaki ekleme bağlantısında **Tepki ekle**'yi işaretlemediysen tepki izni sunucunun `@everyone` rolünden gelir; sunucuda kapatılmışsa `403 missing_permissions` alırsın.

```
# Tepki ver (204)
curl -i -X PUT
"https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages/$MESSAGE_ID/reactions/%F0%9F%91%8B/@me"
\
  -H "Authorization: Bot $BOT_TOKEN"

# Düzenle (200)
curl -X PATCH "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages/$MESSAGE_ID" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"content":"Merhaba, bu mesaj duzenlendi."}'

# Son mesajları oku (200)
curl "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages?limit=5" \
  -H "Authorization: Bot $BOT_TOKEN"

# Sil (204)
curl -i -X DELETE "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages/$MESSAGE_ID" \
  -H "Authorization: Bot $BOT_TOKEN"
```

7. Küçük bir bot yaz

Şimdiye kadar her isteği sen gönderdin. Gerçek bir bot ise kanalda olanları duyar ve kendisi yanıt verir. Bunu gateway sağlar: programın N'oluyo'ya bir `WebSocket` bağlantısı açar, yeni mesajlar bu bağlantıdan anında gelir.

Önce Mesaj içeriği'ni aç. Bot, `!ping` yazıldığını ancak mesajların metnini okuyabilirse anlar ve bu ayrı bir intent'tir. **Intent**, botun hangi olayları almak istediğini söylediği seçimdir. Portalda uygulama sayfasına dön, **Ayrıcalıklı intent'ler** bölümünde **Mesaj içeriği**'ni aç. Açmazsan program `4014` koduyla kapanır.

Örnek Node.js 22 ya da üstü ister: bu sürümlerde `WebSocket` hazırdır, ek paket gerekmez. `bot.mjs` adıyla kaydet:

```
const API = "https://noluyo.net/api/v1";
const { BOT_TOKEN } = process.env;
const headers = { Authorization: `Bot ${BOT_TOKEN}`, "Content-Type": "application/json" };

// Sunucu mesajları (1 << 9) ve mesaj içeriği (1 << 15).
const INTENTS = (1 << 9) | (1 << 15);
// Bu kodlarda yeniden bağlanmak işe yarar; diğerlerinde kodu düzeltmek gerekir.
const RETRY = [4000, 4008, 4010];

const res = await fetch(`${API}/gateway/bot`, { headers });
if (!res.ok) throw new Error(`GET /gateway/bot: ${res.status}`);
const { url } = await res.json();

let me;

async function reply(message) {
  const sent = await fetch(`${API}/channels/${message.channel_id}/messages`, {
    method: "POST",
    headers,
    body: JSON.stringify({ content: "pong!", message_reference: { message_id: message.id } }),
  });
  if (!sent.ok) console.error("yanıt gitmedi", sent.status, await sent.text());
}

function connect() {
  const ws = new WebSocket(url);
  let heartbeat;
  let seq = null;
  const send = (op, d) => {
    if (ws.readyState === WebSocket.OPEN) ws.send(JSON.stringify({ op, d }));
  };

  ws.onerror = () => {};

  ws.onmessage = (e) => {
    const { op, d, s, t } = JSON.parse(e.data);
    if (s) seq = s;
    if (op === 10) {
      heartbeat = setInterval(() => send(1, seq), d.heartbeat_interval);
      send(2, { token: BOT_TOKEN, intents: INTENTS });
    } else if (t === "READY") {
      me = d.user.id;
      console.log(d.user.username, "bağlandı");
    } else if (t === "MESSAGE_CREATE" && d.author.id !== me && d.content === "!ping") {
      reply(d).catch(console.error);
    }
  };
```

```
  ws.onclose = (e) => {
    clearInterval(heartbeat);
    console.error("bağlantı kapandı", e.code, e.reason);
    if (e.code >= 4000 && !RETRY.includes(e.code)) process.exit(1);
    setTimeout(connect, 5000);
  };
}

connect();
```

```
BOT_TOKEN="$BOT_TOKEN" node bot.mjs
```

Terminalde botun adını görünce kanala `!ping` yaz, bot yanıt versin.

Bağlantıdan gidip gelen her JSON mesajına **çerçeve** denir. Çerçevenin `op` alanındaki sayı, onun ne olduğunu söyler: `op 10` sunucunun selamı, `op 1` heartbeat, `op 2` kendini tanıtma (identify). Kod sırayla şunları yapar:

1. `GET /gateway/bot` ile bağlanacağı adresi sorar. Bu adresi koda yazma, buradan öğren.
2. Bağlantı açılınca gelen `op 10` çerçevesindeki aralıkla heartbeat göndermeye başlar. Bu, bağlantının canlı kalmasını sağlar.
3. `op 2` (identify) ile token'ını ve istediği olayları söyler.
4. `READY` gelince kendi ID'sini öğrenir. Bot kendi mesajlarını da duyduğu için bu gerekli.
5. Her `MESSAGE_CREATE`'te metne bakar, `!ping` ise REST ile yanıt gönderir. Yanıt için **Mesaj geçmişini oku** izni gerekir; kanalın mesajlarını duyması için de **Kanalları görüntüle**.

Bağlantı koparsa bu küçük bot 5 saniye sonra baştan bağlanır ve arada yazılanları kaçırır. Kaçırmamak için oturumu sürdürmek gerekir; Gateway sayfası bunu anlatır ve sonunda hazır bir istemci verir.

> **Not: Mesaj içeriği**'ni açmak istemezsen `INTENTS` değerini `1 << 9` yap, koşuldaki `d.content === "!ping"` yerine `d.content.includes("!ping")` yaz ve kanala `@botun !ping` diye botu etiketleyerek yaz. Bu intent olmadan bot yalnızca kendisini etiketleyen mesajların metnini görür, diğerlerinde `content` boş gelir. Kural Intent'ler sayfasında.

REST hâlâ gerekli: gateway yalnızca o andan sonra olanları anlatır ve yalnızca bot dinler. Mesaj göndermek, silmek ve geçmiş mesajları okumak REST işidir. Geçmiş okumalar da aynı içerik kuralına uyar.

Sonraki adım: ilk butonun

Botun mesajına buton da ekleyebilir. Bu istek kanala tek butonlu bir mesaj atar:

```
curl -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"content":"Hazır mısın?","components":[{"type":1,"components":
[{"type":2,"style":1,"label":"Hazırım","custom_id":"hazir"}]}]}'
```

Butona şimdi basarsan birkaç saniye sonra "Bot yanıt vermedi." görürsün: basma botuna `INTERACTION_CREATE` olarak gider ve bot 3 saniye içinde yanıt vermelidir. Yanıt veren ilk botun için Örnekler sayfasındaki anketle başla. Akışın tamamı Bileşenler, yanıt türleri Etkileşimler, kartlar Embed'ler sayfasında.

Resmi kütüphaneler

Ham HTTP ve WebSocket çerçevelerini kendin yönetmek yerine bağlantı sürdürmeyi (`resume`), hız sınırı kuyruklarını ve shard aralıklarını senin yerine yöneten resmi paketleri kullanabilirsin:

- Node.js 22+ için: JavaScript kütüphanesi
- Python 3.11+ için: Python kütüphanesi
- Kopyala-çalıştır dokuz örnek: Tarifler

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu hızlı başlangıç sayfasını bir yapay zeka asistanına verirken şu kuralları ekle:
>
> - Taban adres `https://noluyo.net/api/v1` olmalı; bot token'ı koda yazılmadan `process.env.BOT_TOKEN` ile okunmalıdır.
> - Mesaj beklemek için döngüde kanal okunmaz (`fetch` ile yoklama yapılmaz); canlı olaylar WebSocket gateway ya da resmi `noluyo-bot` kütüphanesi üzerinden dinlenir.
> - `MESSAGE_CONTENT` açık değilse etiketlenmeyen mesajlarda `content` boş gelir (`4014` hatası almamak için ayrıcalıklı intent önce portalda açılır).

Sırada ne var

- Gateway: bağlantının tamamı, kopunca sürdürmek, hazır istemci
- Ses: ses kanalına girip dosya ya da akış yayınlamak; bot kimseyi duyamaz
- Intent'ler: hangi olayları istediğin, mesaj içeriği kuralı
- Kimlik doğrulama: token ve istek kuralları
- Uygulamalar: portal, bot rolü, herkese açık botlar
- İzinler: izin değerleri ve rol sıralaması
- Hız sınırları: sınırlar ve 429 yanıtları
- Hatalar: tüm hata kodları
- Değişiklikler: API'de neler değişti
- Örnekler: zamanlı duyuru, sözcük filtresi, butonlu anket ve rol seçici gibi küçük botlar
- REST referansı: tüm rotalar; her birinde örnek istek, örnek yanıt ve hata kodları var

Tarifler

Bu sayfadaki dokuz tarif, sunucularda en sık istenen bot işlerinin baştan sona çalışan halleridir. Her tarif hem **JavaScript** (`noluyo-bot`, Node.js 22+) hem **Python** (`noluyo_bot`, Python 3.11+) sekmesiyle yazılmıştır; istediğin sekmeyi kopyalayıp `.env` dosyanla doğrudan çalıştırabilirsin.

Kütüphaneyi henüz kurmadıysan önce Kurulum sayfasındaki beş dakikalık hazırlığı tamamla. Paket kurmadan düz `fetch` ve `WebSocket` kullanan örnekler istersen Örnekler sayfasına; kart, buton, form, komut ve ses ayrıntıları için Embed'ler, Bileşenler, Etkileşimler, Komutlar, Formlar ve Ses rehberlerine bak.

1. Hoş geldin mesajı

Sunucuya yeni bir üye katıldığında belirlediğin kanala karşılama kartı gönderir.

- **Gereken intent:** `GUILD_MEMBERS` (`2`) — **ayrıcalıklıdır**, portalda **Sunucu üyeleri**'ni aç.
- **Gereken izinler:** `ViewChannel` + `SendMessages` (`izin=3072`).
- **Hız sınırı notu:** `bot_message` sınıfına tabidir (kanal başına 5 saniyede 5 mesaj); toplu girişlerde resmi SDK bucket kuyruğu istekleri sıraya dizer.

```
import { Client, Intents } from "noluyo-bot";

const bot = new Client({
  token: process.env.BOT_TOKEN,
  intents: Intents.GUILD_MEMBERS,
});

bot.on("GUILD_MEMBER_ADD", async (member) => {
  await bot.rest.createMessage(process.env.WELCOME_CHANNEL_ID, {
    content: `<@${member.user.id}> aramıza katıldı!`,
    embeds: [
      {
        title: "Hoş geldin!",
        description: "Kurallar kanalına göz atmayı ve kendini tanıtmayı unutma.",
        color: 0x3b82f6,
      },
    ],
  });
});

await bot.login();
```

```python
import os
import noluyo_bot

bot = noluyo_bot.Client(
    token=os.environ["BOT_TOKEN"],
    intents=noluyo_bot.Intents.GUILD_MEMBERS,
)

@bot.on("GUILD_MEMBER_ADD")
async def on_member_join(member):
    await bot.rest.create_message(
        os.environ["WELCOME_CHANNEL_ID"],
        {
            "content": f"<@{member['user']['id']}> aramıza katıldı!",
            "embeds": [
                {
                    "title": "Hoş geldin!",
                    "description": "Kurallar kanalına göz atmayı ve kendini tanıtmayı unutma.",
                    "color": 0x3B82F6,
                }
            ],
        },
    )

bot.run()
```

2. Otomatik rol verme

Sunucuya katılan her yeni üyeye (botlar hariç) belirlediğin başlangıç rolünü kendiliğinden verir.

- **Gereken intent:** `GUILD_MEMBERS` (`2`) — portalda **Sunucu üyeleri** açık olmalıdır.
- **Gereken izinler:** `ManageRoles` (`izin=268435456`). Sunucu ayarlarında **Roller** listesinde botun kendi rolünü, vereceği rolün **üstüne** sürükle; aksi halde `403 role_hierarchy` döner.
- **Hız sınırı notu:** `bot_write` sınıfındadır; SDK `X-RateLimit-Reset-After` süresine kendiliğinden uyar.

```
import { Client, Intents } from "noluyo-bot";

const bot = new Client({
  token: process.env.BOT_TOKEN,
  intents: Intents.GUILD_MEMBERS,
});

bot.on("GUILD_MEMBER_ADD", async (member) => {
  if (member.user.bot) return;
  await bot.rest.addMemberRole(member.guild_id, member.user.id, process.env.AUTO_ROLE_ID);
});

await bot.login();
```

```
import os
import noluyo_bot

bot = noluyo_bot.Client(
    token=os.environ["BOT_TOKEN"],
    intents=noluyo_bot.Intents.GUILD_MEMBERS,
)

@bot.on("GUILD_MEMBER_ADD")
async def on_member_join(member):
    if member["user"].get("bot"):
        return
    await bot.rest.add_member_role(
        member["guild_id"],
        member["user"]["id"],
        os.environ["AUTO_ROLE_ID"],
    )

bot.run()
```

3. Butonla rol seçici

`/rol-paneli` komutuyla kanala butonlu bir mesaj bırakır; butona basan üyede rol yoksa verir, varsa geri alır ve sonucu yalnızca basan kişiye (`ephemeral`) gösterir.

- **Gereken intent:** `0` (`INTERACTION_CREATE` hiçbir intent istemez).
- **Gereken izinler:** `ViewChannel` + `SendMessages` + `ManageRoles` (`izin=268438528`), ayrıca botun rolü verilecek rolün üstünde olmalıdır.
- **Hız sınırı notu:** İlk etkileşim yanıtı (`i.reply`) **3 saniye içinde** gönderilmelidir; bu yüzden önce rol işlemi değil, ya hızlı yanıt ya da ertelenmiş yanıt (`defer`) kullanılır.

```
import { Client } from "noluyo-bot";

const ROLE_ID = process.env.PICKER_ROLE_ID;
const bot = new Client({ token: process.env.BOT_TOKEN, intents: 0 });

bot.on("READY", async () => {
  await bot.commands.sync([
    {
      name: "rol-paneli",
      description: "Duyuru rolü seçme butonunu gönderir",
      default_member_permissions: "268435456",
      contexts: [0],
    },
  ]);
});

bot.interactions.on("rol-paneli", async (i) => {
  await i.reply({
    content: "Duyuru bildirimlerini açıp kapatmak için aşağıdaki butona bas:",
    components: [
      {
        type: 1,
        components: [{ type: 2, style: 1, label: "Duyuru Rolü", custom_id: "toggle_duyuru" }],
      },
    ],
  });
});

bot.interactions.on("toggle_duyuru", async (i) => {
  await i.defer({ ephemeral: true });
  const hasRole = i.member.roles.includes(ROLE_ID);
  if (hasRole) {
    await bot.rest.removeMemberRole(i.guild_id, i.member.user.id, ROLE_ID);
    await i.editReply({ content: "Duyuru rolü senden alındı." });
  } else {
    await bot.rest.addMemberRole(i.guild_id, i.member.user.id, ROLE_ID);
    await i.editReply({ content: "Duyuru rolü sana verildi." });
  }
});

await bot.login();
```

```
import os
import noluyo_bot

ROLE_ID = os.environ["PICKER_ROLE_ID"]
bot = noluyo_bot.Client(token=os.environ["BOT_TOKEN"], intents=0)

@bot.on("READY")
async def on_ready(_):
    await bot.commands.sync(
        [
            {
                "name": "rol-paneli",
                "description": "Duyuru rolü seçme butonunu gönderir",
                "default_member_permissions": "268435456",
                "contexts": [0],
            }
        ]
    )

@bot.interactions.on("rol-paneli")
async def rol_paneli(i):
    await i.reply(
        {
            "content": "Duyuru bildirimlerini açıp kapatmak için aşağıdaki butona bas:",
            "components": [
                {
                    "type": 1,
                    "components": [
                        {"type": 2, "style": 1, "label": "Duyuru Rolü", "custom_id":
"toggle_duyuru"}
                    ],
                }
            ],
        }
    )

@bot.interactions.on("toggle_duyuru")
async def toggle_duyuru(i):
    await i.defer(ephemeral=True)
    has_role = ROLE_ID in i.member["roles"]
    user_id = i.member["user"]["id"]
    if has_role:
        await bot.rest.remove_member_role(i.guild_id, user_id, ROLE_ID)
        await i.edit_reply({"content": "Duyuru rolü senden alındı."})
    else:
        await bot.rest.add_member_role(i.guild_id, user_id, ROLE_ID)
        await i.edit_reply({"content": "Duyuru rolü sana verildi."})
```

```
bot.run()
```

4. Sözcük süzgeci

Kanalda yazılan mesajlarda yasaklı listedeki sözcüklerden biri geçerse mesajı siler ve kısa bir uyarı bırakır.

- **Gereken intent'ler:** `GUILD_MESSAGES | MESSAGE_CONTENT` (`33280`) — portalda **Mesaj içeriği** açık olmalıdır.
- **Gereken izinler:** `ViewChannel` + `SendMessages` + `ManageMessages` + `ReadMessageHistory` (`izin=76800`).
- **Hız sınırı notu:** Tek tek silme `bot_write` , uyarı mesajı `bot_message` sınıfındadır; bot kendi mesajlarını görmezden gelmelidir (`author.bot` kontrolü) yoksa sonsuz döngüye girer.

```
import { Client, Intents } from "noluyo-bot";

const WORDS = (process.env.BLOCKED_WORDS ?? "reklam,bedava ödül")
  .split(",")
  .map((w) => w.trim().toLocaleLowerCase("tr-TR"))
  .filter(Boolean);

const bot = new Client({
  token: process.env.BOT_TOKEN,
  intents: Intents.GUILD_MESSAGES | Intents.MESSAGE_CONTENT,
});

bot.on("MESSAGE_CREATE", async (msg) => {
  if (msg.author.bot || !msg.content) return;
  const text = msg.content.toLocaleLowerCase("tr-TR");
  if (!WORDS.some((w) => text.includes(w))) return;

  await bot.rest.deleteMessage(msg.channel_id, msg.id);
  const warn = await bot.rest.createMessage(msg.channel_id, {
    content: `<@${msg.author.id}> bu sözcük sunucuda yasak.`,
  });
  setTimeout(() => {
    bot.rest.deleteMessage(msg.channel_id, warn.id).catch(() => {});
  }, 8000);
});

await bot.login();
```

```
import asyncio
import os
import noluyo_bot

WORDS = [
    w.strip().lower()
    for w in os.environ.get("BLOCKED_WORDS", "reklam,bedava ödül").split(",")
    if w.strip()
]

bot = noluyo_bot.Client(
    token=os.environ["BOT_TOKEN"],
    intents=noluyo_bot.Intents.GUILD_MESSAGES | noluyo_bot.Intents.MESSAGE_CONTENT,
)

@bot.on("MESSAGE_CREATE")
async def on_message(msg):
    if msg["author"].get("bot") or not msg.get("content"):
        return
    text = msg["content"].lower()
    if not any(w in text for w in WORDS):
        return

    await bot.rest.delete_message(msg["channel_id"], msg["id"])
    warn = await bot.rest.create_message(
        msg["channel_id"],
        {"content": f"<@{msg['author']['id']}> bu sözcük sunucuda yasak."},
    )
    await asyncio.sleep(8)
    try:
        await bot.rest.delete_message(msg["channel_id"], warn["id"])
    except noluyo_bot.ApiError:
        pass

bot.run()
```

5. Zamanlı hatırlatma ve duyuru

Belirlediğin kanala her saat başı (ya da seçtiğin aralıkla) hatırlatma mesajı gönderir.

- **Gereken intent:** `0` (yalnızca REST ile mesaj gönderdiği için gateway olayı dinlemesi bile gerekmez).
- **Gereken izinler:** `ViewChannel` + `SendMessages` (`izin=3072`).
- **Hız sınırı notu:** Kanalda yavaş mod (`slowmode`) açıksa ve aralığın ondan kısaysa `429 slowmode` döner; resmi SDK `retry_after` süresini bekleyip gönderir.

```
import { Client } from "noluyo-bot";

const bot = new Client({ token: process.env.BOT_TOKEN, intents: 0 });
const CHANNEL_ID = process.env.REMINDER_CHANNEL_ID;
const INTERVAL_MS = 60 * 60 * 1000; // 1 saat

async function sendReminder() {
  try {
    await bot.rest.createMessage(CHANNEL_ID, {
      content: "Saatlik hatırlatma: su içmeyi ve kısa bir mola vermeyi unutma!",
    });
  } catch (err) {
    console.error("Hatırlatma gönderilemedi:", err.message);
  }
}

await sendReminder();
setInterval(sendReminder, INTERVAL_MS);
```

```
import asyncio
import os
import noluyo_bot

CHANNEL_ID = os.environ["REMINDER_CHANNEL_ID"]
INTERVAL_SECONDS = 3600  # 1 saat

async def main():
    bot = noluyo_bot.Client(token=os.environ["BOT_TOKEN"], intents=0)
    try:
        while True:
            try:
                await bot.rest.create_message(
                    CHANNEL_ID,
                    {"content": "Saatlik hatırlatma: su içmeyi ve kısa bir mola vermeyi
unutma!"},
                )
            except noluyo_bot.ApiError as err:
                print("Hatırlatma gönderilemedi:", err)
            await asyncio.sleep(INTERVAL_SECONDS)
    finally:
        await bot.close()

asyncio.run(main())
```

6. Butonlu anket

`/anket` komutuyla iki seçenekli bir oylama kartı açar; her üyenin tek oyu vardır, diğer butona basarsa oyu taşınır ve kart anında güncellenir.

- **Gereken intent:** `0`.
- **Gereken izinler:** `ViewChannel` + `SendMessages` (`izin=3072`).
- **Hız sınırı notu:** Buton yanıtları `interaction` sınıfındadır; mesajı ayrı bir `PATCH` yerine doğrudan etkileşim yanıtı (`i.update` / `type: 7`) ile güncellemek ek mesaj kotası harcamaz.

```
import { Client } from "noluyo-bot";

const polls = new Map();
let nextPollId = 1;
const bot = new Client({ token: process.env.BOT_TOKEN, intents: 0 });

function renderPoll(id, poll) {
  return {
    embeds: [
      {
        title: poll.question,
        fields: [
          { name: poll.a, value: `${poll.votesA.size} oy`, inline: true },
          { name: poll.b, value: `${poll.votesB.size} oy`, inline: true },
        ],
        color: 0x10b981,
      },
    ],
    components: [
      {
        type: 1,
        components: [
          { type: 2, style: 1, label: poll.a, custom_id: `vote:${id}:a` },
          { type: 2, style: 2, label: poll.b, custom_id: `vote:${id}:b` },
        ],
      },
    ],
  };
}

bot.on("READY", async () => {
  await bot.commands.sync([
    {
      name: "anket",
      description: "İki seçenekli butonlu anket başlatır",
      contexts: [0],
      options: [
        { type: 3, name: "soru", description: "Anket sorusu", required: true },
        { type: 3, name: "secenek_a", description: "Birinci seçenek", required: true },
        { type: 3, name: "secenek_b", description: "İkinci seçenek", required: true },
      ],
    },
  ]);
});

bot.interactions.on("anket", async (i) => {
  const id = String(nextPollId++);
  const poll = {
```

```
    question: String(i.option("soru")),
    a: String(i.option("secenek_a")),
    b: String(i.option("secenek_b")),
    votesA: new Set(),
    votesB: new Set(),
  };
  polls.set(id, poll);
  await i.reply(renderPoll(id, poll));
});

bot.interactions.on(/^vote:/, async (i) => {
  const [, id, side] = i.data.custom_id.split(":");
  const poll = polls.get(id);
  if (!poll) return i.reply({ content: "Bu anket sona erdi.", ephemeral: true });
  const uid = i.person.id;
  poll.votesA.delete(uid);
  poll.votesB.delete(uid);
  (side === "a" ? poll.votesA : poll.votesB).add(uid);
  await i.update(renderPoll(id, poll));
});

await bot.login();
```

```
import itertools
import os
import re
import noluyo_bot

polls = {}
poll_seq = itertools.count(1)
bot = noluyo_bot.Client(token=os.environ["BOT_TOKEN"], intents=0)

def render_poll(pid, poll):
    return {
        "embeds": [
            {
                "title": poll["question"],
                "fields": [
                    {"name": poll["a"], "value": f"{len(poll['votes_a'])} oy", "inline": True},
                    {"name": poll["b"], "value": f"{len(poll['votes_b'])} oy", "inline": True},
                ],
                "color": 0x10B981,
            }
        ],
        "components": [
            {
                "type": 1,
                "components": [
                    {"type": 2, "style": 1, "label": poll["a"], "custom_id": f"vote:{pid}:a"},
                    {"type": 2, "style": 2, "label": poll["b"], "custom_id": f"vote:{pid}:b"},
                ],
            }
        ],
    }

@bot.on("READY")
async def on_ready(_):
    await bot.commands.sync(
        [
            {
                "name": "anket",
                "description": "İki seçenekli butonlu anket başlatır",
                "contexts": [0],
                "options": [
                    {"type": 3, "name": "soru", "description": "Anket sorusu", "required": True},
                    {"type": 3, "name": "secenek_a", "description": "Birinci seçenek",
"required": True},
                    {"type": 3, "name": "secenek_b", "description": "İkinci seçenek", "required":
True},
                ],
            }
```

```
        ]
    )

@bot.interactions.on("anket")
async def create_poll(i):
    pid = str(next(poll_seq))
    poll = {
        "question": str(i.option("soru")),
        "a": str(i.option("secenek_a")),
        "b": str(i.option("secenek_b")),
        "votes_a": set(),
        "votes_b": set(),
    }
    polls[pid] = poll
    await i.reply(render_poll(pid, poll))

@bot.interactions.on(re.compile(r"^vote:"))
async def handle_vote(i):
    _, pid, side = i.data["custom_id"].split(":", 2)
    poll = polls.get(pid)
    if not poll:
        await i.reply({"content": "Bu anket sona erdi."}, ephemeral=True)
        return
    uid = i.person["id"]
    poll["votes_a"].discard(uid)
    poll["votes_b"].discard(uid)
    poll["votes_a" if side == "a" else "votes_b"].add(uid)
    await i.update(render_poll(pid, poll))

bot.run()
```

7. Basit seviye ve mesaj sayacı

Üyelerin sunucuda kaç mesaj yazdığını sayar (mesajların metnini okumadan!) ve `/seviye` komutu sorulduğunda kişinin puanını ve seviyesini gösterir.

- **Gereken intent:** `GUILD_MESSAGES` (`512`) — metin okunmadığı için ayrıcalıklı `MESSAGE_CONTENT` intent'ine **ihtiyaç yoktur**!
- **Gereken izinler:** `ViewChannel` + `SendMessages` (`izin=3072`).
- **Hız sınırı notu:** Her mesaja kanalda otomatik "Seviye atladın!" yazmak kalabalık kanallarda `bot_message` kotasını doldurur ve sohbeti böler; sayacı sessizce tutup `/seviye` komutuyla göstermek çok daha temizdir.

```
import { Client, Intents } from "noluyo-bot";

const counts = new Map();
const bot = new Client({
  token: process.env.BOT_TOKEN,
  intents: Intents.GUILD_MESSAGES,
});

bot.on("READY", async () => {
  await bot.commands.sync([
    { name: "seviye", description: "Mesaj sayını ve seviyeni gösterir", contexts: [0] },
  ]);
});

bot.on("MESSAGE_CREATE", (msg) => {
  if (msg.author.bot || !msg.guild_id) return;
  const key = `${msg.guild_id}:${msg.author.id}`;
  counts.set(key, (counts.get(key) ?? 0) + 1);
});

bot.interactions.on("seviye", async (i) => {
  const key = `${i.guild_id}:${i.member.user.id}`;
  const total = counts.get(key) ?? 0;
  const level = Math.floor(Math.sqrt(total / 5)) + 1;
  await i.reply({
    content: `Toplam mesajın: **${total}** — Seviyen: **${level}**`,
    ephemeral: true,
  });
});

await bot.login();
```

```
import math
import os
import noluyo_bot

counts = {}
bot = noluyo_bot.Client(
    token=os.environ["BOT_TOKEN"],
    intents=noluyo_bot.Intents.GUILD_MESSAGES,
)

@bot.on("READY")
async def on_ready(_):
    await bot.commands.sync(
        [{"name": "seviye", "description": "Mesaj sayını ve seviyeni gösterir", "contexts": [0]}]
    )

@bot.on("MESSAGE_CREATE")
async def on_message(msg):
    if msg["author"].get("bot") or not msg.get("guild_id"):
        return
    key = f"{msg['guild_id']}:{msg['author']['id']}"
    counts[key] = counts.get(key, 0) + 1

@bot.interactions.on("seviye")
async def seviye(i):
    key = f"{i.guild_id}:{i.member['user']['id']}"
    total = counts.get(key, 0)
    level = int(math.sqrt(total / 5)) + 1
    await i.reply(
        {
            "content": f"Toplam mesajın: **{total}** — Seviyen: **{level}**",
            "ephemeral": True,
        }
    )

bot.run()
```

8. Duyuru komutu (`/duyuru`)

Yalnızca `ManageGuild` yetkisi olan yöneticilerin kullanabildiği `/duyuru` komutuyla seçilen yazı kanalına düzenli bir duyuru embed'i gönderir.

- **Gereken intent:** `0`.
- **Gereken izinler:** Hedef kanalda `ViewChannel` + `SendMessages` (`izin=3072`).
- **Hız sınırı notu:** Binlerce üyeli sunucularda çok büyük embed göndermek ortak yayılma bütçesine (`fanout_too_large`) takılabilir; duyuru metnini öz ve okunaklı tut.

```
import { Client } from "noluyo-bot";

const bot = new Client({ token: process.env.BOT_TOKEN, intents: 0 });

bot.on("READY", async () => {
  await bot.commands.sync([
    {
      name: "duyuru",
      description: "Seçilen kanala resmi duyuru kartı gönderir",
      default_member_permissions: "32", // ManageGuild
      contexts: [0],
      options: [
        { type: 7, name: "kanal", description: "Duyuru kanalı", required: true, channel_types:
[0] },
        { type: 3, name: "baslik", description: "Duyuru başlığı", required: true },
        { type: 3, name: "metin", description: "Duyuru metni", required: true },
      ],
    },
  ]);
});

bot.interactions.on("duyuru", async (i) => {
  const channelId = String(i.option("kanal"));
  const title = String(i.option("baslik"));
  const description = String(i.option("metin"));

  await i.defer({ ephemeral: true });
  await bot.rest.createMessage(channelId, {
    embeds: [
      {
        title,
        description,
        color: 0xf59e0b,
        footer: { text: `Duyuran: ${i.member.user.username}` },
      },
    ],
    allowed_mentions: { parse: [] },
  });
  await i.editReply({ content: `Duyuru <#${channelId}> kanalına gönderildi.` });
});

await bot.login();
```

```
import os
import noluyo_bot

bot = noluyo_bot.Client(token=os.environ["BOT_TOKEN"], intents=0)

@bot.on("READY")
async def on_ready(_):
    await bot.commands.sync(
        [
            {
                "name": "duyuru",
                "description": "Seçilen kanala resmi duyuru kartı gönderir",
                "default_member_permissions": "32",  # ManageGuild
                "contexts": [0],
                "options": [
                    {
                        "type": 7,
                        "name": "kanal",
                        "description": "Duyuru kanalı",
                        "required": True,
                        "channel_types": [0],
                    },
                    {"type": 3, "name": "baslik", "description": "Duyuru başlığı", "required":
True},
                    {"type": 3, "name": "metin", "description": "Duyuru metni", "required":
True},
                ],
            }
        ]
    )

@bot.interactions.on("duyuru")
async def duyuru(i):
    channel_id = str(i.option("kanal"))
    title = str(i.option("baslik"))
    description = str(i.option("metin"))

    await i.defer(ephemeral=True)
    await bot.rest.create_message(
        channel_id,
        {
            "embeds": [
                {
                    "title": title,
                    "description": description,
                    "color": 0xF59E0B,
                    "footer": {"text": f"Duyuran: {i.member['user']['username']}"},
                }
```

```
            ],
            "allowed_mentions": {"parse": []},
        },
    )
    await i.edit_reply({"content": f"Duyuru <#{channel_id}> kanalına gönderildi."})

bot.run()
```

9. Zaman aşımı moderasyon komutu (`/sustur`)

`ModerateMembers` izni olan moderatörlerin `/sustur` komutuyla bir üyeyi belirlenen dakika boyunca zaman aşımına almasını sağlar.

- **Gereken intent:** `0`.
- **Gereken izinler:** `ModerateMembers` (`izin=1099511627776`). Botun rolü, susturulacak üyenin en yüksek rolünden yukarıda olmalıdır; sunucu sahibi ve `Administrator` olanlar susturulamaz (`403 cannot_timeout`).
- **Hız sınırı notu:** `PATCH /guilds/{guild_id}/members/{user_id}` isteği `bot_write` sınıfındadır; komut kaydetme (`commands.sync`) ise uygulama başına dakikada 20 oluşturma sınırına sahip olduğundan yalnızca başlangıçta bir kez çağrılmalıdır.

```
import { ApiError, Client } from "noluyo-bot";

const bot = new Client({ token: process.env.BOT_TOKEN, intents: 0 });

bot.on("READY", async () => {
  await bot.commands.sync([
    {
      name: "sustur",
      description: "Bir üyeyi belirtilen dakika boyunca zaman aşımına alır",
      default_member_permissions: "1099511627776", // ModerateMembers
      contexts: [0],
      options: [
        { type: 6, name: "uye", description: "Susturulacak üye", required: true },
        {
          type: 4,
          name: "dakika",
          description: "Süre (1-1440 dakika)",
          required: true,
          min_value: 1,
          max_value: 1440,
        },
      ],
    },
  ]);
});

bot.interactions.on("sustur", async (i) => {
  const targetId = String(i.option("uye"));
  const minutes = Number(i.option("dakika"));
  const until = new Date(Date.now() + minutes * 60_000).toISOString();

  await i.defer({ ephemeral: true });
  try {
    await bot.rest.updateMember(i.guild_id, targetId, {
      communication_disabled_until: until,
    });
    await i.editReply({ content: `<@${targetId}> ${minutes} dakika boyunca susturuldu.` });
  } catch (err) {
    if (err instanceof ApiError && err.code === "role_hierarchy") {
      await i.editReply({ content: "Bu üyenin rolü benim rolümden yüksek ya da eşit." });
      return;
    }
    throw err;
  }
});

await bot.login();
```

```python
from datetime import datetime, timedelta, timezone
import os
import noluyo_bot

bot = noluyo_bot.Client(token=os.environ["BOT_TOKEN"], intents=0)

@bot.on("READY")
async def on_ready(_):
    await bot.commands.sync(
        [
            {
                "name": "sustur",
                "description": "Bir üyeyi belirtilen dakika boyunca zaman aşımına alır",
                "default_member_permissions": "1099511627776",  # ModerateMembers
                "contexts": [0],
                "options": [
                    {"type": 6, "name": "uye", "description": "Susturulacak üye", "required":
True},
                    {
                        "type": 4,
                        "name": "dakika",
                        "description": "Süre (1-1440 dakika)",
                        "required": True,
                        "min_value": 1,
                        "max_value": 1440,
                    },
                ],
            }
        ]
    )

@bot.interactions.on("sustur")
async def sustur(i):
    target_id = str(i.option("uye"))
    minutes = int(i.option("dakika"))
    until = (datetime.now(timezone.utc) + timedelta(minutes=minutes)).isoformat()

    await i.defer(ephemeral=True)
    try:
        await bot.rest.update_member(
            i.guild_id,
            target_id,
            {"communication_disabled_until": until},
        )
        await i.edit_reply({"content": f"<@{target_id}> {minutes} dakika boyunca susturuldu."})
    except noluyo_bot.ApiError as err:
        if err.code == "role_hierarchy":
            await i.edit_reply({"content": "Bu üyenin rolü benim rolümden yüksek ya da eşit."})
```

```
            return
        raise

bot.run()
```

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu tarifleri birleştirip daha kapsamlı bir bot yazdırırken yapay zekaya şunları hatırlat:
>
> - Birden çok tarifi tek botta toplarken `intents` bitlerini `|` ile birleştir (örneğin `Intents.GUILD_MEMBERS | Intents.GUILD_MESSAGES | Intents.MESSAGE_CONTENT`) ve ayrıcalıklı olanları portalda aç.
> - `bot.commands.sync([...])` dizisine tüm komutları tek seferde ver; ayrı ayrı `sync` çağrısı yapmak önceki komutların üzerine yazar.
> - Rol verme ve susturma işlemlerinde botun kendi rolü hedef rolden veya hedef üyeden yukarıda olmalıdır (`role_hierarchy`).

Sırada ne var

- JavaScript kütüphanesi ve Python kütüphanesi: `Client`, `rest`, `commands`, `interactions` ve `voice` yöntemlerinin tam referansı
- Sık hatalar: rol sırası, boş `content`, `4014` ve etkileşim süresi sorunlarının hızlı çözümü
- Botu açık tut: hazırladığın botu bir Linux sunucuda 7/24 çalıştırmak

Botu açık tut

Botunu kendi bilgisayarında başlattığında, ekranı indirdiğin, bilgisayarı uyku moduna aldığın ya da internetin kesildiği an program durur ve botun çevrimdışı olur. Botun günün her saati yanıt vermesini istiyorsan onu hiç kapanmayan küçük bir Linux sanal sunucuda (**VPS**) çalıştırırsın.

Bir Linux VPS seçmek

Belirli bir barındırma şirketine ihtiyacın yok; herhangi bir standart Linux sanal sunucu sağlayıcısı işini görür. Seçerken şu ölçütlere bak:

- **İşletim sistemi:** Güncel bir Ubuntu LTS ya da Debian kararlı sürümü en sade kurulumu sunar.
- **Bellek ve işlemci:** Metin, komut ve buton kullanan normal bir bot için **1 vCPU ve 512 MB – 1 GB RAM** fazlasıyla yeterlidir. Ses kanalında `ffmpeg` ile ses çözen bir bot çalıştıracaksan en az **1–2 GB RAM** seç.
- **Ağ:** Dışarıya `443` portundan HTTPS ve WebSocket (`wss://noluyo.net`) çıkışı açık olmalıdır. Etkileşimleri HTTP ile teslim modunda almayacaksan sunucuna dışarıdan port açman bile gerekmez; gateway bağlantısını bot içeriden kendisi başlatır.

Ücretsiz katmanların sınırı

Bazı bulut platformları "ücretsiz web servisi" sunar ama gelen HTTP isteği olmadığında konteyneri 15 dakika sonra uyutur. **Uyuyan servis gateway botları için uygun değildir:** WebSocket bağlantısı kesilir, bot çevrimdışı düşer ve uyanıp tekrar bağlanana dek sunucudaki mesajları kaçırır (üstelik sürekli baştan `identify` yaparsa günlük 1000 oturum hakkını da eritir). Yalnızca gerçek anlamda kesintisiz çalışan bir sanal makine kullan ya da etkileşimlerini HTTP ile teslim moduna taşı.

1. Sunucuya SSH ile bağlan ve ayrı kullanıcı aç

Sunucu sağlayıcının verdiği IP adresiyle kendi bilgisayarının terminalinden bağlan:

```
ssh root@203.0.113.10
```

Botu `root` yetkisiyle çalıştırma. Yalnızca bota ait kısıtlı bir sistem kullanıcısı ve klasör oluştur, ayrıca Node.js 22+ veya Python 3.11+ paketlerini kur:

```
adduser --system --group --home /opt/noluyo-bot botuser
apt update && apt install -y git curl python3 python3-venv python3-pip
```

(Ses botu çalıştıracaksan aynı komuta `ffmpeg` paketini de ekle.)

2. Dosyaları sunucuya aktar (`.env` hariç)

Kodunu Git deposunda tutuyorsan (`.env` dosyasının `.gitignore` içinde olduğundan emin olarak!) sunucuya çek; Git kullanmıyorsan kendi bilgisayarından `scp` ile yalnız kod dosyalarını kopyala:

```
sudo -u botuser git clone https://ornek-git-adresin/ilk-botum.git /opt/noluyo-bot/app
cd /opt/noluyo-bot/app
sudo -u botuser npm ci --omit=dev
```

```
# Kendi bilgisayarının terminalinden çalıştır (.env dosyasını kopyalama):
scp package.json package-lock.json bot.mjs root@203.0.113.10:/tmp/
# Sonra sunucuda:
mkdir -p /opt/noluyo-bot/app
mv /tmp/package.json /tmp/package-lock.json /tmp/bot.mjs /opt/noluyo-bot/app/
chown -R botuser:botuser /opt/noluyo-bot/app
cd /opt/noluyo-bot/app && sudo -u botuser npm ci --omit=dev
```

Python kullanıyorsan `/opt/noluyo-bot/app` içinde `sudo -u botuser python3 -m venv .venv` ve `sudo -u botuser .venv/bin/pip install -r requirements.txt` çalıştır.

3. Sunucuda `.env` dosyasını oluştur ve kilitle

Token'ı sunucuda doğrudan `/opt/noluyo-bot/app/.env` içine yaz ve dosya izinlerini yalnız `botuser` okuyabilecek şekilde (`chmod 600`) kilitle:

```
nano /opt/noluyo-bot/app/.env
```

İçine `BOT_TOKEN=...` satırını yapıştırıp kaydet (`Ctrl + O` , `Enter` , `Ctrl + X`), ardından izinleri sıkılaştır:

```
chown botuser:botuser /opt/noluyo-bot/app/.env
chmod 600 /opt/noluyo-bot/app/.env
```

`chmod 600` sayesinde aynı sunucudaki başka kullanıcılar `.env` dosyasını okuyamaz.

4. `systemd` servisi ile arka planda çalıştır

Botu SSH penceresinin içinde düz `node bot.mjs` ile bırakırsan SSH bağlantın koptuğunda süreç kapanır. Linux'un yerleşik `systemd` yöneticisi ise botu arka planda başlatır, sunucu yeniden başlarsa kendiliğinden açar ve beklenmeyen bir hatada birkaç saniye bekleyip yeniden kaldırır.

`/etc/systemd/system/noluyo-bot.service` dosyasını oluştur:

```
[Unit]
Description=N'oluyo Bot
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=botuser
Group=botuser
WorkingDirectory=/opt/noluyo-bot/app
EnvironmentFile=/opt/noluyo-bot/app/.env
ExecStart=/usr/bin/node /opt/noluyo-bot/app/bot.mjs
Restart=on-failure
RestartSec=10
NoNewPrivileges=true
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

> **Not:** Python kullanıyorsan `ExecStart` satırını `/opt/noluyo-bot/app/.venv/bin/python /opt/noluyo-bot/app/bot.py` olarak değiştir. Node'u farklı bir yola kurduysan `which node` çıktısındaki tam yolu yaz. `RestartSec=10` önemlidir: hata anında saniyede yüzlerce kez bağlanmaya çalışıp `4008` veya IP engeli yemeni önler; `4004` ve `4014` gibi kalıcı yapılandırma hatalarında ise resmi SDK çıkış kodunu kontrol etmen için net log basar.

Servisi etkinleştir ve başlat:

```
systemctl daemon-reload
systemctl enable --now noluyo-bot
```

5. Durumu ve canlı logları izlemek

Botun çalışıp çalışmadığını ve `console.log` / hata çıktılarını görmek için:

```
# Anlık durum (active (running) görmelisin)
systemctl status noluyo-bot

# Canlı akan loglar (çıkmak için Ctrl + C)
journalctl -u noluyo-bot -f

# Son 100 log satırı
journalctl -u noluyo-bot -n 100 --no-pager
```

6. Güncellemek, yeniden başlatmak ve durdurmak

Kodunda değişiklik yaptığında sunucuda yeni kodu çekip servisi yeniden başlatman yeterlidir:

```
cd /opt/noluyo-bot/app
sudo -u botuser git pull
sudo -u botuser npm ci --omit=dev
systemctl restart noluyo-bot
```

> **Uyarı:** Aynı token ile çalışan botu **hem kendi bilgisayarında hem sunucuda aynı anda açık bırakma.** İki kopya aynı shard için birbirini `4011` (`TOO_MANY_SESSIONS`) koduyla düşürür ya da aynı mesaja iki kez yanıt verir. Kendi bilgisayarında deneme yaparken ya sunucudaki servisi `systemctl stop noluyo-bot` ile geçici olarak durdur ya da denemelerin için portaldan ayrı bir test uygulaması aç.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekadan sunucu kurulumu isterken `.env` dosyasını `chmod 600` ile korumasını, `systemd` biriminde `root` yerine kısıtlı kullanıcı ve `RestartSec=10` kullanmasını ve geliştirme makinesiyle sunucuda aynı token'ı aynı anda çalıştırmamasını iste.

Sırada ne var

- Güvenlik: token sızıntılarını önlemek, izinleri en azda tutmak ve imza doğrulamak
- Sık hatalar: `4011` , `4014` , `4004` ve bağlantı sorunlarını teşhis etmek
- Doğrulama: botun 100 sunucuya yaklaştığında doğrulama başvurusu yapmak

Temel kavramlar

Bot yazarken karşına önce sekiz sözcük çıkar: uygulama, bot, token, sunucu, kanal, rol, izin ve hız sınırı. Olayları dinlemeye başlayınca üç sözcük daha gelir: gateway, intent ve shard. Mesajına kart ve buton eklediğinde dört sözcük daha: embed, bileşen, etkileşim ve ephemeral mesaj. Bu sayfa her birinin ne olduğunu ve birbirine nasıl bağlandığını anlatır. Daha önce bot yazmadıysan buradan başla.

> **Mimari özeti (Uygulama -> Bot -> Sunucu):**
>
> 1. **Geliştirici Portalı (`https://noluyo.net/gelistirici/uygulamalar`):** Bir uygulama açtığında bot hesabı, gizli `BOT_TOKEN` ve ekleme bağlantısı üretilir.
> 2. **Bot Programın:** `BOT_TOKEN` değerini `.env` dosyasında tutar, canlı olaylar için Gateway WebSocket bağlantısı açar ve `https://noluyo.net/api/v1` adresine `Authorization: Bot <TOKEN>` başlığıyla hız sınırına tabi REST istekleri gönderir.
> 3. **Sunucu (`guild`):** Bot ekleme bağlantısıyla sunucuya üye olarak katılır ve kendi rolünü alır; kanal izin ayarları bu rolün izinlerini daraltabilir.

Kısaca: portalda bir **uygulama** açarsın, bu sana bir **bot** hesabı ve bir **token** verir. Botu bir **sunucuya** eklersin; orada kendi **rolünü** alır, rolündeki **izinler** ise **kanallarda** neler yapabileceğini belirler. Programın token'la istek gönderir ve **hız sınırları** içinde kaldığı sürece istediği kadar çalışır.

Uygulama

Uygulama, botunun geliştirici portalındaki kaydıdır: adı, resmi, açıklaması, token'ı ve ekleme bağlantısı burada durur. Her uygulamanın tek bir botu ve tek bir token'ı vardır. Bir hesap en çok 25 uygulamaya sahip olabilir.

Uygulama başlangıçta özeldir, yani botu yalnızca sen ekleyebilirsin. **Herkese açık** seçeneğini açarsan sunucu yöneten herkes ekleme bağlantısıyla ekleyebilir. Ayrıntılar Uygulamalar ve botlar sayfasında.

Bot

Bot, bir programın kullandığı N'oluyo hesabıdır. Kullanıcı adı ve resmi vardır, üye listesinde ve profilinde **BOT** etiketi görünür. Sunuculara üye olarak katılır ve izinleri bir üyeninki gibi hesaplanır.

Bir insandan birkaç farkı vardır: kendi başına sunucuya katılamaz, davet kullanamaz, özel mesaj açamaz ve arkadaş ekleyemez. Bir kişi botla DM açabilir; bot orada yazar ama asla ilk o yazmaz. Yaptığı her şey REST referansındaki rotalardan geçer, olup biteni de gateway bağlantısından duyar.

> **Not:** Bot programın N'oluyo'da çalışmaz. Onu sen kendi bilgisayarında veya kiraladığın bir sunucuda çalıştırırsın. Program durursa bot da durur.

Token

Token, botunun şifresidir. Programın her isteğe şu başlığı ekler ve N'oluyo isteğin hangi bottan geldiğini böyle anlar:

```
Authorization: Bot <token>
```

Token yalnızca uygulamayı oluştururken ve yenilerken gösterilir. Onu bilen herkes botun adına iş yapabilir; nasıl saklayacağın Güvenlik sayfasında.

Sunucu (guild)

Sunucu, insanların toplandığı topluluktur: üyeleri, rolleri ve kanalları vardır. API'de sunucunun adı **guild**'dir; rotalarda `/guilds/{guild_id}`, yanıtlarda `guild_id` görürsün.

Bir bot, ancak o sunucuyu yönetme izni (`ManageGuild`) olan biri ekleme bağlantısını açıp eklerse sunucuya girer. Üyesi olmadığı sunucuları göremez: böyle bir istek `404 unknown_guild` döner.

Kanal

Kanal, sunucunun içinde konuşmaların geçtiği yerdir ve kendi ID'si vardır. Mesajlar kanallara gönderilir, bu yüzden mesaj rotaları `/channels/{channel_id}/messages` biçimindedir. Metin kanallarının `type` değeri `0`'dır; botlar sunucularda yalnızca bunlara mesaj gönderebilir, sesli kanala gönderilen mesaj `400 invalid_channel_type` döner. Ses kanallarının `type` değeri `2`'dir; botlar `op 4` ile bu kanallara girip ses yayınlayabilir ama kanaldaki kimseyi duyamaz, bkz. Ses.

Kanalların izin ayarları olabilir. Sunucunun genelinde bir iznin olsa bile, kanal o izni senden alabilir.

Rol

Rol, üyelere verilen izin ve rütbe paketidir. Bir üyenin birden çok rolü olabilir. Her sunucuda bir de `@everyone` rolü vardır ve herkes için geçerlidir.

Bot sunucuya eklenince kendi adını taşıyan bir rol alır. Bu role **yönetilen rol** denir; botu eklerken seçilen izinler bu rolde durur.

Roller sıralıdır. Üstteki rol alttakini yönetebilir, tersi olmaz. Yeni botun rolü `@everyone`'ın hemen üstüne konur; bu yüzden bir moderasyon botunun, işlem yapacağı üyelerin rollerinin üstüne taşınması gerekir.

İzin

İzin, tek bir yetkidir: mesaj göndermek (`SendMessages`), başkasının mesajını silmek (`ManageMessages`), üye yasaklamak (`BanMembers`) gibi. Bir rolün izinleri tek bir sayı olarak taşınır ve API bunu metin olarak yazar: `"68608"`.

Botunun bir kanaldaki izinleri üç adımda oluşur:

1. Sunucudaki `@everyone` rolünün izinleri,

2. bunlara eklenen, botun kendi rollerinin izinleri,
3. kanalın izin ayarlarının bunlardan çıkardıkları ve bunlara ekledikleri.

Gereken izin yoksa istek `403 missing_permissions` ile, kanalı görme izni yoksa `403 missing_access` ile reddedilir. Hesabın tamamı İzinler sayfasında.

> **İpucu:** Bota yalnızca gerçekten kullanacağı izinleri ver. Sonradan izin eklemek kolaydır: sunucu yöneticisi botun rolünü düzenleyebilir.

Hız sınırı

Hız sınırı, belirli bir sürede en çok kaç istek gönderebileceğindir. Örneğin bir kanala 5 saniyede en çok 5 mesaj gönderilebilir. Sınırı aşınca `429` yanıtı ve ne kadar beklemen gerektiğini söyleyen bir `retry_after` değeri gelir. Bekleyip yeniden denersen sorun çıkmaz. Sayılar Hız sınırları sayfasında.

Gateway

REST, programının sorduğunda yanıt alan yoldur. Gateway ise tersi: programının açık tuttuğu bir WebSocket bağlantısıdır ve sunucularda bir şey olduğunda N'oluyo haberi bu bağlantıdan kendisi gönderir. Her habere **olay** denir: yeni mesaj için `MESSAGE_CREATE`, katılan üye için `GUILD_MEMBER_ADD` gibi.

Bot gateway'den duyar, REST ile iş yapar. Ayrıntılar Gateway sayfasında, tüm olaylar Olaylar sayfasında.

Intent

Intent, botun hangi olay gruplarını istediğini söyleyen seçimdir. Yalnızca mesajlarla ilgilenen bir bot, `GUILD_MESSAGES` ister ve üye olaylarını hiç almaz. Üç intent hassas veri açtığı için **ayrıcalıklıdır** ve önce portalda açılması gerekir. Bunlardan biri olan `MESSAGE_CONTENT` olmadan bot, mesajların yazıldığını duyar ama metnini göremez. Ayrıntılar Intent'ler sayfasında.

İzinle karıştırma: izin botun **ne yapabileceğini** belirler ve sunucu yöneticisi verir. Intent, botun **ne duymak istediğini** söyler ve sen seçersin.

Shard

Shard, çok büyük bir botun sunucularını birkaç gateway bağlantısına bölmesidir. Küçük bir botun tek bağlantısı olur ve bunu düşünmesi gerekmez. Ayrıntılar Shard'lar sayfasında.

Embed

Embed, botun mesajında metnin altında duran karttır: başlık, açıklama, ad ve değer alanları, renkli bir şerit ve görsel taşır. Mesajın `embeds` listesinde gider, bir mesajda en çok 10 tane olur. Görseli yalnızca botun aynı mesajla yüklediği bir dosya olabilir. Ayrıntılar Embed'ler sayfasında.

Bileşen

Bileşen, botun mesajına eklediği buton ya da menüdür. Satırlara (`type: 1`) dizilir; her satır en çok 5 buton ya da tek bir menü taşır. Her bileşenin bir `custom_id` 'si olur, bot basmayı bu değerden tanır. Bağlantı butonu ise yalnızca bir adres açar, bota bir şey göndermez. Ayrıntılar Bileşenler sayfasında.

Etkileşim

Biri butona bastığında ya da menüden seçim yaptığında botuna `INTERACTION_CREATE` olayı gelir; buna etkileşim denir. Bot 3 saniye içinde yanıt vermelidir: yeni bir mesajla, basılan mesajı güncelleyerek ya da yalnızca "aldım" diyerek. Olayın `token` 'ı 15 dakika boyunca o etkileşimin mesajlarını düzenlemeye ve takip mesajı atmaya yarar. Ayrıntılar Etkileşimler sayfasında.

Etkileşimler gateway yerine botun kendi https adresine de gelebilir; N'oluyo her isteği imzalar, bot yanıtı aynı isteğin cevabında verir. Ayrıntılar HTTP ile teslim sayfasında.

Komut

Komut, botun sunduğu ve kişinin `/` yazınca listede gördüğü iştir: `/selam` gibi bir eğik çizgi komutu, ya da bir kişiye veya mesaja sağ tıklayınca çıkan menü satırı. Bot komutlarını REST ile bir kez kaydeder; kişi çalıştırınca botuna bir etkileşim gelir. Komut globaldir ya da tek sunucuya özeldir. Ayrıntılar Komutlar sayfasında.

Ephemeral mesaj

Ephemeral mesaj, bir etkileşime verilen ve yalnızca basan kişinin gördüğü yanıttır (`flags: 64`). N'oluyo onu saklamaz: kişi sayfayı yenileyince gider, kanalda başka kimse görmez. Dosya taşıyamaz. Ayrıntılar Etkileşimler sayfasında.

Hepsi bir arada: bir isteğin yolculuğu

Programın bir kanala mesaj göndermek istediğinde olanlar şunlar:

1. Program `POST /channels/{channel_id}/messages` isteğini `Authorization: Bot <token>` başlığıyla gönderir.
2. N'oluyo token'a bakar. Geçersizse yanıt `401 unauthorized` olur.
3. Hız sınırlarına bakar. Aşıldıysa yanıt `429 rate_limited` olur.
4. Kanalı bulur. Yoksa veya bot o sunucunun üyesi değilse yanıt `404 unknown_channel` olur.
5. Botun bu kanaldaki izinlerini hesaplar. `ViewChannel` yoksa `403 missing_access` , `SendMessages` yoksa `403 missing_permissions` döner.
6. Mesajı kaydeder, kanaldakilere ulaştırır ve `200` ile mesaj nesnesini döndürür.

Bu yolculuğu kendin görmek için Hızlı başlangıç sayfasına geç. Çalışan küçük botlar Örnekler sayfasında.

Sözlük

Burada	API'de	Anlamı
Uygulama	`application`	Botun portaldaki kaydı
Bot	`bot`, `user`	Programın kullandığı hesap
Token	`Authorization: Bot`	Botun şifresi
Sunucu	`guild`	Topluluk
Kanal	`channel`	Konuşmaların geçtiği yer
Üye	`member`	Bir sunucudaki kullanıcı veya bot
Rol	`role`	İzin ve rütbe paketi
Yönetilen rol	`application_id` dolu rol	Bota ait rol
İzin ayarı	`permission_overwrites`	Kanalın izinleri değiştirmesi
Hız sınırı	`X-RateLimit-*`, `429`	İstek sayısının üst sınırı
Gateway	`GET /gateway/bot`, WebSocket	Olayların geldiği açık bağlantı
Olay	`op 0`, `t` alanı	Gateway'den gelen tek bir haber
Intent	identify'da `intents`	Botun istediği olay grupları
Shard	identify'da `shard`	Botun sunucularını bölen bağlantı
Oturum	`session_id`	Bir identify ile başlayan, kopunca sürdürülebilen bağlantı durumu
Embed	`embeds`	Mesajdaki kart
Bileşen	`components`	Mesajdaki buton ya da menü
Etkileşim	`INTERACTION_CREATE`	Bir basma ve botun ona yanıtı
Ephemeral mesaj	`flags: 64`	Yalnızca basan kişinin gördüğü yanıt
Komut	`/applications/{id}/commands`	Botun `/` listesinde ya da sağ tık menüsünde sunduğu iş

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya N'oluyo terimlerini tanıtırken snowflake kimliklerinin 64 bitlik sayısal dizgiler (`string`) olarak taşındığını, `parseInt` veya `Number` ile yuvarlanmaması gerektiğini ve botların ses kanallarında yalnızca yayın yapabildiğini belirt.

Sırada ne var

- Sözlük: tüm geliştirici terimlerinin iki cümlelik hızlı tanımları
- Hızlı başlangıç: ilk REST isteği ve canlı gateway bağlantısı
- Uygulamalar ve botlar: portal ayarları, bot rolü ve sunucuya ekleme

Kimlik doğrulama ve genel kurallar

Bot token'ı

Botun her istekte token'ını şu başlıkla gönderir:

```
Authorization: Bot <token>
```

Token'ı geliştirici portalında uygulamanı oluştururken veya token'ı yenilerken alırsın. `Bot` sözcüğü ve ardından gelen tek boşluk tam bu şekilde yazılmalıdır.

Token, bota açık rotalarda geçerlidir; bu rotaların tamamı REST referansında listelenir. Bota açık olmayan bir rota `403 bot_forbidden` döner. Botun olayları dinlemek için bağlandığı gateway aynı token'ı kullanır ama başlıkta değil, identify çerçevesinde ve `Bot` öneki olmadan. Adresini `GET /gateway/bot` söyler; kişilerin kullandığı `/gateway` adresi bot token'ını `4004` ile reddeder.

Token'ını koru

Token botun kendisidir: onu ele geçiren, botunun adına mesaj atabilir ve botun izinlerini kullanabilir.

- Token'ı koda yazma, depoya gönderme, ortam değişkeninde veya bir sır yöneticisinde tut.
- Token yalnızca oluşturulduğunda ve yenilendiğinde gösterilir. Daha sonra görüntülenemez.
- Sızdığından şüpheleniyorsan hemen portaldan **Token'ı yenile** ile yenile. Eski token o anda çalışmaz olur, açık gateway bağlantıları da `4004` ile kapanır.

Token belirli bir biçim taşımaz; ayrıştırmaya çalışma, olduğu gibi saklayıp gönder.

Şu durumlarda geçerli görünen bir token da `401 unauthorized` alır: uygulama silinmişse, token yenilenmişse, bot hesabı ya da uygulama sahibinin hesabı kapatılmış veya silinmişse.

İstek kuralları

Adres. Tüm rotalar `https://noluyo.net/api/v1` altındadır.

Gövde. JSON gövdeli isteklerde tek bir JSON nesnesi gönder. En fazla 1 MiB olabilir; aşarsa `413 body_too_large` döner. Bozuk JSON `400 invalid_body` verir. Gövde isteyen bir rotaya boş gövde göndermek de `invalid_body` verir; hiçbir alanı değiştirmeyeceksen bile `{}` gönder.

Bilinmeyen alanlar. Bot isteklerinde tanınmayan alanlar yok sayılır, hata vermez.

ID'ler metindir. Tüm ID'ler 64 bitlik sayılardır ve yanıtta da istekte de metin olarak yazılır: `"111600000000000000"`. Gövdede ID'yi sayı olarak gönderirsen `invalid_body` alırsın. ID'ler oluşturulma zamanına göre sıralanır.

İzinler metindir. İzin değerleri ondalık metindir: `"68672"`. Bkz. İzinler.

Zamanlar. Yanıtlardaki zamanlar RFC 3339 biçimindedir. İstekte de aynı biçimi kullan: `"2026-11-05T12:00:00Z"`.

`@me`. `/users/@me` botun kendisidir. Üye rotalarının çoğunda `user_id` yerine `@me` yazabilirsin; ban rotaları gerçek bir kullanıcı ID'si ister.

Kısmi güncelleme. `PATCH` isteklerinde yalnızca gönderdiğin alanlar değişir. Boş bırakılabilen alanlarda `null` değeri temizler.

Yanıt kodları. Başarılı bir yanıt `200`, `201` veya gövdesiz `204` olabilir. Beklediğin kodu her rotanın referansında bulursun. Mesaj göndermek `201` değil, `200` döner.

Mesajlarda biçim

- Kullanıcı etiketi: `<@kullanici_id>`
- Rol etiketi: `<@&rol_id>`. Rolün etiketlenebilir olması veya botun `MentionEveryone` iznine sahip olması gerekir.
- Herkesi etiketleme: `@everyone`, `@here`, `@herkes`, `@burada`. `MentionEveryone` izni olmadan metin olarak kalır, kimseye bildirim gitmez.
- Sunucu emojisi: `<:ad:id>`, hareketliyse `<a:ad:id>`. Botun kullanamadığı emoji `:ad:` metnine dönüşür.

Bir mesajda en çok 50 kullanıcı ve 50 rol etiketlenir; fazlası sayılmaz. Mesaj metni en çok 2000 karakterdir.

Dosya göndermek

Dosyalar `multipart/form-data` ile gönderilir. JSON, `payload_json` adlı parçaya; dosyalar `files[0]`, `files[1]` gibi adlandırılmış parçalara konur. Bir mesajda en çok 10 dosya olur ve bunun için `AttachFiles` izni gerekir.

```
curl -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -F 'payload_json={"content":"Rapor ekte"}' \
  -F 'files[0]=@rapor.txt'
```

Başka adla bir parça `invalid_body` verir. Bir dosya yükleme sınırını aşarsa `413 file_too_large` döner. Yanıttaki dosya adresi siteye göreli, imzalı bir bağlantıdır ve en az 24 saat geçerlidir. Süresi dolunca mesajı yeniden okuyarak yenisini al.

Yalnızca portalın yaptıklarını otomatikleştirmek istersen bir başka kimlik daha vardır: kişi oturumu. Bot kurmak için gerekmez; bkz. API ile yönetmek.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya doğrudan HTTP isteği yazdırırken şunları ekle:
>
> - Bot isteklerinde `Authorization: Bot <token>` başlığı kullanılır (`Bearer` yalnızca OAuth2 kullanıcı token'ları içindir).
> - Sunucu belgelenmemiş JSON alanlarını sessizce yok saymaz; `invalid_body` ile reddeder, bu yüzden yalnızca şemadaki alanlar gönderilir.

Sırada ne var

- Hız sınırları: `X-RateLimit-*` başlıkları ve `429` bekleme kuralları
- Güvenlik: token saklama ve sızıntı durumunda yenileme
- Hatalar: kimlik ve doğrulama hata kodları

Uygulamalar ve botlar

Bir **uygulama**, botunu temsil eder: adı, resmi, token'ı ve sunuculara ekleme bağlantısı onda durur. Her uygulamanın tek bir bot hesabı ve tek bir token'ı vardır, botun sunuculardaki görünen adı da uygulamanın adıdır.

Uygulamalarını geliştirici portalından yönetirsin. N'oluyo hesabınla giriş yapman yeterlidir.

Kim uygulama oluşturabilir

Herkes uygulama oluşturabilir (herkese açık beta). Uygulama oluşturmak için:

- E-posta adresin doğrulanmış olmalı,
- Geliştirici sözleşmesini kabul etmiş olmalısın; portal ilk seferde sorar,
- Hesabın misafir hesabı olmamalı; misafir hesaba önce ayarlardan e-posta ve şifre bağla.

Bir hesap en çok 25 uygulamaya sahip olabilir. Portal, bir koşul eksikse ne yapman gerektiğini söyler.

Uygulama oluşturma

Uygulamalarım sayfasında **Yeni uygulama**'ya bas.

Alan	Kural
Uygulama adı	En çok 32 karakter. Botun görünen adı olur.
Bot kullanıcı adı	4 ile 32 karakter; küçük harf, rakam, `_` ve `.`; art arda iki nokta olamaz. Sonradan değiştirilemez, başkası almışsa kullanılamaz.

Oluştur'a basınca token gösterilir. **Kopyala** ile al: bir daha gösterilmez. Uygulama özel başlar, yani botu yalnızca sen ekleyebilirsin.

Uygulama sayfası

Genel

Botun resmini, **Uygulama adı**'nı ve en çok 400 karakterlik **Açıklama**'yı burada değiştirirsin. Açıklama, bot eklenirken gösterilir. Resim PNG, JPEG, WebP veya GIF olabilir; en çok 5 MB (hareketli GIF 10 MB) ve 4096×4096 piksel.

Herkese açık açıksa sunucu yöneten herkes botu bağlantıyla ekleyebilir. Kapalıysa yalnızca sen eklersin.

Ayrıcalıklı intent'ler

Sunucu üyeleri, **Presence** ve **Mesaj içeriği** hassas veri açar, bu yüzden bot bunları ancak burada açtıysan identify'da isteyebilir. Yeni açtığın bir intent, çalışan oturuma kendiliğinden geçmez: programını yeniden başlat. Kapattığında botun bağlantıları hemen düşer. Bot 100 ya da daha çok sunucudaysa bu intent'ler için onay gerekir. Ne işe yaradıkları Intent'ler sayfasında.

Token

Kayıtlı token güvenlik için gösterilmez. Kaybettiysen ya da sızdığından şüpheleniyorsan **Token'ı yenile**'ye bas: yeni token yalnızca o anda gösterilir, eskisi hemen çalışmaz olur ve botun açık gateway bağlantıları `4004` ile kapanır. Token'ı nasıl koruyacağın Kimlik doğrulama sayfasında.

Etkileşim adresi

Etkileşimleri gateway yerine kendi https adresine almak istersen **Etkileşimler** bölümünde **Etkileşim adresi**'ne yaz ve **Kaydet ve test et**'e bas. N'oluyo adrese iki imzalı `PING` yollar; doğru yanıtlanırsa adres kaydedilir ve etkileşimler artık oraya gider. **Herkese açık anahtar** isteklerin imzasını doğrulamana yarar. **Adresi kaldır** gateway'e döner, **Teslim günlüğü** son teslimleri gösterir. Adresi yalnızca sen görürsün. Nasıl kurulduğu HTTP ile teslim sayfasında.

Ekleme bağlantısı

Bota vereceğin izinleri seç; bağlantı buna göre oluşur. **Bağlantıyı kopyala** ile paylaş ya da **Bağlantıyı aç** ile kendin kullan. Yalnızca botunun gerçekten ihtiyaç duyduğu izinleri iste.

Bağlantının biçimi şöyledir; `izin`, seçilen izinlerin değeridir:

```
https://noluyo.net/bot-ekle/<application_id>?izin=68608
```

Tehlikeli bölge

Uygulamayı sil: Bot bulunduğu tüm sunuculardan çıkar ve token'ı çalışmayı bırakır. Gönderdiği mesajlar kalır. Hesabını silersen uygulamaların da silinir.

Oluşturma, token yenileme ve resim yüklemenin kendi sınırları vardır; bkz. Hız sınırları.

Botu sunucuya ekleme

Bağlantıyı açan kişi giriş yapar, botun adını, açıklamasını ve istediği izinleri görür, **Sunucu** listesinden bir sunucu seçer ve **Sunucuya ekle**'ye basar. Listede yalnızca yönettiği sunucular görünür.

- Kişinin kendisinde olmayan izinler bota verilmez; sayfa bunları işaretler.
- Sayfa yalnızca listelediği izinleri verir. Portaldaki listede olmayan bir izin gerekirse, sunucu yöneticisi onu bot rolüne sonradan ekleyebilir. Embed göndermek için ayrı bir izin gerekmez; Embed'ler ve bileşenler için `SendMessages` yeter.

Şu durumlarda bot eklenemez:

Durum	Hata kodu
Kişide sunucuda `ManageGuild` izni yok	`missing_permissions`
Bot özel ve kişi sahibi değil	`bot_private`
Bot bu sunucudan yasaklı	`banned`
Bot zaten sunucuda	`bot_in_guild`
Doğrulanmamış bot 100 sunucuya ulaşmış	`bot_guild_limit`
Sunucu rol sınırında	`max_roles`

100 sunucu sınırını kaldırmak için Doğrulama, botunu `/kesfet` dizininde göstermek için Rehbere girmek sayfasındaki adımları izle.

Botun rolü

Bir sunucuya eklenen bot, otomatik olarak kendi rolünü alır. API'de buna **yönetilen rol** denir:

- Rolün adı uygulamanın adıdır, izinleri eklenirken seçilen izinlerdir.
- Rolün `application_id` alanı doludur; bu rolü böyle tanırsın. Uygulamada "Bot rolü" etiketiyle görünür.
- Rol, doğrudan `@everyone`'ın üstüne konur. Bu yüzden bot başlangıçta rolü olan üyelere karışamaz; bkz. rol sıralaması.
- Rol silinemez, başka birine verilemez, bottan alınamaz (`managed_role`). Otomatik atanan role çevrilemez.
- Sunucu yöneticileri rolün adını, rengini, izinlerini ve sırasını, normal rol kurallarıyla değiştirebilir.
- Bot sunucudan ayrılırsa, atılırsa veya yasaklanırsa rol de silinir. Yeniden eklenince yeni bir rol oluşur.

Botun izinleri yalnızca bu rolden gelmez: `@everyone` rolünün izinleri ve kanal ayarları da geçerlidir.

Botun yapamadıkları

Bot sunucuya kendisi katılamaz, sunucu oluşturamaz, davet kullanamaz, DM kanalı açamaz (kişi açar, bkz. Özel mesajlar), arkadaş ekleyemez ve kendi profilini API ile değiştiremez. Bu rotalar `403 bot_forbidden` döner. Bot, sunucudan `DELETE /users/@me/guilds/{guild_id}` ile ayrılabilir.

API ile yönetmek

Bot kurmak için bu bölüme ihtiyacın yok. Portalın yaptığı her şeyi otomatikleştirmek istersen aynı rotaları kendin çağırabilirsin.

Bu rotalar bot token'ını kabul etmez (`403 bot_forbidden`), kişi oturumu ister: `Authorization: Bearer <oturum token'ı>`. Oturum token'ı hesabının tamamına erişim sağlar; botun koduna veya sunucusuna asla koyma.

Oturum token'ını girişten alırsın:

```
curl -X POST "https://noluyo.net/api/v1/auth/login" \
  -H "Content-Type: application/json" \
  -d '{"email":"sen@example.com","password":"parolan"}'
```

İki adımlı doğrulama açıksa yanıt `{"mfa":true,"ticket":"..."}` olur; `POST /api/v1/auth/mfa/totp` isteğine `{"ticket":"...","code":"123456"}` göndererek tamamlarsın. Google veya Apple ile açılmış hesabın, parola belirlenene kadar parolası yoktur.

İşlem	Rota
Uygulama oluştur	`POST /applications`
Uygulamaları listele	`GET /applications`
Birini gör	`GET /applications/{application_id}`
Ad, açıklama, `public` değiştir	`PATCH /applications/{application_id}`
Sil	`DELETE /applications/{application_id}`
Token'ı yenile	`POST /applications/{application_id}/token`
Avatar yükle (`file` parçası)	`POST /applications/{application_id}/avatar`
Avatarı kaldır	`DELETE /applications/{application_id}/avatar`
Ekleme ekranının bilgileri	`GET /applications/{application_id}/public`
Botu sunucuya ekle	`POST /applications/{application_id}/authorize`

Botu API ile eklemek şöyledir; başarıda `204` döner:

```
curl -i -X POST "https://noluyo.net/api/v1/applications/$APP_ID/authorize" \
  -H "Authorization: Bearer $SESSION_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"guild_id":"'"$GUILD_ID"'","permissions":"68608"}'
```

Sayfadan farklı olarak API, kişide olmayan bir izni çıkarmaz, isteği `403 missing_permissions` ile reddeder; listede olmayan izinleri de verebilir. `permissions` yoksa rolün izni olmaz. Başkasının uygulaması bu rotalarda `404 unknown_application` verir (herkese açık botlarda `public` ve `authorize` hariç).

Ayrıntılar ve örnekler REST referansında.

Bu sayfayı yapay zekaya verirsen

İpucu: Uygulama yönetimi için yapay zekaya kod yazdırırken şu ayrımı hatırlat:

- Botun mesaj ve sunucu işlemleri `Authorization: Bot <token>` kullanır; `/applications` yönetim uçları ise kişi oturumu (`Bearer`) ister ve normal bir botun kodunda çağrılmaz.
- Bot bir sunucuya kendiliğinden giremez; sunucu yöneticisi `/bot-ekle/<application_id>?izin=...` adresinden ekler.

Sırada ne var

- İzinler: ekleme bağlantısındaki `izin` değerinin hesaplanması ve rol sırası
- Doğrulama: 100 sunucu sınırını aşmak ve ayrıcalıklı intent onayı almak
- N'oluyo ile giriş: aynı uygulamayla web sitene kullanıcı girişi eklemek

İzinler

İzinler bir bit alanıdır. Her iznin bir biti vardır ve API değeri **ondalık metin** olarak taşır: `"68672"`. Metin yerine sayı göndermek veya tanımlı olmayan bir bit kullanmak `400 invalid_body` verir.

Değeri hesaplamak

Portaldaki **Ekleme bağlantısı** bölümünde izinleri seçtiğinde değer senin için hesaplanır ve bağlantıdaki `izin` olur.

Kendin hesaplamak istersen izinlerin değerlerini topla (veya bitlerini `OR` ile birleştir). Bazı izinler 32 bitin üstünde olduğu için JavaScript'te `BigInt` kullan:

```
const VIEW_CHANNEL = 1n << 10n;
const SEND_MESSAGES = 1n << 11n;
const value = VIEW_CHANNEL | SEND_MESSAGES;
console.log(value.toString()); // "3072"
```

Başlangıç rehberindeki `68608` şu toplamdır: `ViewChannel` (1024) + `SendMessages` (2048) + `ReadMessageHistory` (65536). Ses kanalına girip yayın yapan bir bot için `3146752` (`ViewChannel` + `Connect` + `Speak`) gerekir; ayrıntılar Ses sayfasında.

Ekleme bağlantısı bölümü **Tepki ekle** (`AddReactions` , 64) ve **@herkes ve rolleri etiketle** (`MentionEveryone` , 131072) izinlerini de sunar; bot bunları diğer izinler gibi isteyebilir.

İzin listesi

Bit	Değer	Ad	Anlamı
0	1	`CreateInstantInvite`	Davet oluşturma
1	2	`KickMembers`	Üyeyi sunucudan çıkarma
2	4	`BanMembers`	Üye yasaklama
3	8	`Administrator`	Tüm izinler; kanal ayarlarından etkilenmez
4	16	`ManageChannels`	Kanal yönetme
5	32	`ManageGuild`	Sunucu yönetme, bot ekleme
6	64	`AddReactions`	Mesaja yeni emoji ile tepki verme
7	128	`ViewAuditLog`	Denetim kaydını görme
8	256	`PrioritySpeaker`	Öncelikli konuşmacı
9	512	`Stream`	Yayın açma
10	1024	`ViewChannel`	Kanalı görme
11	2048	`SendMessages`	Mesaj gönderme
12	4096	`SendTTSMessages`	Sesli okunan mesaj gönderme
13	8192	`ManageMessages`	Başkalarının mesajlarını silme, sabitleme, tepki temizleme
14	16384	`EmbedLinks`	Bağlantı önizlemesi; botun embed'i için gerekmez, `SendMessages` yeter
15	32768	`AttachFiles`	Dosya gönderme
16	65536	`ReadMessageHistory`	Geçmiş mesajları okuma
17	131072	`MentionEveryone`	Herkesi ve her rolü etiketleme
18	262144	`UseExternalEmojis`	Başka sunucuların emojilerini kullanma
19	524288	`ViewGuildInsights`	Sunucu istatistiklerini görme
20	1048576	`Connect`	Ses kanalına bağlanma
21	2097152	`Speak`	Seste konuşma
22	4194304	`MuteMembers`	Üyeyi sunucuda susturma
23	8388608	`DeafenMembers`	Üyenin sesini kapatma

Bit	Değer	Ad	Anlamı
24	16777216	`MoveMembers`	Üyeyi ses kanalları arasında taşıma
25	33554432	`UseVAD`	Ses algılamayla konuşma
26	67108864	`ChangeNickname`	Kendi takma adını değiştirme
27	134217728	`ManageNicknames`	Başkalarının takma adını değiştirme
28	268435456	`ManageRoles`	Rol yönetme ve verme
29	536870912	`ManageWebhooks`	Webhook yönetme
30	1073741824	`ManageGuildExpressions`	Emoji, çıkartma ve ses yönetme
31	2147483648	`UseApplicationCommands`	Uygulama komutlarını kullanma; kanalda yoksa kişi orada hiçbir komutu görmez ve çalıştıramaz, bkz. Komut izinleri
32	4294967296	`RequestToSpeak`	Konuşma isteme
33	8589934592	`ManageEvents`	Etkinlik yönetme
34	17179869184	`ManageThreads`	Alt başlık yönetme
35	34359738368	`CreatePublicThreads`	Herkese açık alt başlık oluşturma
36	68719476736	`CreatePrivateThreads`	Özel alt başlık oluşturma
37	137438953472	`UseExternalStickers`	Başka sunucuların çıkartmalarını kullanma
38	274877906944	`SendMessagesInThreads`	Alt başlıklarda mesaj gönderme
39	549755813888	`UseEmbeddedActivities`	Gömülü etkinlikleri kullanma
40	1099511627776	`ModerateMembers`	Üyeyi zamanlı susturma

Bu aşamadaki bot rotalarının kullandığı izinler, REST referansında her rotanın altında yazar. Listedeki bazı izinlerin bu aşamada bota açık bir rotası yoktur.

Bir üyenin izinleri nasıl hesaplanır

Bot da diğer üyeler gibi hesaplanır.

Sunucu düzeyinde:

1. Sunucunun sahibi tüm izinlere sahiptir.
2. Diğer üyeler için `@everyone` rolünün izinleri ile üyenin tüm rollerinin izinleri birleştirilir.
3. Sonuçta `Administrator` varsa üye tüm izinlere sahiptir.

`@everyone` rolünün ID'si sunucunun ID'siyle aynıdır. Üyenin `roles` listesinde yer almaz, ama izinleri her zaman sayılır.

Bir rolün ID'sini sunucu ayarlarında **Roller** sekmesinden rolü açıp **Kopyala** ile alırsın (Android ve iOS'ta **Rol ID'sini kopyala**). Bir bot aynı kimlikleri `GET /guilds/{id}/roles` ile de listeleyebilir.

Kanalda. `Administrator` olanlar kanal ayarlarından etkilenmez. Diğerleri için kanalın izin geçersiz kılmaları (`permission_overwrites`) şu sırayla uygulanır; her adımda önce `deny` bitleri çıkarılır, sonra `allow` bitleri eklenir:

1. `@everyone` için olan geçersiz kılma
2. Üyenin rolleri için olanlar, birleştirilerek
3. Üyenin kendisi için olan

Bağımlı izinler. Kanalda `ViewChannel` yoksa hiçbir izin kalmaz. `SendMessages` yoksa `SendTTSMessages`, `MentionEveryone`, `AttachFiles` ve `EmbedLinks` de geçersiz olur. `Connect` yoksa `Speak`, `Stream`, `MuteMembers`, `DeafenMembers`, `MoveMembers`, `UseVAD`, `PrioritySpeaker` ve `RequestToSpeak` de geçersiz olur.

Susturma. Zamanlı susturma sürerken üyede yalnızca `ViewChannel` ve `ReadMessageHistory` kalır. Bu, susturulan bot için de geçerlidir. Sahip ve yöneticiler susturulamaz.

Bir izin geçersiz kılmasının türü (`type`) rol için `0`, üye için `1`'dir.

Rol sıralaması

Her rolün bir `position` değeri vardır, büyük olan daha üsttedir. Üyenin rütbesi, en yüksek rolünün konumudur. Sunucu sahibi herkesten üsttedir.

- Bir üyeyi çıkarmak, yasaklamak, susturmak veya takma adını değiştirmek için rütben hedefin rütbesinden **büyük** olmalıdır. Eşitlik yetmez; aksi halde `403 role_hierarchy` döner.
- Bir rolü düzenlemek, silmek, vermek veya almak için rütben o rolün konumundan büyük olmalıdır.
- Rolün `position` değerini kendi rütbenden küçük bir değere ayarlayabilirsin; en küçük değer 1'dir.
- Sahip olmadığın bir izni kimseye veremezsin (`403 missing_permissions`). Kanal geçersiz kılmaları için de aynısı geçerlidir.

Botun için bunun anlamı: Bot sunucuya eklenince rolü doğrudan `@everyone`'ın üstüne konur. Bu haliyle yalnızca rolü olmayan üyeler üzerinde işlem yapabilir. Moderasyon yapacak botun rolünü, bir sunucu yöneticisi **Sunucu ayarları** içindeki **Roller** listesinde sürükleyerek işlem yapacağı rollerin üstüne taşımalıdır.

Bot kendi rolünü değiştiremez, çünkü rütbesi o rolün konumundan büyük değildir; daha yüksek bir rol verilmişse değiştirebilir.

Sık görülen durumlar

Bot şunu alıyor	Muhtemel neden
Kanalı okurken boş liste	`ReadMessageHistory` yok. Liste rotaları bu durumda hata vermez.
`403 missing_access`	Kanalda `ViewChannel` yok.
`403 missing_permissions`	Gereken izin yok ya da bot susturulmuş.
`403 role_hierarchy`	Hedefin rolü botun rolünden aşağıda değil.
`404 unknown_guild`	Bot o sunucunun üyesi değil.

Bu sayfayı yapay zekaya verirsen

İpucu: Yapay zekadan ekleme bağlantısı veya komut tanımlamasını isterken şu kuralları ekle:

- İzin bitleri 64 bit tam sayı olduğu için JSON içinde metin dizgisi (`"permissions": "3072"`), JavaScript hesabında `BigInt` (`1n << 40n`) kullanılır.
- Kolaya kaçıp `Administrator` (`8`) istemek yerine yalnızca kullanılan rotaların gerektirdiği en küçük izin bitleri toplanır ve rol işlemlerinde bot rolünün hedef rolden yukarıda olması gerektiği (`role_hierarchy`) unutulmaz.

Sırada ne var

- Komut izinleri: komutların hangi rol ve kanalda çalışacağını yönetmek
- Uygulamalar ve botlar: `/bot-ekle` bağlantısı ve yönetilen bot rolü
- Sık hatalar: `missing_access` , `missing_permissions` ve `role_hierarchy` farkı

Hız sınırları

Bot isteklerinin sınırları tek yerde, bu sayfada yazılıdır. Sınırı aşan istek `429` alır. Yanıttaki bekleme süresine uyan bir bot sorun yaşamaz.

Rota sınıfları

Her bot rotası bir sınıfa aittir. Rotanın sınıfı REST referansında yazar.

Sınıf	İstek	Saniye	Hangi işlemler
`read`	30	5	Okuma rotaları
`write`	10	10	Oluşturma, değiştirme, silme
`message`	5	5	Mesaj gönderme
`reaction`	4	1	Tepki ekleme, kaldırma, temizleme
`bulk`	1	1	Toplu mesaj silme
`interaction`	50	1	Etkileşim yanıtları; ayrıca ephemeral düzenleme etkileşim başına 5 saniyede 5
`command`	5	20	Komut oluşturma, değiştirme, silme; ayrıca kapsam başına günde 200 yeni komut
`oauth_read`	30	5	`Authorization: Bearer <access_token>` ile çağrılan `/oauth2/@me` ve `/oauth2/@me/guilds` rotaları (token başına)

Sınır şunların her biri için ayrı sayılır: **bot, rota ve kanal**. Rotada kanal yoksa kanal yerine **sunucu** sayılır; ikisi de yoksa yalnızca bot ve rota. Örneğin iki ayrı kanala aynı anda 5'er mesaj gönderebilirsin.

Genel sınır

Bir bot, tüm rotalar toplamında saniyede en çok **50** istek gönderebilir. Aşınca `X-RateLimit-Scope: global` ile `429` alır.

Tüm botların ortak tavanı

Tüm botlar birlikte saniyede en çok **70** yazma (`read` dışındaki her sınıf) yapabilir; ayrıca aynı anda en çok **8** bot isteği (okuma dahil) çalışır. Etkileşim yanıtında yalnızca kanala mesaj yazanlar yazma sayılır:

İstek	70 yazmaya sayılır mı
`read` sınıfı	Hayır
`callback` `type: 4`, `5`	Evet; `flags: 64` ile ephemeral ise hayır
`callback` `type: 7`	Evet; ephemeral mesajın üstündeyse hayır
`callback` `type: 6`, `type: 8`	Hayır
Takip mesajı, `@original` düzenleme ve silme	Evet
Diğer her yazma rotası	Evet

Aynı anda 8 istek sınırına bunların hepsi girer. Biri aşılırsa bot `X-RateLimit-Scope: shared` ile `429` alır; yanıtta `retry_after` ve `Retry-After` bulunur (eşzamanlılık sınırında 1 saniye). Bu `429` botunun suçu değildir: geçersiz istek sayılmaz ve geçici engele yol açmaz. `retry_after` kadar bekleyip isteği yeniden gönder.

Etkileşimleri HTTP ile alıyorsan adresinin `200` yanıtı da bu tavanlara, `interaction` sınıfına ve yayılma bütçesine aynen sayılır; bir `callback` çağrısı gibi. Tavan doluysa yanıt yazılmaz ve kişi `429 interaction_busy` görür. Bu da botunun suçu olmayabilir: başka botlar tavanı doldurmuş olabilir. Teslim günlüğünde sonuç `busy` olur ve devre kesiciye sayılmaz. Tekrar deneme yoktur; kişi isterse yeniden çalıştırır.

Embed ve bileşenlerin yayılma bütçesi

Bir mesaj, kanaldaki herkese ayrı ayrı gönderilir; embed ve bileşenler mesajı büyütür. Bu yüzden her mesajın embed ve bileşenleri (metin sayılmaz) sunucunun üye sayısıyla (DM'de kişi sayısıyla) çarpılır ve tüm botlarla webhook'ların paylaştığı saniyelik bir bütçeden (**700000** bayt) düşer:

Parça	Bayt	Anlamı
Senin payın	Saniyede 175000	Bir sahibin tüm botları ve açtığı webhook'lar birlikte. Aşarsan `X-RateLimit-Scope: bot` ile `429` alırsın.
Yedek	Saniyede 70000	Maliyeti en çok 17500 bayt olan mesajlar için; bir sahip saniyede bunun 17500 baytını kullanabilir. Yedekte yer oldukça bekletme onları durdurmaz.
Tek mesaja en çok	21000000	Bütçenin 30 saniyesi. Aşarsan `400 fanout_too_large` alırsın; `max_bytes` o kanalın alabileceği en büyük embed ve bileşen boyutunu söyler (2000 kişilik bir sunucuda 10500 bayt).

Maliyeti payından büyük olan mesaj büyük mesajdır. Ardından bir sonraki büyük mesajın için, her sunucuda, mesajın ödenmesi için gereken sürenin dört katı beklersin; o sunucuya da o süre hiçbir bot büyük mesaj gönderemez: sana `X-RateLimit-Scope: bot`, sunucu için `shared` ile `429` gelir. Havuzdan (bütçe eksi yedek, saniyede 630000 bayt) fazla tutan büyük mesaj o saniyede kalanı alır, sonra o ve o saniyenin önceden harcadığı ödenene kadar tüm botların sayılan diğer mesajları bekler: 600 kişilik bir sunucuya 32 KiB'lık bir mesaj onları yaklaşık 31 saniye bekletir, hiçbir bekletme yaklaşık 34 saniyeden uzun sürmez. Aynı anda tek bekletme olur. Bütçe dolu ya da bekletmedeyken bot `X-RateLimit-Scope: shared` ile `429` alır. Bu `429` botunun suçu değildir: geçersiz istek sayılmaz ve geçici engele yol açmaz. `bot` kapsamlı `429` ise senindir ve diğer sınırların gibi sayılır.

`retry_after` ne kadar bekleyeceğini söyler. Düzenleme istekleri ve webhook'la gönderilen embed'ler de aynı bütçeyi kullanır.

Özel mesajlar

Botunun DM'lerde kendiliğinden yazdıkları (mesaj, tepki, sabitleme, yazıyor) tüm DM'leri toplamında ayrıca sayılır:

Sınır	İstek	Saniye	Ne sayılır
`dm_send`	30	10	Botun tüm DM'lerindeki serbest yazmaları; etkileşim yanıtları sayılmaz

Aşınca `X-RateLimit-Scope: bot` ile `429` alırsın. `retry_after` kadar bekleyip yeniden gönder. `message` sınıfının kanal başına sınırı DM'de de geçerlidir.

Başlıklar

Bot isteklerinin yanıtlarında şu başlıklar gelir:

Başlık	Anlamı
`X-RateLimit-Limit`	Pencerenin izin verdiği istek sayısı
`X-RateLimit-Remaining`	Bu pencerede, bu kanal veya sunucu için kalan istek
`X-RateLimit-Reset`	Pencerenin bittiği an; üç ondalıklı Unix saniyesi
`X-RateLimit-Reset-After`	Pencerenin bitmesine kalan saniye; üç ondalıklı
`X-RateLimit-Bucket`	Rotayı ve sınıfını adlandıran 16 onaltılık basamak

`X-RateLimit-Bucket` bir rotanın tüm kanallarında aynıdır ama `Remaining` kanal başına sayılır. Bu yüzden sınırı izlerken kovayı kanal veya sunucu ID'siyle birlikte tut.

Genel sınıra takılan yanıtta yalnızca `X-RateLimit-Scope` ve `Retry-After` gelir. Sınır sistemi geçici olarak çalışmıyorsa istekler sınırlanmaz ve bu başlıklar hiç gelmez; kodun bu başlıkların her zaman var olduğunu varsaymasın.

429 yanıtı

```
HTTP/1.1 429 Too Many Requests
Retry-After: 2
X-RateLimit-Scope: bot
X-RateLimit-Limit: 5
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1793966400.250
X-RateLimit-Reset-After: 1.250
X-RateLimit-Bucket: 3f9a1c7e5b2d4860
Content-Type: application/json; charset=utf-8

{"code":"rate_limited","message":"You are being rate limited.","retry_after":1.25}
```

- `retry_after`, gövdede beklenecek süredir; saniye cinsinden, üç ondalığa kadar.
- `Retry-After`, başlıktaki aynı süredir; yukarı yuvarlanmış tam saniye.
- `X-RateLimit-Scope`, sınıf sınırında `bot`, genel sınırda `global`, yukarıdaki tavanda `shared` olur. Aşağıdaki ek sınırlarda bu başlık gelmez.

Beklemeden tekrar denemek işe yaramaz; aşağıdaki gibi süreyi bekle:

```
if (res.status === 429) {
  const { retry_after } = await res.json();
  await new Promise((done) => setTimeout(done, retry_after * 1000));
  // sonra isteği yeniden gönder
}
```

Ek sınırlar

Bazı işlemlerde sınıf sınırına ek olarak şu sınırlar da geçerlidir. Bunlar da `429` verir ve `retry_after` içerir.

İşlem	Sınır	Kod
Mesaj gönderme	Kanal başına 5 saniyede 5	`rate_limited`
Mesaj düzenleme	Kanal başına 5 saniyede 5	`rate_limited`
Dosyalı mesaj	Tüm kanallarda dakikada 20	`rate_limited`
Yazıyor göstergesi	Kanal başına 5 saniyede 5	`rate_limited`
Sabitleme, sabitleme kaldırma	Kanal başına dakikada 10	`rate_limited`
Tepki ekleme	Kanal başına 5 saniyede 20	`rate_limited`
Dosya yükleme miktarı	Saatte 1 GiB, günde 3 GiB	`upload_quota`
Yavaş mod açık kanalda mesaj	Kanalın `rate_limit_per_user` süresi	`slowmode`
Ephemeral mesaj	Tüm botlar birlikte, basan kişi başına 5 saniyede 262144 bayt	`rate_limited`

`ManageMessages` veya `ManageChannels` iznine sahip olanlar yavaş moddan muaftır.

Portaldaki uygulama işlemlerinin de sınırı vardır: uygulama oluşturma ve token yenileme için hesap başına, bot resmini yükleme ve kaldırma için bot başına 10 dakikada 5 işlem.

Gateway sınırları

Gateway bağlantısının sınırları REST'ten ayrıdır:

İşlem	Sınır	Aşılınca
Oturum başlatma (identify)	Uygulama başına 5 saniyede 1	Bağlantı `4008` ile kapanır
Oturum başlatma (identify)	Uygulama başına günde 1000	Bağlantı `4016` ile kapanır
Gönderilen çerçeve	Bağlantı başına dakikada 120	Bağlantı `4008` ile kapanır
Durum değiştirme (`op 3`)	Oturum başına dakikada 5	`RATE_LIMITED` gelir, güncelleme yok sayılır
Üye isteği (`op 8`)	Oturum başına dakikada 10, aynı anda 1	`RATE_LIMITED` gelir
Tam üye listesi (`op 8`)	Bot ve sunucu başına 30 saniyede 1	`RATE_LIMITED` gelir

Resume oturum başlatma sayılmaz. Kalan başlatma hakkını `GET /gateway/bot` gösterir.

Geçersiz istekler ve geçici engel

Bir IP adresinden `Bot` şemasıyla gönderilen ve reddedilen (`401` , `403` veya `429` alan) istekler sayılır. Sayı 10 dakika içinde **10000**'e ulaşırsa, o adresten gelen `Bot` şemalı tüm istekler pencere bitene kadar reddedilir, geçerli token'lar dahil:

```
{"code":"temporarily_banned","message":"Too many invalid requests. Try again later.","retry_after":312.5}
```

Yanıt `403` 'tür ve `Retry-After` başlığı içerir. Aynı adresten siteyi ve uygulamayı kullanan kişiler bundan etkilenmez.

Bundan kaçınmak için: `401` alınca isteği tekrarlamak yerine botu durdur, `403` alan isteği aynen tekrarlama ve `429` alınca süreyi bekle.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya hız sınırı yönetimi yazdırırken `429` yanıtlarında gövdedeki `retry_after` saniyesi kadar beklemesini, `X-RateLimit-Scope: shared` geldiğinde tüm yazma isteklerini yavaşlatmasını ve `401` / `403` yanıtlarını döngüde tekrarlayarak geçici IP engeline girmemesini söyle.

Sırada ne var

- JavaScript kütüphanesi ve Python kütüphanesi: otomatik kova kuyruğu ve `Retry-After` yönetimi
- Hatalar: `rate_limited` , `slowmode` ve `upload_quota` kodları
- Güvenlik: geçersiz istek engelinden kaçınma ve güvenli çalışma

JavaScript kütüphanesi (`noluyo-bot`)

`noluyo-bot`, N'oluyo bot platformu için geliştirilmiş resmi JavaScript ve TypeScript kütüphanesidir. **Node.js 22 veya üstü** üzerinde sıfır çalışma zamanı bağımlılığıyla çalışır (yerleşik `fetch`, `WebSocket`, `FormData` ve `node:crypto` kullanır), tam `.d.ts` tip tanımlarıyla gelir ve MIT lisanslıdır.

Kurulum

Proje klasöründe `package.json` dosyanın `"type": "module"` kullandığından emin ol ve paketi kur:

```
npm install noluyo-bot
# Yalnızca ses kanallarında yayın yapacaksan isteğe bağlı ek paket:
npm install @livekit/rtc-node@^1.2.1
```

> **Not:** Sıfırdan proje klasörü ve `.env` kurulumu için Kurulum rehberini adım adım izleyebilirsin.

İlk bot

Token'ı kaynak koda yazma; `.env` dosyasındaki `BOT_TOKEN` değişkeninden oku ve `node --env-file=.env bot.mjs` ile çalıştır:

```
import { Client, Intents } from "noluyo-bot";

const bot = new Client({
  token: process.env.BOT_TOKEN,
  intents: Intents.GUILDS | Intents.GUILD_MESSAGES | Intents.MESSAGE_CONTENT,
});

bot.on("READY", (ready, meta) => {
  console.log(`${ready.user.username} bağlandı (shard ${meta.shardId})`);
});

bot.on("MESSAGE_CREATE", async (message) => {
  if (message.author.bot) return;
  if (message.content === "!merhaba") {
    await bot.rest.createMessage(message.channel_id, { content: "Merhaba!" });
  }
});

bot.interactions.on("ping", async (i) => {
  await i.reply({ content: "Pong!", ephemeral: true });
});

await bot.login();
await bot.commands.sync([
  { name: "ping", description: "Botun yanıt verip vermediğini sınar" },
]);
```

Kapatırken açık WebSocket ve ses bağlantılarını temizlemek için `await bot.close()` çağrılır. `new Client({ token, intents, applicationId, baseUrl, presence })` yapılandırıcısı isteğe bağlı olarak `applicationId` (`bot.login()` öncesinde komut eşitlemek istersen), `baseUrl` ve başlangıç `presence` nesnesini kabul eder. İzin bitleri için `Permissions` nesnesi hem `Permissions.ModerateMembers` hem `Permissions.MODERATE_MEMBERS` biçiminde `BigInt` sabitleri sunar (`String(Permissions.ModerateMembers)` ile ondalık dizgeye çevrilir).

Gateway olayları ve durum yönetimi

`bot.on(olayAdi, dinleyici)` ile Gateway üzerindeki tüm dağıtım olaylarını dinlersin. İkinci parametre `(data, meta)` içinde `meta.shardId` ve `meta.seq` gelir:

- **Bağlantı ve sunucular:** `READY`, `RESUMED`, `GUILD_CREATE`, `GUILD_UPDATE`, `GUILD_DELETE`
- **Mesajlar ve tepkiler:** `MESSAGE_CREATE`, `MESSAGE_UPDATE`, `MESSAGE_DELETE`, `MESSAGE_REACTION_ADD`, `MESSAGE_REACTION_REMOVE`
- **Üyeler, roller ve kanallar:** `GUILD_MEMBER_ADD`, `GUILD_MEMBER_UPDATE`, `GUILD_MEMBER_REMOVE`, `CHANNEL_CREATE`, `CHANNEL_UPDATE`, `CHANNEL_DELETE`, `GUILD_ROLE_CREATE`, `GUILD_ROLE_UPDATE`, `GUILD_ROLE_DELETE`
- **Etkileşim ve ses:** `INTERACTION_CREATE`, `VOICE_STATE_UPDATE`, `VOICE_SERVER_UPDATE`, `VOICE_JOIN_ERROR`

Durum güncellemek için `bot.setPresence("online", [{ name: "/yardim", type: 0 }])`, büyük sunucuda üye aramak için `bot.requestGuildMembers(guildId, { query: "ada", limit: 25 })` yardımcılarını kullanabilirsin.

İhtiyacın olan bitleri Intent'ler tablosundan seçip `Intents` bayraklarıyla birleştir.

REST çağrıları (`bot.rest`)

OpenAPI şemasındaki her `operationId`, `bot.rest` üzerinde `camelCase` adıyla hazır bir metottur. Yol parametreleri sırayla ilk argümanlar olarak, JSON gövdesi veya sorgu parametreleri son nesne olarak verilir:

```
// Basit mesaj ve embed gönderme:
await bot.rest.createMessage(channelId, {
  content: "Haftalık özet hazır.",
  embeds: [{ title: "Özet", description: "12 yeni üye katıldı.", color: 0x3366ff }],
});

// Dosya ekiyle birlikte attachment:// görselli embed gönderme:
import { readFile } from "node:fs/promises";
await bot.rest.createMessage(channelId, {
  embeds: [{ title: "Grafik", image: { url: "attachment://grafik.png" } }],
  files: [{ name: "grafik.png", data: await readFile("./grafik.png"), contentType: "image/png" }],
});

// Üyeye rol verme ve 10 dakikalık zaman aşımı uygulama:
await bot.rest.addMemberRole(guildId, userId, roleId);
await bot.rest.updateMember(guildId, userId, {
  communication_disabled_until: new Date(Date.now() + 10 * 60_000).toISOString(),
});
```

Gateway bağlantısı açmadan yalnızca HTTP çağrısı yapmak istediğinde `import { Rest } from "noluyo-bot"` ile doğrudan `new Rest({ token: process.env.BOT_TOKEN })` oluşturabilirsin. Zengin mesaj sınırları Embed'ler rehberindedir.

Komut kaydı ve etkileşimler

`bot.commands.sync(liste)` metodu `PUT /applications/{id}/commands` rotasını çağırarak tüm küresel komutlarını tek adımda eşitler (`{ guildId }` verirsen yalnızca o sunucunun komutlarını günceller). `bot.login()` sonrasında `bot.applicationId` otomatik dolduğu için ayrıca kimlik yazman gerekmez.

`bot.interactions.on(adVeyaKalip, isleyici)` hem eğik çizgi komutlarını (`name`), hem Bileşenler üzerindeki `custom_id` değerlerini, hem de Formlar gönderimlerini yakalar; dize yerine `RegExp` (`/ ^rol:/`) de verebilirsin:

```
bot.interactions.on("duyuru", async (i) => {
  const baslik = i.option("baslik");
  await i.defer({ ephemeral: true });
  await bot.rest.createMessage(i.channel_id, {
    embeds: [{ title: String(baslik), description: "Yeni duyuru yayınlandı." }],
  });
  await i.editReply({ content: "Duyuru kanala gönderildi." });
});

bot.interactions.on(/^anket:/, async (i) => {
  const secim = i.data.custom_id.split(":")[1];
  await i.reply({ content: `Oyun kaydedildi: ${secim}`, ephemeral: true });
});
```

`InteractionContext` üzerindeki veri alanları (`i.id`, `i.guild_id`, `i.channel_id`, `i.member`, `i.user`, sunucu veya DM fark etmeksizin tetikleyen kullanıcıyı veren `i.person`, `i.data`) ve yardımcılar: `i.option(ad)`, `i.fields()`, `i.reply()`, `i.defer({ ephemeral })`, `i.deferUpdate()`, `i.update()`, `i.editReply()`, `i.deleteReply()`, `i.followUp()`, `i.showModal()` ve Otomatik tamamlama için `i.autocomplete(secenekler)`. Tüm etkileşim türlerinin kuralları Komutlar ve Etkileşimler sayfalarında yer alır.

HTTP modu ve imza doğrulama

Etkileşimleri WebSocket yerine kendi HTTPS sunucundan alıyorsan (HTTP ile teslim), gelen her isteğin Ed25519 imzasını ek paket kurmadan `verifyInteraction` ile doğrula:

```
import { verifyInteraction } from "noluyo-bot";

const gecerli = verifyInteraction(
  process.env.NOLUYO_PUBLIC_KEY,
  req.headers["x-signature-ed25519"],
  req.headers["x-signature-timestamp"],
  rawBodyBuffer,
);
if (!gecerli) {
  res.writeHead(401).end("invalid signature");
  return;
}
```

Ses kanallarında yayın (`bot.voice`)

`@livekit/rtc-node@^1.2.1` kurulu olduğunda `bot.voice.join(guildId, channelId)` senin yerine gateway `op 4` gönderir, `VOICE_SERVER_UPDATE` yanıtını bekler ve yayın odasına bağlanır:

```
const conn = await bot.voice.join(guildId, voiceChannelId);
await conn.playFile("./muzik.mp3"); // Yerel ses dosyasını 48 kHz PCM olarak yayınlar
await conn.leave();
```

N'oluyo'da botlar seste **yalnızca yayın yapar** (`canSubscribe: false`); kütüphane hiçbir zaman ses aboneliği açmaz ve konuşulanları duyamaz. Ses sınırları ve `VOICE_JOIN_ERROR` kodları Ses rehberindedir.

Hız sınırı kuyruğu, yeniden deneme ve shard davranışı

Kütüphane platform kurallarını varsayılan olarak uygular:

- **Kova kuyruğu:** `X-RateLimit-Bucket` , `X-RateLimit-Remaining` ve `X-RateLimit-Reset-After` başlıklarını izler; aynı kovaya giden istekleri sırayı bozmadan bekletir (Hız sınırları).
- **`429` ve ortak tavan:** `429` yanıtında `Retry-After` süresince bekleyip isteği tekrarlar; `X-RateLimit-Scope: shared` geldiğinde tüm rotaları birlikte yavaşlatır.
- **Güvenli yeniden deneme:** `5xx` sunucu hatalarında yalnızca `GET` isteklerini en çok 3 kez yeniden dener; `POST` , `PATCH` , `PUT` ve `DELETE` yazma isteklerini çift mesaj oluşmaması için tekrarlamaz.
- **Shard ve sürdürme (`RESUME`):** `bot.login()` önerilen shard sayısını `GET /gateway/bot` ucundan alır ve `IDENTIFY` çerçevelerini en az **5,1 saniye** arayla gönderir (Shard'lar). Bağlantı koptuğunda `op 6` (`RESUME`) ile kaçan olayları tamamlar.
- **Kalıcı kapanış koruması ve token gizleme:** `4004` (geçersiz token) veya `4014` (onaysız ayrıcalıklı intent) gibi `reconnect: false` kapanış kodlarında sonsuz döngüye girmez; portal yönlendirmesi içeren `GatewayCloseError` fırlatır. Hata mesajlarında, `JSON.stringify` ve `util.inspect` çıktılarında bot token'ı otomatik olarak `[REDACTED]` ile maskelenir.

Hata sınıfları, test ve sürüm politikası

Yakalamak için dışa aktarılan hata sınıfları:

- `ApiError` : `4xx` / `5xx` yanıtları (`err.status` , `err.code` , `err.errors` , `err.path`)
- `RateLimitedError` : yeniden deneme tavanı aşıldığında dönen `429` hatası (`err.retryAfter` , `err.scope`)
- `GatewayCloseError` : yeniden bağlanılmaması gereken gateway kapanışı (`err.code` , `err.closeName` , `err.hint`)
- `VoiceJoinError` : ses kanalına katılma reddi (`err.code` , `err.guildId`)

Birim testlerinde `new Rest({ token: "test", fetch: sahteFetch })` ile gerçek ağa çıkmadan REST çağrılarını sınayabilirsin. Kütüphane anlamsal sürümleme (SemVer) izler; OpenAPI veya olay şemasına yeni alan eklendiğinde geriye dönük uyumlu küçük sürüm (`0.x`) çıkar.

Tam çalışan örnek projeler depoda hazır bulunur:

- `sdks/examples/js/moderasyon` : sözcük süzgeci, `/sustur` zaman aşımı komutu ve denetim embed'i
- `sdks/examples/js/anket` : süreli buton oylaması
- `sdks/examples/js/muzik` : yerel ses dosyası çalan ses botu
- `sdks/examples/js/oauth-site` : PKCE tabanlı "N'oluyo ile giriş" web sunucusu

Bu sayfayı yapay zekaya verirsen

İpucu: Yapay zekaya Node.js ile N'oluyo botu yazdırırken bu sayfayı ver ve şu kuralları hatırlat:

- Yalnızca `import { Client, Intents } from "noluyo-bot"` kullanılır; başka sohbet platformlarının paketleri kurulmaz.
- Node.js 22+ yerleşik `--env-file=.env` bayrağıyla çalışır; `dotenv` bile şart değildir ve `BOT_TOKEN` asla koda yazılmaz.
- `bot.rest.<camelCaseOperationId>` metotları doğrudan OpenAPI `operationId` adlarını taşır (`createMessage`, `editMessage`, `addMemberRole`, `updateMember`).
- Embed görselleri yalnızca `files` dizisiyle yüklenen dosyalara `attachment://dosya.png` ile bağlanır.

Sırada ne var

- Hazır tarifler: `noluyo-bot` ile yazılmış 9 kopyala-çalıştır bot tarifi
- Python kütüphanesi: aynı özelliklerin Python `asyncio` karşılığı
- Botu açık tut: Node.js botunu Linux sunucuda `systemd` ile 7/24 çalıştırma

Python kütüphanesi (`noluyo-bot`)

`noluyo-bot` (içe aktarma adıyla `noluyo_bot`), N'oluyo bot platformunun **Python 3.11 ve üstü** için geliştirilmiş resmi `asyncio` kütüphanesidir. Çalışma zamanında yalnızca `aiohttp` kullanır, PEP 561 (`py.typed`) tip bildirimleriyle gelir ve MIT lisanslıdır.

Kurulum

Sanal ortam oluşturup paketi kur (HTTP imza doğrulaması veya ses yayını kullanacaksan isteğe bağlı ekleri ekle):

```
python3 -m venv .venv
source .venv/bin/activate   # Windows PowerShell: .venv\Scripts\Activate.ps1
pip install noluyo-bot

# İsteğe bağlı ekler:
pip install "noluyo-bot[http]"    # Ed25519 HTTP etkileşim doğrulaması (cryptography)
pip install "noluyo-bot[voice]"   # Ses kanallarında yayın (livekit)
```

> **Not:** Sıfırdan Python, sanal ortam ve `.env` kurulumu için Kurulum sayfasındaki adımları izleyebilirsin.

İlk bot

Paket PyPI'da `noluyo-bot` adıyla kurulur, Python kodunda alt çizgiyle `import noluyo_bot` olarak içe aktarılır. Token'ı koda yazma; `os.environ["BOT_TOKEN"]` ile ortamdan oku:

```python
import os
import noluyo_bot

bot = noluyo_bot.Client(
    token=os.environ["BOT_TOKEN"],
    intents=noluyo_bot.Intents.GUILDS
    | noluyo_bot.Intents.GUILD_MESSAGES
    | noluyo_bot.Intents.MESSAGE_CONTENT,
)

@bot.on("READY")
async def on_ready(ready):
    print(f"{ready['user']['username']} bağlandı")

@bot.on("MESSAGE_CREATE")
async def on_message(message):
    if message["author"].get("bot"):
        return
    if message["content"] == "!merhaba":
        await bot.rest.create_message(message["channel_id"], {"content": "Merhaba!"})

@bot.interactions.on("ping")
async def on_ping(i):
    await i.reply({"content": "Pong!"}, ephemeral=True)

@bot.on("READY")
async def sync_commands(_ready):
    await bot.commands.sync([
        {"name": "ping", "description": "Botun yanıt verip vermediğini sınar"},
    ])

bot.run()
```

Zaten çalışan bir `asyncio` döngüsünün içindeysen `bot.run()` yerine `await bot.start()` ve kapatırken `await bot.close()` kullanabilirsin.

Gateway olayları ve TypedDict veri modeli

`@bot.on("OLAY_ADI")` dekoratörü Gateway üzerindeki tüm olayları dinler. Python kütüphanesi gelen JSON verisini ağır sınıf hiyerarşilerine sarmalamaz; alan adları dokümandaki JSON anahtarlarıyla (`message["channel_id"]`, `member["user"]["id"]`) birebir aynı `TypedDict` sözlükleri olarak gelir:

- **Bağlantı ve sunucular:** `READY`, `RESUMED`, `GUILD_CREATE`, `GUILD_UPDATE`, `GUILD_DELETE`
- **Mesajlar ve tepkiler:** `MESSAGE_CREATE`, `MESSAGE_UPDATE`, `MESSAGE_DELETE`, `MESSAGE_REACTION_ADD`, `MESSAGE_REACTION_REMOVE`
- **Üyeler, roller ve kanallar:** `GUILD_MEMBER_ADD`, `GUILD_MEMBER_UPDATE`, `GUILD_MEMBER_REMOVE`, `CHANNEL_CREATE`, `CHANNEL_UPDATE`, `CHANNEL_DELETE`, `GUILD_ROLE_CREATE`, `GUILD_ROLE_UPDATE`, `GUILD_ROLE_DELETE`

- **Etkileşim ve ses:** `INTERACTION_CREATE`, `VOICE_STATE_UPDATE`, `VOICE_SERVER_UPDATE`, `VOICE_JOIN_ERROR`

Dinleyici fonksiyon tek argüman (`data`) ya da iki argüman (`data, meta` — `meta["shard_id"]` ve `meta["seq"]`) alabilir. İhtiyacın olan intent bitlerini Intent'ler rehberinden seç.

REST çağrıları (`bot.rest`)

OpenAPI şemasındaki her `operationId`, `bot.rest` üzerinde `snake_case` adıyla `async` bir metottur:

```
from pathlib import Path

# Metin ve embed gönderme:
await bot.rest.create_message(
    channel_id,
    {
        "content": "Haftalık özet hazır.",
        "embeds": [{"title": "Özet", "description": "12 yeni üye katıldı.", "color": 0x3366FF}],
    },
)

# Dosya ekiyle birlikte attachment:// görselli embed gönderme:
await bot.rest.create_message(
    channel_id,
    {
        "embeds": [{"title": "Grafik", "image": {"url": "attachment://grafik.png"}}],
        "files": [
            {
                "name": "grafik.png",
                "data": Path("./grafik.png").read_bytes(),
                "content_type": "image/png",
            }
        ],
    },
)

# Üyeye rol verme ve 10 dakikalık zaman aşımı uygulama:
from datetime import datetime, timedelta, timezone

until = (datetime.now(timezone.utc) + timedelta(minutes=10)).isoformat()
await bot.rest.add_member_role(guild_id, user_id, role_id)
await bot.rest.update_member(guild_id, user_id, {"communication_disabled_until": until})
```

Gateway açmadan yalnızca HTTP istekleri atmak için `async with noluyo_bot.Rest(token=os.environ["BOT_TOKEN"]) as rest:` bağlam yöneticisini kullanabilirsin. Görsel ve alan sınırları Embed'ler rehberindedir.

Komut kaydı ve etkileşimler

`await bot.commands.sync(liste)` küresel komutlarını tek çağrıda eşitler (`guild_id=...` verirsen yalnızca o sunucunun komutlarını günceller).

`@bot.interactions.on(ad_veya_desen)` dekoratörü Komutlar, Bileşenler (`custom_id`) ve Formlar etkileşimlerini yönlendirir; tam metin ya da derlenmiş `re.compile(r"^anket:")` verebilirsin:

```
import re

@bot.interactions.on("duyuru")
async def on_announce(i):
    baslik = i.option("baslik")
    await i.defer(ephemeral=True)
    await bot.rest.create_message(
        i.channel_id,
        {"embeds": [{"title": str(baslik), "description": "Yeni duyuru yayınlandı."}]},
    )
    await i.edit_reply({"content": "Duyuru kanala gönderildi."})

@bot.interactions.on(re.compile(r"^anket:"))
async def on_poll_vote(i):
    secim = i.data["custom_id"].split(":", 1)[1]
    await i.reply({"content": f"Oyun kaydedildi: {secim}"}, ephemeral=True)
```

`InteractionContext` üzerindeki yardımcılar: `i.option(ad)`, `i.fields()`, `await i.reply(..., ephemeral=...)`, `await i.defer(ephemeral=...)`, `await i.defer_update()`, `await i.update(...)`, `await i.edit_reply(...)`, `await i.delete_reply()`, `await i.follow_up(...)`, `await i.show_modal(...)` ve Otomatik tamamlama için `await i.autocomplete(secenekler)`. Ayrıntılar Etkileşimler sayfasındadır.

HTTP modu ve imza doğrulama (`[http]`)

Etkileşimleri HTTPS sunucundan karşılıyorsan (HTTP ile teslim), `noluyo-bot[http]` ekini kurup Ed25519 imzasını `verify_interaction` ile doğrula:

```
import os
from noluyo_bot import verify_interaction

gecerli = verify_interaction(
    os.environ["NOLUYO_PUBLIC_KEY"],
    request.headers["X-Signature-Ed25519"],
    request.headers["X-Signature-Timestamp"],
    raw_body_bytes,
)
if not gecerli:
    return Response(status=401, text="invalid signature")
```

Ses kanallarında yayın (`[voice]`)

`noluyo-bot[voice]` kurulu olduğunda `await bot.voice.join(guild_id, channel_id)` gateway `op 4` ve `VOICE_SERVER_UPDATE` adımlarını yürüterek LiveKit odasına bağlanır:

```
conn = await bot.voice.join(guild_id, voice_channel_id)
await conn.play_file("./muzik.mp3")
await conn.leave()
```

N'oluyo'da botlar seste **yalnızca yayın yapar**; kütüphane hiçbir uzak ses izine abone olmaz ve kimseyi duyamaz. Ses kuralları ve hata kodları Ses sayfasındadır.

Hız sınırı kuyruğu, shard ve güvenlik kuralları

JavaScript kütüphanesiyle birebir aynı çalışma kuralları geçerlidir:

- **Kova kuyruğu ve `429`:** `X-RateLimit-Bucket`, `X-RateLimit-Remaining` ve `X-RateLimit-Reset-After` başlıklarına göre kova bazında sıra korur; `429` durumunda `Retry-After` kadar bekler ve `X-RateLimit-Scope: shared` geldiğinde tüm rotaları birlikte yavaşlatır (Hız sınırları).
- **Yalnız `GET` tekrarı:** `5xx` hatalarında yalnız `GET` çağrıları en çok 3 kez yeniden denenir; `POST`, `PATCH`, `PUT` ve `DELETE` tekrar edilmez.
- **Shard aralığı ve `RESUME`:** Shard sayısını `/gateway/bot` ucundan alır, `IDENTIFY` çerçevelerini en az **5,1 saniye** arayla gönderir (Shard'lar) ve kopmalarda `op 6` (`RESUME`) ile devam eder.
- **Kalıcı kapanış koruması ve token gizleme:** `4004` ve `4014` gibi `reconnect: false` kapanış kodlarında yeniden bağlanmaya çalışmaz; çözüm ipucu içeren `GatewayCloseError` fırlatır. İstisna metinlerinde (`str`, `repr`, log çıktıları) bot token'ı otomatik olarak `[REDACTED]` ile gizlenir.

Hata sınıfları ve örnek projeler

Yakalayabileceğin hata sınıfları `noluyo_bot` kökünden dışa aktarılır:

- `NoluyoError`: tüm kütüphane hatalarının taban sınıfı
- `ApiError`: `4xx` / `5xx` yanıtları (`err.status`, `err.code`, `err.errors`, `err.path`)
- `RateLimitedError`: yeniden deneme sınırını aşan `429` yanıtı (`err.retry_after`, `err.scope`)
- `GatewayCloseError`: kalıcı gateway kapanışı (`err.code`, `err.close_name`, `err.hint`)
- `VoiceJoinError`: ses kanalına giriş reddi (`err.code`, `err.guild_id`)

Tam çalışan Python örnek projeleri depoda yer alır:

- `sdks/examples/python/moderasyon`: sözcük süzgeci, `/sustur` zaman aşımı komutu ve denetim embed'i
- `sdks/examples/python/anket`: süreli buton oylaması
- `sdks/examples/python/muzik`: yerel ses dosyası yayınlayan ses botu
- `sdks/examples/python/oauth_site`: PKCE tabanlı "N'oluyo ile giriş" web uygulaması

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya Python ile N'oluyo botu yazdırırken bu sayfayı ver ve şu noktaları hatırlat:
>
> - Paket `pip install noluyo-bot` ile kurulur, kodda `import noluyo_bot` olarak içe aktarılır; başka sohbet platformlarının kütüphaneleri kullanılmaz.
> - Olay verileri sınıf nesnesi değil `TypedDict` sözlüğüdür: `message.content` değil, `message["content"]` ve `message["channel_id"]` yazılır.
> - `bot.rest.<snake_case_operation_id>` metotları `async` çalışır (`await bot.rest.create_message(...)`).
> - Token koda yazılmaz; `os.environ["BOT_TOKEN"]` üzerinden okunur.

Sırada ne var

- Hazır tarifler: hem Python hem JavaScript sekmeli 9 çalışan bot örneği
- JavaScript kütüphanesi: Node.js 22+ için sıfır bağımlılıklı kütüphane rehberi
- Botu açık tut: Python botunu Linux sunucuda sanal ortam ve `systemd` ile çalıştırma

Örnekler

Bu sayfadaki altı bot, bugünkü API ile çalışır ve Node.js 22 veya üstü dışında bir şey istemez: bu sürümlerde `WebSocket` hazırdır, ek paket gerekmez. İlk üçü mesajlarla çalışır; son üçü buton, menü ve embed kullanır. Hepsi aynı küçük yardımcı dosyaları kullanır. Ses kanalında müzik veya ses dosyası çalan bot örneği ise LiveKit paketi kullandığı için Ses sayfasındadır. curl örnekleri için ayrıca jq gerekir.

Başlamadan önce botunu oluşturup sunucuna ekle (Hızlı başlangıç bunu adım adım anlatır) ve token ile kanal ID'sini terminalde tanımla. Kanal ID'si zamanlı duyuru ve butonlu üç bot için gerekir; rol seçici sunucu ID'sini de ister:

```
export BOT_TOKEN='kopyaladigin_token'
export CHANNEL_ID='kanal id degeri'
```

> **Not:** Mesajları okuyan iki bot için portalda uygulama sayfasının **Ayrıcalıklı intent'ler** bölümünden **Mesaj içeriği**'ni aç. Açmazsan bağlantı `4014` ile kapanır. Neden gerektiği Intent'ler sayfasında.

Ortak yardımcılar

Olayları dinlemek için Gateway sayfasının sonundaki `gateway.mjs` dosyasını bu klasöre kaydet.

İstek göndermeyi tek yerde toplayan bu dosyayı `api.mjs` adıyla kaydet. `429` gelince söylenen süre kadar bekleyip yeniden dener, başka bir hatada ise durum ve kodu içeren bir hata fırlatır.

```
const API = "https://noluyo.net/api/v1";
const { BOT_TOKEN } = process.env;

export const sleep = (ms) => new Promise((done) => setTimeout(done, ms));

export async function call(method, path, body) {
  for (;;) {
    const res = await fetch(API + path, {
      method,
      headers: {
        Authorization: `Bot ${BOT_TOKEN}`,
        ...(body ? { "Content-Type": "application/json" } : {}),
      },
      body: body ? JSON.stringify(body) : undefined,
    });
    if (res.status === 204) return null;
    const data = await res.json();
    if (res.status === 429) {
      await sleep(data.retry_after * 1000);
      continue;
    }
    if (!res.ok) throw new Error(`${res.status} ${data.code}: ${data.message}`);
    return data;
  }
}
```

!ping yazılınca yanıt ver

Botun görebildiği her kanalı dinler ve `!ping` yazan mesajlara `pong!` diye yanıt verir.

Gereken izinler: Kanalları görüntüle, Mesaj gönder, Mesaj geçmişini oku. Bunların toplamı `68608` eder: ekleme bağlantısının sonu `izin=68608` olur.

Gereken intent'ler: `GUILD_MESSAGES` ve `MESSAGE_CONTENT`.

```
import { call } from "./api.mjs";
import { connect } from "./gateway.mjs";

let me;

await connect({
  intents: (1 << 9) | (1 << 15),
  async onEvent(t, d) {
    if (t === "READY") me = d.user.id;
    if (t !== "MESSAGE_CREATE" || d.author.id === me || d.content !== "!ping") return;
    // Tek bir hata botu durdurmasın.
    try {
      await call("POST", `/channels/${d.channel_id}/messages`, {
        content: "pong!",
        message_reference: { message_id: d.id },
      });
    } catch (err) {
      console.error(err.message);
    }
  },
});
```

Çalıştır ve kanala `!ping` yaz:

```
node ping.mjs
```

`message_reference`, gönderilen mesajı yanıt yapar; bunun için **Mesaj geçmişini oku** gerekir. Portalda **Mesaj içeriği** kapalıyken bu kod `4014` ile çıkar. `MESSAGE_CONTENT`'i `intents`'ten de çıkarırsan bağlanır, ama `content` boş gelir ve `!ping` hiç eşleşmez; yalnızca botu etiketleyen mesajların metni görünür.

Zamanlı duyuru gönder

Belirli aralıklarla bir kanala aynı mesajı gönderir. Bu bot gateway kullanmaz, REST yeterlidir: hiçbir olaya yanıt vermez, yalnızca saati gelince yazar. Açık bağlantıya ihtiyacı olmadığı için `cron` gibi bir zamanlayıcıyla da çalışır. Kanalı okumadığı için geçmiş iznine de ihtiyacı yoktur.

Gereken izinler: Kanalları görüntüle, Mesaj gönder. Toplamları `3072` eder: `izin=3072`.

```
// duyuru.mjs
import { call } from "./api.mjs";

const EVERY_HOUR = 60 * 60 * 1000;
const path = `/channels/${process.env.CHANNEL_ID}/messages`;

async function announce() {
  // Tek bir hata zamanlayıcıyı durdurmasın.
  try {
    const sent = await call("POST", path, { content: "Etkinlik bu akşam 21.00'de!" });
    console.log("gönderildi", sent.id);
  } catch (err) {
    console.error(err.message);
  }
}

await announce();
setInterval(announce, EVERY_HOUR);
```

```
# Her gün 09.00'da göndermek için crontab'a ekle. BOT_TOKEN ve CHANNEL_ID
# cron'un ortamında tanımlı olmalı; onları crontab'ın başına da yazabilirsin.
0 9 * * * curl -s -o /dev/null -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages"
-H "Authorization: Bot $BOT_TOKEN" -H "Content-Type: application/json" -d '{"content":"Günaydın,
bugünün duyurusu burada."}'
```

Kanalda yavaş mod açıksa ve aralık ondan kısaysa `429 slowmode` alırsın; `api.mjs` söylenen süre kadar bekleyip yeniden dener.

`@everyone` ile herkese bildirim göndermek istersen bota **@herkes ve rolleri etiketle** iznini de ver: `3072 + 131072 = 134144`. Bu izin olmadan `@everyone` düz yazı olarak kalır ve kimseye bildirim gitmez.

Belirli bir sözcüğü içeren mesajları sil

Yeni mesajları dinler ve listedeki sözcüklerden birini içerenleri siler.

Gereken izinler: Kanalları görüntüle, Mesaj geçmişini oku, Mesajları yönet. Toplamları `74752` eder: `izin=74752`.

Gereken intent'ler: `GUILD_MESSAGES` ve `MESSAGE_CONTENT`.

```
import { call } from "./api.mjs";
import { connect } from "./gateway.mjs";

const WORDS = ["reklam", "bedava üyelik"];
const lower = (text) => text.toLocaleLowerCase("tr-TR");

let me;

await connect({
  intents: (1 << 9) | (1 << 15),
  async onEvent(t, d) {
    if (t === "READY") me = d.user.id;
    if (t !== "MESSAGE_CREATE" || d.author.id === me) return;
    const text = lower(d.content);
    if (!WORDS.some((word) => text.includes(word))) return;
    try {
      await call("DELETE", `/channels/${d.channel_id}/messages/${d.id}`);
      console.log("silindi", d.id);
    } catch (err) {
      // Başka biri önce silmiş olabilir: 404 unknown_message.
      console.error(d.id, err.message);
    }
  },
});
```

Olaylar yalnızca bot çalışırken yazılanları getirir. Daha önceki mesajlar için kanalı REST ile bir kez tarayabilirsin. Bu okuma da aynı içerik kuralına uyar: **Mesaj içeriği** kapalıysa metinler boş gelir ve hiçbir şey eşleşmez.

```
# temizle.sh; son 100 mesajı bir kez tarar ve eşleşenleri siler.
API="https://noluyo.net/api/v1"
AUTH="Authorization: Bot $BOT_TOKEN"
MESSAGES="$API/channels/$CHANNEL_ID/messages"

curl -s "$MESSAGES?limit=100" -H "$AUTH" \
  | jq -r '.[] | select(.content | ascii_downcase | contains("reklam")) | .id' \
  | while read -r ID; do
      curl -s -o /dev/null -w "$ID %{http_code}\n" -X DELETE "$MESSAGES/$ID" -H "$AUTH"
      sleep 1
    done
```

Bilmen gerekenler:

- Mesaj silmek için rol sırası önemli değildir; `ManageMessages` yeterlidir. Kanal ayarları bu izni kaldırabilir, o zaman `403 missing_permissions` alırsın.
- Silme rotaları `write` sınıfındadır: kanal başına 10 saniyede 10 istek. `api.mjs` sınır aşılınca bekler.
- Başkasının mesajını silmek, sunucunun denetim kaydına botun adıyla yazılır.

Çok mesajı tek istekte silmek için toplu silme rotası var. 2 ile 100 arası mesaj alır, hiçbiri 14 günden eski olamaz ve saniyede bir istek kabul eder:

```
curl -i -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages/bulk-delete" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"messages":["111600000000000000","111600000000000100"]}'
```

Butonla anket

Kanala üç butonlu bir anket atar. Her basmada oyu sayar ve aynı mesajı `type: 7` ile yeni sayılarla günceller. Kişi başka butona basarsa oyu oraya geçer, iki kez oy veremez. Basmanın kendisi ve yanıt türleri Bileşenler ile Etkileşimler sayfalarında.

Gereken izinler: Kanalları görüntüle, Mesaj gönder. Toplamları `3072` eder: `izin=3072`. Embed için ayrı izin gerekmez.

Gereken intent'ler: Yok. `INTERACTION_CREATE` her bağlantıya gelir.

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const { CHANNEL_ID } = process.env;
if (!CHANNEL_ID) throw new Error("CHANNEL_ID tanımlı olmalı");

const QUESTION = "Bu akşam ne izleyelim?";
const OPTIONS = ["Film", "Dizi", "Maç"];
// Kim neye oy verdi: kişinin kimliği ve seçeneğin sırası. Bot yeniden başlarsa sıfırlanır.
const votes = new Map();

// Mesajın gövdesi: ilk gönderimde ve her güncellemede aynı biçim.
function view() {
  const count = (i) => [...votes.values()].filter((v) => v === i).length;
  return {
    content: QUESTION,
    embeds: [{
      color: parseInt("3366FF", 16),
      fields: OPTIONS.map((name, i) => ({ name, value: `${count(i)} oy`, inline: true })),
      footer: { text: `${votes.size} kişi oy verdi. Fikrin değişirse başka butona bas.` },
    }],
    components: [{ type: 1, components: OPTIONS.map((name, i) => (
      { type: 2, style: 1, label: name, custom_id: `anket:${i}` }
    )) }],
  };
}

async function onPress(d) {
  const [kind, choice] = d.data.custom_id.split(":");
  if (kind !== "anket") return;
  // Sunucuda member.user, DM'de user gelir.
  const who = (d.member?.user ?? d.user).id;
  votes.set(who, Number(choice));
  // type 7: butonun durduğu mesajı yeni sayılarla günceller.
  await call("POST", `/interactions/${d.id}/${d.token}/callback`, { type: 7, data: view() });
}

let posted = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !posted) {
      posted = true;
      call("POST", `/channels/${CHANNEL_ID}/messages`, view()).catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onPress(d).catch(console.error);
```

```
  },
});
```

Çalıştır, kanalda butonlara bas:

```
node anket.mjs
```

Bilmen gerekenler:

- Oylar programın belleğinde durur; bot yeniden başlarsa sıfırlanır ve yeni bir anket atar. Kalıcı olması gerekiyorsa oyları kendi veritabanında tut.
- Embed değiştiği için mesaj "(düzenlendi)" olarak işaretlenir.
- Kanalı gören herkes oy verebilir, `SendMessages` izni olmayan biri de.

Rol seçici

Kanala bir buton atar. Basan kişi, yalnızca kendisinin gördüğü bir menüde rollerini işaretler. Bot işaretlenen rolleri verir, işareti kaldırılanları geri alır. Menü ephemeral olduğu için kanal kalabalıklaşmaz: herkes kendi menüsünü görür.

Gereken izinler: Kanalları görüntüle, Mesaj gönder, Rolleri yönet. Toplamları `268438528` eder: `izin=268438528`. Dağıtılacak roller botun rolünün altında olmalı; değilse rol istekleri `403 role_hierarchy` döner.

Gereken intent'ler: Yok.

```
export GUILD_ID='sunucu id degeri'
export ROLE_IDS='rol id,diger rol id'
```

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const { CHANNEL_ID, GUILD_ID, ROLE_IDS } = process.env;
if (!CHANNEL_ID || !GUILD_ID || !ROLE_IDS) throw new Error("CHANNEL_ID, GUILD_ID ve ROLE_IDS
tanımlı olmalı");

// Menüde rollerin adları görünür; açılışta bir kez okunur.
const wanted = ROLE_IDS.split(",");
const roles = (await call("GET", `/guilds/${GUILD_ID}/roles`)).filter((r) =>
wanted.includes(r.id));
if (roles.length === 0) throw new Error("ROLE_IDS içindeki roller sunucuda yok");

// Yalnızca basan kişinin gördüğü menü. Kişinin şimdiki rolleri işaretli gelir.
const menu = (content, has) => ({
  content,
  flags: 64,
  components: [{ type: 1, components: [{
    type: 3,
    custom_id: "roller",
    placeholder: "Rol seç",
    min_values: 0,
    max_values: roles.length,
    options: roles.map((r) => ({ label: r.name, value: r.id, default: has.includes(r.id) })),
  }] }],
});

async function onPress(d) {
  if (!d.member) return;
  const answer = (body) => call("POST", `/interactions/${d.id}/${d.token}/callback`, body);
  const has = d.member.roles;

  if (d.data.custom_id === "roller-ac") {
    await answer({ type: 4, data: menu("Almak istediğin rolleri işaretle, bırakacaklarının
işaretini kaldır.", has) });
    return;
  }
  if (d.data.custom_id !== "roller") return;

  const picked = d.data.values;
  const add = roles.filter((r) => picked.includes(r.id) && !has.includes(r.id));
  const drop = roles.filter((r) => !picked.includes(r.id) && has.includes(r.id));
  const names = roles.filter((r) => picked.includes(r.id)).map((r) => r.name).join(", ") ||
"hiçbiri";

  // Önce yanıt ver: rol istekleri 3 saniyeyi aşabilir. Ephemeral mesajın tamamı gönderilir.
  await answer({ type: 7, data: menu(`Rollerin: ${names}`, picked) });
```

```
  const path = (r) => `/guilds/${d.guild_id}/members/${d.member.user.id}/roles/${r.id}`;
  try {
    for (const r of add) await call("PUT", path(r));
    for (const r of drop) await call("DELETE", path(r));
  } catch (err) {
    console.error(err.message);
    // Takip mesajı: ilk yanıttan sonra 15 dakika gönderilebilir. Kişiye hatanın ayrıntısı değil,
sade bir metin gider.
    await call("POST", `/interactions/${d.id}/${d.token}/messages`, { content: "Rollerin şu an
değiştirilemedi. Biraz sonra yeniden dene.", flags: 64 });
  }
}

let posted = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !posted) {
      posted = true;
      call("POST", `/channels/${CHANNEL_ID}/messages`, {
        content: "Sunucudaki rollerini buradan seçebilirsin.",
        components: [{ type: 1, components: [
          { type: 2, style: 1, label: "Rollerimi seç", custom_id: "roller-ac" },
        ] }],
      }).catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onPress(d).catch(console.error);
  },
});
```

```
node rol-secici.mjs
```

Bilmen gerekenler:

- Rol istekleri ilk yanıttan **sonra** gider. Bot önce `type: 7` ile cevap verir, böylece 3 saniye kuralı rol isteklerinin süresinden etkilenmez. Bir istek başarısız olursa kişi ayrı bir ephemeral takip mesajı görür.
- Ephemeral mesajı düzenlerken göndermediğin alan silinir. Bu yüzden `type: 7` her seferinde metni, `flags`'i ve menünün tamamını gönderir.
- Menünün işaretli seçenekleri olaydaki `member.roles`'dan gelir, yani kişinin o anki rolleridir. Kişi başka yerden rol alıp bırakırsa bir sonraki basışta doğru görünür.
- Ephemeral mesaj sayfa yenilenince gider. Kişi menüyü kaybederse butona yeniden basar.

Sayfalı liste

Uzun bir listeyi dörder satırlık sayfalar halinde gösterir. "Önceki" ve "Sonraki" butonları aynı mesajı `type: 7` ile değiştirir. Son basmadan 2 dakika sonra bot iki butonu da kapatır. Kapalı buton soluk görünür ve basılmaz; ekranı

henüz güncellenmemiş biri basarsa "Bu bileşen kapalı." görür ve bota bir şey gelmez.

Gereken izinler: Kanalları görüntüle, Mesaj gönder. `izin=3072`.

Gereken intent'ler: Yok.

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const { CHANNEL_ID } = process.env;
if (!CHANNEL_ID) throw new Error("CHANNEL_ID tanımlı olmalı");

// Kendi listenle değiştir.
const ITEMS = [
  "Herkese saygılı ol.", "Reklam yapma.", "Spoiler'ı gizle.", "Kanalın konusunda kal.",
  "Aynı mesajı tekrar tekrar atma.", "Kişisel bilgi paylaşma.", "Yetkiliye DM'den yaz.",
  "Ses kanalında müzik çalarken sor.", "Takma adın okunabilir olsun.", "Kurallar herkes için
geçerli.",
];
const PER_PAGE = 4;
const PAGES = Math.ceil(ITEMS.length / PER_PAGE);
// Son basmadan bu kadar sonra butonlar kapanır.
const OPEN_FOR = 2 * 60 * 1000;

function view(page, open = true) {
  const start = page * PER_PAGE;
  const lines = ITEMS.slice(start, start + PER_PAGE).map((item, i) => `${start + i + 1}.
${item}`);
  return {
    content: "Sunucu kuralları",
    embeds: [{ description: lines.join("\n"), footer: { text: `Sayfa ${page + 1} / ${PAGES}` }
}],
    // Gidilecek sayfa custom_id'nin içinde: bot yeniden başlasa da butonlar çalışır.
    components: [{ type: 1, components: [
      { type: 2, style: 2, label: "Önceki", custom_id: `sayfa:${page - 1}`, disabled: !open ||
page === 0 },
      { type: 2, style: 2, label: "Sonraki", custom_id: `sayfa:${page + 1}`, disabled: !open ||
page === PAGES - 1 },
    ] }],
  };
}

// Her mesajın kapanma zamanlayıcısı; her basmada baştan kurulur.
const timers = new Map();

function closeLater(messageId, page) {
  clearTimeout(timers.get(messageId));
  timers.set(messageId, setTimeout(() => {
    timers.delete(messageId);
    // Yalnızca components gönderilir: metin ve embed kalır, mesaj "(düzenlendi)" olmaz.
    call("PATCH", `/channels/${CHANNEL_ID}/messages/${messageId}`, { components: view(page,
false).components })
      .catch(console.error);
  }, OPEN_FOR));
```

```
}

async function onPress(d) {
  const [kind, page] = d.data.custom_id.split(":");
  if (kind !== "sayfa") return;
  await call("POST", `/interactions/${d.id}/${d.token}/callback`, { type: 7, data:
view(Number(page)) });
  closeLater(d.message.id, Number(page));
}

let posted = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !posted) {
      posted = true;
      call("POST", `/channels/${CHANNEL_ID}/messages`, view(0))
        .then((m) => closeLater(m.id, 0))
        .catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onPress(d).catch(console.error);
  },
});
```

```
node sayfali-liste.mjs
```

Bilmen gerekenler:

- Gidilecek sayfa `custom_id`'nin içinde durur, bot ek bir kayıt tutmaz. Sunucu `custom_id`'nin mesajda gerçekten olduğunu doğruladığı için kimse olmayan bir sayfa isteyemez.
- Sayfayı biri değiştirince kanaldaki herkes yeni sayfayı görür. Herkes kendi sayfasında gezinsin istiyorsan listeyi ephemeral bir yanıtla aç; Rol seçici aynı yolu kullanır.
- Butonları kapatan `PATCH` yalnızca `components` gönderir: metin ve embed olduğu gibi kalır, mesaj " (düzenlendi)" olmaz.
- Zamanlayıcılar programın belleğindedir. Bot yeniden başlarsa eski mesajın butonları açık kalır ve çalışmaya devam eder, çünkü sayfa `custom_id`'nin içindedir.

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip bot yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur.
- `INTERACTION_CREATE` gelince ilk yanıt 3 saniye içinde gider. Uzun iş (rol vermek, başka bir servise sormak) yanıttan sonra yapılır; gerekirse önce `type: 5` ya da `type: 6` gönderilir.
- Etkileşim başına tek `callback` gönderilir; sonrası `@original` ya da takip mesajıdır (en çok 5).
- Gereken bilgi `custom_id`'ye yazılır (en çok 100 karakter, mesajda tekil). Gizli bilgi konmaz: kanalı gören herkesin istemcisine gider.
- Basan kişinin yetkisi `member.permissions`'tan denetlenir; sunucu bunu senin yerine yapmaz.
- Ephemeral (`flags: 64`) mesaj dosya almaz; düzenlerken mesajın tamamı gönderilir.
- Bir mesajda en çok 5 satır, satırda en çok 5 buton ya da tek menü olur. Embed sınırları Embed'ler sayfasında.
- `429` gelince `retry_after` kadar beklenir.

Sırada ne var

- Botu sürekli çalışır tutarken token'ı nasıl koruyacağın: Güvenlik
- Bağlantının ayrıntıları: Gateway
- Buton, menü ve embed'in ayrıntıları: Bileşenler, Etkileşimler, Embed'ler
- Bir şey ters gidiyorsa: Sık sorulan sorular
- Başka neler yapılabileceği: REST referansı

Embed'ler ve bileşenler

Embed, bir mesajın metninin altında duran karttır: başlık, açıklama, alanlar, görsel ve renkli bir şerit taşır. Bileşen ise mesaja eklenen buton ve menüdür. İkisini de yalnızca botlar gönderebilir.

İlk embed

Çalışan en küçük istek şudur; `content` yazmak gerekmez:

```
curl -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"embeds":[{"title":"Merhaba","description":"İlk embed","color":3368703}]}'
```

Yanıttaki mesajda `embeds` dizisi gelir ve `application_id` botun uygulamasını gösterir. Aynı mesaj gateway'de `MESSAGE_CREATE` olarak da akar.

Embed alanları

Bir mesajda en çok **10** embed olur. Hepsinin metinleri toplam **6000** karakteri, hepsinin JSON'u **16384** baytı (UTF-8) geçemez. Bir embed'de başlık, açıklama, alan, yazar, alt bilgi, görsel veya küçük görselden en az biri bulunmalıdır. Sınır sütunundaki sayılar karakter ya da öğe sayısıdır.

Alan	Tür	Sınır	Zorunlu	Açıklama
`title`	metin	256	hayır	Başlık
`description`	metin	4096	hayır	Ana metin
`url`	metin	2048	hayır	Başlığın bağlandığı `https` adresi
`color`	sayı	16777215	hayır	Şeridin rengi; ondalık sayı olarak, aşağıya bak
`timestamp`	metin	-	hayır	RFC 3339 zamanı
`fields`	liste	25	hayır	Ad ve değer çiftleri
`fields[].name`	metin	256	evet	Alanın adı
`fields[].value`	metin	1024	evet	Alanın değeri
`fields[].inline`	doğru/yanlış	-	hayır	`true` ise alanlar yan yana durur
`footer.text`	metin	2048	evet	Alt bilgi metni
`footer.icon_url`	metin	-	hayır	`attachment://` ile bir görsel dosyası
`author.name`	metin	256	evet	Yazarın adı
`author.url`	metin	2048	hayır	Yazarın bağlandığı `https` adresi
`author.icon_url`	metin	-	hayır	`attachment://` ile bir görsel dosyası
`image.url`	metin	-	hayır	`attachment://` ile bir görsel dosyası
`thumbnail.url`	metin	-	hayır	`attachment://` ile bir görsel dosyası

Bilmediğin bir alan yazarsan görmezden gelinir ve kaydedilmez. Metinlerdeki kontrol karakterleri (satır sonu ve sekme dışında) hata verir. Başında ve sonundaki boşluklar silinir.

Görsel göndermek

Embed'in görseli, aynı mesajla yüklediğin bir dosyadır. Başka bir sitedeki adres kabul edilmez: görseli okuyanın tarayıcısı dışarıdaki bir sunucuya gitmesin, okuyanın IP adresi üçüncü bir tarafa ulaşmasın diye N'oluyo yalnızca kendi sakladığı dosyaları gösterir. `https://...` ile başlayan bir görsel adresi `invalid` ile reddedilir.

Dosyalı mesaj `multipart/form-data` ile gider. JSON `payload_json` parçasına, dosyalar `files[0]`, `files[1]` gibi parçalara konur:

```
curl -X POST "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -F 'payload_json={"embeds":[{"title":"Haftalık grafik","image":
{"url":"attachment://grafik.png"}}]}' \
  -F 'files[0]=@grafik.png'
```

- `-F 'payload_json=...'`: mesajın JSON'u. Embed'deki `attachment://grafik.png`, aynı istekteki `grafik.png` dosyasına işaret eder.
- `-F 'files[0]=@grafik.png'`: yüklenen dosya. Kanalda `AttachFiles` izni gerekir.
- Yanıtta `image.attachment_id`, dosyanın mesajın `attachments` listesindeki kimliğidir. `image.url` yanıtta da `attachment://` olarak kalır; görseli bulmak için kimliği kullan.

Ad şu kurallarla eşleştirilir:

- Dosya adı yüklenirken temizlenir: harf, rakam, `-`, `_` ve bir `.` kalır, her diğer karakter `_` olur. `haftalık grafik.png` böylece `haftalık_grafik.png` olur. `attachment://` adresindeki ad da aynı biçimde temizlenir, yani iki yazım da aynı dosyayı bulur.
- Yalnızca PNG, JPEG, GIF ve WebP görselleri gösterilebilir. PDF veya başka bir dosyaya başvurmak `invalid` verir.
- Mesaja eklenmemiş bir ada başvurmak `invalid` verir.
- Aynı adı taşıyan iki dosya varsa o ada başvurmak `invalid` verir; dosyaları farklı adlarla yükle.

Renk ve zaman

`color` ondalık bir sayıdır, `#RRGGBB` yazımının sayıya çevrilmiş halidir. Onaltılık renkten çevirmek için:

```
const color = parseInt("3366FF", 16); // 3368703
```

`timestamp` RFC 3339 biçimindedir, örneğin `2026-11-07T18:30:00Z`. JavaScript'te `new Date().toISOString()` doğru biçimi verir. Zaman UTC olarak saklanır ve okuyanın saatine çevrilerek gösterilir.

Metin, markdown ve etiketler

Metinler markdown biçimlendirmesiyle yazılabilir; sunucu metni olduğu gibi saklar, çizimi istemci yapar. Embed içinde yazdığın `<@kullanıcı_id>`, `<@&rol_id>` ve `@everyone` kimseyi etiketlemez; bildirim göndermez, okunmamış sayacını artırmaz.

> **Not:** Birine bildirim göndermek istiyorsan etiketi mesajın `content` alanına yaz.

Kimi etiketlersin

`allowed_mentions` yalnızca daraltır: botun `@everyone` etiketleme izni yoksa `parse` içinde `everyone` yazmak bir şey değiştirmez. Etiket metinde kalır, yalnızca bildirim gitmez.

Gönderdiğin	Sonuç
hiçbiri	Bir kişinin yazdığı mesajla aynı: metindeki her etiket bildirim gönderir.
`"parse": []`	Hiçbir etiket bildirim göndermez.
`"parse": ["users"]`	Yalnızca kullanıcı etiketleri.
`"parse": ["roles"]`, `["everyone"]`	Yalnızca rol etiketleri, yalnızca `@everyone` ve `@here`.
`"users": ["id"]`	Yalnızca listedeki kullanıcılar. `roles` için aynısı. Listeler en çok 100 kimlik taşır.
`"replied_user": false`	Yanıt, yanıtlanan kişiye bildirim göndermez.

`parse` içinde `users` varken `users` listesi (rol için de aynısı) yazılamaz; istek `validation_failed` ile döner, `parse` içinde bilinmeyen bir değer de öyle. Aşağıdaki mesaj metinde kullanıcıyı etiketler ama kimseye bildirim göndermez, okunmamış sayacını da artırmaz:

```
{ "content": "<@111500000000000001> toplantı 15:00'te", "allowed_mentions": { "parse": [] } }
```

Mesajı `PATCH` ile düzenlerken `content` değişiyorsa `allowed_mentions` o istekte de gönder; düzenleme kimseye yeni bildirim göndermez ama kimin etiketli sayıldığı güncellenir. `allowed_mentions` göndermezsen düzenlenen metin, bir kişinin yazdığı metin gibi çözülür.

Özel emojiler (`<:ad:id>`) `content` ve embed metinlerinde botun kullanabildiği emojilerle sınırlıdır; kullanamadığı emoji `:ad:` düz metnine döner. Embed'ler ve bileşenler birlikte en çok **50** özel emoji taşır, fazlası `too_long` döner. Emoji sunucudaki güncel adıyla yazılır; ad senin yazdığından uzunsa 16384 bayt sınırı bu haliyle denetlenir.

Webhook'tan embed

Webhook bağlantısına (`POST /api/v1/webhooks/{id}/{token}`) gönderdiğin `embeds` de aynı kurallarla gösterilir; `content` yazmak gerekmez. İki fark var:

- Görseller düşer. `image`, `thumbnail`, `author.icon_url` ve `footer.icon_url` hata vermez ama saklanmaz; dış adres okuyanlara hiç ulaşmaz. Başlık, açıklama ve alanlar durur. Yalnızca görselden oluşan bir embed görselle birlikte kaybolur.
- Bileşen yoktur ve mesaj bir uygulamaya bağlanmaz; `application_id` gelmez. `flags` ve `allowed_mentions` eskisi gibi alınıp yok sayılır.

Bugün çalışan bir isteğin kırılmaması için embed'ler hoşgörüyle okunur. Bilmediğimiz alanlar (`type`, `video` gibi) yok sayılır. Kurallara uymayan bir embed varsa (`title` 257 karakter, `http://` adres, yalnızca `color` taşıyan embed ya da liste olmayan `embeds` gibi) mesajın **bütün** embed'leri düşer ve `content` yine gönderilir. `400` yalnızca geriye hiçbir şey kalmayınca gelir: `content` boşsa hata embed'lerin neyinin yanlış olduğunu söyler. Embed'ler botlarınkiyle aynı yayılma bütçesine sayılır. Sunucunun otomatik moderasyonu embed metnine de bakar.

Kişi ne görür

> **Not:** Bu bölüm webde gözlenen davranışı anlatır. Masaüstü uygulaması aynı web kodunu kullanır ama ayrıca çalıştırılmadı. Android ve iOS uygulamalarında henüz gözlenmedi.

Embed, mesajın metninin altında bir kart olarak çizilir. Kartın sol kenarında `color` renginde ince bir şerit durur; `color` yoksa şerit kişinin seçtiği temanın rengini alır. Yukarıdan aşağıya sıra şudur: yazar, başlık, açıklama, alanlar, görsel, altbilgi ve zaman. `thumbnail` kartın sağ üstünde küçük bir kare olur, dar ekranda biraz daha küçülür.

`inline: true` alanlar yan yana dizilir: geniş kartta üç sütun, telefonda tek sütun. Satır içi olmayan alan tüm genişliği kaplar. `url` verdiysen `title` ve `author.name` yeni sekmede açılan bağlantı olur. Açıklamada ve alan değerlerinde markdown çalışır, HTML etiketleri düz metin olarak görünür. Uzun bir kelime kartın dışına taşmaz, satıra bölünür.

Embed'in gösterdiği dosya (`image`, `thumbnail`, `author` ve `footer` simgesi) mesajın dosya listesinde ikinci kez görünmez. Embed'e bağlanmayan dosyalar kartın altında her zamanki gibi durur. Görsele basınca büyür. Bir mesajdaki 10 embed alt alta dizilir.

Metni olmayan bir mesajda sabitlenmiş mesajlar, arama sonuçları, yanıt satırı ve bildirim ilk embed'in başlığını, başlık yoksa açıklamasını gösterir.

Eski bir uygulama sürümü embed'i çizmez: dosyaları gösterir, yalnızca embed'den oluşan mesaj boş görünür. Kaçırılmaması gereken bir bilgiyi `content`'e de yaz.

Bir mesajda neler birlikte olur

Birleşim	Olur mu	Not
`content` ve embed	Evet	Metin üstte, embed altta durur
Yalnız embed	Evet	`content` gerekmez
Embed ve dosya	Evet	Embed, dosyayı `attachment://` ile gösterir; dosya `attachments` listesinde de kalır
Yalnız bileşen	Hayır	`content`, embed veya dosya gerekir; `content` için `required` döner
Embed ve anket, iletme ya da çıkartma	Hayır	`embeds` için `invalid` döner
Bileşen ve anket, iletme ya da çıkartma	Hayır	`embeds` için `invalid` döner

Bir kişi `embeds`, `components`, `flags` veya genişletilmiş `allowed_mentions` gönderirse istek `validation_failed` ile reddedilir. Bir mesaj iletilirse metni ve dosyaları gider, embed ve bileşenleri gitmez; yalnızca embed'den oluşan bir mesaj iletilemez.

Bileşenler

Bileşenler eylem satırlarına konur. Bir mesajda en çok **5** satır olur; satır 1-5 buton ya da tam bir menü taşır:

Alan	Tür	Sınır	Zorunlu	Açıklama
`components`	liste	5	hayır	Eylem satırları; her satırın `type` değeri `1`
`components[].components`	liste	5	evet	Satırdaki butonlar ya da tek bir menü
`type`	sayı	-	evet	`2` buton, `3` metin menüsü, `5` kullanıcı, `6` rol, `8` kanal menüsü
`style`	sayı	-	buton için	`1` birincil, `2` ikincil, `3` başarılı, `4` tehlike, `5` bağlantı
`label`	metin	80	buton için	Buton yazısı
`custom_id`	metin	100	bağlantı dışında	Mesajda tekil olmalı
`url`	metin	2048	bağlantı için	`https` adresi
`emoji`	nesne	-	hayır	`{"name":"□"}` ya da botun kullanabildiği `{"id":"..."}`
`placeholder`	metin	150	hayır	Menüde seçim yokken görünen yazı
`options`	liste	25	metin menüsü için	Seçenekler
`options[].label`	metin	100	evet	Seçenek yazısı
`options[].value`	metin	100	evet	Seçenek değeri; menüde tekil
`min_values`, `max_values`	sayı	25	hayır	Seçilebilecek en az ve en çok değer; varsayılan 1

Tüm bileşenlerin JSON'u **16384** baytı geçemez. Bileşenler `content`, embed veya dosyayla birlikte gönderilir.

Bir kişi butona bastığında bot `INTERACTION_CREATE` alır. Akış, menü türleri ve botun alacakları Bileşenler sayfasında.

Düzenleme

`PATCH /channels/{channel_id}/messages/{message_id}` ile botun kendi mesajı düzenlenir. `content`, `embeds` ve `components` ayrı ayrı gönderilebilir:

```
curl -X PATCH "https://noluyo.net/api/v1/channels/$CHANNEL_ID/messages/$MESSAGE_ID" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"components":[]}'
```

- Göndermediğin alan olduğu gibi kalır. `components` yazıp başka bir şey yazmazsan metin ve embed'ler aynı kalır.
- Boş liste (`"embeds":[]` ya da `"components":[]`) o alanı kaldırır.
- Yalnızca `components` değişirse mesaj düzenlendi sayılmaz; `content` veya `embeds` değişirse `edited_timestamp` dolar.
- Embed'deki `attachment://` yalnızca mesajın var olan dosyalarına bakar. Yeni dosya yüklemek isteyen `multipart` istek `invalid_body` ile reddedilir; yeni görsel yeni bir mesajla gönderilir.
- Mesajda metin, embed ve dosya kalmıyorsa `content` için `required` döner.

Mesaj içeriği intent'i

`MESSAGE_CONTENT` intent'i olmayan bir bot, başka kişilerin ve botların mesajlarındaki `embeds` ve `components` alanlarını görmez; gateway olaylarında da REST yanıtlarında da bu alanlar gelmez. Kendi mesajları ve kendisini etiketleyen mesajlar eksiksiz gelir. `flags` ve `application_id` her zaman gelir. Ayrıntı için Intent'ler sayfasına bak.

Limit aşımı

Sınırı aşan alan, adıyla birlikte döner:

```
{
  "code": "validation_failed",
  "message": "One or more fields are invalid.",
  "errors": { "embeds[0].title": "too_long" }
}
```

Hata anahtarı sorunlu alanın yoludur. Yapılacak şey metni kısaltmak ya da içeriği birkaç mesaja bölmektir. Toplam sınırlar aşılırsa anahtar `embeds` veya `components` olur.

Embed'ler ayrıca ortak bir tavana sayılır: her mesajın embed ve bileşenleri sunucunun üye sayısıyla çarpılarak tüm botların ve webhook'ların birlikte kullandığı bir bütçeden düşer; bir sahibin botlarına bunun bir payı düşer. Payın dolunca `X-RateLimit-Scope: bot`, bütçenin tamamı dolunca `shared` başlığıyla `429` gelir; ikincisi botunun suçu değildir. `retry_after` kadar bekleyip yeniden dene. Bütçenin 30 saniyesinden fazla tutan mesaj `400 fanout_too_large` alır; `max_bytes` o kanalın alabileceği en büyük embed ve bileşen boyutunu söyler. Kalabalık bir sunucuya giden büyük bir mesaj o saniyede kalanı alır, bütçe onu ödeyene kadar tüm botların sayılan mesajlarını bekletir, sen de bir sonraki büyük mesajından önce bunun dört katı beklersin. Ayrıntı Hız sınırları sayfasında.

Node.js ile tam örnek

Node.js 22 veya üstü yeter, ek paket gerekmez. Token'ı koda yazma; ortam değişkeninden okunur. `grafik.png` aynı klasörde olmalı:

```
import { readFile } from "node:fs/promises";

const API = "https://noluyo.net/api/v1";
const token = process.env.BOT_TOKEN;
const channelId = process.env.CHANNEL_ID;
if (!token || !channelId) throw new Error("BOT_TOKEN ve CHANNEL_ID tanımlı olmalı");

async function call(method, path, body) {
  for (;;) {
    const res = await fetch(API + path, {
      method,
      headers: { Authorization: `Bot ${token}`, ...(body instanceof FormData ? {} : { "Content-
Type": "application/json" }) },
      body: body instanceof FormData ? body : JSON.stringify(body),
    });
    const data = await res.json();
    if (res.status === 429) {
      await new Promise((done) => setTimeout(done, data.retry_after * 1000));
      continue;
    }
    if (!res.ok) throw new Error(`${res.status} ${data.code} ${JSON.stringify(data.errors ??
{})}`);
    return data;
  }
}

const form = new FormData();
form.set(
  "payload_json",
  JSON.stringify({
    embeds: [
      {
        title: "Haftalık grafik",
        description: "Geçen haftanın özeti",
        color: parseInt("3366FF", 16),
        timestamp: new Date().toISOString(),
        fields: [{ name: "Toplam", value: "128", inline: true }],
        image: { url: "attachment://grafik.png" },
      },
    ],
    components: [
      { type: 1, components: [{ type: 2, style: 5, label: "Ayrıntılar", url: "https://noluyo.net"
}] },
    ],
  }),
);
form.set("files[0]", new Blob([await readFile("grafik.png")], { type: "image/png" }),
"grafik.png");
```

```
const message = await call("POST", `/channels/${channelId}/messages`, form);
console.log("gönderildi:", message.id);

// Başlığı değiştir; görsel ve bileşenler olduğu gibi kalır.
const embeds = message.embeds.map((e) => ({ ...e, title: "Güncel grafik" }));
await call("PATCH", `/channels/${channelId}/messages/${message.id}`, { embeds });
```

Düzenleme isteğinde `image.attachment_id` gibi yanıttan gelen alanlar gönderilince yok sayılır; görsel `attachment://` adresiyle aynı dosyaya yeniden bağlanır.

Sık hatalar

Hata	Neden	Ne yapmalı
`embeds[0].image.url: invalid`	Adres `attachment://` ile başlamıyor, dosya mesajla gönderilmemiş, görsel değil ya da aynı adlı iki dosya var	Dosyayı aynı istekte, görsel olarak ve tekil adla yükle
`embeds: too_long`	Toplam 6000 karakter ya da 16384 bayt aşıldı, ya da 50'den çok özel emoji var	Metni kısalt ya da birkaç mesaja böl
`embeds[0]: required`	Embed boş	En az bir alan doldur
`content: required`	Yalnız bileşen gönderildi ya da düzenleme sonrası mesaj boş kalıyor	Metin, embed veya dosya ekle
`components[0].components[0].emoji: invalid`	Botun kullanamadığı özel emoji	Botun kullanabildiği bir emoji ya da `{"name":"□"}` yaz
`ephemeral_forbidden`	Kanala giden mesajda `flags: 64`	`flags` alanını çıkar
`automod_blocked`	Sunucunun otomatik moderasyonu embed metnini ya da buton yazısını engelledi	Metni değiştir
`invalid_body`	`embeds` liste olmalı ama başka tür gönderildi	Gövdeyi kontrol et
`429` ve `X-RateLimit-Scope: shared`	Ortak bütçe dolu	`retry_after` kadar bekle

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur ve depoya girmeyen bir yerde tutulur.
- Embed görselleri yalnızca `attachment://` ile, aynı mesajla yüklenen PNG, JPEG, GIF ya da WebP dosyalarından gelir. Dış adres kullanılmaz.
- Renk `#RRGGBB` değil, ondalık sayıdır.
- Embed sınırları: mesajda en çok 10 embed, hepsinde toplam 6000 karakter ve 16384 bayt; `title` 256, `description` 4096 karakter, en çok 25 alan, alan adı 256, alan değeri 1024 karakter.
- Bileşen sınırları: en çok 5 satır, satırda 5 buton ya da tek menü, `custom_id` 100 karakter ve mesajda tekil, JSON'u en çok 16384 bayt. Yalnızca bileşenden oluşan mesaj gönderilmez.
- Butona basılınca ilk yanıt 3 saniye içinde gider; uzun iş yanıttan sonra yapılır.
- Bildirim gerekiyorsa etiket `content` içine yazılır, embed içine değil.
- `429` gelince `retry_after` kadar beklenir; hemen yeniden denenmez.

Sırada ne var

- Bileşenler: mesajın altına eklenen buton ve seçim menüleri
- Etkileşimler: bileşen ve komut yanıtlarının yaşam döngüsü
- Hız sınırları: embed yayılma bütçesi ve `429` başlıkları

Bileşenler

Bileşen, botun mesajına eklediği buton ve menüdür. Bir kişi butona bastığında ya da menüden seçim yaptığında bot bunu `INTERACTION_CREATE` olayı olarak alır. Bu sayfa akışı baştan sona anlatır. Botla açılmış DM'de de aynı çalışır; tek fark etkileşimin `context: 1` ile, `member` yerine `user` ile gelmesi ve kullanıcı menüsünde yalnızca DM'nin iki kişisinin, rol menüsünde hiçbir şeyin seçilememesidir. Bileşenin mesajla nasıl gönderildiği ve JSON sınırları Embed'ler ve bileşenler sayfasındadır.

Akış



Sunucu her basmada mesajın **o anki** bileşenlerine bakar. Bot butonu kaldırdıysa eski ekranda duran butona basmak bota hiçbir şey ulaştırmaz, kişi `component_not_found` alır. Bot olaya gelen `custom_id`'nin kendi gönderdiği bir değer olduğuna bu yüzden güvenebilir.

`INTERACTION_CREATE` için intent gerekmez; bağlantın ne seçmiş olursa olsun gelir. Çok shard'lı botta olay, sunucunun shard'ına gider. Alanların tamamı Olaylar sayfasında.

Olaya nasıl yanıt vereceğin Etkileşimler sayfasında.

Satır ve buton

Bileşenler eylem satırlarına konur (`type: 1`). Bir mesajda en çok 5 satır, bir satırda en çok 5 buton olur:

```
{
  "content": "Bu akşam geliyor musun?",
  "components": [
    {
      "type": 1,
      "components": [
        { "type": 2, "style": 3, "label": "Geliyorum", "custom_id": "katilim:evet" },
        { "type": 2, "style": 2, "label": "Gelemem", "custom_id": "katilim:hayir" },
        { "type": 2, "style": 5, "label": "Etkinlik sayfası", "url":
"https://example.com/etkinlik" }
      ]
    }
  ]
}
```

Beş stil vardır:

`style`	Ad	Görünüm	Ne zaman
`1`	Birincil	Vurgu rengi, dolu	Mesajın beklediği asıl eylem
`2`	İkincil	Gri, dolu	Yan seçenekler, vazgeçme
`3`	Başarılı	Yeşil, dolu	Onay, katılma, kabul
`4`	Tehlike	Kırmızı, dolu	Silme, yasaklama, geri dönmeyen işler
`5`	Bağlantı	Gri, dışarı oklu	Bir adres açar, bota bir şey gitmez

Bağlantı butonunda `custom_id` olmaz, `url` zorunludur. Basıldığında bot olay almaz. Bir butonu kapatmak için `"disabled": true` yaz; kapalı bileşene basma isteği `component_disabled` ile döner.

Menüler

Menü bir satırı tek başına doldurur. Dört türü vardır. Hepsinde `custom_id`, `placeholder`, `min_values`, `max_values` ve `disabled` alanları ortaktır. `min_values` ve `max_values` yazılmazsa ikisi de 1 sayılır; `min_values` 0 ise kişi hiçbir şey seçmeden gönderebilir ve `values` boş liste gelir. Menüler bir formun içinde de durabilir; orada etiket alırlar.

Metin menüsü

`type: 3`. Seçenekleri sen yazarsın, bot seçilen `value` değerlerini alır:

```
{
  "type": 1,
  "components": [
    {
      "type": 3,
      "custom_id": "renk",
      "placeholder": "Renk seç",
      "max_values": 2,
      "options": [
        { "label": "Kırmızı", "value": "kirmizi" },
        { "label": "Mavi", "value": "mavi", "description": "Sakin olan" }
      ]
    }
  ]
}
```

Bot şunu alır:

```
{ "custom_id": "renk", "component_type": 3, "values": ["kirmizi", "mavi"] }
```

Bir seçeneğe `"default": true` yazarsan menü o seçenek işaretli açılır; kişi değiştirmeden gönderebilir.

Listede olmayan bir değer, tekrar eden bir değer ya da `min_values` ve `max_values` dışında bir sayı `validation_failed` ile döner ve bota ulaşmaz.

Kullanıcı menüsü

`type: 5`. Kişi sunucunun üyeleri arasından seçer. `default_values` ile baştan seçili kişiler verebilirsin:

```
{
  "type": 1,
  "components": [
    { "type": 5, "custom_id": "uyar", "placeholder": "Kimi uyaralım?", "default_values": [{ "id":
"14631410073600003", "type": "user" }] }
  ]
}
```

Bot, `values` içinde kimlikleri, `resolved` içinde kişilerin bilgisini alır:

```
{
  "custom_id": "uyar",
  "component_type": 5,
  "values": ["14631410073600003"],
  "resolved": {
    "users": { "14631410073600003": { "id": "14631410073600003", "username": "ayse",
"display_name": "Ayşe", "avatar": null, "cosmetics": null } },
    "members": { "14631410073600003": { "user": { "id": "14631410073600003", "username": "ayse",
"display_name": "Ayşe", "avatar": null, "cosmetics": null }, "nick": null, "roles": [],
"joined_at": "2026-11-01T10:00:00Z", "mute": false, "deaf": false,
"communication_disabled_until": null } }
  }
}
```

Sunucunun üyesi olmayan biri seçilemez. `resolved`, seçilen kişi için `GUILD_MEMBERS` intent'in olmasa da gelir. DM'de `members` yer almaz.

Rol menüsü

`type: 6`. Kişi sunucunun rolleri arasından seçer:

```
{ "type": 1, "components": [{ "type": 6, "custom_id": "rol-ver", "placeholder": "Rol seç" }] }
```

Bot şunu alır:

```
{
  "custom_id": "rol-ver",
  "component_type": 6,
  "values": ["21531879014400009"],
  "resolved": {
    "roles": { "21531879014400009": { "id": "21531879014400009", "name": "Oyuncu", "color":
3368703, "hoist": false, "mentionable": true, "position": 2, "permissions": "68672",
"auto_assign": false } }
  }
}
```

Başka bir sunucunun rolü seçilemez.

Kanal menüsü

`type: 8`. Kişi yalnızca **kendisinin gördüğü** kanalları seçebilir. `channel_types` listeyi kanal türüne göre daraltır: `0` metin, `2` ses, `4` kategori.

```
{ "type": 1, "components": [{ "type": 8, "custom_id": "duyuru-kanali", "channel_types": [0] }] }
```

Bot şunu alır:

```
{
  "custom_id": "duyuru-kanali",
  "component_type": 8,
  "values": ["21531887403008007"],
  "resolved": {
    "channels": { "21531887403008007": { "id": "21531887403008007", "type": 0, "name": "genel",
"parent_id": null } }
  }
}
```


Bot o kanalı görmüyor olabilir, bu yüzden `resolved.channels` yalnızca kimlik, tür, ad ve üst kategoriyi taşır. Konu ya da izin ayarları gelmez.

custom_id tasarımı

`custom_id` en çok 100 karakterdir ve mesajda tekil olmalıdır. Bot yeniden başlasa bile basmayı anlayabilsin diye gereken bilgiyi içine yaz:

- `rol:21531879014400009`: hangi rol olduğu değerin içinde, ek kayıt gerekmez.
- `anket:42:evet`: önce iş, sonra kayıt, sonra seçenek. `:` ile bölmek yeter.
- Kişinin göreceği metin ya da gizli bilgi koyma. Değer mesajla birlikte kanalı görebilen herkesin istemcisine gider.

Sınırlar

Sınır	Değer
Mesajdaki satır	5
Satırdaki buton	5
`custom_id`	100 karakter
Buton yazısı	80 karakter
Metin menüsü seçeneği	25
Seçenek yazısı ve değeri	100 karakter
`placeholder`	150 karakter
Seçilen değer	25
Basma, kişi başına kanalda	5 sn'de 5
Basma, kişi başına tüm kanallarda	5 sn'de 10
Basma, uygulama başına	saniyede 22
Basma, tüm uygulamalar	saniyede 67

Basma sınırları kişiyi korur, botunu cezalandırmaz: dolduğunda `429` yalnızca basan kişiye gider, mesaj akışı etkilenmez. Kişi başına sınırlar `rate_limited` verir; tüm kanallardaki sınır, birkaç hesabın ortak sınırı tek başına doldurmasını önler. Uygulama ve ortak sınır `interaction_busy` kodunu verir; ortak sınır doluyken o anda başka bir bota basan kişi de bu kodu alabilir. Sınırı aşan basma bota ulaşmaz.

Mesajın bileşenleri, embed'leriyle birlikte yayılma bütçesine sayılır: boyutları sunucunun üye sayısıyla çarpılır. Sunucu büyüdükçe en büyük mesaj küçülür; onu aşan mesaj `max_bytes` ile `400 fanout_too_large` alır.

Kişi ne görür

> **Not:** Bu bölüm webde gözlenen davranışı anlatır. Masaüstü uygulaması aynı web kodunu kullanır ama ayrıca çalıştırılmadı. Android ve iOS uygulamalarında henüz gözlenmedi.

Butonlar mesajın altında satır satır, uygulamanın kendi buton stilleriyle durur: birincil temanın rengini alır, ikincil gri, başarılı yeşil, tehlike kırmızıdır. Kapalı bileşen soluk görünür ve basılmaz. Bağlantı butonunun yazısının yanında bir bağlantı simgesi olur, üzerine gelince hedefin alan adı görünür. Basınca adres yeni sekmede açılır. Masaüstü uygulamasında, koda göre, varsayılan tarayıcıda açılması gerekir; bu henüz gözlenmedi. Yalnızca `https` adresli bağlantı butonu çizilir.

Kişi basınca buton dönen bir simgeye döner ve bot yanıtlayana kadar o mesajın bütün bileşenleri kapalı kalır: bir mesajda aynı anda tek basma olur. Bot 3 saniye içinde yanıtlamazsa kişi yaklaşık 3,5 saniye sonra bileşenlerin

altında "Bot yanıt vermedi." görür. Botun çevrimdışıysa olay kaybolur ve kişi aynı metni görür: olaylar bot bağlanınca sonradan gönderilmez. Hata metni 5 saniye sonra kendiliğinden gider.

Menü satırı boydan boya kaplar ve seçim yokken `placeholder`'ı gösterir; yazmadıysan "Seçim yap" yazar. `max_values` değeri 1 olan metin menüsü uygulamanın açılır listesidir, seçim yapılır yapılmaz gönderilir. Çok seçimli metin menüsü ile kullanıcı, rol ve kanal menüleri bir pencerede liste açar: kişi seçenekleri işaretler ve "Onayla"ya basar. `min_values`'tan azıyla "Onayla" kapalı kalır, `max_values` dolunca başka seçenek işaretlenemez. Seçeneğin `emoji`'si adının önünde, `description`'ı altında küçük yazıyla görünür. Değerler bota listedeki sırayla gider.

Kullanıcı, rol ve kanal menüsünün penceresinde arama kutusu olur. Kullanıcı menüsü sunucunun üyelerini, rol menüsü `@everyone` dışındaki rolleri en üstteki rolden başlayarak listeler. Kanal menüsünde yalnızca kişinin gördüğü ve `channel_types`'a uyan kanallar çıkar. Gönderdikten sonra menü seçilenlerin adlarını gösterir. Mesaj güncellenince menüler yeniden `default` ve `default_values` ile başlar.

Klavyeyle de kullanılır: `Tab` bileşenler arasında gezer, `Enter` ya da boşluk basar. Açılır listede oklar seçeneği değiştirir, pencerede `Enter` ya da boşluk işaretler, `Esc` kapatır.

Android uygulamasında, koda göre, her menü ekranın altından açılan bir panelde listelenir; bu davranış telefonda henüz gözlenmedi. Tek seçim isteyen menüde (`min_values` ve `max_values` 1) seçeneğe dokunmak onu hemen gönderir, öbürlerinde "Onayla" vardır. Butonların dokunma alanı en az 48 dp'dir. Bağlantı butonu adresi telefonun tarayıcısında açar.

Bot `type: 7` ile yanıt verince mesaj kişinin ekranında yerinde değişir: kapattığın buton soluklaşır, kaldırdığın buton gider.

Şunlar basmayı daha baştan durdurur ve bota hiçbir şey gitmez. Kişi hatayı bileşenlerin altında görür:

Durum	Yanıt	Kişinin gördüğü
Kanalı görmüyor	`403 missing_access`	"Bu kanalı görme yetkin yok."
Zaman aşımında	`403 missing_permissions`	"Bunun için yetkin yok."
Bot butonu kaldırmış	`404 component_not_found`	"Bu bileşen artık geçerli değil."
Bileşen kapalı	`400 component_disabled`	"Bu bileşen kapalı."
Bot sunucudan çıkmış ya da uygulama silinmiş	`409 application_unavailable`	"Bu mesajı gönderen bot artık burada değil."
Ephemeral mesajın süresi dolmuş	`400 interaction_expired`	"Bu etkileşimin süresi doldu."
Kişi çok hızlı basıyor	`429 rate_limited`	"Çok hızlı gidiyorsun, biraz yavaşla."
Uygulama ya da ortak sınır dolu	`429 interaction_busy`	"Bot şu an yoğun. Biraz sonra yeniden dene."

Ekranı henüz güncellenmemiş biri botunun kaldırdığı bir butona basarsa basma bota ulaşmaz, kişi "Bu bileşen artık geçerli değil." görür. Uygulama `429`'u beklemez ve yeniden denemez: geç kalan bir basma botunun 3 saniyesini zaten kaçırmış olur.

Kanalı gören herkes basabilir, `SendMessages` gerekmez: salt okunur bir duyuru kanalındaki rol butonu böyle çalışır.

Yalnızca basan kişinin göreceği butonlu mesaj için Etkileşimler sayfasına bak.

Yetkiyi sen denetlersin

> **Uyarı:** Sunucu `custom_id`'nin mesajda gerçekten olduğunu doğrular, ama basan kişinin o işi yapmaya yetkili olduğunu doğrulamaz. Kanalı gören herkes "Yasakla" butonuna basabilir. İş yetki istiyorsa `member.permissions`'a bak.

`member.permissions` kişinin bu kanaldaki izinleridir, ondalık metin olarak gelir. `BigInt` ile denetle:

```
const BAN_MEMBERS = 1n << 2n;

if (!(BigInt(d.member.permissions) & BAN_MEMBERS)) {
  // Yetkisi yok: işi yapma.
}
```

`app_permissions` de botun kendi izinleridir. DM'de `member` yerine `user` gelir ve izinler DM'in sabit izinleridir.

Tam örnek: rol seçme botu

Node.js 22 veya üstü yeter. Gateway sayfasındaki `gateway.mjs` ile Örnekler sayfasındaki `api.mjs` aynı klasörde olmalı. Botun `ManageRoles` izni olmalı ve rolü kendi rolünün altında kalmalı. Bot her açıldığında kanala bir buton atar, basan kişiye rolü verir, yeniden basana geri alır. Ekranda değişen bir şey olmadığı için yanıtı `type: 6`'dır:

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const { CHANNEL_ID, ROLE_ID } = process.env;
if (!CHANNEL_ID || !ROLE_ID) throw new Error("CHANNEL_ID ve ROLE_ID tanımlı olmalı");

let posted = false;

async function onPress(d) {
  // Sunucu custom_id'yi doğruladı: içindeki rol bizim yazdığımız roldür.
  const [kind, roleId] = d.data.custom_id.split(":");
  if (kind !== "rol" || !d.member) return;
  await call("POST", `/interactions/${d.id}/${d.token}/callback`, { type: 6 });
  const path = `/guilds/${d.guild_id}/members/${d.member.user.id}/roles/${roleId}`;
  await call(d.member.roles.includes(roleId) ? "DELETE" : "PUT", path);
}

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !posted) {
      posted = true;
      call("POST", `/channels/${CHANNEL_ID}/messages`, {
        content: "Rolü almak ya da bırakmak için bas.",
        components: [{ type: 1, components: [
          { type: 2, style: 1, label: "Oyuncu", custom_id: `rol:${ROLE_ID}` },
        ] }],
      }).catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onPress(d).catch(console.error);
  },
});
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' CHANNEL_ID='kanal id degeri' ROLE_ID='rol id degeri' node rol-botu.mjs
```

Sık hatalar

- **Olay hiç gelmiyor.** Bot bağlı mı? Çevrimdışı bota olay sonradan gönderilmez. Çok shard'lı botta olay sunucunun shard'ına gider, her shard'ı dinle.
- **`component_not_found` alan kişiler var.** Bot mesajı düzenleyip `custom_id`'yi değiştirmiş; ekranı güncellenmemiş kişi eski değere basıyor. Bu doğru davranıştır.
- **`custom_id` iki kez kullanılmış.** Mesaj gönderilirken `validation_failed` döner; değer mesajda tekil olmalıdır.

- **Yönetim butonuna herkes basabiliyor.** Yetkiyi sunucu denetlemez, `member.permissions`'a bak.
- **Rol verilemiyor.** Botun `ManageRoles` izni yoksa ya da rol botun rolünün üstündeyse REST isteği `403` döner.
- **`values` içinde kimlik yerine başka bir şey bekleniyor.** Kullanıcı, rol ve kanal menülerinde `values` kimliklerdir; ayrıntı `resolved` içindedir.

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur.
- `INTERACTION_CREATE` gelince ilk yanıt 3 saniye içinde gider. Önce yanıt verilir, uzun iş ondan sonra yapılır.
- Sunucu `custom_id`'nin mesajda olduğunu doğrular, basanın yetkisini doğrulamaz. Yetki isteyen işte `member.permissions` `BigInt` ile denetlenir.
- Gereken bilgi `custom_id`'ye yazılır; gizli bilgi konmaz, çünkü kanalı gören herkesin istemcisine gider.
- Sınırlar: mesajda en çok 5 satır, satırda 5 buton ya da tek menü; `custom_id` 100, buton yazısı 80 karakter; en çok 25 seçenek, seçenek yazısı ve değeri 100, `placeholder` 150 karakter.
- Bağlantı butonu (`style: 5`) `url` alır, `custom_id` almaz ve bota olay göndermez.
- Bot çevrimdışıyken yapılan basma sonradan gelmez.
- Kullanıcı, rol ve kanal menüsünde `values` kimliklerdir; ayrıntı `resolved` içindedir.

Sırada ne var

- Etkileşimler: butona ve menüye basıldığında gelen olayın yanıt türleri
- Formlar: buton tıklamasıyla en çok beş alanlı pencere açmak
- Hazır tarifler: butonlu rol seçici ve anket örnekleri

Komutlar

Komut, botunun sunduğu ve kişinin `/` yazınca listede ya da bir kişiye veya mesaja sağ tıklayınca menüde gördüğü iştir. Bot komutlarını REST ile bir kez kaydeder; kişi bir komutu çalıştırınca botuna bir etkileşim gelir ve ona Etkileşimler sayfasındaki gibi yanıt verirsin.

Kişi ne zaman görür

Kaydettiğin komut sunucuda hemen saklanır ve REST'te hemen görünür. Kişilerin uygulamaları komut listesini en çok 60 saniyede bir yeniler: yeni, değişen ya da silinen komut kişinin ekranına en geç 60 saniyede gelir. Kişiye ayrıca bir olay gitmez.

Bu arada eski tanımla çalıştırılan komut reddedilmez; kişinin uygulaması yeni tanımı alır ve uyan değerleri korur. Yani tanımı değiştirmek için botunu durdurman gerekmez.

Kişi listede yalnızca o kanalda kullanabileceği komutları görür. Kanalda komut kullanma izni (`UseApplicationCommands`) yoksa ya da kişi zaman aşımındaysa liste boştur. Sunucunun komut izinleri ve komutun `default_member_permissions` değeri, çalıştırmada olduğu gibi listede de uygulanır: kişinin çalıştıramayacağı komut listede çıkmaz. Sunucu sahibi ve `Administrator` izni olanlar hepsini görür. Bot sunucudan çıkarılırsa komutları da listeden kalkar, geri eklenince döner. Komutlar sunucunun yazı kanallarında ve kişinin botunla açtığı DM'de listelenir; DM'de hangilerinin çıktığı DM'de komut bölümünde.

Kişi ne görür

> **Not:** Bu bölüm webde gözlenen davranışı anlatır. Masaüstü uygulaması aynı web kodunu kullanır ama ayrıca çalıştırılmadı. Android uygulamasında henüz gözlenmedi.

Kişi mesaj kutusunun başına `/` yazınca, imleç ilk kelimedeyken, kutunun üstünde bir liste açılır. Komutlar botlara göre gruplanır: her grubun başında botun avatarı ve adı, altında her komutun adı ve açıklaması durur. Yazdıkça liste daralır; adın başıyla eşleşenler önce gelir. Arama büyük küçük harfe ve i, ı, I, İ farkına bakmaz: `/IS` yazan kişi `/ısınma`'yı da görür. Yerelleştirdiğin ad da aranır.

Hiçbir komut eşleşmezse liste hiç çizilmez ve Enter mesajı düz metin olarak gönderir: `/home/kero yedek nerede` ya da `/shrug` olduğu gibi gider. Tek başına `/` yazıp Enter'a basan kişi de düz `/` gönderir; listeden seçmek için `Tab`, `Enter` (en az bir harf yazılmışken) ya da tıklama gerekir. Klavyede `↑` ve `↓` satır değiştirir, `Esc` listeyi kapatır.

Alt komutu olan komut seçilince liste o komutun satırlarına geçer: `/mod kisi ban`, `/mod at` gibi, her biri kendi açıklamasıyla. Grup ve alt komut adları satırda yan yana yazılır.

Komut seçilince mesaj kutusunun yerine bir form gelir. Başında botun avatarı, komutun tam yolu ve açıklaması, sağda kapatma düğmesi durur. Zorunlu seçenekler önce gelir ve adlarının yanında kırmızı bir yıldız vardır. Her seçeneğin altında açıklaması yazar. Seçenek türüne göre denetim:

Tür	Kişi ne görür
`3` metin, `4` tam sayı, `10` sayı	Yazı kutusu. Sayıda virgül de olur: `0,5` sana `0.5` olarak gelir.
`choices` olan seçenek	Uygulamanın açılır listesi, seçeneklerin yerelleştirilmiş adlarıyla.
`autocomplete: true`	Yazı kutusu; üstünde botunun önerileri açılır (Otomatik tamamlama).
`5` mantıksal	Aç kapa düğmesi. Zorunlu değilse ve kişi dokunmazsa hiç gönderilmez.
`6` kullanıcı, `8` rol, `7` kanal, `9` kişi ya da rol	Bir düğme; basınca aramalı bir liste penceresi açılır, kişi birini işaretleyip "Onayla"ya basar. Kanal listesinde yalnızca kişinin gördüğü ve `channel_types`'a uyan kanallar çıkar. `9`'da roller kişilerin altında, adlarının önünde `@` ile listelenir.
`11` dosya	"Dosya seç" düğmesi; kişinin yükleme sınırını aşan dosya seçilemez. Seçilen dosya adı ve boyutuyla görünür, çarpıyla kaldırılır.

Kişi Enter'a ya da "Gönder"e basınca uygulama önce zorunlu alanları, uzunlukları, aralıkları ve listeyi denetler; hata alanın altında kırmızı yazar ("Bu alan gerekli", "1 ile 10 arasında olmalı." gibi) ve komut gitmez. Her şey tamsa "Gönder" dönen bir simgeye döner ve bot yanıtlayana kadar bekler. Yanıt gelince form kapanır, mesaj kutusu boşalır. Yanıtın üstünde "{kişi}, /{komut} komutunu kullandı" yazar; `type: 5` ile ertelersen botun mesajı düşünüyor hâlini gösterir.

Botun 3 saniye içinde yanıt vermezse ya da çevrimdışıysa form açık kalır ve altında "Bot yanıt vermedi." yazar; kişi yeniden gönderebilir. Sunucudan gelen hata da formun altında görünür, form kapanmaz. Tanımı kişi formu doldururken değiştirirsen kişinin uygulaması yeni tanımı alır, adı ve türü aynı kalan değerleri korur, gideni düşürür ve "Komut güncellendi, kontrol edip yeniden gönder." yazar. `Esc` ya da kapatma düğmesi formu kapatır ve kişinin yazdığı metin mesaj kutusuna geri gelir.

Kullanıcı ve mesaj menüsü komutların (`type: 2` ve `3`) sağ tık menüsünde "Uygulamalar" adlı bir alt menüde çıkar: mesaja sağ tıklayınca mesaj komutları, bir kişiye (mesajdaki adına, üye listesinde ya da seste) sağ tıklayınca kullanıcı komutları. Komutlar ada göre sıralanır; birden çok botun komutu varsa her satırın sonunda botun adı yazar. Alt menü yalnızca bir sunucunun yazı kanalında ya da botla açılmış DM'de, o türden en az bir komut varken görünür; ana menü yalnız bir satır uzar, uzun alt menü kayar. Kullanıcı menüsü açık olan kanalda çalışır, bu yüzden yalnız o sunucunun üyesine (DM'de DM'nin kişilerine) sağ tıklayınca çıkar. Menü açılırken ağı beklemez: uygulama listeyi henüz almadıysa onu o an ister ve alt menü bir sonraki sağ tıkta görünür. Komut seçilince hemen çalışır; yanıt kanalda belirir, botun yanıt vermezse ya da bir hata olursa ekranın altında kısa bir uyarı çıkar.

İlk komut: /selam

Uygulama ID'si portalda uygulamanı açınca adres satırının sonunda durur. Bu istek global bir `/selam` komutu kaydeder:

```
curl -X POST "https://noluyo.net/api/v1/applications/$APPLICATION_ID/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"name":"selam","description":"Birini selamlar"}'
```

Yanıt `201` ve komuttur:

```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": null,
  "type": 1,
  "name": "selam",
  "name_localizations": null,
  "description": "Birini selamlar",
  "description_localizations": null,
  "options": [],
  "default_member_permissions": null,
  "contexts": [0, 1]
}
```

Aynı istek ikinci kez `200` döner: aynı türde ve adda komut varsa tanımı değişir, `id` aynı kalır. `version` tanım her değiştiğinde değişir.

Global mi, sunucu mu

Konu	Global	Sunucu komutu
Rota	`/applications/{application_id}/commands`	`/applications/{application_id}/guilds/{guild_id}/com`
Nerede görünür	Botun bulunduğu her sunucuda	Yalnızca o sunucuda
Şart	Yok	Bot o sunucunun üyesi olmalı, değilse `403 missing_acc`
Sınır	100 eğik çizgi komutu, her menü türünden 15	Sunucu başına aynısı, globalden ayrı

Geliştirirken sunucu komutu kullan. Yarım kalmış komut yalnızca kendi deneme sunucunda görünür, botunu kullanan diğer sunucular onu görmez. Bitince aynı tanımı global rotaya gönder ve deneme sunucusundakini sil. İkisi aynı adı taşıyabilir: kapsamlar ayrıdır.

Sunucu silinince oradaki komutlar da silinir. Bot sunucudan çıkarılırsa komutları gizlenir ama silinmez; bot geri eklenince döner. Bot üye değilken o sunucunun rotaları `403` döner.

Üç tür

`type`	Tür	Kişi nerede görür	`description` ve `options`
`1`	Eğik çizgi komutu	`/` yazınca açılan liste	Açıklama zorunlu, seçenek alabilir
`2`	Kullanıcı menüsü	Bir kişiye sağ tıklayınca	İkisi de gönderilmez
`3`	Mesaj menüsü	Bir mesaja sağ tıklayınca	İkisi de gönderilmez

`type` gönderilmezse `1` sayılır. Tür sonradan değişmez: değiştirmek için komutu silip yeniden oluştur.

Alanlar

Gönderdiğin gövde:

Alan	Tür	Zorunlu	Kural
`type`	tam sayı	Hayır	`1`, `2` ya da `3`; varsayılan `1`
`name`	metin	Evet	En çok 32 karakter, ad kuralları
`name_localizations`	nesne	Hayır	Yalnızca `tr` ve `en` anahtarları
`description`	metin	Eğik çizgi komutunda	1 ile 100 karakter; menüde boş
`description_localizations`	nesne	Hayır	Yalnızca `tr` ve `en`
`options`	dizi	Hayır	En çok 25 seçenek; menüde yok
`default_member_permissions`	metin ya da `null`	Hayır	Ondalık izin bit alanı, aşağıda
`contexts`	dizi	Hayır	Nerede çalıştığı: `0` sunucu, `1` botla DM; varsayılan `[0, 1]`, aşağıda

Yanıtta bunlara ek olarak `id`, `application_id`, `version` ve `guild_id` (global komutta `null`) gelir. Bilinmeyen alanlar botun isteğinde yok sayılır ve saklanmaz.

Seçenek alanları

Alan	Tür	Kural
`type`	tam sayı	1 ile 11, türler
`name`	metin	Komut adıyla aynı kural, en çok 32 karakter
`name_localizations`	nesne	`tr` ve `en`
`description`	metin	1 ile 100 karakter
`description_localizations`	nesne	`tr` ve `en`
`required`	evet/hayır	Zorunlu seçenekler diğerlerinden önce gelir
`choices`	dizi	Yalnızca 3, 4 ve 10 türünde; en çok 25 seçim, her birinin `name` alanı en çok 100 karakter, `value` seçeneğin türünde ve metinse en çok 100 karakter
`autocomplete`	evet/hayır	Yalnızca 3, 4 ve 10 türünde, `choices` ile birlikte olmaz
`options`	dizi	Alt komutun seçenekleri ya da grubun alt komutları
`channel_types`	tam sayı dizisi	Yalnızca 7 türünde: seçilebilecek kanal türleri
`min_value`, `max_value`	sayı	Yalnızca 4 ve 10 türünde; ±9007199254740992 içinde, 4 türünde tam sayı
`min_length`, `max_length`	tam sayı	Yalnızca 3 türünde; 0 ile 6000

Seçenek türleri

`type`	Tür	Ne işe yarar	Botun aldığı `value`
`1`	Alt komut	`/rol ver` gibi bir dal	Değer yok, kendi `options` listesi gelir
`2`	Alt komut grubu	Alt komutları toplar	Değer yok, içindeki alt komut gelir
`3`	Metin	Serbest yazı, istersen `choices` ile sınırlı	Metin, en çok 6000 karakter
`4`	Tam sayı	Sayım, adet	Tam sayı
`5`	Evet/hayır	Bir açma kapama	`true` ya da `false`
`6`	Kullanıcı	Sunucunun bir üyesini seçtirir	Kullanıcının ID'si, metin olarak
`7`	Kanal	Kişinin görebildiği bir kanalı seçtirir	Kanalın ID'si
`8`	Rol	Sunucunun bir rolünü seçtirir	Rolün ID'si
`9`	Kullanıcı ya da rol	İkisinden birini seçtirir	Seçilenin ID'si
`10`	Sayı	Ondalıklı değer	Sayı
`11`	Dosya	Kişiden dosya ister	Dosyanın ID'si

Sunucu kişinin gönderdiği değeri tanıma karşı denetler: tür, zorunluluk, `choices`, aralık ve uzunluk. Seçilen kişi, kanal ya da rolün ayrıntısı etkileşimle birlikte gelir. `autocomplete` açık seçeneğin değeri bir listeye karşı denetlenmez; onu botun denetlemeli. Ayrıntı Otomatik tamamlama sayfasında. Uzun ya da çok satırlı metin istiyorsan seçenek yerine komuta bir form ile yanıt ver.

Alt komutlar ve gruplar

Bir komut ya düz seçenekler taşır ya da alt komutlar; ikisi aynı seviyede karışmaz. Grup alt komut taşır, alt komut düz seçenek taşır. Daha derin seviye yoktur:

```
/rol
├── ver      (alt komut: kisi, rol)
├── al       (alt komut: kisi, rol)
└── ayar     (grup)
    ├── goster   (alt komut)
    └── sifirla  (alt komut)
```

Bu ağacın tanımı:

```
{
  "name": "rol",
  "description": "Rolleri yönetir",
  "options": [
    { "type": 1, "name": "ver", "description": "Bir kişiye rol verir", "options": [
      { "type": 6, "name": "kisi", "description": "Kime", "required": true },
      { "type": 8, "name": "rol", "description": "Hangi rol", "required": true }
    ] },
    { "type": 1, "name": "al", "description": "Bir kişiden rolü alır", "options": [
      { "type": 6, "name": "kisi", "description": "Kimden", "required": true },
      { "type": 8, "name": "rol", "description": "Hangi rol", "required": true }
    ] },
    { "type": 2, "name": "ayar", "description": "Rol ayarları", "options": [
      { "type": 1, "name": "goster", "description": "Ayarları gösterir" },
      { "type": 1, "name": "sifirla", "description": "Ayarları sıfırlar" }
    ] }
  ]
}
```

Kişi her çalıştırmada tek bir alt komut seçer. Alt komutlu bir komutun kendisi çalıştırılamaz.

Ad kuralları

- Eğik çizgi komutu ve seçenek adı: harf, rakam, `-` ve `_`; en çok 32 karakter; **yalnızca küçük harf**. Türkçe harfler çalışır: `/ısınma`, `/çekiliş`, `/iade`.
- Büyük harf reddedilir, Türkçe büyük harfler de: `/İade` ve `/Selam` `400 validation_failed` döner, alan `name`, kod `invalid`. `/iade` ile `/ısınma` farklı komutlardır; sunucu `i` ile `ı`'yı birleştirmez.
- Harfleri birleşik (NFC) biçimde gönder. Bir harfin noktasını ayrı bir işaretle yazan (NFD) metin reddedilir. Çoğu klavye ve dil zaten NFC yazar; emin değilsen `name.normalize("NFC")` kullan.
- Menü adı (tür 2 ve 3): en çok 32 karakter, büyük harf ve tek boşluk olabilir, başta ve sonda boşluk olamaz: `Profili aç`.
- Ad, aynı seviyede her dilde tekil olmalı. İki seçeneğin ya da aynı türden iki komutun adı, `tr` ya da `en` karşılığı dahil, çakışırsa çakışan `invalid` alır.

Yerelleştirme

`name_localizations` ve `description_localizations` yalnızca `tr` ve `en` anahtarlarını alır. Kişi uygulamayı hangi dilde kullanıyorsa o dildeki adı görür; karşılık yoksa `name` görünür. Botuna her zaman `name` gelir, dili ne olursa olsun.

```
{
  "name": "selam",
  "name_localizations": { "en": "hello" },
  "description": "Birini selamlar",
  "description_localizations": { "en": "Greets someone" }
}
```

Karşılıklar da ad kurallarına uyar. Başka bir dil anahtarı `invalid` döner.

Toplu güncelleme: PUT

`PUT` listeyi kapsamın tüm komutları yapar, tek adımda:

```
curl -X PUT "https://noluyo.net/api/v1/applications/$APPLICATION_ID/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '[{"name":"selam","description":"Birini selamlar"},{"type":2,"name":"Profili aç"}]'
```

- Türü ve adı eşleşen komutun `id`'si kalır. Tanımı değişmediyse `version`'ı da kalır.
- Listede olmayan komut silinir. Boş liste `[]` hepsini siler.
- Bir tanım geçersizse hiçbir şey değişmez, eski küme yerinde kalır. Hata alanları sırayla işaretlenir: `[1].name` ikinci komutun adıdır.
- Yanıt `200` ve kapsamın yeni tam listesidir.

Botun her açılışta komutlarını `PUT` ile göndersin, tek tek `POST` ile değil. `POST` silinen komutu silmez, eski bir komut listede kalır; `PUT` kodundaki listeyi olduğu gibi yazar. Kapsam başına günde 200 **yeni** komut oluşturulabilir. `PUT` yalnızca listede yeni olanları sayar; aynı listeyi her açılışta göndermek bu sınırı harcamaz. Her açılışta komutu silip yeniden oluşturan bot ise sınıra çarpar ve `429` alır.

Kim kullanabilir: default_member_permissions

Sunucu kendi kurallarını koyana dek komutu kimin kullanacağını söyler:

Değer	Kim kullanır
`null` ya da hiç gönderilmez	Kanalda komut kullanma izni olan herkes
`"0"`	Yalnızca sunucu sahibi ve `Administrator` izni olanlar
`"6"` gibi bir sayı	Kanalda bu bitlerin hepsine sahip olanlar, ayrıca sahip ve `Administrator`

Değer, İzinler sayfasındaki bitlerin toplamıdır, ondalık metin olarak. `KickMembers` (bit 1, 2) ile `BanMembers` (bit 2, 4) birlikte `2 + 4 = 6` eder. JavaScript'te `BigInt` ile hesapla:

```
const kick = 1n << 1n;
const ban = 1n << 2n;
const value = (kick | ban).toString(); // "6"
```

Bu yalnızca bir varsayılandır, güvenlik değildir. Sunucu yöneticileri izin kurallarıyla komutu başkalarına açabilir ya da kapatabilir. Botun yetki isteyen işte yine kendi denetimini yapmalı.

DM'de komut

Kişi botunla bir DM açarsa (Özel mesajlar) global komutların orada da çalışır: eğik çizgi komutu, kullanıcı menüsü ve mesaj menüsü. Nerede çalışacağını `contexts` söyler:

Değer	Nerede çalışır
`0`	Sunucunun yazı kanalları
`1`	Kişinin botunla açtığı DM

`contexts` gönderilmezse `[0, 1]` sayılır: komut ikisinde de çalışır. Boş liste, `0` ve `1` dışında bir değer ya da aynı değerin iki kez yazılması `validation_failed`, `contexts`: `invalid` döner. Yanıttaki komut `contexts`'i her zaman taşır. `PATCH` ile yalnızca `contexts` gönderip komutu DM'den çekebilirsin.

Bir komut DM'de ancak üç şart birlikteyse görünür ve çalışır:

Şart	Neden
Global komut	Sunucu komutu bir sunucuya aittir; DM'nin sunucusu yoktur.
`contexts` içinde `1`	Sen DM'ye açtın.
`default_member_permissions` `null`	DM'de kimsenin izin biti yoktur. Yetki isteyen komut (`"0"` dahil) `contexts` ne derse desin DM'de görünmez ve çalışmaz: `/ban` DM'ye düşmez.

Kişinin DM'deki listesinde yalnızca o botun bu şartlara uyan komutları çıkar. İki kişinin DM'sinde ve grup DM'de komut yoktur. Sunucunun komut izinleri DM'de uygulanmaz; DM'de kişiyi durduran onun kapatması ve engelidir.

`PUT` ile komutun birini yalnız sunucuya, birini yalnız DM'ye ayır:

```
curl -X PUT "https://noluyo.net/api/v1/applications/$APPLICATION_ID/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '[{"name":"selam","description":"Birini selamlar"},{"name":"ayarlar","description":"Sunucu ayarları","contexts":[0]},{"name":"hatirlat","description":"Sana hatırlatırım","contexts":[1]}]'
```

DM'den gelen etkileşimde `context` `1`'dir, `guild_id` ve `member` yoktur, kişi `user` içindedir, `data.guild_id` de yoktur. Kodunu ikisine birden yaz: Sunucuda mı DM'de mi.

Komutu yanıtla

Kişi komutu çalıştırınca botuna `INTERACTION_CREATE` gelir, `type` değeri `2`'dir. Intent gerekmez; olay sunucunun shard'ına gider. Ona bir butona basma gibi 3 saniye içinde `callback` ile yanıt verirsin. Gerçek bir çalıştırma şöyle görünür (`/zar yuz:20`):

```
{
  "id": "1118042062848001 20",
  "application_id": "14631410073600008",
  "type": 2,
  "token": "eyJpZCI6IjExMTgwNDIwNjI4NDgwMDEyMCJ9.bWFj",
  "version": 1,
  "guild_id": "21531879014400001",
  "channel_id": "21531887403008007",
  "member": {
    "user": { "id": "14631410073600003", "username": "ayse", "display_name": "Ayşe", "avatar": null, "cosmetics": null },
    "nick": null,
    "roles": [],
    "joined_at": "2026-11-01T10:00:00Z",
    "mute": false,
    "deaf": false,
    "communication_disabled_until": null,
    "permissions": "1071698529857"
  },
  "app_permissions": "1071698529857",
  "locale": "tr",
  "data": {
    "id": "1118042062848000 50",
    "name": "zar",
    "type": 1,
    "guild_id": "21531879014400001",
    "options": [{ "name": "yuz", "type": 4, "value": 20 }]
  }
}
```

`data` alanı	Ne taşır
`id`	Komutun kimliği
`name`	Komutun tanımdaki adı. Kişi İngilizce kullanıyor olsa da çevirisi değil, `zar` gelir.
`type`	`1` eğik çizgi komutu, `2` kullanıcı menüsü, `3` mesaj menüsü
`guild_id`	Sunucu komutunda o sunucu; global komutta yer almaz
`options`	Kişinin doldurdukları: `name`, `type`, `value`. Alt komut ve grup `value` yerine kendi `options` listesini taşır, yol tanımdaki gibi gelir: `rol > renk > ver` için `options[0].options[0].options`. Boşsa yer almaz.
`target_id`	Menülerde kişinin sağ tıkladığı kullanıcı ya da mesaj
`resolved`	Seçeneklerdeki ve menüdeki kimliklerin karşılıkları: `users`, `members`, `roles`, `channels`, `messages`, `attachments`

Sunucu senin yerine şunları denetledi: kişinin bu kanalda komutu kullanabilmesi (`UseApplicationCommands`, zaman aşımı, komut izinleri, `default_member_permissions`), seçeneklerin tanıma uyması (tür, zorunlu, seçim listesi, aralık, uzunluk) ve her kullanıcı, rol, kanal ve mesajın kişinin gösterebileceği biri olması. Kullanıcı seçeneği sunucunun üyesidir, rol bu sunucunundur, kanal kişinin gördüğü ve `channel_types` içindedir. Uymayan çalıştırma kişiye hata döner, sana hiç gelmez. Kişinin uygulaması eski bir tanımla göndermişse sunucu onu reddeder, uygulama yeni tanımı alıp yeniden dener.

Kullanıcı menüsünde `target_id` kişinin seçtiği üyedir, `resolved.users` ve `resolved.members` onu taşır. Mesaj menüsünde `target_id` bu kanalın mesajıdır ve `resolved.messages` mesajı içeriğiyle birlikte taşır, botunun `MESSAGE_CONTENT` intent'i olmasa da. Kişinin o kanalda `ReadMessageHistory` izni olmalıdır; ephemeral mesaj hedef olamaz.

> **Not:** Kullanıcı menüsü ve mesaj menüsü, hedefi kişinin seçimiyle botuna gönderir. Hedef kişi ya da mesajın yazarı bundan haberdar edilmez, ve sen `GUILD_MEMBERS` ya da `MESSAGE_CONTENT` almasan da onun profilini ve mesajın içeriğini görürsün. Bunu yalnızca menünün işi için kullan, saklama.

Yanıt `type: 4` ya da `type: 5` olur, `flags: 64` ile yalnızca çalıştıran kişi görür. Komut çalıştırmanın güncellenecek bir mesajı olmadığı için `type: 6` ve `type: 7` `400 interaction_response_invalid` döner. Yanıtın üstünde "Ayşe, /zar kullandı" yazar; mesajın `interaction` alanı `name: "zar"` taşır. Takip mesajları ve `@original` Etkileşimler sayfasındaki gibidir.

Dosya seçeneği

Tür `11` seçenek kişiden komutla birlikte bir dosya ister. Kişi dosyayı seçip komutu çalıştırınca N'oluyo dosyayı saklar ve sana yalnızca bir indirme bağlantısı verir. Seçeneğin `value` alanı dosyanın kimliğidir, ayrıntısı `resolved.attachments` içindedir:

```
"data": {
  "name": "oku",
  "type": 1,
  "options": [{ "name": "dosya", "type": 11, "value": "1118042062848003 00" }],
  "resolved": {
    "attachments": {
      "1118042062848003 00": {
        "id": "1118042062848003 00",
        "filename": "resim.png",
        "content_type": "image/png",
        "size": 5120,
        "width": 640,
        "height": 480,
        "url": "/attachments/2153188740300 8007/1118042062848003 00/resim.png?
ex=1793000000&sig=..."
      }
    }
  }
}
```

Alan	Ne taşır
`id`	Dosyanın kimliği, seçeneğin `value` değeriyle aynı
`filename`	Dosyanın adı, sunucunun temizlediği hâliyle
`content_type`	Dosyanın içeriğinden çıkarılan tür; kişinin söylediği değil
`size`	Bayt cinsinden boyut
`width`, `height`	Yalnızca resimlerde, piksel
`url`	İmzalı indirme bağlantısı. Başına `https://noluyo.net` ekleyip token göndermeden `GET` ile indirirsin.

Kurallar:

- Bağlantı **15 dakika** geçerlidir. Sonra `403 link_invalid` döner ve dosya kısa süre sonra silinir. Dosyayı o süre içinde indir; saklamak istiyorsan kendi diskinde saklamak senin işin.
- Dosya bir mesaj değildir: kanalda görünmez, kişi de başkası da onu mesaj listesinde bulamaz. Bağlantıyı yalnızca sen alırsın.
- Yanıtında dosyayı göstermek istiyorsan indirip yanıtla birlikte yeniden yükle; bu bağlantıyı mesaja koyma, 15 dakikada ölür.
- Kişi dosyayı kendi yükleme sınırlarıyla yükler: boyut sınırı, saatlik ve günlük yükleme bütçesi ve kanalda `AttachFiles` izni. Bunlar tutmazsa çalıştırma sana hiç gelmez.
- Bir seviyede en çok 3 dosya seçeneği olur. Zorunlu dosya seçeneği boş kalırsa çalıştırma reddedilir.

Limitler

Anahtar	Değer	Anlamı
`chat_commands`	100	Kapsam başına eğik çizgi komutu
`menu_commands`	15	Kapsam başına her menü türünden (kullanıcı ve mesaj ayrı sayılır)
`options`	25	Seviye başına seçenek
`choices`	25	Seçenek başına seçim
`attachment_options`	3	Seviye başına dosya seçeneği
`permission_rules`	100	Komut başına izin kuralı
`name`	32	Ad, karakter
`description`	100	Açıklama, karakter
`choice_name`	100	Seçim adı ve metin değeri, karakter
`string_value`	6000	Kişinin metin seçeneğine yazabileceği, karakter
`autocomplete_value`	100	Otomatik tamamlamada yazılan metin, karakter
`bytes`	8192	Bir komutun JSON boyutu
`creates`	200	Uygulama ve kapsam başına günde yeni komut

Yazma rotaları (`POST` , `PUT` , `PATCH` , `DELETE`) `command` sınıfındadır: 20 saniyede 5 istek. Okuma `read` sınıfındadır. Aynı sınırlar `openapi.json` içinde `x-command-limits` olarak durur.

Tam örnek: komutları kaydeden betik

Node.js 22 veya üstü yeter, ek paket gerekmez. Betik komut listesini `PUT` ile gönderir ve kaydedilenleri yazar. `GUILD_ID` verirsen o sunucuya, vermezsen global kaydeder. Botun her açılışında çalıştırabilirsin:

```
const API = "https://noluyo.net/api/v1";
const { BOT_TOKEN, APPLICATION_ID, GUILD_ID } = process.env;
if (!BOT_TOKEN || !APPLICATION_ID) throw new Error("BOT_TOKEN ve APPLICATION_ID tanımlı olmalı");

const commands = [
  {
    name: "selam",
    name_localizations: { en: "hello" },
    description: "Birini selamlar",
    description_localizations: { en: "Greets someone" },
    options: [
      { type: 6, name: "kisi", name_localizations: { en: "person" }, description: "Selamlanacak
kişi", required: true },
    ],
  },
  { type: 2, name: "Profili aç" },
];

const path = GUILD_ID
  ? `/applications/${APPLICATION_ID}/guilds/${GUILD_ID}/commands`
  : `/applications/${APPLICATION_ID}/commands`;

for (;;) {
  const res = await fetch(API + path, {
    method: "PUT",
    headers: { Authorization: `Bot ${BOT_TOKEN}`, "Content-Type": "application/json" },
    body: JSON.stringify(commands),
  });
  const data = await res.json();
  if (res.status === 429) {
    await new Promise((done) => setTimeout(done, data.retry_after * 1000));
    continue;
  }
  if (!res.ok) throw new Error(`${res.status} ${data.code}: ${JSON.stringify(data.errors ??
data.message)}`);
  for (const c of data) console.log(`${c.name} kaydedildi (${c.id})`);
  break;
}
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' APPLICATION_ID='uygulama id degeri' GUILD_ID='deneme sunucusu id
degeri' node komutlar.mjs
```

Tam örnek: /zar botu

Node.js 22 veya üstü yeter. Gateway sayfasındaki `gateway.mjs` ile Örnekler sayfasındaki `api.mjs` aynı klasörde olmalı. Bot açılınca `/zar` komutunu global olarak kaydeder. `/zar yuz:20` 1 ile 20 arasında bir sayı atar; `gizli:true` verilirse sonucu yalnızca çalıştıran kişi görür:

```
import { randomInt } from "node:crypto";
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const commands = [{
  name: "zar",
  name_localizations: { en: "dice" },
  description: "Zar atar",
  description_localizations: { en: "Rolls a die" },
  options: [
    { type: 4, name: "yuz", name_localizations: { en: "sides" }, description: "Kaç yüzlü, yoksa
6",
      description_localizations: { en: "How many sides, 6 if left out" }, min_value: 2,
max_value: 100 },
    { type: 5, name: "gizli", name_localizations: { en: "secret" }, description: "Sonucu yalnızca
sen gör",
      description_localizations: { en: "Only you see the result" } },
  ],
}];

// Seçenekler tanımdaki adlarıyla gelir: kişi İngilizce kullansa da "yuz", "sides" değil.
const option = (d, name) => d.data.options?.find((o) => o.name === name)?.value;

async function onCommand(d) {
  if (d.type !== 2 || d.data.name !== "zar") return;
  const sides = option(d, "yuz") ?? 6;
  const roll = randomInt(1, sides + 1);
  const content = d.locale === "en" ? `You rolled ${roll} (1-${sides}).` : `${roll} geldi
(1-${sides}).`;
  await call("POST", `/interactions/${d.id}/${d.token}/callback`, {
    type: 4,
    data: { content, flags: option(d, "gizli") ? 64 : 0 },
  });
}

let registered = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !registered) {
      registered = true;
      call("PUT", `/applications/${d.application.id}/commands`, commands)
        .then(() => console.log("/zar kaydedildi"))
        .catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onCommand(d).catch(console.error);
```

```
  },
});
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' node zar-botu.mjs
```

Komut kişilerin listesine en geç 60 saniyede gelir. Sunucu `yuz`'un 2 ile 100 arasında olduğunu zaten denetledi; botun yalnızca seçeneğin hiç gönderilmediği durumu düşünür.

Sık hatalar

Kod	Ne oldu	Ne yap
`validation_failed`, `name`: `invalid`	Adda büyük harf, boşluk ya da izin verilmeyen işaret var, ya da başka bir adla çakışıyor	Küçük harf kullan: `/İade` değil `/iade`. `errors` hangi alanı söyler.
`validation_failed`, `name`: `taken`	`PATCH` ile verdiğin ad aynı türden başka bir komutta	Başka bir ad seç ya da önce diğerini sil.
`validation_failed`, `type`: `invalid`	`PATCH` türü değiştirmeye çalıştı	Komutu silip yeni türle oluştur.
`validation_failed`, `""`: `too_long`	Komutun JSON'u 8192 baytı aşıyor	Seçenekleri ya da açıklamaları kısalt.
`max_commands`	Kapsam bu türden komut sınırında	Kullanmadığın komutu sil ya da sunucu komutu kullan.
`missing_access`	Bot o sunucunun üyesi değil	Botu sunucuya ekle ya da global rotayı kullan.
`unknown_command`	Komut yok ya da başka kapsamda	Sunucu komutunu sunucu rotasıyla, global komutu global rotayla çağır.
`unknown_application`	Yoldaki uygulama senin değil	Kendi uygulamanın ID'sini kullan.
`rate_limited`	20 saniyede 5 yazmayı ya da günde 200 yeni komutu aştın	`retry_after` kadar bekle. Her açılışta komutları silip yeniden oluşturma, `PUT` gönder.
`interaction_response_invalid`	Komut çalıştırmaya `type: 6` ya da `type: 7` ile yanıt verdin	`type: 4` kullan, ya da `type: 5` gönderip sonra `PATCH @original`.
`link_invalid`	Dosya seçeneğinin bağlantısı 15 dakikayı geçti	Dosyayı etkileşim gelir gelmez indir.

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur. Uygulama ID'si de `process.env.APPLICATION_ID`'den gelir.
- Komutlar bot açılırken tek `PUT` ile kaydedilir, tek tek `POST` ile değil. Komutları her açılışta silip yeniden oluşturan kod yazılmaz.
- Eğik çizgi komutu ve seçenek adları küçük harftir: harf, rakam, `-`, `_`, en çok 32 karakter. `İ` gibi büyük Türkçe harf reddedilir.
- Yerelleştirme yalnızca `tr` ve `en` alır.
- Zorunlu seçenekler önce gelir; alt komut ve düz seçenek aynı seviyede karışmaz.
- `default_member_permissions` ondalık metindir ve `BigInt` ile hesaplanır; güvenlik değildir, bot yetkiyi kendisi denetler.
- DM'de izin biti yoktur: yetki isteyen komut `contexts: [0]` ile yalnızca sunucuya konur. DM'den gelen etkileşimde `member` ve `guild_id` yoktur, kişi `user` içindedir.
- Değişiklik kişilere en geç 60 saniyede ulaşır.
- Komut çalıştırılınca gelen etkileşime 3 saniye içinde yanıt verilir.
- Çalıştırma `INTERACTION_CREATE` olayında `type: 2` ile gelir; komut `data.name` ile, seçenekler `data.options` içindeki tanımdaki adlarıyla bulunur, çevirileriyle değil.
- Komut çalıştırmaya yalnızca `type: 4` ya da `type: 5` ile yanıt verilir; `type: 6` ve `type: 7` reddedilir.
- Kullanıcı, rol, kanal ve mesajın bilgisi `data.resolved` içinden kimliğiyle okunur; ayrıca istek atılmaz.
- Menüyle gelen kişi ve mesaj bilgisi yalnızca o işte kullanılır, saklanmaz.
- Dosya seçeneğinin dosyası `data.resolved.attachments[value].url` bağlantısından 15 dakika içinde indirilir; yanıtta gösterilecekse yeniden yüklenir, bağlantı mesaja konmaz.

Sırada ne var

- Komut izinleri: sunucu yöneticilerinin komut erişimini özelleştirmesi
- Otomatik tamamlama: kullanıcı yazarken dinamik seçenek önerileri sunmak
- Formlar: komuta yanıt olarak metin girişli pencere açmak

Komut izinleri

Komut izinleri, botunun komutlarını bir sunucuda kimin görüp çalıştırabileceğini ve hangi kanallarda kullanılabileceğini söyleyen kurallardır: kuralları sunucunun yöneticileri koyar, botun okur.

Kimin işi

Kişi	Ne yapar
Sunucu yöneticisi	Kuralları koyar, değiştirir ve kaldırır. `ManageGuild` ve `ManageRoles` izinlerinin ikisi de gerekir.
Geliştirici	`default_member_permissions` ile komutun varsayılanını seçer, kuralları okur ve değişince olay alır. Kural yazamaz.

Kuralı sunucu uygular: kişinin kullanamayacağı komut ona listede gösterilmez ve çalıştırılırsa botuna hiç ulaşmaz.

Sunucu yöneticisi için

Nereden açılır

1. Sunucunun adına tıkla ve **Sunucu ayarları**'nı aç.
2. Soldaki **Bot komutları** sekmesine gir. Sekme yalnız `ManageGuild` ile `ManageRoles` izinlerinin ikisi birden sende varsa görünür.
3. **Bot** kutusundan bir bot seç. Altındaki **Komut izinleri** kartında önce **Tüm komutlar** satırı gelir: botun bütün komutlarına uyan kurallar. Altında her komut kendi satırındadır; satırda komutun açıklaması, kural yokken kimin kullanabildiği (`default_member_permissions`) ve bugünkü kuralların özeti yazar.
4. Satırdaki **Düzenle**'ye bas. **Rol**, **Üye** ya da **Kanal** seç, hedefi seç (aramak için yazabilirsin), **Kural ekle**'ye bas. Yeni kural **İzin ver** olarak eklenir; yanındaki kutudan **Engelle**'ye çevirebilirsin. **Varsayılan** ya da çöp kutusu kuralı kaldırır: o hedef için komutun varsayılanı geçerli olur.
5. **Kaydet**'e bas. Kaydetmeden önce kuralların özeti çıkar ("İzin ver: @Moderatör · Engelle: @herkes, #genel" gibi); onaylayınca kaydedilir. Kişilerin uygulamaları değişikliği en geç 60 saniyede görür: komut listelerinden kaybolur ve çalıştırılamaz.

Listeyi okuyan rota `ManageGuild` ister, kural değiştirmek `ManageGuild` ile `ManageRoles`'ü birlikte ister; sekme ikisi birden olmadan görünmez. Komutun istediği bir izin sende yoksa kaydederken "Bu izin sende olmadığı için bu komutu açamazsın." yazar (aşağıda **Sende olmayan yetkiyi açamazsın**). Her değişiklik denetim kaydına "Komut izni güncelleme" olarak düşer: kimin, hangi komutta, eski ve yeni kuralları.

Kural türleri

`type`	Hedef	`id`
`1`	Rol	Sunucunun bir rolünün ID'si; sunucunun kendi ID'si `@everyone` demektir
`2`	Üye	Sunucunun bir üyesinin kullanıcı ID'si
`3`	Kanal	Sunucunun bir kanalının ID'si; `0` tüm kanallar demektir

Bir komutta ve **Tüm komutlar** satırında ayrı ayrı en çok 100 kural olur. Aynı hedef bir listede iki kez geçemez. Başka sunucunun rolü ya da kanalı, üye olmayan bir kişi `400 validation_failed` döner.

Hangi kural kazanır

Sunucu, kişi bir komutu açınca ve çalıştırınca bu sırayla bakar:

1. Sunucunun sahibi ve `Administrator` izni olanlar her komutu her kanalda kullanır. Hiçbir kural onları durdurmaz.
2. Kişinin o kanalda `UseApplicationCommands` izni yoksa ya da zaman aşımındaysa orada hiçbir komutu kullanamaz.
3. Kanal kuralı: önce komutun kendi kuralları, orada kanal kuralı yoksa **Tüm komutlar** kuralları. İçlerinde önce o kanalın kuralı, yoksa "tüm kanallar" (`0`). Bulunan kural engelse komut o kanalda kullanılamaz. Hiç kanal kuralı yoksa her kanal açıktır.
4. Kişi kuralı: önce komutun kendi kuralları, sonra **Tüm komutlar** kuralları. Her birinde önce kişinin kendi kuralı, sonra rolleri (rollerinden biri izin veriyorsa izin, hepsi engelliyorsa engel), sonra `@everyone` . İlk bulunan karar verir. Bir istisna: **Tüm komutlar**'daki bir izin, `default_member_permissions` değeri olan komutu açmaz; o komutta 5. adıma geçilir. **Tüm komutlar**'daki engel ise her komutta geçerlidir.
5. Hiç kişi kuralı yoksa komutun `default_member_permissions` değeri karar verir.

Komutun kendi kuralı **Tüm komutlar** kuralını ezer. Örneğin **Tüm komutlar**'da `@everyone` engelli ama `/yardım` 'da `@everyone` izinliyse, herkes `/yardım` 'ı kullanır, botun diğer komutlarını kullanamaz.

Dört örnek

İstek	Nereye	Kurallar
Herkes kullansın ama #genel'de olmasın	Komuta ya da **Tüm komutlar**'a	Kanal `#genel` : Engelle
Yalnız Moderatör	Komuta	`@everyone` : Engelle; rol Moderatör: İzin ver
Bir kişi hariç	Komuta	O üye: Engelle
Yalnız yönetici	Komuta	`@everyone` : Engelle. Sahip ve `Administrator` zaten geçer

"Bir kişi hariç" kişi kuralı olduğu için rolden önce gelir: o kişinin Moderatör rolü olsa bile engellenir. "Yalnız Moderatör"de Moderatör olmayan herkes `@everyone` kuralına düşer.

Komutları bir kanalda tamamen kapatmak

`UseApplicationCommands` artık denetlenir. Kanalın izin ayarlarında bir rolden bu izni kaldırırsan o roldeki kişiler o kanalda hiçbir botun hiçbir komutunu görmez ve çalıştıramaz. Bütün sunucuda kapatmak için **Roller**'de `@everyone` rolünden kaldır. Yeni sunucularda `@everyone` bu izinle başlar.

Sende olmayan yetkiyi açamazsın

Bir komutun `default_member_permissions` değeri varsa, örneğin `BanMembers`, o komutun kurallarını değiştirmek için bu izinlere senin de sahip olman gerekir; yoksa `403 command_permission_escalation` alırsın. Değeri `"0"` olan komutun kurallarını yalnızca sahip ve `Administrator` değiştirir. **Tüm komutlar** satırında bu, botun bütün komutlarının izinleri demektir. Sebep: aksi halde yalnızca `ManageGuild` izni olan biri `/ban`'ı herkese açabilirdi. Aynı sebeple **Tüm komutlar** kuralı yalnızca varsayılan izni olmayan komutları açar; bot sonradan `BanMembers` isteyen bir komut eklese de o komut kendiliğinden açılmaz. Yetki isteyen bir komutu bir role açmak için kuralı o komutun kendisine koy.

Geliştirici için

Varsayılan: default_member_permissions

Sunucu bir kural koyana dek komutu kimin kullanacağını Komutlar sayfasındaki `default_member_permissions` söyler. Bir kişi kuralı varsayılanın önüne geçer: yönetici `/ban`'ı bir role açabilir, herkese açık bir komutu bir role kapatabilir. Varsayılanı yetki isteyen komutlar için doğru seç; ama tek güvencen o olmasın.

Bot neden kural yazamaz

Kim neyi kullanır, sunucunun kararıdır. Bot kendi komutunu herkese açabilseydi sunucu bu kararı kaybederdi. Bu yüzden kural yazma rotası yalnızca kişinin oturumuyla çalışır; bot token'ıyla `403 bot_forbidden` döner. Botun kuralları okur ve değişince olay alır.

Kuralları okumak

```
curl
"https://noluyo.net/api/v1/applications/$APPLICATION_ID/guilds/$GUILD_ID/commands/permissions" \
  -H "Authorization: Bot $BOT_TOKEN"
```

Yanıt, kuralı olan her komut için bir kayıttır. `id` uygulamanın ID'siyse kayıt **Tüm komutlar** kurallarıdır:

```
[
  {
    "id": "111540000000000000",
    "application_id": "111540000000000000",
    "guild_id": "111500000000000000",
    "permissions": [{ "type": 1, "id": "111500000000000000", "allow": false }]
  },
  {
    "id": "111560000000000000",
    "application_id": "111540000000000000",
    "guild_id": "111500000000000000",
    "permissions": [
      { "type": 1, "id": "111520000000000000", "allow": true },
      { "type": 3, "id": "0", "allow": false }
    ]
  }
]
```

Rota	Ne döner
`GET .../guilds/{guild_id}/commands/permissions`	Botun o sunucudaki bütün kuralları; kuralı olmayan komut listede yok
`GET .../guilds/{guild_id}/commands/{command_id}/permissions`	Bir komutun kuralları, yoksa boş liste; **Tüm komutlar** kuralları burada yok

İkisi de `read` sınıfındadır. Bot o sunucunun üyesi değilse `403 missing_access` döner.

Alanlar

Alan	Tür	Anlamı
`id`	ID	Komut; **Tüm komutlar** kurallarında uygulamanın ID'si
`application_id`	ID	Uygulaman
`guild_id`	ID	Sunucu
`permissions[].type`	tam sayı	`1` rol, `2` üye, `3` kanal
`permissions[].id`	ID	Hedef; rol olarak sunucunun ID'si `@everyone`, kanal olarak `0` tüm kanallar
`permissions[].allow`	evet/hayır	`true` izin verir, `false` engeller

Değişince: APPLICATION_COMMAND_PERMISSIONS_UPDATE

Bir yönetici kuralları değiştirince botuna APPLICATION_COMMAND_PERMISSIONS_UPDATE gelir. Gövdesi yukarıdaki kayıtlardan biridir ve kümenin tamamını taşır; boş `permissions` kuralların kaldırıldığını söyler. Intent

istemez, yalnızca botuna gider. Kuralları sunucu zaten uygular; olayı yalnızca bilgi için kullan, örneğin bir kayıt kanalına yazmak için.

Tam örnek: kuralları okuyan betik

Node.js 22 veya üstü yeter, ek paket gerekmez. Betik bir sunucudaki kuralları okur ve okunur biçimde yazar:

```
const API = "https://noluyo.net/api/v1";
const { BOT_TOKEN, APPLICATION_ID, GUILD_ID } = process.env;
if (!BOT_TOKEN || !APPLICATION_ID || !GUILD_ID) throw new Error("BOT_TOKEN, APPLICATION_ID ve
GUILD_ID tanımlı olmalı");

const headers = { Authorization: `Bot ${BOT_TOKEN}` };
const get = async (path) => {
  for (;;) {
    const res = await fetch(API + path, { headers });
    const data = await res.json();
    if (res.status === 429) {
      await new Promise((done) => setTimeout(done, data.retry_after * 1000));
      continue;
    }
    if (!res.ok) throw new Error(`${res.status} ${data.code}: ${data.message}`);
    return data;
  }
};

const names = new Map();
for (const c of await get(`/applications/${APPLICATION_ID}/commands`)) names.set(c.id, c.name);
for (const c of await get(`/applications/${APPLICATION_ID}/guilds/${GUILD_ID}/commands`))
names.set(c.id, c.name);

const kinds = { 1: "rol", 2: "üye", 3: "kanal" };
for (const entry of await
get(`/applications/${APPLICATION_ID}/guilds/${GUILD_ID}/commands/permissions`)) {
  const title = entry.id === APPLICATION_ID ? "Tüm komutlar" : `/${names.get(entry.id) ??
entry.id}`;
  console.log(title);
  for (const p of entry.permissions) {
    let target = `${kinds[p.type]} ${p.id}`;
    if (p.type === 1 && p.id === GUILD_ID) target = "@everyone";
    if (p.type === 3 && p.id === "0") target = "tüm kanallar";
    console.log(`  ${target}: ${p.allow ? "izin" : "engel"}`);
  }
}
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' APPLICATION_ID='uygulama id degeri' GUILD_ID='sunucu id degeri'
node kurallar.mjs
```

> **Uyarı:** İzin, komutu görmeyi ve çalıştırmayı sınırlar; işin yetkisini yine sen denetle. Bir yönetici `/ban`'ı herkese açabilir. Komut gelince etkileşimdeki `member.permissions`'a bak ve işi kişinin yetkisi varsa yap: `BanMembers` yoksa yasaklama.

Limitler

Konu	Değer
Kural	Komut başına 100; **Tüm komutlar** için ayrıca 100
Kural yazma	Kişi ve sunucu başına dakikada 10
Kural okuma	`read` sınıfı: 5 saniyede 30 istek
Yayılma	Kişilerin uygulamaları en geç 60 saniyede

Hatalar

Kod	Ne oldu	Ne yap
`missing_permissions`	Kural değiştiren kişide `ManageGuild` ve `ManageRoles` birlikte yok	Kişiye iki izni de veren bir rol ver.
`command_permission_escalation`	Kişi komutun `default_member_permissions` bitlerine sahip değil	Kuralı o izinlere sahip biri, sahip ya da bir `Administrator` değiştirsin.
`validation_failed`, `permissions[0].id: invalid`	Hedef bu sunucunun değil, iki kez geçiyor ya da ID geçersiz	Bu sunucunun rolünü, kanalını ya da üyesini seç; tüm kanallar için `0`.
`validation_failed`, `permissions: too_long`	100'den çok kural	Kişi kuralları yerine rol kullan.
`unknown_application`	Bot bu sunucuda değil ya da kapatılmış	Botu sunucuya ekle.
`unknown_command`	Komut yok ya da başka sunucunun komutu	Global komutu ya da bu sunucunun komutunu seç.
`bot_forbidden`	Bot kural yazmaya çalıştı	Bot yazamaz; sunucunun yöneticisinden iste.
`missing_access`	Okuyan bot o sunucunun üyesi değil	Botu sunucuya ekle.
`rate_limited`	Dakikada 10 değişikliği aştın	`retry_after` kadar bekle.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:
>
> - Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur.
> - Bot komut kuralı yazmaz; yalnızca okur. Kural yazan kod bot token'ıyla `bot_forbidden` alır.
> - Kurallar sunucuda uygulanır; bot onları yeniden uygulamaz ama yetki isteyen işte `member.permissions`'ı denetler.
> - Kanal kuralında `0` tüm kanallar, rol kuralında sunucunun ID'si `@everyone` demektir. ID'ler metindir.
> - `id` uygulamanın ID'si olan kayıt botun bütün komutlarının kurallarıdır.
> - Komut çalıştırılınca gelen etkileşime 3 saniye içinde yanıt verilir.

Sırada ne var

- Komutlar: `default_member_permissions` alanı ve komut tanımları
- İzinler: izin bitlerinin tam listesi ve rol hiyerarşisi

- Hatalar: `command_forbidden` ve `command_permission_escalation` yanıtları

Otomatik tamamlama

Otomatik tamamlama, kişi bir komut seçeneğine yazarken botunun ona öneriler sunmasıdır: sunucu yazılanı botuna gönderir, botun en çok 25 öneriyle yanıt verir ve kişi listeden seçer ya da kendi yazdığını gönderir.

> **Uyarı:** Kişinin bitmemiş yazısı, Enter'a basmadan sana gelir. Onu yalnızca öneri bulmak için kullan: saklama, loglama, başka bir yere gönderme. Kişi yazarken fikrini değiştirebilir; yazdığı her harf bir karar değildir.

Akış

Adım	Ne olur
1	Seçeneği `autocomplete: true` ile tanımlarsın. Yalnızca metin (`3`), tam sayı (`4`) ve sayı (`10`) seçeneklerinde olur, `choices` ile birlikte olmaz.
2	Kişi o seçeneğe yazar. Uygulaması yazma durduktan 300 ms sonra yazılanı sunucuya gönderir; kişi yazdıkça sana 300 ms'de bir yeni istek gelebilir.
3	Sana `INTERACTION_CREATE` gelir, `type: 4`. `data.options` içinde `focused: true` olan seçenek kişinin yazdığı yerdir.
4	3 saniye içinde `callback` ile `type: 8` ve `data.choices` gönderirsin. Her otomatik tamamlama tek yanıt alır.
5	Öneriler yalnızca yazan kişiye gider. Kişi birini seçer ya da kendi yazdığıyla devam eder. Enter'a basınca komut sana her zamanki gibi `type: 2` olarak gelir.

Geliştirici `autocomplete` açmadıkça kişinin yazdığı Enter'dan önce hiçbir bota gitmez.

Kişi ne görür

> **Not:** Bu bölüm webde gözlenen davranışı anlatır. Masaüstü uygulaması aynı web kodunu kullanır ama ayrıca çalıştırılmadı. Android uygulamasında henüz gözlenmedi.

Kişi alana yazar; yazma 300 ms durunca önerilerin alanın üstünde bir listede açılır, her öneri yerelleştirilmiş adıyla (`name_localized`) görünür. `↑` ve `↓` öneri değiştirir, `Enter` ya da `Tab` seçer, `Esc` listeyi kapatır, tıklamak da seçer. Seçilen önerinin adı alanda görünür, sana çalıştırmada `value`'su gelir. Kişi sonra yeniden yazarsa değer onun yazdığına döner.

Kişi yazdıkça eski yazıya ait geç gelen yanıt listeyi değiştirmez; yalnızca son isteğin yanıtı gösterilir. Botun 1 saniye geç yanıt verse de öneriler gelir. Botun hiç yanıt vermezse ya da çevrimdışıysa liste açılmaz, kişiye hata gösterilmez ve alan serbest yazıya açık kalır: kişi Enter'a basınca komut yazdığı metinle gelir. Bu yüzden çalıştırmada değeri her zaman yeniden doğrula.

Odaktaki alanla birlikte yalnızca geçerli değeri olan diğer seçenekler gelir; yarım yazılmış sayı ya da dosya gelmez.

En küçük örnek

Seçeneği tanımla:

```
{
  "name": "hava",
  "description": "Bir şehrin havası",
  "options": [
    { "type": 3, "name": "sehir", "description": "Şehir", "required": true, "autocomplete": true
}
  ]
}
```

Kişi `ist` yazınca sana şu gelir (kısaltıldı):

```
{
  "id": "1118042062848001 20",
  "type": 4,
  "token": "eyJpZCI6IjExMTgwNDIwNjI4NDgwMDEyMCJ9.bWFj",
  "locale": "tr",
  "data": {
    "id": "1115600000000000000",
    "name": "hava",
    "type": 1,
    "options": [{ "name": "sehir", "type": 3, "value": "ist", "focused": true }]
  }
}
```

Önerilerle yanıt ver:

```
curl -X POST "https://noluyo.net/api/v1/interactions/$INTERACTION_ID/$INTERACTION_TOKEN/callback"
\
  -H "Authorization: Bot $BOT_TOKEN" -H 'Content-Type: application/json' \
  -d '{"type":8,"data":{"choices":[{"name":"İstanbul","name_localizations":
{"en":"Istanbul"},"value":"istanbul"}]}}'
```

Yanıt `204` ve gövdesizdir.

Gelen etkileşim

Alan	Anlamı
`type`	`4` : otomatik tamamlama
`data.id` , `data.name`	Komutun kimliği ve tanımdaki adı
`data.options`	Şimdiye dek yazılanlar, tanımına göre denetlenmiş. Alt komut varsa içindeki `options` 'tadır.
`data.options[].focused`	Yalnızca kişinin yazdığı tek seçenekte `true`
`data.options[].value`	Odaktaki seçenekte bitmemiş yazı: en çok 100 karakter metin, tam sayı ve sayı seçeneğinde sayı ya da `"1."` gibi metin
`data.resolved`	Kişinin diğer seçeneklerde seçtiği kullanıcı, rol, kanal
`member` , `locale`	Yazan kişi ve dili; DM'de `member` yerine `user`
`context`	`0` sunucu, `1` botla DM. DM'de de aynı çalışır; tek fark kişinin `user` 'da olmasıdır.
`token`	Yalnızca `callback` için. Takip mesajı ve düzenleme almaz.

Zorunlu seçenekler henüz eksik olabilir. Dosya seçeneği otomatik tamamlamada gönderilmez; dosya ancak çalıştırmada gelir.

Yanıt: type 8

Alan	Kural
`choices`	Zorunlu, en çok 25 öğe. Boş liste "eşleşen yok" demektir.
`choices[].name`	Kişinin gördüğü, en çok 100 karakter
`choices[].name_localizations`	`tr` ve `en` karşılığı; kişi kendi dilindekini görür
`choices[].value`	Odaktaki seçeneğin türünde; metin en çok 100 karakter. Seçilince bu gider, ad değil.

`type: 8` yalnızca otomatik tamamlamaya, otomatik tamamlama yalnızca `type: 8` 'e uyar. Öneriler kanala yazılmaz, mesaj değildir; bu yüzden tüm botların saniyede 70 yazmalık ortak tavanına sayılmaz. Yine `interaction` sınıfına ve aynı anda 8 bot isteği sınırına girer.

Değeri çalıştırırken yeniden doğrula

Kişi listeden seçmek zorunda değildir: kendi yazdığını gönderebilir, öneri gelmeden Enter'a basabilir ya da başka bir istemci kullanabilir. Sunucu çalıştırmadaki değeri önerilerine karşı **denetlemez**. `type: 2` gelince değeri kendi

kaynağında yeniden ara; bulamazsan kişiye `flags: 64` ile ne yapacağını söyle. Önerdiğin bir değerin hâlâ geçerli olduğunu da varsayma: liste o arada değişmiş olabilir.

Hızlı kal

- 3 saniye üst sınırdır; kişi o sırada yazmaya devam ediyor. 300 ms içinde yanıt vermeyi hedefle.
- Uygulama yalnızca son yazının yanıtını gösterir. Eski bir yazıya geç gelen yanıt atılır; yanıt vermezsen alan serbest yazıma açık kalır.
- Listeni bellekte tut, her istekte veritabanına ya da dış bir servise gitme. Dış kaynağa gitmen gerekiyorsa sonucu kısa süre önbellekle ve aynı yazı için kopya sorgu yapma.
- Hata gelince yeniden deneme: 3 saniye geçtiyse yanıt artık işe yaramaz. Sıradaki yazı zaten gelecek.

Yerelleştirme

`locale` kişinin dilidir. Öneriyi `name` ile Türkçe yaz, `name_localizations.en` ile İngilizcesini ver; kişi kendi dilindekini görür. `value` dilden bağımsız kalsın ki çalıştırmada aynı değeri tanıyasın. Türkçe harfleri karşılaştırırken `toLocaleLowerCase("tr")` kullan: `"İST"` küçültülünce `"ist"` olur.

Tam örnek: şehir arama botu

Node.js 22 veya üstü yeter. Gateway sayfasındaki `gateway.mjs` ile Örnekler sayfasındaki `api.mjs` aynı klasörde olmalı. Bot açılınca `/hava` komutunu kaydeder; kişi şehir yazdıkça eşleşen şehirleri önerir, çalıştırılınca şehri listede yeniden arar:

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const cities = [
  { value: "istanbul", tr: "İstanbul", en: "Istanbul" },
  { value: "izmir", tr: "İzmir", en: "Izmir" },
  { value: "ankara", tr: "Ankara", en: "Ankara" },
  { value: "bursa", tr: "Bursa", en: "Bursa" },
  { value: "antalya", tr: "Antalya", en: "Antalya" },
  { value: "eskisehir", tr: "Eskişehir", en: "Eskisehir" },
];

const commands = [{
  name: "hava",
  name_localizations: { en: "weather" },
  description: "Bir şehrin havası",
  description_localizations: { en: "The weather of a city" },
  options: [{ type: 3, name: "sehir", name_localizations: { en: "city" }, description: "Şehir",
    description_localizations: { en: "City" }, required: true, autocomplete: true }],
}];

const fold = (s) => s.toLocaleLowerCase("tr");
// Aynı yazı ikinci kez gelince liste yeniden taranmaz.
const cache = new Map();
function search(typed) {
  const key = fold(String(typed)).trim();
  if (!cache.has(key)) {
    if (cache.size > 1000) cache.clear();
    cache.set(key, cities.filter((c) => fold(c.tr).startsWith(key) ||
c.en.toLowerCase().startsWith(key)).slice(0, 25));
  }
  return cache.get(key);
}

// Odaktaki seçenek alt komutun içinde de olabilir.
const focused = (options = []) => options.find((o) => o.focused) ?? options.map((o) =>
focused(o.options)).find(Boolean);

async function onInteraction(d) {
  if (d.data.name !== "hava") return;
  const path = `/interactions/${d.id}/${d.token}/callback`;
  if (d.type === 4) {
    const choices = search(focused(d.data.options)?.value ?? "").map((c) => ({
      name: c.tr, name_localizations: { en: c.en }, value: c.value,
    }));
    await call("POST", path, { type: 8, data: { choices } });
    return;
  }
```

```
  if (d.type !== 2) return;
  // Kişi listeden seçmek zorunda değil: değer burada yeniden doğrulanır.
  const value = d.data.options?.find((o) => o.name === "sehir")?.value;
  const city = cities.find((c) => c.value === value) ?? search(value)[0];
  const en = d.locale === "en";
  if (!city) {
    await call("POST", path, { type: 4, data: { flags: 64,
      content: en ? "I do not know that city. Pick one from the list." : "Bu şehri bilmiyorum.
Listeden birini seç." } });
    return;
  }
  await call("POST", path, { type: 4, data: { content: en ? `Weather in ${city.en}: sunny.` :
`${city.tr} için hava: güneşli.` } });
}

let registered = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !registered) {
      registered = true;
      call("PUT", `/applications/${d.application.id}/commands`, commands)
        .then(() => console.log("/hava kaydedildi"))
        .catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onInteraction(d).catch((err) => console.error(err.message));
  },
});
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' node sehir-botu.mjs
```

Örnek kişinin yazdığını konsola yazmaz; hata mesajında da yalnızca kod ve açıklama vardır.

Limitler

Anahtar	Değer	Saniye	Anlamı
`user`	5	1	Kişi başına otomatik tamamlama isteği
`app`	20	1	Uygulaman başına, tüm kişilerden
`shared`	200	1	Tüm botlar birlikte
`choices`	25		Yanıt başına öneri
`autocomplete_value`	100		Odaktaki yazı, karakter

İlk üçü kişinin isteğini sınırlar: aşılırsa sana bir şey gelmez, kişi `429` alır ve uygulaması bir sonraki yazıda yeniden dener. İnsanların mesajlaşması bundan etkilenmez.

Öneri yanıtların gerçek yanıtlarınla aynı bütçeyi paylaşır: `callback` rotası `interaction` sınıfındadır (saniyede 50) ve botunun genel sınırına (saniyede 50) sayılır. Bu yüzden uygulama sınırı 20'dedir; öneriler bütçenin tamamını yemez, komut ve basma yanıtlarına yer kalır. Öneri yanıtlarını hızlı ve tek tut ki bütçe gerçek yanıtlara kalsın.

Hatalar

Kod	Ne oldu	Ne yap
`interaction_response_invalid`	Otomatik tamamlamaya `type: 8` dışında yanıt verdin, `type: 8`'de `choices` yok ya da yanında `content`, `embeds`, `flags` gibi mesaj alanı var, başka bir etkileşime `type: 8` gönderdin ya da bu token'la takip mesajı veya düzenleme denedin	Otomatik tamamlamaya yalnızca `{"type":8,"data":{"choices":[...]}}` gönder.
`validation_failed`, `choices`: `too_long`	25'ten çok öneri	İlk 25'i gönder.
`validation_failed`, `choices[0].value`: `invalid`	Değer seçeneğin türünde değil, örneğin metin seçeneğine sayı	Seçeneğin türünde değer gönder.
`validation_failed`, `choices[0].name`: `too_long`	Ad 100 karakteri aşıyor	Adı kısalt.
`interaction_already_acknowledged`	Aynı yazıya ikinci kez yanıt verdin	Her otomatik tamamlamaya bir kez yanıt ver.
`interaction_timeout`	3 saniye geçti	Listeyi bellekte tut, önbellekle. Yeniden deneme.
`rate_limited`, `X-RateLimit-Scope: shared`	Aynı anda 8 bot isteği sınırı doldu	Bu yazıyı bırak; sıradaki gelecek.
`rate_limited`, `X-RateLimit-Scope: bot` ya da `global`	Botunun `interaction` sınıfı ya da genel sınırı doldu; öneriler ve gerçek yanıtlar aynı bütçeden harcar	Öneriyi bırak, bekleme. Gerçek yanıtlar için `retry_after` kadar bekle.

Kişinin alabileceği hatalar (`validation_failed`, `command_forbidden`, `command_outdated`, `rate_limited`, `interaction_busy`) sana gelmez; Hatalar sayfasında.

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur.
- Otomatik tamamlama `INTERACTION_CREATE` olayında `type: 4` ile gelir ve 3 saniye içinde tek bir `{"type":8,"data":{"choices":[...]}}` ile yanıtlanır.
- Yazılan, `data.options` içinde `focused: true` olan seçeneğin `value`'sudur; alt komutun `options`'ında da olabilir. Zorunlu seçenekler eksik olabilir.
- En çok 25 öneri; `value` seçeneğin türünde, metin en çok 100 karakter. Boş liste geçerlidir.
- Kişinin yazdığı saklanmaz ve loglanmaz.
- Liste bellekte tutulur ya da önbelleklenir; her tuşta veritabanına veya dış servise gidilmez, aynı yazı için kopya sorgu yapılmaz. Hata gelince yeniden denenmez.
- Komut çalıştırılınca (`type: 2`) değer yeniden doğrulanır: kişi listeden seçmek zorunda değildir.
- Otomatik tamamlamanın token'ı takip mesajı ya da düzenleme için kullanılmaz.

Sırada ne var

- Komutlar: seçenek türleri, alt komutlar ve kayıt rotaları
- Etkileşimler: komut seçildikten sonra gönderilen asıl yanıt türleri
- Hız sınırları: otomatik tamamlama ve etkileşim hız sınırları

Formlar

Form, botunun bir komuta ya da butona yanıt olarak kişiye açtığı pencere: kişi en çok beş alanı doldurur, gönderir ve doldurduğu her şey sana yeni bir etkileşim olarak gelir.

> **Uyarı:** Formda şifre, doğrulama kodu, kart bilgisi ya da token isteme. Bu geliştirici koşullarına aykırıdır ve botun kapatılır. Kişi her formun altında senin kaldıramayacağın bir satır görür: "Bu form {bot} botuna gider. Şifreni ya da doğrulama kodunu asla yazma." Yalnızca işin için gereken alanı sor.

Form mu, komut seçeneği mi?

Durum	Seç
Kısa, tek satırlık birkaç değer: bir sayı, bir kişi, listeden bir şey	Komut seçeneği
Uzun ya da çok satırlı metin: şikayet, başvuru, açıklama	Form
Değeri bir butona basınca sormak	Form (butonun seçeneği yoktur)
Kişinin yazdığını önceden doldurup düzelttirmek	Form, `value` ile
Yazarken öneri göstermek	Otomatik tamamlama; formda öneri yok

Akış

Adım	Ne olur
1	Kişi komutunu çalıştırır ya da butonuna basar. Sana `INTERACTION_CREATE` gelir, `type: 2` ya da `type: 3`.
2	3 saniye içinde `callback` ile `type: 9` ve formu gönderirsin. Yanıt `204`'tür. Bu, etkileşimin tek yanıtıdır.
3	Form kişinin ekranında açılır; altında sabit uyarı satırı durur. Kişi istediği kadar uzun yazabilir.
4	Kişi gönderir. Sana yeni bir `INTERACTION_CREATE` gelir, `type: 5`: kendi `id`'si ve 15 dakikalık yeni `token`'ı vardır.
5	3 saniye içinde bu yeni etkileşime yanıt verirsin: `4`, `5`, form bir butondan açıldıysa `6` ya da `7` de olur.

DM'de de aynıdır: komut ya da buton botla açılmış DM'deyse gönderim `context: 1` ile ve `member` yerine `user` ile gelir; kişi gönderim arasında botu engellerse gönderim reddedilir.

Form açıldıktan sonra 1 saat gönderilebilir. Açan etkileşimin 15 dakikalık token'ı bunu kısaltmaz: kişi 40 dakika yazsa da gönderim çalışır.

Kişi ne görür

> **Not:** Bu bölüm webde gözlenen davranışı anlatır. Masaüstü uygulaması aynı web kodunu kullanır ama ayrıca çalıştırılmadı. Android uygulaması formu kendi tasarımıyla çizer.

Form yalnızca komutu çalıştıran ya da butona basan cihazda açılır; kişinin öbür oturumları aynı olayı alır ama yok sayar. Pencerenin başlığı senin `title`'ındır. Altında botunun avatarı, adı, `BOT` etiketi ve uygulaman doğrulanmışsa "Doğrulanmış" etiketi durur. Alanlar senin sıranla gelir; zorunlu olanların adının yanında kırmızı bir yıldız vardır.

Alan	Kişi ne görür
Metin girişi, `style: 1`	Tek satırlık yazı kutusu, `placeholder` ve `value` ile. Altında kalan karakter sayısı yazar; sınırı geçince sayı kırmızıya döner. `max_length` göndermezsen sınır 4000'dir.
Metin girişi, `style: 2`	Çok satırlı, boyu uzatılabilen yazı kutusu; sayaç aynı.
Metin menüsü	Tek seçimliyse uygulamanın açılır listesi, çok seçimliyse işaretlenen bir liste penceresi. `default` seçenekler seçili gelir.
Kullanıcı, rol, kanal, karma menü	Bir düğme; basınca aramalı bir liste penceresi açılır, kişi işaretleyip "Onayla"ya basar. Kanal listesinde yalnızca kişinin gördüğü ve `channel_types`'a uyan kanallar çıkar. `required: false` menü boş bırakılabilir.

Pencerenin altında, alanların dışında, kişinin her zaman gördüğü satır durur: "Bu form {bot} botuna gider. Şifreni ya da doğrulama kodunu asla yazma." Alanlar uzunsa yalnızca alanlar kayar; bu satır ve düğmeler yerinde kalır.

Kişi "Gönder"e basınca uygulama önce zorunlu alanları ve uzunlukları sunucunun kurallarıyla denetler: yalnızca boşluk yazılmış zorunlu alan boş sayılır. Hata alanın altında kırmızı yazar ve form gitmez. Her şey tamsa "Gönder" dönen bir simgeye döner ve botun yanıtlayana kadar bekler; yanıt gelince form kapanır.

Gönderim başarısız olursa form açık kalır ve kişinin yazdığı her şey yerinde durur:

Ne oldu	Kişi ne görür
Botun 3 saniyede yanıt vermedi	Formun altında "Bot yanıt vermedi." Kişi yeniden gönderebilir.
Sunucu bir alanı reddetti (`validation_failed`)	Hata o alanın altında.
Form süresi doldu ya da üç kez gönderildi (`modal_expired`)	"Süre doldu, metni kopyalayıp komutu yeniden çalıştır." "Gönder" kapanır, yanında bir "Kopyala" düğmesi çıkar; yazılan metin alan adlarıyla birlikte panoya kopyalanır.
Başka bir hata	Hatanın metni formun altında.

Kişi bir şey yazmışsa kapatma düğmesi, `Esc`, dışarı tıklama ya da "Vazgeç" önce "Yazdıkların silinsin mi?" diye sorar. Yalnızca senin `value` ile doldurduğun metin duruyorsa form sormadan kapanır.

En küçük örnek

Komuta form ile yanıt ver:

```
curl -X POST "https://noluyo.net/api/v1/interactions/$INTERACTION_ID/$INTERACTION_TOKEN/callback"
\
  -H "Authorization: Bot $BOT_TOKEN" -H 'Content-Type: application/json' \
  -d '{"type":9,"data":{"custom_id":"sikayet","title":"Şikayet","components":
[{"type":1,"components":
[{"type":4,"custom_id":"konu","style":1,"label":"Konu","max_length":100}]},
{"type":1,"components":[{"type":4,"custom_id":"detay","style":2,"label":"Ne
oldu?","required":false}]}]}}'
```

Aynı form JSON olarak:

```
{
  "type": 9,
  "data": {
    "custom_id": "sikayet",
    "title": "Şikayet",
    "components": [
      { "type": 1, "components": [
        { "type": 4, "custom_id": "konu", "style": 1, "label": "Konu", "max_length": 100 }
      ] },
      { "type": 1, "components": [
        { "type": 4, "custom_id": "detay", "style": 2, "label": "Ne oldu?", "required": false }
      ] }
    ]
  }
}
```

Formun alanları

Alan	Kural
`custom_id`	Zorunlu, en çok 100 karakter. Gönderimde `data.custom_id` olarak geri gelir.
`title`	Zorunlu, en çok 45 karakter
`components`	1-5 eylem satırı (`type: 1`); her satırda tek bir metin girişi ya da tek bir menü

Formun JSON'u en çok 16 KiB olabilir. Alanların `custom_id` 'leri form içinde tekil olmalı. Otomatik moderasyon başlığı, etiketleri, yer tutucuları ve ön doldurduğun değerleri okur; kuralı tutan form `automod_blocked` ile reddedilir.

Metin girişi

Alan	Kural
`type`	`4`
`custom_id`	Zorunlu, en çok 100 karakter
`style`	`1` tek satır, `2` paragraf
`label`	Zorunlu, en çok 45 karakter
`placeholder`	Boşken görünen yazı, en çok 100 karakter
`min_length`	0-4000, varsayılan 0
`max_length`	1-4000, varsayılan 4000
`required`	Varsayılan `true`
`value`	Form açılırken alanda dolu görünen metin, en çok `max_length`; kişi değiştirebilir

Zorunlu alana yalnız boşluk yazılırsa gönderim reddedilir ve kişi alanı doldurması gerektiğini görür. Bunun dışında metin yazıldığı gibi gelir: baştaki ve sondaki boşluk silinmez, satır sonları korunur. İsteğe bağlı alan boş bırakıldıysa `value` boş dizedir.

Formda menü

Bir satır metin girişi yerine bir menü de taşıyabilir: metin (`3`), kullanıcı (`5`), rol (`6`), kullanıcı ya da rol (`7`), kanal (`8`). Bileşenler sayfasındaki kurallar geçerlidir; formda iki fark vardır:

Alan	Formda
`label`	Zorunlu, en çok 45 karakter
`required`	Varsayılan `true`; `false` ise kişi menüyü boş bırakabilir, o zaman `values` boş liste gelir
`disabled`	Kullanılmaz

Sunucu seçilenleri kişinin erişebildiğine göre denetler: kullanıcı sunucunun üyesi, rol bu sunucunun, kanal kişinin görebildiği olmalı. Seçilen kullanıcılar, üyeler, roller ve kanallar `data.resolved` içinde gelir.

Ön doldurma

`value` ile kişinin daha önce yazdığını ya da senin bildiğin bir değeri alana koyarsın; kişi onaylar ya da düzeltir. Örneğin bir butondan "düzenle" formu açarken eski metni geri ver:

```
{ "type": 4, "custom_id": "aciklama", "style": 2, "label": "Açıklama", "value": "Eski açıklama burada." }
```

Kullanıcı, rol ve kanal menüsünde `default_values` aynı işi görür.

Gönderim sana nasıl gelir

```
{
  "id": "1118042062848007777",
  "application_id": "14631410073600008",
  "type": 5,
  "token": "eyJpZCI6IjExMTgwNDIwNjI4NDgwMDc3NyJ9.bWFj",
  "version": 1,
  "guild_id": "21531879014400001",
  "channel_id": "21531887403008007",
  "member": {
    "user": { "id": "14631410073600003", "username": "ayse", "display_name": "Ayşe", "avatar": null, "cosmetics": null },
    "nick": null, "roles": [], "joined_at": "2026-10-01T12:00:00Z", "permissions": "2147483648"
  },
  "app_permissions": "2048",
  "locale": "tr",
  "data": {
    "custom_id": "sikayet",
    "components": [
      { "type": 1, "components": [{ "type": 4, "custom_id": "konu", "value": "Spam yapan biri" }] },
      { "type": 1, "components": [{ "type": 4, "custom_id": "detay", "value": "Genel kanalda\nreklam atıyor." }] },
      { "type": 1, "components": [{ "type": 5, "custom_id": "kim", "values": ["14631410073600005"] }] }
    ],
    "resolved": {
      "users": { "14631410073600005": { "id": "14631410073600005", "username": "spamci", "display_name": null, "avatar": null, "cosmetics": null } }
    }
  }
}
```

Alan	Anlamı
`type`	`5` : gönderilen form
`token`	Bu gönderimin kendi token'ı, 15 dakika geçerli; yanıt ve takip mesajları bununla
`data.custom_id`	Açtığın formun `custom_id` 'si
`data.components`	Formun her alanı kendi sırasıyla, doldurulmuş ya da boş; metin girişi `value` , menü `values` taşır
`data.resolved`	Menülerde seçilen kullanıcı, üye, rol ve kanal; menü yoksa yer almaz
`member` ya da `user` , `locale`	Gönderen kişi ve dili

Formu bir komut açtıysa yanıtın mesajı "X, /komut kullandı" diye gösterilir.

Yanıt seçenekleri

Etkileşim	`type: 4`	`type: 5`	`type: 6`	`type: 7`	`type: 9`
Komut çalıştırma (`type: 2`)	Evet	Evet	Hayır	Hayır	Evet
Butona basma, menüden seçme (`type: 3`)	Evet	Evet	Evet	Evet	Evet
Otomatik tamamlama (`type: 4`)	Hayır	Hayır	Hayır	Hayır	Hayır
Komuttan açılmış formun gönderimi (`type: 5`)	Evet	Evet	Hayır	Hayır	Hayır
Butondan açılmış formun gönderimi (`type: 5`)	Evet	Evet	Evet	Evet	Hayır

`type: 7` butondan açılmış formda butonun durduğu mesajı günceller; o mesaj ephemeral ise ephemeral mesaj bütünüyle değişir. `flags: 64` ile `4` ya da `5` yanıtı yalnızca gönderen görür. Ayrıntılar Etkileşimler sayfasında.

Kurallar

- **Formu açan bir form açamaz.** Gönderilen forma `type: 9` ile yanıt veremezsin. Başka soruların varsa ilk formu en çok beş alanla tasarla ya da yanıtına bir buton koy; buton yeni bir form açabilir.
- **Formu açmak ilk yanıttır.** Önce `type: 5` ya da `type: 6` verip sonra form açamazsın: 3 saniye içinde doğrudan `type: 9` gönder. Formu hazırlamak için dışarıya gitme; tanımı kodunda hazır tut.
- **Form yazmaz.** `type: 9` kanala mesaj yazmaz ve tüm botların ortak yazma tavanına sayılmaz.
- **Bir form üç kez gönderilebilir.** Botun yanıt vermezse kişi yazdığını kaybetmeden yeniden gönderebilir. Her gönderim sana ayrı bir etkileşim olarak gelir; aynı içerik ikinci kez gelirse işini iki kez yapma. Üçüncüden sonra kişi `modal_expired` alır ve formu yeniden açması gerekir.
- **Gönderimde her şey yeniden denetlenir.** Kişi kanala hâlâ erişebilmeli, zaman aşımında olmamalı, botun sunucuda olmalı ve form bir komuttan açıldıysa komutun izin kuralları kişiye hâlâ izin vermeli. Form sen açtığından beri değişmemiştir: kişi alan ekleyemez, `custom_id` değiştiremez.

- **Yazılan saklanmaz.** N'oluyo formun içeriğini saklamaz; yalnızca kişinin bir form gönderdiğini erişim kaydına yazar. İçerik sana gelir; yalnızca işin için kullan.

Tam örnek: şikayet formu botu

Node.js 22 veya üstü yeter. Gateway sayfasındaki `gateway.mjs` ile Örnekler sayfasındaki `api.mjs` aynı klasörde olmalı. Bot `/sikayet` komutunu kaydeder, form açar, gönderilen şikayeti moderatör kanalına yazar ve kişiye yalnızca onun göreceği bir teşekkür gönderir:

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

// Moderatörlerin gördüğü kanalın kimliği.
const modChannel = process.env.MOD_CHANNEL_ID;

const commands = [{
  name: "sikayet",
  name_localizations: { en: "report" },
  description: "Moderatörlere şikayet gönder",
  description_localizations: { en: "Send a report to the moderators" },
}];

const form = (en) => ({
  custom_id: "sikayet",
  title: en ? "Report" : "Şikayet",
  components: [
    { type: 1, components: [{ type: 5, custom_id: "kim", label: en ? "Who?" : "Kim?" }] },
    { type: 1, components: [{ type: 4, custom_id: "konu", style: 1, label: en ? "Subject" :
"Konu", max_length: 100 }] },
    { type: 1, components: [{ type: 4, custom_id: "detay", style: 2, label: en ? "What happened?"
: "Ne oldu?",
      required: false, max_length: 1500 }] },
  ],
});

// Gönderilen formun alanları: custom_id -> value ya da values.
function fields(data) {
  const out = {};
  for (const row of data.components) for (const f of row.components) out[f.custom_id] = f.value
?? f.values;
  return out;
}

async function onInteraction(d) {
  const path = `/interactions/${d.id}/${d.token}/callback`;
  const en = d.locale === "en";
  if (d.type === 2 && d.data.name === "sikayet") {
    // Formu açmak ilk yanıttır: önce 5 verilmez.
    await call("POST", path, { type: 9, data: form(en) });
    return;
  }
  if (d.type !== 5 || d.data.custom_id !== "sikayet") return;
  const f = fields(d.data);
  const who = d.data.resolved?.users?.[f.kim[0]];
  const from = d.member?.user ?? d.user;
  await call("POST", path, { type: 5, data: { flags: 64 } });
  try {
```

```
    await call("POST", `/channels/${modChannel}/messages`, {
      embeds: [{
        title: f.konu,
        description: f.detay || "-",
        fields: [
          { name: "Şikayet edilen", value: `<@${who.id}>`, inline: true },
          { name: "Gönderen", value: `<@${from.id}>`, inline: true },
        ],
      }],
      allowed_mentions: { parse: [] },
    });
    await call("PATCH", `/interactions/${d.id}/${d.token}/messages/@original`, {
      content: en ? "Thanks, the moderators got your report." : "Teşekkürler, şikayetin
moderatörlere ulaştı.",
    });
  } catch (err) {
    console.error(err.message);
    await call("PATCH", `/interactions/${d.id}/${d.token}/messages/@original`, {
      content: en ? "Your report could not be delivered. Try again later." : "Şikayetin
iletilemedi. Biraz sonra yeniden dene.",
    });
  }
}

let registered = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !registered) {
      registered = true;
      call("PUT", `/applications/${d.application.id}/commands`, commands)
        .then(() => console.log("/sikayet kaydedildi"))
        .catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onInteraction(d).catch((err) => console.error(err.message));
  },
});
```

Çalıştırmak için botunun moderatör kanalında `SendMessages` ve `EmbedLinks` izni olsun:

```
BOT_TOKEN='kopyaladigin_token' MOD_CHANNEL_ID='kanal_kimligi' node sikayet-botu.mjs
```

Örnek şikayetin içeriğini konsola yazmaz. Moderatör kanalına yazma 3 saniyeyi aşabilir; bu yüzden önce `type: 5` ile yanıt verip sonucu `@original` ile doldurur.

Limitler

Anahtar	Değer	Saniye	Anlamı
`title`	45		Başlık, karakter
`rows`	5		Formdaki alan sayısı
`label`	45		Alan etiketi, karakter
`placeholder`	100		Metin girişinin yer tutucusu, karakter
`value`	4000		Metin girişine yazılabilen, karakter
`bytes`	16384		Formun JSON'u
`sends`	3	3600	Bir formun gönderilebileceği sayı ve formun ömrü
`user`	5	5	Kişi başına kanalda basma, komut ve form gönderimi birlikte

Kişinin gönderimleri basma ve komutlarla aynı sınırları paylaşır; aşılırsa sana bir şey gelmez, kişi `429` alır ve form açık kalır.

Hatalar

Formu açarken sana dönenler:

Kod	Ne oldu	Ne yap
`interaction_response_invalid`	`type: 9`'u otomatik tamamlamaya ya da gönderilen forma verdin, `data` yok, ya da yanında `content`, `embeds`, `flags` gibi mesaj alanı var	Formu yalnızca komuta ya da basmaya, yalnızca `custom_id`, `title` ve `components` ile gönder.
`validation_failed`	Form geçersiz: örneğin başlık yok, beşten çok satır, bir satırda iki alan, etiket 45 karakteri aşıyor	`errors` alanına bak. Hakkın gitmedi, düzeltip yeniden gönder.
`automod_blocked`	Başlık, etiket ya da ön doldurduğun değer sunucunun otomatik moderasyon kuralına takıldı	Metni değiştir.
`interaction_already_acknowledged`	Bu etkileşime zaten yanıt verdin, örneğin önce `type: 5`	Formu ilk yanıt olarak gönder.
`interaction_timeout`	3 saniye geçti	Formu kodunda hazır tut, açmadan önce dışarıya gitme.

Kişinin gönderirken alabileceği hatalar sana gelmez:

Kod	Ne oldu
`modal_expired`	Form 1 saatten eski, üç kez gönderildi, ya da başka bir kişinin veya kanalın
`validation_failed`	Zorunlu alan boş ya da yalnız boşluk, uzunluk dışında, menüde kişinin erişemediği bir değer
`missing_access` , `missing_permissions`	Kişi kanalı artık göremiyor ya da zaman aşımında
`application_unavailable`	Botun sunucudan çıkarıldı ya da kapatıldı
`command_forbidden`	Form bir komuttan açıldı ve komutun izin kuralları artık kişiye izin vermiyor
`rate_limited` , `interaction_busy`	Kişinin ya da tüm botların sınırı doldu

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:
>
> - Token koda yazılmaz, `process.env.BOT_TOKEN` 'dan okunur.
> - Formda şifre, doğrulama kodu, kart bilgisi ya da token istenmez; yalnızca işin gerektirdiği alanlar sorulur.
> - Form, komuta (`type: 2`) ya da basmaya (`type: 3`) 3 saniye içinde ilk yanıt olarak `{"type":9,"data":{"custom_id","title","components"}}` ile açılır; önce `type: 5` verilmez.
> - En çok 5 satır, her satırda tek metin girişi (`type: 4`) ya da tek menü; başlık ve etiket en çok 45 karakter.
> - Gönderim `INTERACTION_CREATE` `type: 5` olarak kendi `id` ve `token` 'ıyla gelir ve 3 saniye içinde yanıtlanır; yanıt `type: 9` olamaz.
> - Alanlar `data.components[].components[]` içindedir: metin girişi `value` , menü `values` taşır; seçilenler `data.resolved` 'dadır.
> - Aynı form üç kez gelebilir; iş iki kez yapılmaz. Formun içeriği konsola yazılmaz.

Sırada ne var

- Komutlar: form açabilen eğik çizgi, kullanıcı ve mesaj komutları
- Bileşenler: form açabilen butonlar ve seçim menüleri
- Etkileşimler: form gönderimine verilen yanıt türleri ve takip mesajları

Etkileşimler

Biri botunun butonuna bastığında ya da komutunu çalıştırdığında bot `INTERACTION_CREATE` alır ve ona yanıt verir. Bu sayfa yanıtın dört türünü, süreleri ve sonradan gelen mesajları anlatır. Basmanın kendisi ve bileşenlerin nasıl yazıldığı Bileşenler sayfasında.

Zaman çizelgesi

Ne zaman	Ne olur
0 sn	Kişi basar, bot `INTERACTION_CREATE` alır. Olayda `id` ve `token` vardır.
3 sn	İlk yanıtın son anı. Sonrası `400 interaction_timeout` döner, kişi "Bot yanıt vermedi" görür.
15 dk	Token'ın ömrü. `@original` düzenlemesi ve takip mesajları buraya kadar çalışır, sonrası `400 interaction_expired` döner.

Her etkileşim tek bir ilk yanıt alır. İkincisi `400 interaction_already_acknowledged` döner. Reddedilen bir yanıt hakkını harcamaz: hatayı düzeltip süre içinde yeniden gönderebilirsin.

Gateway mı HTTP mi

Etkileşim botuna iki yoldan gelebilir. Uygulamanın etkileşim adresi yoksa gateway'den, varsa yalnızca o adrese. İçerik ikisinde aynıdır; değişen, yanıtı nasıl verdiğindir:

	Gateway	HTTP adresi
Etkileşim nasıl gelir	`INTERACTION_CREATE` olayı	Adresine imzalı `POST`
İlk yanıt	`callback` rotası, bot token'ıyla	Aynı isteğin `200` gövdesi; ya da `202` dönüp `callback` rotası
3 saniye	Etkileşimin kimliğinden sayılır	Adresine bağlanma ve yanıtın tamamı dahil
Reddedilen yanıt	Hata kodunu sen alırsın, süre içinde düzeltip yeniden gönderebilirsin	Kişi "Bot zamanında yanıt vermedi." görür, kod teslim günlüğüne yazılır
Bot kapalıysa	Etkileşim kaybolur, kişi 3 saniye sonra hatayı görür	Kişi hatayı hemen görür
Takip, `@original`	Aynı rotalar	Aynı rotalar, bot token'ıyla
Sınırlar	`interaction` sınıfı ve ortak tavanlar	Aynısı

Kişi hangi yolu kullandığını görmez.

Sunucuda mı DM'de mi

Her etkileşim `context` alanını taşır: `0` sunucu kanalı, `1` kişinin botunla açtığı DM. Basma, komut, otomatik tamamlama ve form DM'de de aynı çalışır; fark yalnızca kimin nerede olduğundadır:

Alan	Sunucuda	DM'de
`context`	`0`	`1`
`guild_id`	Sunucunun kimliği	Yok
`member`	Kişi, `permissions` dolu	Yok
`user`	Yok	Kişi
`app_permissions`	Botun bu kanaldaki izinleri	DM'nin sabit izinleri

```
// Aynı kod sunucuda ve DM'de çalışır.
const guildId = interaction.guild_id ?? null; // DM'de null
const person = interaction.member?.user ?? interaction.user;
const inDM = interaction.context === 1;
if (inDM) {
  // Rol yok, izin yok: yetki isteyen işi burada yapma.
}
```

Yanıt rotaları DM'de de aynıdır. Kişi DM'yi kapatır ya da botu engellerse yanıt `403 dm_closed` alır (Özel mesajlar).

Hangi yanıtı seçmeliyim?

İşin 3 saniyeden uzun sürer mi?	Neyi değiştiriyorsun?	Seç
Hayır	Kanala yeni bir mesaj yazıyorsun	`type: 4`
Hayır	Butonun durduğu mesajı değiştiriyorsun	`type: 7`
Evet	Sonuç yeni bir mesaj olacak	`type: 5`, sonra `PATCH @original`
Evet	Butonun durduğu mesajı değiştireceksin	`type: 6`, sonra kanal mesajını düzenle
Evet ya da hayır	Görünen hiçbir şey değişmiyor	`type: 6`
Hayır	Bir komuta yanıt veriyorsun	`type: 4`
Evet	Bir komuta yanıt veriyorsun	`type: 5`, sonra `PATCH @original`

Her yanıt her etkileşime uymaz. Uymayanı göndermek `400 interaction_response_invalid` döner:

Etkileşim	`type: 4`	`type: 5`	`type: 6`	`type: 7`	`type: 8`	`type: 9`
Butona basma, menüden seçme (`type: 3`)	Evet	Evet	Evet	Evet	Hayır	Evet
Komut çalıştırma (`type: 2`)	Evet	Evet	Hayır	Hayır	Hayır	Evet
Otomatik tamamlama (`type: 4`)	Hayır	Hayır	Hayır	Hayır	Evet	Hayır
Gönderilen form (`type: 5`)	Evet	Evet	Butondan açıldıysa	Butondan açıldıysa	Hayır	Hayır

Komut çalıştırmanın üzerinde durduğu bir mesaj yoktur; değiştirilecek bir şey olmadığı için `6` ve `7` almaz. `type: 8` kişi yazarken önerdiğin seçeneklerdir (`data.choices`) ve yalnızca otomatik tamamlamaya uyar. `type: 9` kişiye bir form açar; ilk yanıt olmalıdır ve gönderilen form sana yeni bir etkileşim olarak gelir.

Yanıtı yalnızca basan kişi görsün istiyorsan `data` içine `flags: 64` ekle: Yalnızca basan kişi görsün.

Yanıt rotası

Dört tür de aynı rotaya gider:

```
curl -X POST "https://noluyo.net/api/v1/interactions/$INTERACTION_ID/$INTERACTION_TOKEN/callback"
\
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"type":4,"data":{"content":"Kaydın alındı."}}'
```

type 4: yeni mesajla yanıt

Kanala yeni bir mesaj yazar. `data` bir mesaj gibidir: `content`, `embeds`, `components`, `allowed_mentions`. Metin ya da embed zorunludur.

```
{ "type": 4, "data": { "content": "Kaydın alındı." } }
```

Yanıt `200` ve mesajdır. Mesajın `interaction` alanı kimin bastığını söyler:

```
{
  "message": {
    "id": "111600000000000500",
    "type": 0,
    "channel_id": "111500000000000100",
    "content": "Kaydın alındı.",
    "application_id": "111540000000000000",
    "interaction": {
      "id": "111600000000000400",
      "type": 3,
      "user": { "id": "111540000000000009", "username": "ayse" }
    }
  }
}
```

type 5: önce "düşünüyor", sonra sonuç

İşin uzun sürecekse hemen bunu gönder. Kanala boş bir mesaj düşer ve `flags: 128` taşır; kişi onu "düşünüyor" olarak görür. `data` içinden yalnızca `flags` okunur: `content`, `embeds` ve `components` gönderirsen yok sayılır, onları `PATCH @original` ile gönder.

```
{ "type": 5 }
```

Yanıt `200` ve mesajdır:

```
{
  "message": {
    "id": "111600000000000500",
    "type": 0,
    "content": "",
    "flags": 128,
    "interaction": {
      "id": "111600000000000400",
      "type": 3,
      "user": { "id": "111540000000000009", "username": "ayse" }
    }
  }
}
```

İş bitince mesajı doldur. `flags`'teki `128` kendiliğinden silinir, mesaj "(düzenlendi)" olarak işaretlenmez:

```
curl -X PATCH
"https://noluyo.net/api/v1/interactions/$INTERACTION_ID/$INTERACTION_TOKEN/messages/@original" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"content":"Rapor hazır."}'
```

> **Uyarı:** Botun hiç dönmezse mesaj kanalda boş kalır, `flags: 128` da silinmez. İstemciler böyle bir mesajı süre dolunca "Bot yanıt vermedi" diye gösterir; bu kuralı kendi istemcinde de uygula. Bu mesajı doldurmak bildirim göndermez: içindeki etiket kişiye haber vermez. Haber vermek istiyorsan bir takip mesajı at.

type 6: sessiz onay

Bastığını aldığını söyler, ekranda hiçbir şey değişmez. Mesaj oluşmaz, `SendMessages` da gerekmez. `data` gönderirsen yok sayılır.

```
{ "type": 6 }
```

Yanıt `204`, gövdesizdir. Sonra kanala bir şey yazmak istersen takip mesajı kullan. Butonun durduğu mesajı değiştirmek istersen mesaj düzenleme rotası çalışır; `@original` burada işe yaramaz, çünkü bu yanıtın mesajı yoktur: istek `200` döner ama ekranda bir şey değişmez.

type 7: basılan mesajı güncelle

Butonun durduğu mesajı değiştirir. Göndermediğin alan olduğu gibi kalır, boş `components` listesi butonları kaldırır. Yalnızca `components` değişirse mesaj "(düzenlendi)" olarak işaretlenmez.

```
{ "type": 7, "data": { "content": "Oylama bitti.", "components": [] } }
```

Yanıt `200` ve güncellenen mesajdır:

```
{
  "message": {
    "id": "111600000000000000",
    "content": "Oylama bitti.",
    "edited_timestamp": "2026-11-07T20:00:00.000Z"
  }
}
```

Bu rotalarda yeni dosya gönderilemez. Görsel göndermek istiyorsan takip mesajı kullan.

@original ve takip mesajları

İlk yanıtı verdikten sonra token 15 dakika boyunca çalışır:

İstek	Ne yapar	Yanıt
`POST /interactions/{id}/{token}/messages`	Yeni takip mesajı yazar, dosya da alır	`200` ve mesaj
`PATCH /interactions/{id}/{token}/messages/@original`	`type: 4` ya da `type: 5` mesajını düzenler	`200` ve mesaj
`PATCH /interactions/{id}/{token}/messages/{message_id}`	Bu etkileşimin bir mesajını düzenler	`200` ve mesaj
`DELETE /interactions/{id}/{token}/messages/{message_id}`	Bu etkileşimin bir mesajını siler, `@original` da olur	`204`

Token yalnızca kendi etkileşiminin mesajlarına ulaşır. Aynı kanaldaki başka bir mesaj `404 unknown_message` döner. Bir etkileşim en çok 5 takip mesajı atabilir, altıncısı `429` döner. Takip mesajı dosya almak için `multipart/form-data` kullanır; biçimi mesaj gönderme ile aynıdır.

```
curl -X POST "https://noluyo.net/api/v1/interactions/$INTERACTION_ID/$INTERACTION_TOKEN/messages"
\
  -H "Authorization: Bot $BOT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"content":"Bir şey daha: rapor e-postana da gitti."}'
```

Ephemeral takip mesajı dosya almaz ama beşe aynı sayılır. Bir etkileşimin ephemeral mesajları 5 saniyede en çok 5 kez düzenlenebilir, fazlası `429` döner. Ayrıca tüm botlar birlikte bir kişiye 5 saniyede en çok 262144 bayt ephemeral mesaj gönderebilir (metin, embed'ler ve bileşenlerin üç katı sayılır); fazlası `429 rate_limited` döner ve kişiye bir şey gitmez. Yanıtlara yavaş mod ve mesaj sınırı işlemez, çünkü onları bir kişinin basması tetikler. `SendMessages` izni, otomatik moderasyon ve `interaction` sınıfı ile ortak tavanlar yine geçerlidir.

Yalnızca basan kişi görsün

`data` içine `flags: 64` eklersen yanıtı yalnızca basan kişi görür. Buna ephemeral mesaj denir. `type: 4`, `type: 5` ve takip mesajında çalışır:

```
{ "type": 4, "data": { "content": "Kaydın alındı. Bunu yalnızca sen görüyorsun.", "flags": 64 } }
```

Yanıt `200` ve `{"message": ...}`, takip mesajında yalnızca mesajdır. `flags` `64` gelir, `mentions` her zaman boştur. İlk yanıtın kimliği etkileşimin kimliğidir, takip mesajı yeni kimlik alır.

`type: 5` ile `flags: 64` gönderirsen kişi "düşünüyor" görür ve `flags` `192` gelir. `PATCH @original` ile doldurduğunda `64` olur. Bu `PATCH` ve diğerleri yalnızca basan kişiye gider. Kişi artık kanalı görmüyorsa `404 unknown_channel` döner, ona bir şey gitmez.

Ne zaman kaybolur

N'oluyo bu mesajı saklamaz. Kişi sayfayı yenilerse ya da uygulamayı yeniden açarsa mesaj gider. O anda açık olan başka cihazında da görünür, sonradan açtığı cihazda görünmez. Kişi mesajdaki "Kapat" düğmesiyle onu kendi ekranından kaldırabilir, bu sana haber vermez. Mesaj geçmişte çıkmaz, okunmadı sayacını ve `last_message_id`'yi oynatmaz, bildirim göndermez. Diğer üyeler ve botlar, senin botun da, `MESSAGE_CREATE` almaz.

Kalıcı olması gereken bir şeyi burada söyleme. Kişi mesajı kaybetse de sonucu bulabilmeli.

Ne yapamaz

Yapılamayan	Olan	Yerine
Görsel ya da dosya	`400 ephemeral_files`, hem `attachment://` hem `multipart/form-data`	Dosya bir mesaj satırına bağlıdır, burada satır yok. Görseli herkese açık mesajla gönder.
Tepki, sabitleme	Satırı olmayan mesajda çalışmaz	Bileşen kullan.
Kanala doğrudan atma	`POST /channels/{id}/messages` `400 ephemeral_forbidden` döner	Bir etkileşime yanıt ver: kime gösterileceğini basma söyler.
Kanal mesajı rotalarıyla okuma, düzenleme	`404 unknown_message`	Etkileşim rotalarını kullan.
15 dakika sonra basma	Kişi `400 interaction_expired` alır, sana bir şey gelmez	Mesajı değiştirirsen süre yeniden başlar.

> **Uyarı:** Ephemeral mesajı düzenlerken göndermediğin alan **boş** sayılır, olduğu gibi kalmaz. Birleştirilecek bir satır yok. `PATCH` ya da `type: 7` içinde yalnızca `embeds` gönderirsen metin gider, yalnızca `components` gönderirsen `400 validation_failed` alırsın. Bedeli şu: metni unutursan kişi onu kaybeder. Her seferinde mesajın tamamını gönder.

Butonlu ephemeral

Bileşen eklersen kişi basabilir. Mesaj saklanmadığı için botuna gelen `message` kısadır: `id`, `channel_id`, `flags`, `author` ve `components`. **Metni gelmez**, gerekiyorsa bilgiyi `custom_id`'ye koy ya da kendi veritabanında tut.

```
{
  "type": 4,
  "data": {
    "content": "Silinsin mi?",
    "flags": 64,
    "components": [{ "type": 1, "components": [
      { "type": 2, "style": 4, "label": "Sil", "custom_id": "sil:42" }
    ] }]
  }
}
```

Kişi bastığında:

```
{
  "id": "1116000000000000700",
  "type": 3,
  "token": "...",
  "channel_id": "1115000000000000100",
  "data": { "custom_id": "sil:42", "component_type": 2 },
  "message": {
    "id": "1116000000000000400",
    "channel_id": "1115000000000000100",
    "flags": 64,
    "author": { "id": "1115400000000000001", "username": "botun" },
    "components": [{ "type": 1, "components": [
      { "type": 2, "style": 4, "label": "Sil", "custom_id": "sil:42" }
    ] }]
  }
}
```

`type: 7` basılan mesajı değiştirir. `type: 4` ile `flags: 64` yeni bir ephemeral açar, `flags` olmadan herkese açık yanıt verir. `type: 7` sonrası düzenleme için `@original` değil, `message.id` kullan: `PATCH /interactions/{id}/{token}/messages/{message.id}`.

```
{ "type": 7, "data": { "content": "Silindi.", "components": [] } }
```

> **Not:** Bileşenler mesajla birlikte mühürlenip kişinin istemcisine gider, basarken `ephemeral_state` olarak geri gelir. Kişi onu kurcalarsa, başkasının durumunu ya da mesajda olmayan bir `custom_id` gönderirse `404 component_not_found` alır ve sana bir şey gelmez. Yani gelen `custom_id` senin gönderdiğindir. Ama **güncel olmayabilir**: mesajı değiştirdikten sonra kişi eski durumla, 15 dakika içinde, eski butona yeniden basabilir. Bir kez yapılması gereken işi kendi tarafında denetle. `ephemeral_state` yanıtında sana gelmez.

Satırı olmayan kimlik ephemeral sayıldığı için `type: 6` ya da `type: 7` sonrası `@original` `200` döner ama ekranda bir şey değişmez: o kimlikte mesaj yoktur. İstemci ephemeral `MESSAGE_UPDATE` ve `MESSAGE_DELETE` olayını yalnızca elinde `flags: 64` ile tuttuğu ve `application_id`'si aynı olan mesaja uygular. Böylece bir bot başka botun mesajını değiştiremez.

Kişi ne görür

> **Not:** Bu bölüm webde gözlenen davranışı anlatır. Masaüstü uygulaması aynı web kodunu kullanır ama ayrıca çalıştırılmadı. Android ve iOS uygulamalarında henüz gözlenmedi.

Etkileşime yanıt olan mesajın üstünde, yanıt satırı gibi, "ad kullandı" yazar. Herkese açık yanıtta bunu kanaldaki herkes görür. Böyle bir mesaj botunun bir önceki mesajıyla tek blokta birleşmez, adı ve avatarıyla ayrı başlar.

`type: 5` sonrası mesajda üç nokta oynar ve "Botun adı düşünüyor…" yazar. `PATCH @original` gelince yerini yanıt alır. Bot 15 dakika içinde doldurmazsa yazı "Bot yanıt vermedi." olur; kişi sonsuza dek bekleyen bir mesaj görmez.

Ephemeral mesaj hafif renkli bir zeminde, solunda temanın renginde bir çizgiyle durur, altında "Yalnızca sen görüyorsun · Kapat" yazar. Üzerinde mesaj araç çubuğu ve sağ tık menüsü yoktur: yanıtlanamaz, iletilemez, sabitlenemez, tepki eklenemez ve bildirilemez. "Kapat" mesajı yalnızca o ekrandan kaldırır. Sayfa yenilenince ephemeral mesajlar gider; okunmadı sayacı ve kanalın son mesajı yerinde kalır.

Ephemeral mesajın butonları ve menüleri herkese açık mesajdakiler gibi çizilir ve basılır. Basınca uygulama mesajla gelen `ephemeral_state`'i geri gönderir, sunucu bileşenleri onunla doğrular. `type: 7` ile yanıt verirsen ephemeral mesaj yerinde değişir. Mesaj gelişinden ya da son güncellenişinden 15 dakika sonra basan kişi "Bu etkileşimin süresi doldu." görür.

Token'ı koru

> **Uyarı:** Etkileşim token'ı adres satırında durur, bu yüzden log'a kolayca düşer. Kendi log'larında tam adresi yazma. Token'ı kullanıcıya gösterme, başka bir sisteme aktarma.

Token tek başına yetmez: dört rota da `Authorization: Bot` başlığı ister ve token'ı yalnızca etkileşimin gittiği bot kullanabilir. Başka bir botun göndermesi `400 interaction_token_invalid` döner. Sunucu bu rotaların adresindeki token'ı kendi log'larına yazmaz.

Hatalar

Kod	Ne oldu	Ne yap
`interaction_timeout`	İlk yanıt 3 saniyeyi geçti	İşi yanıttan önce yapma. Önce `type: 5` ya da `type: 6` gönder, sonra işe başla.
`interaction_already_acknowledged`	Bu etkileşime zaten yanıt verdin	Sonrası için takip mesajı ya da `@original` kullan.
`interaction_not_acknowledged`	Takip ya da düzenleme ilk yanıttan önce geldi	Önce `callback` gönder ve yanıtını bekle.
`interaction_response_invalid`	Bu etkileşime uymayan bir `type` gönderdin, `type: 7`'de `data` yok, `choices` `type: 8` dışında geldi, ya da `type: 9` otomatik tamamlamaya veya gönderilen forma gitti	Basmaya `4`, `5`, `6`, `7` ya da `9`, komuta `4`, `5` ya da `9`, otomatik tamamlamaya `8` gönder.
`interaction_token_invalid`	Token bozuk, adresteki kimlikle uyuşmuyor ya da başka bir botun	`id` ve `token`'ı aynı olaydan al, kendi bot token'ınla gönder.
`interaction_expired`	Token'ın 15 dakikası doldu	Kanala normal mesaj gönder.
`ephemeral_files`	Ephemeral mesajda dosya ya da `attachment://` var	Görseli herkese açık mesajla gönder.
`validation_failed`	`data` bir mesaj olarak geçersiz, örneğin metin yok	`errors` alanına bak. Hakkın gitmedi, düzeltip yeniden gönder.
`missing_permissions`	Botun bu kanalda `SendMessages` izni yok	`type: 6` izin istemez. Diğerleri için izni ver.
`rate_limited`	Sınır doldu; `X-RateLimit-Scope: shared` ise tüm botların ortak tavanı	`retry_after` kadar bekle. 3 saniye geçtiyse ilk yanıt artık kabul edilmez.

Her tür için hangisi gelebilir:

Kod	`type: 4`	`type: 5`	`type: 6`	`type: 7`
`interaction_timeout`	Evet	Evet	Evet	Evet
`interaction_already_acknowledged`	Evet	Evet	Evet	Evet
`validation_failed`	Evet	Yalnızca `flags`	Hayır	Evet
`missing_permissions`	Evet	Evet	Hayır	Evet
`automod_blocked`	Evet	Hayır	Hayır	Evet

Tam örnek: sayaç botu

Node.js 22 veya üstü yeter. Gateway sayfasındaki `gateway.mjs` ile Örnekler sayfasındaki `api.mjs` aynı klasörde olmalı. Bot açılınca kanala bir buton atar. Her basmada sayıyı bir artırıp aynı mesajı `type: 7` ile günceller. Sayı `custom_id`'nin içinde durur, bot yeniden başlasa da kaybolmaz:

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const { CHANNEL_ID } = process.env;
if (!CHANNEL_ID) throw new Error("CHANNEL_ID tanımlı olmalı");

// Mesajın gövdesi: hem ilk gönderimde hem güncellemede aynı biçim.
const view = (count) => ({
  content: `Sayaç: ${count}`,
  components: [{ type: 1, components: [
    { type: 2, style: 1, label: "Artır", custom_id: `sayac:${count}` },
  ] }],
});

async function onPress(d) {
  const [kind, count] = d.data.custom_id.split(":");
  if (kind !== "sayac") return;
  await call("POST", `/interactions/${d.id}/${d.token}/callback`, {
    type: 7,
    data: view(Number(count) + 1),
  });
}

let posted = false;

await connect({
  intents: 0,
  onEvent(t, d) {
    if (t === "READY" && !posted) {
      posted = true;
      call("POST", `/channels/${CHANNEL_ID}/messages`, view(0)).catch(console.error);
    }
    if (t === "INTERACTION_CREATE") onPress(d).catch(console.error);
  },
});
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' CHANNEL_ID='kanal id degeri' node sayac-botu.mjs
```

İki kişi aynı anda basarsa ikisi de aynı sayıyı görüp aynı sonucu yazar. Sayım önemliyse sayıyı kendi veritabanında tut.

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur.
- İlk yanıt 3 saniye içinde gider. `INTERACTION_CREATE` gelince önce yanıt verilir; iş daha uzun sürecekse `type: 5` ya da `type: 6` gönderilir, iş ondan sonra yapılır.
- Etkileşim başına tek `callback` gönderilir; sonrası `@original` ya da takip mesajıdır.
- Etkileşim token'ı 15 dakika yaşar. Etkileşim başına en çok 5 takip mesajı olur; ephemeral mesajlar 5 saniyede en çok 5 kez düzenlenir.
- Dört rotada da `Authorization: Bot` başlığı zorunludur.
- Etkileşim token'ı ve tam adres asla log'a yazılmaz.
- Yetki isteyen işte önce `member.permissions`'a bakılır. DM'de (`context: 1`) `member` yoktur, kişi `user` içindedir: kişi `interaction.member?.user ?? interaction.user` ile, sunucu `interaction.guild_id ?? null` ile okunur.
- `flags: 64` dosya almaz; düzenlerken mesajın tamamı gönderilir, gönderilmeyen alan silinir.
- Ephemeral mesajın metni basınca geri gelmez; gereken bilgi `custom_id`'ye konur.

Sırada ne var

- Komutlar: `/` menüsü, kullanıcı ve mesaj komutlarının tanımı
- Formlar: `type: 9` ile çok satırlı form penceresi açmak
- HTTP ile teslim: etkileşimleri gateway yerine kendi HTTPS adresine almak

Özel mesajlar

Bir kişi botunla özel mesaj (DM) açabilir; orada sana yazar, komut çalıştırır ve butonlarına basar, sen de ona yanıt verirsin, ama **ilk yazan asla bot olmaz**.

Kişi DM'yi nereden açar

Kişi botunun profilinde ya da sunucunun üye listesinde **Mesaj gönder**'e basar. Bunun için ikisinin en az bir ortak sunucusu olmalı ve kişi botu engellememiş olmalı. Açılış senin değil kişinin işidir: botun `POST /users/@me/channels` çağırırsa `403 bot_forbidden` alır.

DM açılınca sana bir şey gelmez. Kişi ilk mesajını yazdığında ya da butona bastığında haberin olur.

Bot neden ilk yazamaz

N'oluyo'da bot kimseye kendiliğinden DM atamaz. Böylece kişiler istemedikleri reklamı ya da duyuruyu hiç görmez; bir botla konuşmak her zaman kişinin seçimidir. Bu yüzden "botumu ekleyen herkese hoş geldin mesajı" gibi bir şey yapamazsın. Onun yerine sunucuda bir kanala yaz ya da kişiye "Bana DM'den yaz" de.

Ne zaman yazabilirsin

Ne	Ne zaman	Kişi kapattıysa ya da engellediyse
Etkileşime yanıt: `callback`, takip mesajı, `@original` düzenleme ve silme	Token'ın 15 dakikası boyunca, her zaman	`dm_closed`
Kendiliğinden mesaj: `POST /channels/{channel_id}/messages`	Kişinin bu DM'deki son mesajı, komutu, basması ya da form gönderiminden sonraki 24 saat içinde	`dm_closed`
Tepki, sabitleme, yazıyor göstergesi	Kendiliğinden mesajla aynı: 24 saat içinde	`dm_closed`
Kendi eski mesajını düzenleme	Her zaman	`dm_closed`
Kendi eski mesajını silme	Her zaman	Her zaman

Kişi her yazdığında, komut çalıştırdığında ya da bastığında 24 saat baştan başlar. Kişi 24 saattir bir şey yapmadıysa kendiliğinden yazdığın her şey `403 dm_window_closed` alır; kişinin yeniden yazmasını bekle.

Kişi DM'yi açtıktan sonra ortak sunucudan ayrılsa da DM sürer: sen yazmaya, o yazmaya devam edebilir.

Kapatma ve engel

Kişi DM'yi kapatırsa ya da botu engellerse botun o DM'ye hiçbir şey yazamaz, etkileşim yanıtı da dahil. İki durumda da aynı yanıtı alırsın; hangisi olduğunu bilemezsin:

```
{ "code": "dm_closed", "message": "The person closed this DM or blocked the bot." }
```

Ne yap: `dm_closed` gelince o kişiye yazmayı bırak ve yeniden deneme. Kişi DM'yi yeniden açınca ya da engeli kaldırınca yazdığında sana yeniden mesaj gelir; o zaman yine yanıt verebilirsin. Senin mesajın kapatılmış DM'yi kişinin listesinde yeniden açmaz.

Kişi botu engellediyse kendisi de o DM'de bota yazamaz ve butonlara basamaz: `403 cannot_dm` alır.

DM'yi nasıl tanırsın

DM'den gelen mesajda ve etkileşimde `guild_id` yoktur. Etkileşimde `member` yerine `user` gelir ve `app_permissions` DM'nin sabit izinleridir.

```
{
  "id": "111804206284800099",
  "type": 0,
  "channel_id": "21531887403008042",
  "author": { "id": "14631410073600003", "username": "ayse", "display_name": "Ayşe", "avatar":
null, "cosmetics": null },
  "content": "Merhaba bot",
  "timestamp": "2026-11-08T12:30:00Z",
  "edited_timestamp": null,
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false
}
```

Komutlar DM'de

Global komutların DM'de de çalışır, `contexts` alanında `1` varsa ve `default_member_permissions` koymadıysan (DM'de komut). Kişinin komutu, menüsü, basması ve form gönderimi 24 saatlik pencereyi yeniden açar; otomatik tamamlama açmaz, yazmak henüz bir iş değildir. Kişi DM'yi kapattıysa yeniden açana dek komut çalıştıramaz; botu engellediyse komut da bileşen de çalışmaz.

DM'de bir komut çalışınca gelen etkileşim:

```
{
  "id": "112793931510448128",
  "application_id": "112793931040686080",
  "type": 2,
  "token": "eyJpZCI6IjExMjc5MzkzMTUxMDQ0ODEyOCJ9.1vpG",
  "version": 1,
  "context": 1,
  "channel_id": "112793931443339264",
  "user": { "id": "112793930608672768", "username": "ayse", "display_name": "Ayşe", "avatar":
null, "cosmetics": null },
  "app_permissions": "137476034112",
  "locale": "tr",
  "data": { "id": "112793931397201920", "name": "zar", "type": 1, "options": [{ "name": "yuz",
"type": 4, "value": 6 }] }
}
```


Seçeneklerde ve menülerde DM'nin kuralları:

Ne	DM'de
Kullanıcı seçeneği, kullanıcı menüsü	Yalnızca DM'nin iki kişisi: kişinin kendisi ya da botun
Rol seçeneği	Her zaman `invalid_target`
Kanal seçeneği	Yalnızca bu DM
Mesaj menüsü	Bu DM'nin ephemeral olmayan bir mesajı, içeriğiyle
Dosya seçeneği	Sunucudaki gibi

Intent ve içerik

DM mesajları için identify'da `DIRECT_MESSAGES` (`1 << 12` , `4096`) iste. DM'deki tepkiler için `DIRECT_MESSAGE_REACTIONS` , yazıyor bilgisi için `DIRECT_MESSAGE_TYPING` gerekir. Etkileşimler intent istemez.

`MESSAGE_CONTENT` DM'de uygulanmaz: kişinin bota yazdığı mesaj her zaman `content` ve dosyalarıyla tam gelir. Kişi mesajı sana yazıyor.

Shard

Sunucusu olmayan her şey, DM mesajları ve DM'den gelen etkileşimler, **shard 0**'a gelir. Birden çok shard açıyorsan DM'leri shard 0 oturumunda dinle. Ayrıntılar Shard'lar sayfasında.

Tam örnek: yankı botu

Node.js 22 veya üstü yeter. Gateway sayfasındaki `gateway.mjs` ile Örnekler sayfasındaki `api.mjs` aynı klasörde olmalı. Bot DM'ye gelen her mesajı aynen geri yazar; `dm_closed` gelirse o DM'ye bir daha yazmaz:

```
import { connect } from "./gateway.mjs";
import { call } from "./api.mjs";

const DIRECT_MESSAGES = 1 << 12;
let me;
// Kapatılmış ya da engellenmiş DM'ler: bunlara bir daha yazılmaz.
const closed = new Set();

async function echo(d) {
  if (closed.has(d.channel_id) || !d.content) return;
  try {
    await call("POST", `/channels/${d.channel_id}/messages`, {
      content: d.content.slice(0, 2000),
      message_reference: { message_id: d.id },
      allowed_mentions: { parse: [] },
    });
  } catch (err) {
    if (err.message.startsWith("403 dm_closed")) {
      closed.add(d.channel_id);
      return;
    }
    console.error(err.message);
  }
}

await connect({
  intents: DIRECT_MESSAGES,
  onEvent(t, d) {
    if (t === "READY") me = d.user.id;
    // DM'de guild_id yoktur; botun kendi mesajları da gelir.
    if (t !== "MESSAGE_CREATE" || d.guild_id || d.author.id === me) return;
    // Kişi yeniden yazdıysa DM yine açıktır.
    closed.delete(d.channel_id);
    echo(d);
  },
});
```

Çalıştırmak için:

```
BOT_TOKEN='kopyaladigin_token' node yanki-botu.mjs
```

Sonra botunun bulunduğu bir sunucuda botun profilinden **Mesaj gönder**'e bas ve bir şey yaz. Bot kişiye yalnızca kişi yazdığında yanıt verdiği için 24 saat kuralına hiç takılmaz.

Limitler

Anahtar	Değer	Saniye	Anlamı
`window`	86400		Kişinin son mesajı, komutu ya da basmasından sonra kendiliğinden yazabileceğin süre
`dm_send`	30	10	Tüm DM'lerinde toplam kendiliğinden yazma (mesaj, tepki, sabitleme, yazıyor); etkileşim yanıtları sayılmaz
`message`	5	5	Kanal başına mesaj, sunucudaki gibi

`dm_send` aşılınca `X-RateLimit-Scope: bot` ile `429` gelir. Diğer sınırlar Hız sınırları sayfasında.

Hatalar

Kod	Ne oldu	Ne yap
`dm_closed`	Kişi DM'yi kapattı ya da botu engelledi	O kişiye yazmayı bırak, yeniden deneme. Kişi yeniden yazarsa yanıt verirsin.
`dm_window_closed`	Kişinin son mesajı, komutu ya da basmasından beri 24 saat geçti	Kendiliğinden yazma. Kişi yeniden yazana dek bekle.
`rate_limited`	`X-RateLimit-Scope: bot` : `dm_send` doldu	`retry_after` kadar bekle; çok kişiye birden yazıyorsan yavaşla.
`bot_forbidden`	Bot DM açmaya çalıştı	DM'yi yalnızca kişi açar. Kişiye sunucuda "Bana DM'den yaz" de.
`unknown_channel`	Bu DM'nin alıcısı değilsin	Yalnızca kişinin seninle açtığı DM'ye yazabilirsin.

Kişinin alabileceği hata sana gelmez:

Kod	Ne oldu
`cannot_dm`	Kişi botu engellemiş, ortak sunucu yok ya da bot kapatılmış: DM açılamaz, engelliyken yazılamaz ve basılamaz

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:
>
> - Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur.
> - Etkileşime ilk yanıt 3 saniye içinde verilir.
> - Kimseye kendiliğinden DM atma; `dm_closed` ve `dm_window_closed`'ta dur, yeniden deneme.
> - DM'yi bot açamaz; `POST /users/@me/channels` çağrılmaz.
> - DM'den gelen olayda `guild_id` yoktur; etkileşimde `context: 1` ve `member` yerine `user` gelir.
> - DM'de yalnızca `contexts` içinde `1` olan ve `default_member_permissions` koymayan global komutlar çalışır; yetki isteyen komut `contexts: [0]` yapılır.
> - DM mesajları `DIRECT_MESSAGES` intent'iyle ve shard 0'a gelir; içerik `MESSAGE_CONTENT` olmadan da tamdır.

Kişi ne görür

Botun profilindeki veya üye menüsündeki **Mesaj gönder** ile konuşmayı aç. Başlıkta BOT etiketi görünür; sesli ve görüntülü arama yoktur. Üstteki sabit bilgi satırı 24 saatlik yanıt kuralını açıklar. Botu engellersen yazı alanının yerini **Engeli kaldır** düğmesi alır. `/` komutları ve **Uygulamalar** menüsü yalnız bu DM için sunulan komutları gösterir.

Sırada ne var

- Komutlar: `contexts: [0, 1]` ile DM destekleyen küresel komutlar
- Etkileşimler: DM içinde buton, menü ve form yanıtları
- Hatalar: `dm_closed`, `dm_window_closed` ve `cannot_dm` kodları

HTTP ile teslim

Etkileşimleri gateway yerine kendi https adresine aldırabilirsin: N'oluyo her komutu, basmayı ve form gönderimini imzalı bir `POST` isteğiyle adresine yollar, sen de aynı isteğin yanıtında cevap verirsin.

Hangi modu seçmeliyim

İhtiyacın	Gateway	HTTP
Bot sürekli çalışan bir program	Uygun	Olur, ama gerek yok
Sunucusuz bir fonksiyon, yalnız istek gelince çalışan bir sunucu	Olmaz: gateway açık bir bağlantı ister	Uygun
Mesaj, üye, tepki gibi olayları dinlemek	Uygun	Olmaz: HTTP'ye yalnızca etkileşimler gelir
Yalnızca komut, buton, menü ve form	Uygun	Uygun

Adres kaydettiğin an etkileşimler **yalnızca** adresine gelir, gateway'e `INTERACTION_CREATE` gelmez. Olayları da dinlemek istiyorsan gateway bağlantın açık kalır; ikisi birlikte çalışır. Adresi kaldırınca etkileşimler yeniden gateway'e döner.

En küçük çalışan örnek

Node.js 22 veya üstü yeter, ek paket gerekmez. Program `3000` portunda düz HTTP dinler; önüne https sağlayan bir ters vekil (Caddy, nginx gibi) koyarsın, çünkü adres https olmalıdır. `/selam` komutuna yanıt verir:

```
import { createServer } from "node:http";
import { createPublicKey, verify } from "node:crypto";

// Portaldaki "Herkese açık anahtar": 64 onaltılık karakter. Gizli değildir, ama koda gömme.
const PUBLIC_KEY = process.env.NOLUYO_PUBLIC_KEY;
const PORT = Number(process.env.PORT ?? 3000);

// Ham 32 baytlık Ed25519 anahtarını Node'un okuduğu SPKI biçimine saran sabit önek.
const SPKI_PREFIX = Buffer.from("302a300506032b6570032100", "hex");
const key = createPublicKey({ key: Buffer.concat([SPKI_PREFIX, Buffer.from(PUBLIC_KEY, "hex")]),
format: "der", type: "spki" });

const MAX_AGE = 5 * 60; // saniye
const seen = new Map(); // etkileşim kimliği -> geldiği an

function signatureOk(req, raw) {
  const sig = req.headers["x-signature-ed25519"];
  const ts = req.headers["x-signature-timestamp"];
  if (typeof sig !== "string" || !/^[0-9a-f]{128}$/.test(sig)) return false;
  if (typeof ts !== "string" || !/^\d+$/.test(ts)) return false;
  // Eski bir isteğin yeniden yollanması reddedilir.
  if (Math.abs(Date.now() / 1000 - Number(ts)) > MAX_AGE) return false;
  return verify(null, Buffer.concat([Buffer.from(ts), raw]), key, Buffer.from(sig, "hex"));
}

function answer(i) {
  if (i.type === 1) return { type: 1 }; // PING
  if (i.type === 2 && i.data.name === "selam") {
    const user = i.member?.user ?? i.user;
    return { type: 4, data: { content: `Selam ${user.display_name ?? user.username}!` } };
  }
  return { type: 4, data: { content: "Bunu tanımıyorum.", flags: 64 } };
}

createServer((req, res) => {
  if (req.method !== "POST") return void res.writeHead(405).end();
  const chunks = [];
  let size = 0;
  req.on("data", (c) => {
    size += c.length;
    if (size > 1 << 20) req.destroy();
    else chunks.push(c);
  });
  req.on("end", () => {
    const raw = Buffer.concat(chunks);
    // Önce imza, ham gövdeyle. JSON'a ancak imza tutunca çevrilir.
    if (!signatureOk(req, raw)) return void res.writeHead(401).end("invalid request signature");
    let i;
```

```
    try {
      i = JSON.parse(raw);
    } catch {
      return void res.writeHead(400).end();
    }
    // Aynı istek ikinci kez gelirse iş ikinci kez yapılmaz.
    const now = Date.now();
    for (const [id, at] of seen) if (now - at > 2 * MAX_AGE * 1000) seen.delete(id);
    if (seen.has(i.id)) return void res.writeHead(204).end();
    seen.set(i.id, now);
    res.writeHead(200, { "Content-Type": "application/json" }).end(JSON.stringify(answer(i)));
  });
}).listen(PORT, () => console.log(`Dinleniyor: ${PORT}`));
```

Çalıştırmak için:

```
NOLUYO_PUBLIC_KEY='portaldaki_acik_anahtar' node http-bot.mjs
```

`/selam` komutunu önce Komutlar sayfasındaki gibi kaydet. Sonra adresini portala yaz (aşağıda). Bu programın bot token'ına ihtiyacı yoktur; takip mesajı ya da `@original` düzenlemesi yapacaksan token'ı `process.env.BOT_TOKEN`'dan okursun.

Adres kuralları

Adres kaydedilirken ve her bağlantıda şunlar denetlenir. Uymayan adres `endpoint_not_allowed` alır ve ona hiç istek gitmez.

Kural	Neden
Yalnızca `https://`	İstek kişilerin verisini taşır; düz HTTP'de yolda okunabilir.
Port yok ya da `443`	Başka portlarda çoğunlukla iç servisler durur; N'oluyo bir port tarayıcısına dönmesin.
Kullanıcı adı ve şifre yok (`https://ad:sifre@...`)	Adresin içindeki şifre loglara düşer, ayrıca asıl host'u gizlemenin bilinen bir yoludur.
`#` yok	`#` sonrası sunucuya hiç gitmez; adresin ne olduğunu karıştırır.
Host bir ad olmalı, IP olmaz	IP adresi, ada bakan denetimleri atlatır. `https://203.0.113.5/` reddedilir.
Host yalnızca küçük-büyük İngilizce harf, rakam, `-` ve `.`, en az iki parça (`bot.ornek.com`)	Başka alfabelerden harfler bir adı başka bir ad gibi gösterebilir. `localhost` tek parçadır.
N'oluyo'nun kendi adresleri olmaz (`noluyo.net` ve alt alanları)	N'oluyo kendini çağırmasın.
En çok 2048 karakter	Daha uzunu bir adres değil, veri taşıma girişimidir.
Ad, her bağlantıda genel bir IP'ye çözülmeli	`10.x`, `192.168.x`, `127.0.0.1`, `::1` gibi yerel ve özel ağ adresleri reddedilir. Denetim DNS'ten sonra her bağlantıda yapılır, kayıttan sonra DNS'i değiştirmek işe yaramaz.
Yönlendirme izlenmez	`301`, `302` gibi bir yanıt başarısız sayılır. Adresin son adres olsun.

Host'u küçük harfe çevirip sonundaki `.` ve `:443`'ü atarak kaydederiz. Sorgu dizesi (`?anahtar=...`) serbesttir ve aynen saklanır: N'oluyo onu loglara yazmaz, adresi yalnızca sen görürsün. İstersen adresine kendi gizli değerini koyup isteğin senden geldiğini bir kez daha denetleyebilirsin; asıl kanıt yine imzadır.

Portalda adım adım

1. Programını çalıştır ve adresin dışarıdan https ile açıldığını denetle.
2. Geliştirici portalında uygulamanı aç, **Etkileşimler** bölümüne in.
3. **Herkese açık anahtar**'ı **Kopyala** ile kopyala ve programına `NOLUYO_PUBLIC_KEY` olarak ver. Anahtar her uygulamaya özeldir ve hiç değişmez.
4. **Etkileşim adresi** alanına adresini yaz, **Kaydet ve test et**'e bas.
5. N'oluyo adresine iki `PING` yollar (aşağıda). İkisi de doğru yanıtlanırsa adres kaydedilir ve **Kaydedildi** görürsün; o andan itibaren etkileşimler buraya gelir. Test geçmezse hiçbir şey kaydedilmez ve ne yapman gerektiği yazılır: örneğin imza denetimi yoksa "Geçersiz imzaya 401 dönmüyor. İmza denetimini ekle."
6. Gateway'e dönmek için **Adresi kaldır**'a bas.

Aynı işi otomasyonla yapmak için Uygulamanı güncelle rotasına `interactions_endpoint_url` gönder; `null` adresi kaldırır.

İmzayı doğrula

Her isteğin iki başlığı vardır:

Başlık	İçeriği
`X-Signature-Ed25519`	İmza, 128 onaltılık karakter
`X-Signature-Timestamp`	İsteğin yollandığı an, Unix saniyesi

İmza, Ed25519 ile, `X-Signature-Timestamp`'in metni ve **ham gövdenin** baytları art arda eklenerek atılır. Doğrulamak için:

1. Gövdeyi bayt olarak oku. **Ham gövdeyi JSON'a çevirmeden doğrula**: çevirip yeniden yazarsan boşluklar ve alan sırası değişir, imza tutmaz.
2. Zaman damgası şimdiden 5 dakikadan eskiyse ya da ilerideyse reddet. Böylece yakalanmış eski bir istek yeniden yollanamaz.
3. `timestamp + gövde` üzerinde imzayı uygulamanın açık anahtarıyla doğrula.
4. Herhangi biri tutmazsa `401` dön ve hiçbir şey yapma.

> **Uyarı:** Geçersiz imzaya `401` dön. Adres kaydedilirken N'oluyo bilerek yanlış imzalı bir istek yollar; `401` almazsa adresi kaydetmez. İmzayı denetlemeyen bir adres, adresini bilen herkesin botuna sahte komut yollamasına açıktır.

Python'da doğrulama, standart kütüphanede Ed25519 olmadığı için ek bir paket ister: `pip install cryptography`.

```
from cryptography.exceptions import InvalidSignature
from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PublicKey
import os, time

KEY = Ed25519PublicKey.from_public_bytes(bytes.fromhex(os.environ["NOLUYO_PUBLIC_KEY"]))

def signature_ok(headers, raw: bytes) -> bool:
    sig, ts = headers.get("X-Signature-Ed25519", ""), headers.get("X-Signature-Timestamp", "")
    if not ts.isdigit() or abs(time.time() - int(ts)) > 300:
        return False
    try:
        KEY.verify(bytes.fromhex(sig), ts.encode() + raw)
        return True
    except (InvalidSignature, ValueError):
        return False
```

PING ve iki istekli test

Adresi kaydederken N'oluyo şu gövdeyi iki kez yollar:

```
{ "id": "1127941200000000000", "application_id": "1127939310406860800", "type": 1, "token": "", "version": 1 }
```

İstek	Beklenen yanıt
Doğru imzalı `PING`	`200` ve gövdede `{"type":1}`
Yanlış imzalı `PING`	`401`

Neden iki istek: ilki adresin gerçekten bu botun programı olduğunu gösterir, çünkü yalnızca senin programın `PING`'e `{"type":1}` döner; böylece kimse N'oluyo'yu başkasının adresine yönlendiremez. İkincisi programın imzayı gerçekten denetlediğini gösterir. Test kişiyi bekletir: iki istek birlikte en çok 6 saniye kadar sürer ve kişi başına dakikada en çok 5 test yapılır.

Bot çalışırken de arada bir `PING` gelebilir; her zaman `{"type":1}` dön.

Yanıt ver

İstek, gateway'deki `INTERACTION_CREATE` olayının `d` alanıyla aynı nesnedir; alanlar Etkileşimler ve Komutlar sayfalarında. Yanıtın üç biçimi vardır:

Durum	Gövde	Anlamı
`200`	JSON: `{ "type": ..., "data": ... }`	Yanıtın kendisi, callback rotasına gönderdiğinle aynı biçim: `4`, `5`, `6`, `7`, otomatik tamamlamada `8`, formda `9`.
`202` ya da `204`	Boş	"Yanıtı `callback` rotasıyla vereceğim." Bu bir yanıt sayılmaz; 3 saniye kuralı sürer, `callback`'i yine zamanında göndermelisin.
Başka her şey		Başarısız: kişi botun yanıt vermediğini görür.

Kişinin isteği sen yanıt verene kadar açık bekler, en çok 3 saniye. `200` gövdesi tek bir JSON nesnesidir; tanımadığımız alanlar yok sayılır. Gövde, `callback` rotasına göndereceğin yanıtla aynı denetimlerden ve sınırlardan geçer. Tek fark: reddedilirse hata kodunu sen görmezsin, kişi botun yanıt vermediğini görür ve kod teslim günlüğüne yazılır.

`200` yanıtı dosya taşıyamaz, çünkü JSON'dur. Görselli yanıt için `5` dön, sonra görseli `@original` düzenlemesiyle ya da takip mesajıyla yolla. Yanıt en çok 64 KiB olabilir.

`200` ile mesaj yazan bir yanıt da gateway'den gelen botların sınırlarına sayılır: aynı yazma tavanı, aynı yayılma bütçesi, aynı `interaction` sınıfı. Yoğunlukta kişi `interaction_busy` görür; bu senin botunun suçu olmayabilir, tüm botların ortak tavanı dolu olabilir. HTTP, sınırlardan kaçış yolu değildir. Ayrıntı Hız sınırları sayfasında.

Her tür için gövde

Ne istiyorsun	`200` gövdesi
Komuta ya da basmaya yeni mesajla yanıt	`{"type":4,"data":{"content":"Selam!"}}`
Yalnızca kişi görsün	`{"type":4,"data":{"content":"Yalnızca sen görüyorsun.","flags":64}}`
Uzun iş: önce "düşünüyor"	`{"type":5}`
Butona sessiz onay	`{"type":6}`
Basılan mesajı güncelle	`{"type":7,"data":{"content":"Sayaç: 2"}}`
Otomatik tamamlama önerileri	`{"type":8,"data":{"choices":[{"name":"Ankara","value":"ankara"}]}}`
Form aç	`{"type":9,"data":{"custom_id":"sikayet","title":"Şikayet","components":[{"type":1,"components":[{"type":4,"custom_id":"konu","style":1,"label":"Konu"}]}]}}`

Hangi etkileşimin hangi türü aldığı Etkileşimler sayfasında. Gönderilen form da adresine yeni bir etkileşim (`type: 5`) olarak gelir.

Uzun iş: önce 5, sonra @original

3 saniyeye sığmayacak işte hemen `{"type":5}` dön, işi yanıttan sonra bitir ve mesajı bot token'ıyla doldur. Token'ın ömrü 15 dakikadır. Yukarıdaki `http-bot.mjs`'te `answer` işlevini bununla değiştir:

```
const API = "https://noluyo.net/api/v1";

// Senin uzun işin; burada 5 saniye bekliyor.
async function makeReport() {
  await new Promise((done) => setTimeout(done, 5000));
  return "Rapor hazır.";
}

async function finish(i) {
  const content = await makeReport();
  const res = await fetch(`${API}/interactions/${i.id}/${i.token}/messages/@original`, {
    method: "PATCH",
    headers: { Authorization: `Bot ${process.env.BOT_TOKEN}`, "Content-Type": "application/json" },
    body: JSON.stringify({ content }),
  });
  if (!res.ok) console.error("Doldurulamadı:", res.status, await res.text());
}

function answer(i) {
  if (i.type === 1) return { type: 1 }; // PING
  if (i.type === 2 && i.data.name === "rapor") {
    finish(i).catch(console.error); // yanıt gittikten sonra sürer
    return { type: 5 };
  }
  if (i.type === 2 && i.data.name === "selam") {
    const user = i.member?.user ?? i.user;
    return { type: 4, data: { content: `Selam ${user.display_name ?? user.username}!` } };
  }
  return { type: 4, data: { content: "Bunu tanımıyorum.", flags: 64 } };
}
```

Çalıştırmak için token'ı da ver:

```
NOLUYO_PUBLIC_KEY='portaldaki_acik_anahtar' BOT_TOKEN='bot_tokenin' node http-bot.mjs
```

202 ile, callback rotasından

İstersen yanıtı gövdede değil, gateway'deki gibi `callback` rotasıyla ver: hemen `202` dön, sonra 3 saniye içinde rotayı çağır. Bu, yanıtı başka bir süreçte hazırlayan botlar içindir; tek süreçte gövdeyle yanıt vermek daha kısadır.

```
// createServer içinde, imza tuttuktan sonra, answer() yerine:
res.writeHead(202).end();
fetch(`${API}/interactions/${i.id}/${i.token}/callback`, {
  method: "POST",
  headers: { Authorization: `Bot ${process.env.BOT_TOKEN}`, "Content-Type": "application/json" },
  body: JSON.stringify({ type: 4, data: { content: "Selam!" } }),
}).catch(console.error);
```

Kişi ne görür

Durum	Kişinin isteği	Kişi ne görür
`200` ve yanıt geçti	`200`	Botun yanıtı
`202`, `204`	`200`	`callback` gelene dek bekler; 3 saniyede gelmezse "Bot yanıt vermedi"
Yanıt gelmedi, reddedildi ya da devre kesici açık	`504 application_no_response`	"Bot zamanında yanıt vermedi."
Adresine aynı anda 16 teslim gidiyor ya da tüm botların tavanı dolu	`429 interaction_busy`	"Bot şu an yoğun. Biraz sonra yeniden dene."

Kişi beklerken sayfayı kapatsa da yanıtın işlenir.

Zaman, tekrar ve tekilleştirme

- **3 saniye.** Bağlanma, TLS ve yanıtın tamamı 3 saniyeyi geçerse istek başarısız sayılır. Uzun iş yapacaksan önce `5` dön, işi sonra bitir.
- **Tekrar deneme yok.** Kişi beklediği için her etkileşim adresine en çok bir kez yollanır. Başarısız olursa kişi hatayı hemen görür ve isterse yeniden dener; bu yeni bir etkileşimdir.
- **Kimliğe göre tekilleştir.** İmzalı bir isteği yakalayan biri onu 5 dakika içinde yeniden yollayabilir. Gördüğün `id`'leri en az 10 dakika tut ve aynısı gelirse işi ikinci kez yapma.
- **Aynı anda en çok 16 istek.** Bir uygulamanın adresine aynı anda en çok 16 teslim gider; dolarsa kişi hemen `interaction_busy` görür.

Devre kesici ve teslim günlüğü

Adresin art arda 10 kez başarısız olursa N'oluyo onu 30 saniye aramaz (portal günlüğün üstünde "Adres yanıt vermediği için teslim 30 saniye durdu." yazar); bu sürede kişiler botun yanıt vermediğini hemen görür. Sonra 30 saniyede bir tek bir deneme yapar, biri başarılı olunca normale döner. Yalnızca adresinin kendi hataları sayılır: `timeout`, `dns`, `connect`, `tls`, `redirect`, `status`, `too_large`, `not_allowed` ve `bad_json`. Adresinin yanıt verdiği ama yanıtın uygulanırken reddedildiği teslimler (`busy`, `error` ve hata kodları, örneğin kişi DM'yi kapattığı için `dm_closed`) günlükte görünür, ama bu sayıma girmez ve sayımı sıfırlamaz: onlar adresinin değil, kişinin ya da tavanın sonucudur. Yeni bir adres kaydedince sayım ve günlük sıfırlanır.

Portalda **Etkileşimler** bölümündeki **Teslim günlüğü** (**Yenile** ile tazelenir), adresine yapılan son 50 teslimi en yenisi önce gösterir. Günlük bellekte tutulur, N'oluyo yeniden başlarsa boşalır. Kişi ve içerik yoktur:

Sütun	Anlamı
Zaman	Teslimin anı
Tür	Etkileşimin türü: `2` komut, `3` bileşen, `4` otomatik tamamlama, `5` form
Durum	Adresinin HTTP durumu, hiç yanıt gelmediyse `0`
Süre	Milisaniye
Sonuç	Aşağıdaki tablo

Sonuç	Ne oldu	Ne yap
`ok`	Başarılı	
`timeout`	3 saniyede yanıt gelmedi	İşi kısalt ya da önce `5` dön.
`dns`	Ad çözülemedi	DNS kaydını denetle.
`connect`	Bağlanılamadı	Sunucun ve 443 portu açık mı bak.
`tls`	TLS kurulamadı	Sertifikan geçerli ve adla eşleşiyor mu bak.
`redirect`	`3xx` döndü	Son adresi kaydet.
`status`	`200`, `202` ve `204` dışında bir durum	Programının loguna bak; geçersiz imzaya `401` doğrudur, geçerliye değil.
`too_large`	Yanıt 64 KiB'tan büyük	Yanıtı küçült.
`not_allowed`	Ad yerel ya da özel bir IP'ye çözüldü	Adın genel bir IP göstermeli.
`bad_json`	`200` geldi ama gövde tek bir JSON nesnesi değil	Gövdeyi `JSON.stringify` ile yaz, ardına bir şey ekleme.
`busy`	Yanıt geldi ama tüm botların ortak tavanı doluydu; kişi `interaction_busy` gördü	Bir şey yapma; sürerse Hız sınırları sayfasına bak.
`error`	Bizim tarafımızda bir hata	Sürerse bize bildir.
Bir hata kodu	Yanıt geldi ama reddedildi; `callback` rotası da aynı kodu dönerdi: `interaction_response_invalid`, `validation_failed`, `dm_closed`, `application_unavailable` gibi	Kodun anlamı Hatalar sayfasında.

Aynı bilgiyi Teslim günlüğünü oku rotası da verir; `open: true` devre kesicinin açık olduğunu söyler.

Yerelde nasıl denersin

`https://localhost:3000` kaydedilemez: N'oluyo'nun sunucuları senin bilgisayarına ulaşamaz, yerel adresler de zaten reddedilir. Denemek için şu yollardan birini seç:

- Bilgisayarındaki `3000` portunu genel bir https adrese açan bir tünel servisi kullan ve onun verdiği adresi kaydet. İşin bitince adresi kaldır.
- Programı ucuz bir sunucuya ya da sunucusuz bir fonksiyona koy.
- Kayıttan önce imza kodunu kendi testinle dene: yanlış imzaya `401`, doğru `PING`'e `{"type":1}` dönmeli.

Limitler

Anahtar	Değer	Saniye	Anlamı
`verify`	5	60	Kişi başına adres testi
`timeout`	3		Bir teslimin tamamı
`response`	65536		Yanıtın en büyük boyutu, bayt
`in_flight`	16		Bir uygulamaya aynı anda giden teslim
`trip`	10	30	Bu kadar ardışık hatadan sonra adres bu kadar saniye aranmaz
`freshness`	300		Kabul edeceğin en eski zaman damgası, saniye
`url`	2048		Adresin en çok karakteri

Hatalar

Adres kaydedilirken dönebilecek hatalar. Yalnızca hatanın türü söylenir; adresinin döndüğü durum ya da metin söylenmez.

Kod	Ne oldu	Ne yap
`endpoint_not_allowed`	Adres kurallara uymuyor ya da ad yerel bir IP'ye çözülüyor	Yukarıdaki tabloya bak.
`endpoint_unreachable`	Adrese ulaşılamadı ya da 3 saniyede yanıt gelmedi	Programın açık mı, https ve sertifika doğru mu bak.
`endpoint_bad_response`	Doğru imzalı `PING`'e `200 {"type":1}` dönmedi	`type: 1` gelince `{"type":1}` dön; yönlendirme yapma.
`endpoint_no_signature_check`	Yanlış imzalı isteğe `401` dönmedi	İmzayı her istekte doğrula, tutmazsa `401` dön.
`rate_limited`	Dakikada 5 testi aştın	`retry_after` kadar bekle.

Bu sayfayı yapay zekaya verirsen

İpucu: Bu sayfayı bir yapay zeka asistanına verip kod yazdırıyorsan şu kuralları da ekle:

- Token koda yazılmaz, `process.env.BOT_TOKEN`'dan okunur; açık anahtar `process.env.NOLUYO_PUBLIC_KEY`'den.
- Etkileşime ilk yanıt 3 saniye içinde verilir.
- Her istekte imza, JSON'a çevirmeden önce ham gövde üzerinde doğrulanır: `X-Signature-Timestamp` + gövde, Ed25519; tutmazsa `401`.
- 5 dakikadan eski zaman damgası reddedilir; aynı `id` ikinci kez işlenmez.
- `type: 1` (`PING`) gelince `200` ve `{"type":1}` dönülür.
- Yanıt `200` ile JSON `{type, data}`; dosya yoksa doğrudan, varsa önce `5` sonra takip mesajı. `202` ya da `204` yanıt sayılmaz.
- Uzun iş için önce `{"type":5}` dönülür, iş yanıttan sonra bitirilir ve `PATCH .../messages/@original` ile doldurulur.
- Yönlendirme yapılmaz; adres https ve 443'tür.

Sırada ne var

- Etkileşimler: HTTP yanıt gövdesinde ya da callback rotasında dönen yanıt türleri
- JavaScript kütüphanesi ve Python kütüphanesi: yerleşik `verifyInteraction` / `verify_interaction` yardımcıları
- Güvenlik: imza doğrulama ve gizli anahtar yönetimi

N'oluyo ile giriş

N'oluyo ile giriş (OAuth2), bir kişinin kendi hesabıyla senin web sitene, masaüstü programına ya da mobil uygulamana açık rızasıyla giriş yapmasını sağlar. Şifresini seninle paylaşmaz; N'oluyo ekranında hangi bilgilerin paylaşılacağını görür, onaylar ve senin uygulamana tek kullanımlık bir kodla döner.

Bu nedir ve ne zaman gerekir

Geliştirici portalındaki tek bir uygulama hem bot hem de **N'oluyo ile giriş** barındırabilir; ama ikisi tamamen farklı işler yapar:

- **Bir sunucuya bot eklemek istiyorsan** bu sayfaya ihtiyacın yok. Bot ekleme akışı `/bot-ekle/<uygulama_id>` ekranından yürür; ayrıntısı Bot uygulamaları rehberinde.
- **Kendi sitene veya uygulamana kullanıcı girişi koymak istiyorsan** doğru yer burasıdır. Bot kullanıcısı oluşturman bile gerekmez; yalnız OAuth2 ayarlarını açman yeterlidir.

Kişinin gördüğü izin ekranı

Kullanıcıyı `https://noluyo.net/oauth2/authorize` adresine yönlendirdiğinde karşısına N'oluyo'nun izin ekranı çıkar. Bu ekranda:

- Uygulamanın adı, avatarı, doğrulanmış rozeti (doğrulanmamışsa açık uyarı etiketi), gizlilik politikası (`privacy_url`) ve kullanım koşulları (`terms_url`) bağlantıları yer alır.
- İstenen her kapsam (`identify` , `email` , `guilds`) sade cümlelerle listelenir.
- Ekranın altında **"Uygulama şunları asla göremez"** kutusu sabit olarak gösterilir: şifren, iki aşamalı doğrulama kodların, özel mesajların, sunucu mesajların, sesli sohbetlerin ve ödeme bilgilerin paylaşılmaz; uygulama senin adına mesaj atamaz veya sunucuya katılamaz.

Misafir (kayıtsız) hesaplar OAuth2 izni veremez (`403 guest_not_allowed`). Doğrulanmamış bir uygulamaya en çok **100 farklı kullanıcı** giriş yapabilir; 101. kişi `403 oauth_user_cap` uyarısı alır. Sınırı kaldırmak için Doğrulama başvurusu yapılır.

> **Not:** Kullanıcı dilediği an **Ayarlar → Bağlı uygulamalar** ekranından uygulamanın erişimini tek tıkla kaldırabilir.

Portal ayarları ve yönlendirme adresleri

`/gelistirici/uygulamalar` sayfasında uygulamanı açıp **N'oluyo ile Giriş (OAuth2)** bölümünü yapılandır:

1. **Yönlendirme adresleri (`redirect_uris`):** En çok **10** adres eklenebilir (her biri en çok 512 karakter). Yetkilendirme isteğindeki `redirect_uri` parametresi buradaki kayıtlardan biriyle **birebir** eşleşmelidir. Joker karakter (`*`), `#fragment` , `kullanici@` bilgisi veya `noluyo.net` / `*.noluyo.net` alan adları kabul edilmez. İzin verilen üç biçim vardır:

- **Web siteleri:** `https://` ile başlayan tam adres (`https://ornek.site/giris/geri-donus`).

- **Yerel geliştirme ve masaüstü geri dönüşü:** `http://localhost`, `http://127.0.0.1` veya `http://[::1]`. Port numarası çalışma anında esnektir: portala `http://127.0.0.1/geri-donus` kaydettiğinde çalışma anında `http://127.0.0.1:5173/geri-donus` veya rastgele seçilmiş yerel bir port kabul edilir.
- **Mobil ve masaüstü özel şema:** Nokta içeren ters alan adı (`com.ornek.uygulama:/oauth/geri-donus`; `net.noluyo` ile başlayamaz).

1. **İstemci türü (`oauth_public`):**

- **Gizli istemci (`oauth_public: false`, varsayılan):** Kendi arka uç sunucusu olan web siteleri içindir. Portaldan **Secret oluştur / yenile** düğmesiyle (`POST /api/v1/applications/{application_id}/oauth-secret`) bir `client_secret` üret ve sunucundaki `.env` dosyasında sakla. Secret yalnızca üretildiği anda tek sefer gösterilir.
- **Açık istemci (`oauth_public: true`):** Arka ucu olmayan tek sayfa uygulamalar (SPA), masaüstü uygulamaları ve mobil uygulamalar içindir. Kodu kullanıcının cihazında çalıştığı için `client_secret` kullanılmaz; güvenliği PKCE ve tam yönlendirme adresi sağlar.

1. **Yasal bağlantılar:** İzin ekranında görünmesi için `privacy_url` ve `terms_url` alanlarına `https://` ile başlayan gizlilik politikası ve kullanım koşulları adreslerini ekle.

Yetkilendirme adresi ve parametreler

Kullanıcıyı tarayıcıda şu adrese yönlendirirsin:

```
https://noluyo.net/oauth2/authorize?
client_id=12345678901234567&redirect_uri=https%3A%2F%2Fornek.site%2Fgiris%2Fgeri-
donus&response_type=code&scope=identify+email+guilds&state=RastgeleDurumDegeri&code_challenge=PKC
EOzetDegeri&code_challenge_method=S256
```

Her parametrenin tam karşılığı:

Parametre	Zorunlu	Açıklama
`client_id`	evet	Uygulamanın snowflake kimliği (`application.id`)
`redirect_uri`	evet	Portalda kayıtlı yönlendirme adreslerinden biri (URL kodlu)
`response_type`	evet	Her zaman `code` olmalıdır
`scope`	evet	Boşlukla ayrılmış kapsam listesi; `identify` zorunludur (`identify`, `identify email`, `identify email guilds`)
`state`	evet	Oturumuna bağladığın rastgele tahmin edilemez metin (en az 16 bayt rastgele veri önerilir)
`code_challenge`	evet	`code_verifier` değerinin SHA-256 özeti (base64url, `=` dolgusuz)
`code_challenge_method`	evet	Her zaman `S256` olmalıdır; `plain` reddedilir

> **İpucu:** Geliştirici portalındaki **N'oluyo ile Giriş** bölümünde yer alan bağlantı oluşturucu, seçtiğin kapsamlara ve yönlendirme adresine göre hazır test adresini tek tıkla üretir.

PKCE ve state ne işe yarar

Hem gizli hem de açık istemcilerde **PKCE (`S256`)** ve **`state`** zorunludur:

- **`code_verifier` ve `code_challenge` (PKCE):** Kullanıcıyı yönlendirmeden hemen önce 43–128 karakter uzunluğunda rastgele bir metin (`code_verifier`) üretip kullanıcının oturumunda saklarsın. Bunun SHA-256 özetinin base64url karşılığını (`code_challenge`) yetkilendirme adresine koyarsın. Kullanıcı kodla döndüğünde asıl `code_verifier` değerini `/api/v1/oauth2/token` ucuna gönderirsin. Yolda kodu ele geçiren biri `code_verifier` değerini bilmediği için token alamaz.
- **`state`:** Başka birinin kendi yetkilendirme kodunu senin kullanıcının tarayıcısına enjekte etmesini (CSRF) engeller. Yönlendirmeden önce ürettiğin rastgele `state` değerini HTTP-only çerezde ya da sunucu oturumunda sakla; geri dönüşte gelen `state` birebir eşleşmiyorsa işlemi hemen reddet.

Beş adımda giriş akışı

Tam çalışan web sitesi örneklerini depodaki `sdks/examples/js/oauth-site` ve `sdks/examples/python/oauth_site` klasörlerinde bulabilirsin.

1. PKCE değerlerini üret ve kullanıcıyı yönlendir

```
# PKCE ve state üretimi (OpenSSL ile örnek):
VERIFIER=$(openssl rand -base64 32 | tr '+/' '-_' | tr -d '=')
CHALLENGE=$(printf '%s' "$VERIFIER" | openssl dgst -sha256 -binary | openssl base64 -A | tr '+/' '-_' | tr -d '=')
STATE=$(openssl rand -hex 16)

echo "Oturuma kaydet -> verifier=$VERIFIER state=$STATE"
echo "Tarayıcıda aç -> https://noluyo.net/oauth2/authorize?client_id=$CLIENT_ID&redirect_uri=http%3A%2F%2F127.0.0.1%3A3000%2Fcb&response_type=code&scope=identify+email&state=$STATE&code_challenge=$CHALLENGE&code_challenge_method=S256"
```

```
import { createHash, randomBytes } from "node:crypto";

const codeVerifier = randomBytes(32).toString("base64url");
const codeChallenge = createHash("sha256").update(codeVerifier).digest("base64url");
const state = randomBytes(16).toString("hex");

// codeVerifier ve state değerlerini kullanıcının sunucu oturumuna kaydet.
const url = new URL("https://noluyo.net/oauth2/authorize");
url.searchParams.set("client_id", process.env.CLIENT_ID);
url.searchParams.set("redirect_uri", "http://127.0.0.1:3000/cb");
url.searchParams.set("response_type", "code");
url.searchParams.set("scope", "identify email");
url.searchParams.set("state", state);
url.searchParams.set("code_challenge", codeChallenge);
url.searchParams.set("code_challenge_method", "S256");
```

```
import base64, hashlib, os, secrets
from urllib.parse import urlencode

code_verifier = secrets.token_urlsafe(32)
digest = hashlib.sha256(code_verifier.encode("ascii")).digest()
code_challenge = base64.urlsafe_b64encode(digest).rstrip(b"=").decode("ascii")
state = secrets.token_hex(16)

# code_verifier ve state değerlerini kullanıcının sunucu oturumuna kaydet.
params = urlencode({
    "client_id": os.environ["CLIENT_ID"],
    "redirect_uri": "http://127.0.0.1:3000/cb",
    "response_type": "code",
    "scope": "identify email",
    "state": state,
    "code_challenge": code_challenge,
    "code_challenge_method": "S256",
})
authorize_url = f"https://noluyo.net/oauth2/authorize?{params}"
```

2. Geri dönüşü karşıla ve state kontrolü yap

Kullanıcı **İzin ver** düğmesine bastığında tarayıcı `redirect_uri?code=...&state=...` adresine, **Vazgeç** düğmesine bastığında ise `redirect_uri?error=access_denied&state=...` adresine döner. Yetkilendirme kodu **2 dakika** geçerlidir ve **tek kullanımlıktır**.

```
# Tarayıcının döndüğü adres sorgusundan gelen değerler:
# ?code=nly_ac_...&state=...
# Önce gelen state değerinin 1. adımdaki $STATE ile birebir aynı olduğunu doğrula.
```

```
import { timingSafeEqual } from "node:crypto";

function checkCallback(query, session) {
  const got = Buffer.from(String(query.state ?? ""));
  const expected = Buffer.from(String(session.state ?? ""));
  if (got.length === 0 || got.length !== expected.length || !timingSafeEqual(got, expected)) {
    throw new Error("Geçersiz state");
  }
  if (query.error) {
    throw new Error(`Kullanıcı vazgeçti: ${query.error}`);
  }
  return String(query.code);
}
```

```
import hmac

def check_callback(query: dict[str, str], session: dict[str, str]) -> str:
    got = query.get("state", "")
    expected = session.get("state", "")
    if not got or not hmac.compare_digest(got, expected):
        raise ValueError("Geçersiz state")
    if "error" in query:
        raise ValueError(f"Kullanıcı vazgeçti: {query['error']}")
    return query["code"]
```

3. Kodu access token ile değiştir

`POST https://noluyo.net/api/v1/oauth2/token` ucuna **`application/x-www-form-urlencoded`** gövde gönderilir (en çok 8 KiB; JSON gövde gönderirsen `400 invalid_request` alırsın). Gizli istemciler `client_secret` alanını form içinde ya da HTTP Basic başlığında iletir; açık istemciler (`oauth_public: true`) `client_secret` göndermez. Hız sınırı `client_id` ve IP başına dakikada **20** istektir.

```
curl -X POST "https://noluyo.net/api/v1/oauth2/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=authorization_code" \
  --data-urlencode "code=$CODE" \
  --data-urlencode "redirect_uri=http://127.0.0.1:3000/cb" \
  --data-urlencode "client_id=$CLIENT_ID" \
  --data-urlencode "client_secret=$CLIENT_SECRET" \
  --data-urlencode "code_verifier=$VERIFIER"
```

```
const body = new URLSearchParams({
  grant_type: "authorization_code",
  code,
  redirect_uri: "http://127.0.0.1:3000/cb",
  client_id: process.env.CLIENT_ID,
  client_secret: process.env.CLIENT_SECRET, // Açık istemcide bu satırı çıkar
  code_verifier: session.codeVerifier,
});

const tokenRes = await fetch("https://noluyo.net/api/v1/oauth2/token", {
  method: "POST",
  headers: { "Content-Type": "application/x-www-form-urlencoded" },
  body,
});
const tokens = await tokenRes.json();
if (!tokenRes.ok) throw new Error(`${tokens.error}: ${tokens.error_description}`);
// tokens: { access_token, token_type: "Bearer", expires_in: 3600, refresh_token, scope }
```

```
import os, aiohttp

async with aiohttp.ClientSession() as http:
    async with http.post(
        "https://noluyo.net/api/v1/oauth2/token",
        data={
            "grant_type": "authorization_code",
            "code": code,
            "redirect_uri": "http://127.0.0.1:3000/cb",
            "client_id": os.environ["CLIENT_ID"],
            "client_secret": os.environ["CLIENT_SECRET"],  # Açık istemcide çıkar
            "code_verifier": session["code_verifier"],
        },
    ) as resp:
        tokens = await resp.json()
        if resp.status != 200:
            raise RuntimeError(f"{tokens['error']}: {tokens.get('error_description')}")
```

4. Kullanıcı profilini ve sunucu listesini oku

`access_token` **1 saat** (`expires_in: 3600`) geçerlidir ve `Authorization: Bearer <access_token>` başlığıyla kullanılır (`Bot` önekiyle değil!). Okuma uçlarının hız sınırı 5 saniyede **30** istektir (`oauth_read`).

```
curl "https://noluyo.net/api/v1/oauth2/@me" \
  -H "Authorization: Bearer $ACCESS_TOKEN"

# "guilds" kapsamı istendiyse kullanıcının üye olduğu sunucular:
curl "https://noluyo.net/api/v1/oauth2/@me/guilds" \
  -H "Authorization: Bearer $ACCESS_TOKEN"
```

```
const meRes = await fetch("https://noluyo.net/api/v1/oauth2/@me", {
  headers: { Authorization: `Bearer ${tokens.access_token}` },
});
const me = await meRes.json();
if (!meRes.ok) throw new Error(me.code);
// me.user.id -> kendi veritabanındaki tekil anahtar!
```

```
async with aiohttp.ClientSession() as http:
    async with http.get(
        "https://noluyo.net/api/v1/oauth2/@me",
        headers={"Authorization": f"Bearer {tokens['access_token']}"},
    ) as resp:
        me = await resp.json()
        if resp.status != 200:
            raise RuntimeError(me["code"])
        user_id = me["user"]["id"]
```

5. Token yenile (rotasyon) ve çıkışta iptal et

`refresh_token` **30 gün** geçerlidir ve **her yenilemede döner (rotate olur)**: yanıt sana yeni bir `access_token` ve yeni bir `refresh_token` verir; hemen ikisini birden veritabanında güncelle. Eşzamanlı ağ isteklerinin birbirini kırmaması için eski `refresh_token` **30 saniyelik** bir geçiş penceresi boyunca geçerli kalır; 30 saniye geçtikten sonra eski bir `refresh_token` tekrar gönderilirse çalıntı şüphesiyle o iznin **tüm** açık token'ları anında iptal edilir ve `400 invalid_grant` döner.

Kullanıcı senin sitenden çıkış yaptığında ya da hesabını ayırdığında `POST /api/v1/oauth2/revoke` çağrısıyla token'ı iptal et (her zaman `200` döner):

```
# Yenileme:
curl -X POST "https://noluyo.net/api/v1/oauth2/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "refresh_token=$REFRESH_TOKEN" \
  --data-urlencode "client_id=$CLIENT_ID" \
  --data-urlencode "client_secret=$CLIENT_SECRET"

# İptal etme:
curl -X POST "https://noluyo.net/api/v1/oauth2/revoke" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "token=$REFRESH_TOKEN" \
  --data-urlencode "client_id=$CLIENT_ID" \
  --data-urlencode "client_secret=$CLIENT_SECRET"
```

```
// Yenileme:
const refreshed = await fetch("https://noluyo.net/api/v1/oauth2/token", {
  method: "POST",
  headers: { "Content-Type": "application/x-www-form-urlencoded" },
  body: new URLSearchParams({
    grant_type: "refresh_token",
    refresh_token: storedRefreshToken,
    client_id: process.env.CLIENT_ID,
    client_secret: process.env.CLIENT_SECRET,
  }),
}).then((r) => r.json());

// Çıkışta iptal:
await fetch("https://noluyo.net/api/v1/oauth2/revoke", {
  method: "POST",
  headers: { "Content-Type": "application/x-www-form-urlencoded" },
  body: new URLSearchParams({
    token: refreshed.refresh_token,
    client_id: process.env.CLIENT_ID,
    client_secret: process.env.CLIENT_SECRET,
  }),
});
```

```
async with aiohttp.ClientSession() as http:
    # Yenileme:
    async with http.post(
        "https://noluyo.net/api/v1/oauth2/token",
        data={
            "grant_type": "refresh_token",
            "refresh_token": stored_refresh_token,
            "client_id": os.environ["CLIENT_ID"],
            "client_secret": os.environ["CLIENT_SECRET"],
        },
    ) as resp:
        refreshed = await resp.json()

    # Çıkışta iptal:
    await http.post(
        "https://noluyo.net/api/v1/oauth2/revoke",
        data={
            "token": refreshed["refresh_token"],
            "client_id": os.environ["CLIENT_ID"],
            "client_secret": os.environ["CLIENT_SECRET"],
        },
    )
```

Kapsamlar ve dönen alanlar

N'oluyo kasıtlı olarak yalnızca üç salt-okunur kapsam sunar; mesaj okuma, mesaj yazma ya da sunucuya sokma kapsamı yoktur:

Kapsam	Zorunlu	Döndürdüğü uç ve alanlar
`identify`	evet	`GET /api/v1/oauth2/@me` içinde `application: {id, name}`, `scopes`, `expires_at` ve `user: {id, username, display_name, avatar}`
`email`	hayır	`GET /api/v1/oauth2/@me` içindeki `user` nesnesine `email` (`string` veya `null`) ve `email_verified` (`boolean`) ekler
`guilds`	hayır	`GET /api/v1/oauth2/@me/guilds` ucunu açar; her sunucu için `{id, name, icon, owner, permissions}` dizisi döndürür (`permissions` dizgi olarak izin bit maskesidir)

> **Uyarı:** Kendi veritabanında kullanıcıyı **her zaman `user.id` (snowflake)** ile eşleştir; asla `username` ya da `email` alanını birincil anahtar yapma. Kullanıcı adını veya e-postasını değiştirebilir. Ayrıca `email` kapsamı verilmiş olsa bile kullanıcının e-postası doğrulanmamışsa (`email_verified: false`), `user.email` alanı **`null`** gelir! E-postası doğrulanmamış iki hesabı otomatik birleştirme.

Açık istemciler ve CORS

Tarayıcıda çalışan tek sayfa uygulamalar (SPA), Electron/Tauri masaüstü programları ve mobil uygulamalar kaynak kodunda gizli anahtar saklayamaz. Bu uygulamalarda:

1. Portalda **Açık istemci (`oauth_public`)** seçeneğini aç ve istemci koduna hiçbir zaman `client_secret` gömme.
2. `/api/v1/oauth2/token`, `/api/v1/oauth2/revoke`, `/api/v1/oauth2/@me` ve `/api/v1/oauth2/@me/guilds` uçları `Access-Control-Allow-Origin: *` (çerezsiz CORS) başlığıyla tarayıcıdan doğrudan çağrılabilir.
3. Tarayıcı uygulamasında `access_token` değerini `localStorage` yerine bellek içinde tut; masaüstü ve mobilde işletim sisteminin güvenli anahtar zincirini (Keychain / Keystore / Credential Locker) kullan.

Hata tablosu

`/api/v1/oauth2/token` ve `/revoke` uçları RFC 6749 biçiminde `{"error": "...", "error_description": "..."}` döndürür; diğer tüm uçlar standart `{"code": "...", "message": "..."}` zarfını kullanır:

Durum	Kod / `error`	Neden ve çözüm
400	`invalid_request`	Form kodlaması (`application/x-www-form-urlencoded`) kullanılmadı, PKCE `S256` eksik ya da zorunlu alan boş.
401	`invalid_client`	`client_id` veya `client_secret` yanlış, ya da gizli istemcide secret gönderilmedi.
400	`invalid_grant`	Yetkilendirme kodunun 2 dakikalık süresi doldu, kod ikinci kez kullanıldı, `redirect_uri` veya `code_verifier` eşleşmedi ya da 30 saniyeyi geçmiş eski bir `refresh_token` tekrar gönderildi. Kullanıcıdan yeniden giriş iste.
400	`unsupported_grant_type`	`grant_type` alanı `authorization_code` veya `refresh_token` dışında bir değer.
400	`invalid_scope`	İstenen kapsam tanınmıyor veya `identify` eklenmemiş.
400	`oauth_unknown_client`	Yetkilendirme ekranındaki `client_id` bulunamadı veya uygulama durdurulmuş.
400	`oauth_bad_redirect`	`redirect_uri` portalda kayıtlı adreslerle eşleşmiyor. Portaldaki listeyi kontrol et.
400	`oauth_bad_request`	`response_type=code` , `state` , `code_challenge` veya `code_challenge_method=S256` eksik/geçersiz.
403	`oauth_user_cap`	Doğrulanmamış uygulamanın 100 kullanıcı sınırı doldu. Doğrulama başvurusu yap.
403	`guest_not_allowed`	Misafir hesapla giriş denendi; kullanıcının kalıcı hesap açması gerekir.
401	`invalid_oauth_token`	`access_token` süresi doldu, iptal edildi ya da kullanıcı şifresini değiştirdi. `refresh_token` ile yenile; o da reddedilirse yeniden giriş ekranına yönlendir.
403	`insufficient_scope`	`/oauth2/@me/guilds` çağrıldı ama izin alınırken `guilds` kapsamı istenmemişti.

Oturum iptali, şifre değişimi ve hesap silme

Token yaşam döngüsü kullanıcının hesap güvenliği hareketlerini anında izler:

- **Kullanıcı erişimi kaldırdığında:** Kullanıcı **Ayarlar → Bağlı uygulamalar** ekranından uygulamanı kaldırdığında tüm `access_token` ve `refresh_token` kayıtları anında silinir; bir sonraki `@me` çağrısı `401 invalid_oauth_token` döndürür.
- **Şifre değişimi veya sıfırlama:** Kullanıcı şifresini değiştirdiğinde ya da sıfırladığında açık `access_token` ve `refresh_token` değerleri iptal olur (uygulama Bağlı uygulamalar listesinde kalır; kullanıcı sitene geldiğinde yeniden giriş yapar). "Diğer cihazlardan çık" düğmesi ve iki aşamalı doğrulama değişiklikleri OAuth2 token'larını düşürmez.

- **Hesap veya uygulama kapanışı:** Kullanıcı N'oluyo hesabını sildiğinde/dondurduğunda ya da sen uygulamayı sildiğinde/durdurduğunda tüm token'lar derhal geçersiz kalır.

Güvenlik listesi ve yasal yükümlülüklerin

Yayına çıkmadan önce şu maddeleri doğrula:

- `client_secret` hiçbir zaman tarayıcı koduna, mobil pakete veya Git deposuna girmez; yalnız arka uç `.env` dosyasında durur.
- Her giriş isteğinde yeni bir `state` ve `code_verifier` üretilir ve geri dönüşte `state` doğrulanmadan kod takas edilmez.
- Portalda yalnızca gerçekten kullandığın tam `redirect_uri` adresleri kayıtlıdır.
- Kullanıcı kimliği yalnızca `user.id` üzerinden tutulur ve `user.email === null` durumu hatasız ele alınır.
- **Yasal sorumluluk:** N'oluyo ile giriş üzerinden kendi sistemine çektiğin veriler için kendi kullanıcılarına karşı bağımsız veri sorumlusun. Portalda doğru bir gizlilik politikası (`privacy_url`) yayınlamalı ve sitende hesabını silen kişinin senden tutulmasını istemediği OAuth2 token'larını `/oauth2/revoke` ile iptal edip verilerini silmelisin.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bir yapay zeka aracına "N'oluyo ile giriş" özelliği yazdırırken bu sayfayı ver ve şu kuralları özellikle vurgula:
>
> - Yetkilendirme ekranı `https://noluyo.net/oauth2/authorize` , API uçları ise `https://noluyo.net/api/v1/oauth2/token` , `/revoke` , `/@me` ve `/@me/guilds` adresindedir.
> - PKCE (`code_challenge_method=S256`) ve `state` hem gizli hem açık istemcilerde zorunludur.
> - `/api/v1/oauth2/token` ve `/revoke` uçları JSON değil, yalnızca `application/x-www-form-urlencoded` kabul eder.
> - `@me` çağrılarında `Authorization: Bearer <access_token>` kullanılır (`Bot` değil).
> - Kullanıcılar veritabanında `user.id` ile tutulur; `user.email` alanı `null` dönebilir.

Sırada ne var

- Doğrulama: 100 kullanıcı sınırını kaldırmak için doğrulama başvurusu
- Güvenlik: secret ve token saklama kuralları
- Hatalar: tüm HTTP durum ve hata kodlarının tam listesi

Gateway

REST'te programın sorar, N'oluyo yanıtlar. Yeni bir mesaj yazıldığını öğrenmek için sormak gerekseydi kanalı durmadan okuman gerekirdi. Gateway bunu tersine çevirir: programın açık duran tek bir WebSocket bağlantısı kurar, sunucularda bir şey olduğunda haber kendiliğinden gelir.

Botun iki yolu birlikte kullanır: **gateway'den duyar, REST ile iş yapar.** Mesaj göndermek, silmek ya da rol vermek yine REST isteğidir.

Bu sayfa bir bağlantının başından sonuna ne olduğunu sırayla anlatır. Sayfanın sonunda kopyalayıp kullanabileceğin bir istemci var. Her olayın alanları ve örnekleri Olaylar sayfasında.

> **Not:** Örnekler düz `WebSocket` kullanır. Node.js 22 ve üstünde `WebSocket` hazırdır, ek paket gerekmez. Sürümünü `node --version` ile görebilirsin.

Akışa kısa bakış

1. `GET /gateway/bot` ile bağlanacağın adresi öğrenirsin.
2. O adrese WebSocket açarsın. İlk gelen çerçeve `op 10` (HELLO) olur ve heartbeat aralığını söyler.
3. O aralıkla `op 1` (heartbeat) göndermeye başlarsın.
4. `op 2` (identify) ile kim olduğunu ve hangi olayları istediğini söylersin.
5. `READY` gelir, ardından sunucuların tek tek `GUILD_CREATE` olarak akar.
6. Olaylar gelmeye başlar. Her olayın sıra numarasını saklarsın.
7. Bağlantı koparsa yeniden bağlanıp `op 6` (resume) ile kaldığın yerden sürersin.

Çerçeveler

Her iki yönde de her mesaj, metin olarak yazılmış tek bir JSON nesnesidir:

```
{ "op": 0, "d": { "id": "21531879014400001" }, "s": 42, "t": "GUILD_DELETE" }
```

Alan	Anlamı
`op`	Çerçevenin ne olduğu. Tüm opcode'lar Olaylar sayfasında.
`d`	İçerik. Biçimi `op` değerine ve olaya göre değişir.
`s`	Olayın sıra numarası. Yalnızca `op 0` çerçevelerinde dolu olur.
`t`	Olayın adı, örneğin `MESSAGE_CREATE`. Yalnızca `op 0` çerçevelerinde dolu olur.

Senin gönderdiğin bir çerçeve en çok 16 KiB olabilir ve dakikada en çok 120 çerçeve gönderebilirsin, heartbeat'ler dahil. Bu sayıyı aşan bağlantı `4008` ile kapanır.

Sıkıştırma botlar için yok. Bağlantı adresine `compress` eklesen de bir şey değişmez; `zstd` botlara sunulmaz. Çerçeveler her zaman düz JSON metnidir, kod çözücü gerekmez.

1. Adresi al

Botlar kendi adreslerine bağlanır. Bu adresi koda yazma, her başlangıçta `GET /gateway/bot` ile sor:

```
curl "https://noluyo.net/api/v1/gateway/bot" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
{
  "url": "wss://noluyo.net/bot-gateway",
  "shards": 1,
  "session_start_limit": {
    "total": 1000,
    "remaining": 998,
    "reset_after": 41230000,
    "max_concurrency": 1
  }
}
```

- `url` : Bağlanacağın adres.
- `shards` : Kaç bağlantı açmanın önerildiği. Çoğu bot için `1` ; ayrıntısı Shard'lar sayfasında.
- `session_start_limit` : Bugün kaç oturum başlatabileceğin ve sayacın kaç milisaniye sonra sıfırlanacağı. Bu rotayı okumak oturum hakkı harcamaz.

> **Uyarı:** N'oluyo'nun kendi uygulamalarının kullandığı `/gateway` adresi kişiler içindir: bot token'ını `4004` ile reddeder. Botun için doğru adresi her zaman bu rotadan öğrenirsin.

2. Bağlan ve HELLO'yu bekle

```
const ws = new WebSocket(url);
```

Bağlantı açılınca sunucu hemen `op 10` gönderir:

```
{ "op": 10, "d": { "heartbeat_interval": 41250 }, "s": null, "t": null }
```

Bundan sonra 10 saniye içinde identify ya da resume göndermelisin, yoksa bağlantı `4003` ile kapanır. Aynı IP adresinden, henüz identify etmemiş en çok 20 bağlantı açık kalabilir; fazlası `429` ile reddedilir.

3. Heartbeat gönder

`heartbeat_interval`, milisaniye cinsinden aralıktır. Bu aralıkla `op 1` gönder. `d` alanına son gördüğün `s` değerini koyabilirsin, sunucu onu okumaz:

```
setInterval(() => ws.send(JSON.stringify({ op: 1, d: seq })), heartbeat_interval);
```

Sunucu her heartbeat'e `op 11` ile yanıt verir. İki durumu bilmen gerekir:

- **Yanıt gelmezse** bağlantı ölmüş demektir, kapanmış gibi görünmese de. Bir sonraki heartbeat zamanı geldiğinde hâlâ yanıt yoksa bağlantıyı bırak, yeniden bağlan ve resume yap.
- **Sen göndermezsen** sunucu aralığın bir buçuk katı kadar bekler, sonra bağlantıyı kapatır. Bu kapanış bir kod taşımaz, sen `1006` görürsün. Oturum sürdürülebilir kalır: yeniden bağlanıp resume yap.

4. Identify gönder

Identify, oturumu başlatan çerçevedir:

```
{
  "op": 2,
  "d": {
    "token": "<token>",
    "intents": 33281,
    "shard": [0, 1],
    "presence": { "status": "online", "activities": [] }
  }
}
```

Alan	Gerekli mi	Anlamı
`token`	Evet	Botun token'ı. REST'teki gibi `Bot ` öneki **almaz**.
`intents`	Evet	Hangi olayları istediğin, bitlerin toplamı olarak. Buradaki `33281`: `GUILDS` + `GUILD_MESSAGES` + `MESSAGE_CONTENT`. Bkz. Intent'ler.
`shard`	Hayır	`[id, count]`. Verilmezse `[0, 1]` olur. Bkz. Shard'lar.
`presence`	Hayır	Botun başlangıç durumu: `online`, `idle`, `dnd` ya da `invisible`.

Her başarılı identify günlük oturum haklarından birini harcar, reddedilen identify harcamaz. Oturumlar en çok 5 saniyede bir başlatılabilir; daha sık denersen bağlantı `4008` ile kapanır.

5. READY ve GUILD_CREATE

Identify kabul edilince ilk olay `READY` gelir:

```
{
  "op": 0,
  "s": 1,
  "t": "READY",
  "d": {
    "v": 1,
    "user": { "id": "21879171579904005", "username": "selam_botu", "bot": true },
    "session_id": "6f1c0e2ab7d94c3f8e5a1d0b9c7e4a26",
    "application": { "id": "21879167385600004" },
    "shard": [0, 1],
    "guilds": [{ "id": "21531879014400001", "unavailable": true }]
  }
}
```

`session_id` değerini sakla, resume için gerekir. `user.id` de işine yarar: bot kendi mesajlarını da duyar ve onlara yanıt vermemek için bu ID ile karşılaştırman gerekir.

`guilds` listesi yalnızca ID'leri verir ve hepsi `unavailable` görünür. Bu, sunucuların kapalı olduğu anlamına gelmez, henüz gönderilmedikleri anlamına gelir. Asıl içerikleri ardından **tek tek** `GUILD_CREATE` olarak gelir, senin okuma hızına göre. Yani `READY` yalnızca neyin geleceğini söyler: listedeki her sunucu için bir `GUILD_CREATE` bekle.

Bu akışta üç şey yeni başlayanları şaşırtır:

- **Bir sunucunun olayları, sırası gelene kadar gelmez.** Sıra ona yaklaşınca başlar. Yani `READY`'den hemen sonra her sunucuyu dinliyor sayılmazsın; o arada yazılan mesajları görmek istersen REST ile okumalısın.
- **Olay, `GUILD_CREATE`'ten önce gelebilir.** Bir sunucunun ilk olayı onun `GUILD_CREATE`'inden hemen önce ulaşabilir. Kodun, tanımadığı bir `guild_id` gördüğünde hata vermemeli.
- **`GUILD_CREATE`, aldığın bir olaydan eski olabilir.** Sunucu önce olayları sana açar, sonra sunucunun o anki halini okur, sonra gönderir. Bu arada olan bir değişiklik önce olay olarak gelir, sonra gelen `GUILD_CREATE` onu içermeyebilir. Bazen de tersi olur: aynı değişiklik hem anlık görüntüde hem olayda bulunur.

Sunucuların bilgisini bellekte tutuyorsan güvenli yol şudur: Bir sunucu için `GUILD_CREATE`'ten önce gelen olayları kenara yaz. `GUILD_CREATE` gelince bilgiyi ondan kur, sonra kenardaki olayları sırayla uygula. Kodunu, aynı olay iki kez uygulansa da bir şey bozulmayacak biçimde yaz: "ekle" yerine "yoksa ekle, varsa güncelle" mantığıyla.

> **Not:** `GUILD_CREATE` almak için `GUILDS` intent'i gerekir. Bu intent'i istemezsen `READY`'deki sunucular için `GUILD_CREATE` bekleme, hiç gelmez. Sunucuların diğer olayları yine sırası gelince başlar.

Bot sonradan bir sunucuya eklenirse aynı `GUILD_CREATE` o an gelir. Çıkarılırsa ya da sunucu silinirse `GUILD_DELETE` gelir. Bu, `READY`'de olup henüz gönderilmemiş bir sunucu için de geçerlidir; bu yüzden tüm sunucuları beklerken `GUILD_DELETE`'i de o sunucunun cevabı say. Aynı sunucu için iki kez gelebilir, ikincisi zarar vermeyecek şekilde yaz. `GUILD_CREATE` üye olarak yalnızca botu listeler; diğer üyeler için aşağıdaki `op 8` var. `GUILD_VOICE_STATES` istemediysen `voice_states` boştur.

6. Olayları dinle ve sıra numarasını sakla

Her `op 0` çerçevesindeki `s` değerini sakla, her zaman en son gördüğünü tut. Resume bunu ister.

```
ws.onmessage = (e) => {
  const { op, d, s, t } = JSON.parse(e.data);
  if (s) seq = s;
  if (op === 0 && t === "MESSAGE_CREATE") console.log(d.author.username, d.content);
};
```

Hangi olayların geldiği Intent'ler ile seçtiklerine bağlıdır. `MESSAGE_CREATE`'te `content` alanının boş gelmesi hata değildir, aynı sayfadaki mesaj içeriği kuralına bak.

7. Bağlantı koparsa resume yap

Bağlantılar kopar: ağ değişir, sunucu yeniden başlar, programın yavaş kalır. Olay kaçırmamak için yeniden identify etmek yerine oturumu sürdür. Yeni bir bağlantı aç, HELLO'yu bekle, identify yerine şunu gönder:

```
{
  "op": 6,
  "d": {
    "token": "<token>",
    "session_id": "6f1c0e2ab7d94c3f8e5a1d0b9c7e4a26",
    "seq": 42
  }
}
```

Kaçırdığın olaylar sırayla gelir, ardından `RESUMED` gelir. Resume, oturum hakkı harcamaz. Heartbeat'i yeni bağlantının HELLO'sundaki aralıkla yeniden başlat.

Resume'un sınırları var:

- Oturum, bağlantı koptuktan sonra **2 dakika** bekler. Daha geç kalırsan `op 9` gelir.
- Sunucu oturum başına son 1000 olayı, en çok 2 MiB olacak şekilde saklar. Daha çoğunu kaçırdıysan yine `op 9` gelir.
- Oturum, identify ettiği intent'lerle sürer. Portalda intent açtıysan resume yetmez, yeni identify gerekir.
- Görmediğin `GUILD_CREATE`'ler saklanan kopyadan değil, sunucunun o anki halinden yeniden gönderilir. `GUILD_MEMBERS_CHUNK` ve `RATE_LIMITED` ise hiç gönderilmez; üye listesini yeniden iste.

Kendin kapatırken kapanış koduna dikkat et: `ws.close(1000)` oturumu **bitirir** ve bot çevrimdışı olur. Başka kütüphaneler `1001` de gönderebilir, sunucu onu da aynı sayar. Sonra resume etmek istiyorsan başka bir kod kullan, örneğin `ws.close(4000)`.

op 7 ve op 9

`op 7` **(RECONNECT):** Sunucu kapanmak üzere. Bağlantıyı kendin kapat, yeniden bağlan, resume yap.

`op 9` **(INVALID_SESSION):** Oturum sürdürülemiyor. `d` her zaman `false` olur. Sakladığın `session_id` ve `seq` değerlerini unut, bellekteki sunucu bilgilerini de eskimiş say. Bağlantı açık kalır: birkaç saniye bekle ve aynı bağlantıda `op 2` gönder. Hemen denersen 5 saniyelik sınıra takılabilirsin.

Kapanış kodları

Bağlantı kapanınca `onclose` içindeki `e.code` nedenini söyler. Kararını şu tabloya göre ver. Adları ve daha uzun açıklamaları Olaylar sayfasında.

Kod	Anlamı	Yeniden bağlan	Ne yapmalı
`4000`	Sunucuda bir hata oldu	Evet	Resume dene.
`4001`	Bilinmeyen ya da botlara kapalı opcode	Hayır	Kodunu düzelt.
`4002`	Çerçeve okunamadı ya da değerleri geçersiz	Hayır	Kodunu düzelt.
`4003`	Identify'dan önce heartbeat dışında bir şey gönderildi ya da 10 saniye geçti	Hayır	Sırayı düzelt.
`4004`	Token yanlış ya da yenilendi, veya bot ya da sahibi kapatıldı	Hayır	Yeni token al, programı yeniden başlat.
`4005`	Oturumu olan bağlantıda yeniden identify ya da resume	Hayır	Kodunu düzelt.
`4008`	Çok fazla çerçeve ya da çok sık oturum başlatma	Evet	5 saniye bekle, gönderimini azalt.
`4010`	Platform bakımda	Evet	Bir süre bekleyip identify dene.
`4011`	Daha yeni bir oturum bunun yerini aldı	Hayır	Botun iki kopyası çalışıyor olabilir, birini durdur.
`4012`	`shard` yanlış ya da üst sınırın üstünde	Hayır	`shard` değerini düzelt.
`4013`	`intents` yok, sayı değil ya da tanımsız bit içeriyor	Hayır	`intents` değerini düzelt.
`4014`	İzni olmayan ayrıcalıklı intent	Hayır	Portaldan aç ya da istemeyi bırak.
`4015`	Bu sayı için fazla sunucu	Hayır	Daha çok shard ile bağlan.
`4016`	Günlük oturum hakların bitti	Evet	`reset_after` kadar bekle.

Kapanış kodu olmadan kopan bağlantılar (`1006` gibi) ağ sorunudur ya da sunucu heartbeat'ini zamanında alamamıştır; `1012` (*Service Restart*) ise N'oluyo sunucusu güncellenirken gönderilir ve oturumun sunucu belleğinde korunduğunu gösterir: her ikisinde de `session_id` ile `seq` değerlerini koruyup ~1,5 saniye sonra yeniden bağlanarak `op 6` (`RESUME`) gönder. "Hayır" yazan kodlarda döngüye girme, aynı istek aynı sonucu verir.

Botun durumunu değiştirmek: op 3

```
{ "op": 3, "d": { "status": "idle", "activities": [] } }
```

`status`, `online`, `idle`, `dnd` ya da `invisible` olabilir. Bir oturum dakikada 5 kez gönderebilir. Değişiklik, botun tüm shard'lardaki sunucularında görünür.

Ses kanalına girmek: op 4

```
{ "op": 4, "d": { "guild_id": "21531879014400001", "channel_id": "21531879014400099",
"self_mute": false } }
```

Botu bir ses kanalına sokar; `channel_id: null` gönderirsen kanaldan çıkarır. Yanıt olarak aynı oturuma LiveKit adresini ve 10 dakikalık token'ı taşıyan `VOICE_SERVER_UPDATE`, bir kural ya da sınır engellerse `VOICE_JOIN_ERROR` gelir. Bot seste yalnızca yayın yapar, kimseyi duyamaz. Çalışan örnek ve sınırlar Ses sayfasında.

Üye listesi: op 8

Binlerce üyeyi her bota peşin göndermek israf olurdu, bu yüzden `GUILD_CREATE` üye listesi taşımaz. Üye listesine ihtiyacı olan bot onu `op 8` ile ister:

```
{ "op": 8, "d": { "guild_id": "21531879014400001", "query": "ay", "limit": 10, "nonce": "arama-1"
} }
```

`query` ve `user_ids` alanlarından **tam olarak biri** gönderilir:

İstek	Ne döner	Ne ister
`"query": "ay"`	Adı bu sözcükle başlayanlar. `limit` en çok 100, verilmezse 100.	Bir şey istemez.
`"user_ids": ["..."]`	Bu kullanıcılar, en çok 100 ID. Üye olmayanlar `not_found` içinde.	Bir şey istemez.
`"query": ""`	Tam liste. `limit` verilirse en çok o kadar.	`GUILD_MEMBERS` intent'i

`"presences": true` üyelerin durumlarını da ekler ve `GUILD_PRESENCES` ister. Gereken intent oturumda yoksa bağlantı `4014` ile kapanır. `nonce`, en çok 32 baytlık bir etikettir; yanıtta aynen geri gelir ve hangi isteğin yanıtı olduğunu anlamanı sağlar. `query` en çok 100 karakter olabilir. Bu sınırları ya da tablodakileri aşan istek bağlantıyı `4002` ile kapatır.

Yanıt, her biri en çok 1000 üye taşıyan `GUILD_MEMBERS_CHUNK` parçalarıyla gelir. Bitişi şöyle anlarsın:

```
if (t === "GUILD_MEMBERS_CHUNK") {
  members.push(...d.members);
  if (d.chunk_index + 1 === d.chunk_count) console.log("liste bitti", members.length);
}
```

- **Geçerli olan, son parçadaki `chunk_count` değeridir.** İlk parçalardaki sayı bir tahmindir: liste gönderilirken üyeler ayrılabilir, o zaman son parça daha erken gelir ve kendi `chunk_count` değeriyle bittiğini söyler. Sayıyı başta saklayıp ona güvenme, her parçada yeniden bak.
- **Boş bir son parça, listenin eksik olabileceği anlamına gelir.** Liste yarıda hata alırsa, boşuna beklemeyesin diye `members` alanı boş bir son parça gelir. Ayrı bir hata alanı yoktur. Emin olmak için sınırlara uyarak listeyi yeniden iste.
- Botun içinde olmadığı ya da başka shard'daki bir sunucu için tek bir boş parça gelir.

Aynı anda tek istek yanıtlanır. Bir oturum dakikada 10 istek gönderebilir, tam liste ise sunucu başına 30 saniyede bir istenebilir. Buna uymayan istek bağlantıyı kapatmaz, `RATE_LIMITED` ile yanıtlanır.

RATE_LIMITED

`op 3`, `op 4` ya da `op 8` sınıra takılınca bağlantı kapanmaz, sana bir olay gelir:

```
{ "op": 0, "s": 57, "t": "RATE_LIMITED", "d": { "opcode": 8, "retry_after": 12.5, "meta": { "guild_id": "215318790144000001", "nonce": "arama-1" } } }
```

`retry_after` saniye kadar bekleyip aynı çerçeveyi yeniden gönder. Sayılar Hız sınırları sayfasında.

INTERACTION_CREATE

Biri botunun mesajındaki butona bastığında ya da menüden seçtiğinde gelir. Intent istemez, yalnızca sunucunun shard'ına gider; DM'den gelirse shard 0'a. Bot çevrimdışıysa olay kaybolur. Ayrıntı Bileşenler sayfasında.

Tam bir istemci

Aşağıdaki dosya bu sayfadaki bağlantı işlerinin hepsini yapar: adresi sorar, heartbeat gönderir, kopunca resume yapar, `op 7` ve `op 9`'u işler, kapanış koduna göre karar verir. Örnekler sayfasındaki botlar bu dosyayı kullanır. `gateway.mjs` adıyla kaydet:

```
const API = "https://noluyo.net/api/v1";
const { BOT_TOKEN } = process.env;

// Bu kodlarda yeniden bağlanmak bir şey düzeltmez.
const FATAL = [4001, 4002, 4003, 4004, 4005, 4011, 4012, 4013, 4014, 4015];
const sleep = (ms) => new Promise((r) => setTimeout(r, ms));

async function gatewayInfo() {
  for (let attempt = 0; ; attempt++) {
    try {
      const res = await fetch(`${API}/gateway/bot`, {
        headers: { Authorization: `Bot ${BOT_TOKEN}` },
      });
      if (res.status >= 500 && attempt < 5) {
        await sleep(Math.min(10000, 1000 * 2 ** attempt));
        continue;
      }
      if (!res.ok) throw new Error(`GET /gateway/bot: ${res.status}`);
      return res.json();
    } catch (err) {
      if (attempt >= 5 || String(err.message).startsWith("GET /gateway/bot: 4")) throw err;
      await sleep(Math.min(10000, 1000 * 2 ** attempt));
    }
  }
}

export async function connect({ intents, shard, onEvent }) {
  const { url } = await gatewayInfo();
  let sessionId = null;
  let seq = null;
  let current = null;
  let failures = 0;

  function open() {
    const ws = new WebSocket(url);
    current = ws;
    const send = (op, d) => {
      if (ws.readyState === WebSocket.OPEN) ws.send(JSON.stringify({ op, d }));
    };
    const identify = () => send(2, { token: BOT_TOKEN, intents, shard });
    let heartbeat;
    let reidentifyTimer;
    let acked = true;

    const clearTimers = () => {
      clearInterval(heartbeat);
      clearTimeout(reidentifyTimer);
    };
```

```
    // Bu bağlantıyı bırakır. 4000 ile kapattığımız için oturum sürdürülebilir kalır.
    function reopen() {
      clearTimers();
      ws.onmessage = ws.onclose = ws.onerror = null;
      try {
        if (ws.readyState === WebSocket.OPEN || ws.readyState === WebSocket.CONNECTING)
ws.close(4000);
      } catch {}
      setTimeout(open, 1000);
    }

    ws.onerror = () => {};

    ws.onmessage = (e) => {
      const { op, d, s, t } = JSON.parse(e.data);
      if (s) seq = s;
      if (op === 0) {
        if (t === "READY") sessionId = d.session_id;
        onEvent(t, d);
      } else if (op === 10) {
        failures = 0;
        heartbeat = setInterval(() => {
          // Önceki heartbeat yanıtsız kaldı: bağlantı ölmüş.
          if (!acked) return reopen();
          acked = false;
          send(1, seq);
        }, d.heartbeat_interval);
        if (sessionId) send(6, { token: BOT_TOKEN, session_id: sessionId, seq });
        else identify();
      } else if (op === 11) {
        acked = true;
      } else if (op === 7) {
        reopen();
      } else if (op === 9) {
        sessionId = seq = null;
        clearTimeout(reidentifyTimer);
        reidentifyTimer = setTimeout(identify, 5000);
      }
    };

    ws.onclose = async (e) => {
      clearTimers();
      console.error("gateway kapandı", e.code, e.reason);
      if (FATAL.includes(e.code)) process.exit(1);
      if (e.code === 4010) sessionId = seq = null;
      let wait = e.code === 1012 ? 1500 : Math.min(15000, 1000 * 2 ** Math.min(failures++, 4));
      if (e.code === 4008) wait = Math.max(5000, wait);
      if (e.code === 4016) {
```

```
        try {
          wait = (await gatewayInfo()).session_start_limit.reset_after;
        } catch {
          wait = 60000;
        }
      }
      setTimeout(open, wait);
    };
  }

  open();
  return {
    send: (op, d) => {
      if (current?.readyState === WebSocket.OPEN) current.send(JSON.stringify({ op, d }));
    },
  };
}
```

Kullanımı şöyle:

```
import { connect } from "./gateway.mjs";

await connect({
  intents: (1 << 0) | (1 << 9),
  onEvent(t, d) {
    if (t === "READY") console.log(d.user.username, "bağlandı");
  },
});
```

`onEvent` içinde fırlatılan bir hata programı durdurur, kendi işlerini `try` içinde yap.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya ham WebSocket istemcisi yazdırırken şu kuralları özellikle hatırlat:
>
> - Gateway adresi koda sabit yazılmaz; her açılışta `GET https://noluyo.net/api/v1/gateway/bot` ucundan alınır ve bot gateway'inde zlib akış sıkıştırması yoktur (düz UTF-8 JSON çerçeveleri kullanılır).
> - `op 10` içindeki `heartbeat_interval` süresine uyulur, kopmalarda önce `op 6` (`RESUME`) denenir ve `4004` / `4014` kapanış kodlarında yeniden bağlanma döngüsüne girilmez.

Sırada ne var

- Intent'ler: hangi olayların geleceğini seçmek ve mesaj içeriği kuralı
- Shard'lar: büyük botları birden çok WebSocket bağlantısına bölmek

- JavaScript kütüphanesi ve Python kütüphanesi: hazır gateway ve `RESUME` yönetimi

Ses

Botun bir ses kanalına girip müzik, duyuru ya da ses efekti yayınlayabilir. Ses dosyası ya da akış senin makinende çözülür ve doğrudan medya sunucusuna gönderilir; N'oluyo'nun API sunucularından geçmez.

> **Önemli:** Bir bot seste **yalnızca yayın yapar, kimseyi duyamaz.** Botun aldığı token'da `canSubscribe: false` yazılıdır ve medya sunucusu botun hiçbir ses, görüntü ya da veri izine abone olmasına izin vermez. Kodunda abone olmayı denesen bile tek bir ses karesi gelmez.

Akış

Ses bağlantısı iki adımda kurulur: önce gateway'den ses kanalına girmek istediğini söylersin, sonra gelen adres ve token ile LiveKit odasına bağlanırsın.

1. Gateway bağlantında `op 4` gönderirsin (`intents: 0` yeterlidir; ses için ek bir intent gerekmez):

```
{
  "op": 4,
  "d": {
    "guild_id": "21531879014400001",
    "channel_id": "21531879014400099",
    "self_mute": false
  }
}
```

1. İzinlerin tamamsa ve sınırlar aşılmadıysa aynı oturuma `VOICE_SERVER_UPDATE` olayı gelir:

```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531879014400099",
  "endpoint": "wss://noluyo.net/rtc",
  "token": "<10 dakikalık LiveKit token'ı>",
  "bitrate": 96000
}
```

1. Bir kural ya da sınır engellerse bağlantı kapanmaz; `VOICE_JOIN_ERROR` olayı gelir:

```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531879014400099",
  "code": "channel_bot_limit",
  "message": "This voice channel already has the maximum number of bots."
}
```

1. Kanaldan ayrılmak için önce LiveKit odasından kop, ardından `channel_id: null` ile `op 4` gönder:

```
{
  "op": 4,
  "d": {
    "guild_id": "21531879014400001",
    "channel_id": null
  }
}
```

Bir bot her sunucuda aynı anda tek bir ses kanalında olabilir; farklı sunucularda ise aynı anda birden çok ses kanalında bulunabilir. Odaya girerken aldığın `token` 10 dakika geçerlidir; bağlandıktan sonra açık kalan odalarda LiveKit bağlantıyı kendisi yeniler, 10 dakika içinde hiç bağlanmazsan token geçersiz olur.

Gereken izinler

Ses kanalına girip yayın yapmak için botun o kanalda üç izne ihtiyacı vardır:

- `ViewChannel` (`1024`) — kanalı görme
- `Connect` (`1048576`) — ses kanalına bağlanma
- `Speak` (`2097152`) — seste yayın yapma

Üçünün toplamı `3146752` eder; ekleme bağlantısında `izin=3146752` kullanabilirsin.

Bot odaya girdikten sonra izinleri değişirse şunlar olur:

- **`Speak` kaldırılırsa** (veya bot zaman aşımına alınırsa) bot odada kalır ama yayın izni anında kapatılır; sesi kesilir.
- **`Connect` veya `ViewChannel` kaldırılırsa** bot odadan hemen çıkarılır.

Botlar yalnızca sunuculardaki ses kanallarına (`type: 2`) girebilir; DM aramalarına katılamaz, kamera ya da ekran paylaşımı açamaz ve yalnızca tek bir mikrofon ses izi yayınlayabilir.

Odaya bağlanmak ve dosya çalmak

Aşağıdaki örnek bir ses dosyasını `ffmpeg` ile 48 kHz stereo PCM'e çevirip odaya yayınlar ve dosya bitince kanaldan ayrılır. Yanına Gateway sayfasındaki `gateway.mjs` dosyasını koy.

Önce LiveKit'in Node paketini ve makinenin paket yöneticisinden `ffmpeg` aracını kur:

```
npm install @livekit/rtc-node@1.2.1
```

Dosyayı `cal.mjs` adıyla kaydet:

```
import { spawn } from "node:child_process";
import { AudioFrame, AudioSource, LocalAudioTrack, Room, TrackPublishOptions, TrackSource,
dispose } from "@livekit/rtc-node";
import { connect } from "./gateway.mjs";

const { GUILD_ID, VOICE_CHANNEL_ID, AUDIO_FILE } = process.env;
const RATE = 48000;
const CHANNELS = 2;
const SAMPLES = 960; // 20 ms
const FRAME_BYTES = SAMPLES * CHANNELS * 2;

const gateway = await connect({
  intents: 0, // Ses için intent gerekmez.
  onEvent(type, d) {
    if (type === "READY") gateway.send(4, { guild_id: GUILD_ID, channel_id: VOICE_CHANNEL_ID,
self_mute: false });
    if (type === "VOICE_SERVER_UPDATE") play(d).catch(fail);
    if (type === "VOICE_JOIN_ERROR") fail(new Error(`${d.code}: ${d.message}`));
  },
});

function fail(err) {
  console.error(err.message);
  process.exit(1);
}

async function play({ endpoint, token, bitrate }) {
  const room = new Room();
  await room.connect(endpoint, token, { autoSubscribe: false, dynacast: false });

  const source = new AudioSource(RATE, CHANNELS);
  const track = LocalAudioTrack.createAudioTrack("muzik", source);
  const options = new TrackPublishOptions({
    source: TrackSource.SOURCE_MICROPHONE,
    audioEncoding: { maxBitrate: BigInt(bitrate) }, // Kanalın ses kalitesini aşma.
  });
  await room.localParticipant.publishTrack(track, options);

  const ffmpeg = spawn("ffmpeg", ["-v", "error", "-i", AUDIO_FILE, "-f", "s16le", "-ar",
String(RATE), "-ac", String(CHANNELS), "-"], {
    stdio: ["ignore", "pipe", "inherit"],
  });
  let rest = Buffer.alloc(0);
  for await (const chunk of ffmpeg.stdout) {
    rest = Buffer.concat([rest, chunk]);
    while (rest.length >= FRAME_BYTES) {
      const pcm = rest.subarray(0, FRAME_BYTES);
      const samples = new Int16Array(pcm.buffer.slice(pcm.byteOffset, pcm.byteOffset +
```

```
FRAME_BYTES));
      await source.captureFrame(new AudioFrame(samples, RATE, CHANNELS, SAMPLES));
      rest = rest.subarray(FRAME_BYTES);
    }
  }

  await source.waitForPlayout();
  await track.close();
  await room.disconnect();
  gateway.send(4, { guild_id: GUILD_ID, channel_id: null });
  await dispose();
  process.exit(0);
}
```

Çalıştırmak için:

```
export BOT_TOKEN='kopyaladigin_token'
export GUILD_ID='sunucu_id'
export VOICE_CHANNEL_ID='ses_kanali_id'
export AUDIO_FILE='./ornek.mp3'
node cal.mjs
```

Ses dosyası senin makinende okunur; N'oluyo'nun API sunucusuna yüklenmez ve bizde saklanmaz. Yayınladığın sesin kullanım ve telif haklarına sahip olman senin sorumluluğundadır.

Sınırlar

İnsanların sesli sohbetini korumak için bot ses bağlantıları aşağıdaki sınırlarla çalışır:

Anahtar	Sınır	Açıklama
`channel`	2	Bir ses kanalında aynı anda bulunabilecek en çok bot
`guild`	4	Bir sunucudaki tüm ses kanallarında aynı anda bulunabilecek en çok bot
`app`	50	Bir uygulamanın tüm sunucularda aynı anda tutabileceği ses bağlantısı
`platform`	100	Tüm botların ortak eşzamanlı ses bağlantısı tavanı (`NOLUYO_BOT_VOICE_MAX`)
`alone`	5	Kanalda hiç insan kalmadığında botun çıkarılmadan önce beklediği dakika
`op4`	5 / 20	Bot ve sunucu başına 20 saniyede gönderilebilecek `op 4` çerçevesi
`tracks`	10 / 60	Bot ve oda başına dakikada yapılabilecek ses izi yayını; aşılırsa bot odadan çıkarılır
`bitrate`	96	Botlara verilen en yüksek ses bit hızı (kbps); kanal daha düşükse kanalın değeri gelir

`op 4` hız sınırını aşarsan bağlantı kapanmaz, `RATE_LIMITED` (`opcode: 4`) olayı gelir; `retry_after` saniye bekleyip yeniden gönder. Kanalda tek başına 5 dakika kalan bot kendiliğinden sesten çıkarılır, böylece boş odalar ortak kapasiteyi doldurmaz.

Bot odada ne görür, kişi ne görür

Botun gördükleri: Bot ses alamaz ve kaydedemez. Ancak LiveKit odasında katılımcı olduğu için o odada o an kimin bağlı olduğunu (kullanıcı kimliği ve görünen ad), susturulmuş olup olmadığını ve **kimin ne zaman konuştuğunu** (ses içeriği olmadan yalnızca aktif konuşmacı bilgisini) görür. Ekleme ekranı `Connect` izni isteyen her botta sunucu yöneticisine bunu açıkça söyler.

Kişilerin gördükleri:

- Web, masaüstü (oyun içi arayüz dahil) ve Android'de bot ses listesinde ve arama ekranında **BOT** etiketiyle görünür; etiketin üstüne gelindiğinde ya da bilgi penceresinde botun yalnızca yayın yaptığı ve kimseyi duyamadığı yazar.
- Bot, ses kanalının kişi sayacına (`1 / 5` gibi) **dahil edilmez** ve dolu bir kanala insan girmesini engellemez.
- İsteyen her kullanıcı botun ses seviyesini kendi cihazında kısıp açabilir ya da botu yalnızca kendisi için susturabilir.
- Botun satırında kulaklık kapalı (`self_deaf: true`) durumu her zaman açıktır; bu bir hata değil, botun kimseyi duyamadığının teknik karşılığıdır.

Yönetici ne yapabilir

Sunucuda yetkisi olan yöneticiler seste duran bir bot üzerinde üç işlem yapabilir:

- **Sunucuda sustur (`MuteMembers`):** Botun yayın izni hemen kapatılır, sesi kesilir. Susturma kaldırılınca yeniden yayın yapabilir. **Sunucuda sağırlaştır** seçeneği botlar için gizlidir çünkü bot zaten hiçbir zaman duyamaz.
- **Bağlantıyı kes (`MoveMembers`):** Bot LiveKit odasından çıkarılır (`Disconnected`) ve ses satırı silinir. Botun kendi kendine hemen geri girmemeli; yalnızca bir kullanıcı komutu istediğinde yeniden `op 4` göndermelidir.
- **Başka ses kanalına taşı (`MoveMembers`):** Bot eski odadan çıkarılır ve botun açık gateway oturumuna yeni kanal için yeni bir `VOICE_SERVER_UPDATE` gönderilir. `cal.mjs` gibi `VOICE_SERVER_UPDATE` dinleyen bir bot eski `Room` nesnesini kapatıp gelen yeni `endpoint` ve `token` ile yeni odaya bağlanarak yayını sürdürebilir.

Botun token'ı yenilenirse, uygulaması durdurulursa ya da bot sunucudan çıkarılırsa açık tüm ses bağlantıları anında kesilir.

Hata kodları (VOICE_JOIN_ERROR)

`op 4` isteği reddedildiğinde `VOICE_JOIN_ERROR` olayındaki `code` alanı nedenini söyler:

Kod	Anlamı	Ne yapmalı
`unknown_guild`	Sunucu yok ya da bot bu sunucunun üyesi değil.	Sunucu kimliğini kontrol et, tekrar deneme.
`unknown_channel`	Kanal bu sunucuda yok.	Kanal kimliğini kontrol et.
`invalid_channel_type`	Kanal bir ses kanalı (`type: 2`) değil.	Metin kanalı ya da kategori yerine ses kanalı seç.
`missing_access`	Botun bu kanalı görme (`ViewChannel`) izni yok.	Kanal izinlerini kontrol et.
`missing_permissions`	Botun kanalda `Connect` izni yok ya da zaman aşımında.	Botun rolüne `Connect` ve `Speak` ver.
`channel_bot_limit`	Bu ses kanalında zaten 2 bot var.	Başka bir ses kanalı seç ya da diğer bot çıkana dek bekle.
`guild_bot_limit`	Bu sunucuda zaten 4 bot seste.	Sunucudaki diğer botlardan biri sesten çıkana dek bekle.
`app_voice_limit`	Uygulamanın eşzamanlı 50 ses bağlantısı doldu.	Boşta kalan bağlantıları kapat.
`bot_voice_full`	Platform genelindeki bot ses kapasitesi o an dolu.	Biraz bekleyip yeniden dene, sık döngüye girme.
`voice_unavailable`	Ses hizmeti o an kullanılamıyor ya da geçici bir hata oluştu.	Birkaç saniye bekleyip yeniden dene.

Bu sayfayı yapay zekaya verirsen

Bir yapay zeka aracından N'oluyo için ses veya müzik botu yazmasını isterken bu sayfayı ve Gateway sayfasını birlikte ver. Şu kuralları özellikle hatırlat:

- Bot token'ı koda yazılmaz, `process.env.BOT_TOKEN` ile ortam değişkeninden okunur.
- Ses kanalına girmek HTTP isteği değildir; gateway üzerinden `op 4` gönderilir ve `VOICE_SERVER_UPDATE` beklenir. `POST /channels/{id}/voice` kişi rotasıdır ve bot token'ını `403 bot_forbidden` ile reddeder.
- Bot seste kimseyi duyamaz: `setSubscribed(true)` işe yaramaz, ses tanıma veya kayıt botu istendiğinde "N'oluyo'da botlar seste kimseyi duyamaz, bu özellik yapılamaz" denmelidir.
- `TrackPublishOptions` içinde `source: TrackSource.SOURCE_MICROPHONE` ve `maxBitrate: BigInt(bitrate)` kullanılmalıdır; kamera veya ekran kaynağı reddedilir.

Sırada ne var

- JavaScript kütüphanesi ve Python kütüphanesi: `bot.voice.join` ile tek satırda ses yayını
- Gateway: `op 4` çerçevesinin gittiği WebSocket bağlantısı
- Hatalar: tüm REST ve `VOICE_JOIN_ERROR` hata kodları

Intent'ler

Bot, sunucularında olan biten her şeyi duymak zorunda değil. Yalnızca `!ping`'e yanıt veren bir botun, kimin hangi ses kanalına girdiğini bilmesine gerek yok. **Intent**, botun hangi olay gruplarını almak istediğini söylediği seçimdir. İstemediğin olaylar sana hiç gönderilmez, programın da onlarla boşuna uğraşmaz.

Intent'leri identify çerçevesinde bir sayı olarak gönderirsin. Bu alan zorunludur.

Intent'ler ve değerleri

Intent	Bit	Değer	Ne açar	Ayrıcalıklı
`GUILDS`	0	1	Sunucular, roller, kanallar, sabitlemeler	
`GUILD_MEMBERS`	1	2	Üye katılma, değişme, ayrılma; `op 8` ve `GET /guilds/{guild_id}/members` ile tam liste	Evet
`GUILD_MODERATION`	2	4	Yasaklar	
`GUILD_EXPRESSIONS`	3	8	Emoji, çıkartma, ses paneli	
`GUILD_WEBHOOKS`	5	32	Webhook değişiklikleri	
`GUILD_INVITES`	6	64	Davetler	
`GUILD_VOICE_STATES`	7	128	Seste kimlerin olduğu, orada çalınan sesler	
`GUILD_PRESENCES`	8	256	Üyelerin çevrimiçi durumu ve etkinlikleri	Evet
`GUILD_MESSAGES`	9	512	Sunucu kanallarındaki mesajlar	
`GUILD_MESSAGE_REACTIONS`	10	1024	Sunucu kanallarındaki tepkiler	
`GUILD_MESSAGE_TYPING`	11	2048	Sunucu kanallarında yazıyor bilgisi	
`DIRECT_MESSAGES`	12	4096	Kişilerin botunla açtığı DM'lerdeki mesajlar; içerik `MESSAGE_CONTENT` olmadan da tam gelir	
`DIRECT_MESSAGE_REACTIONS`	13	8192	DM'lerdeki tepkiler	
`DIRECT_MESSAGE_TYPING`	14	16384	DM'lerde yazıyor bilgisi	
`MESSAGE_CONTENT`	15	32768	Mesajların içeriği	Evet
`GUILD_SCHEDULED_EVENTS`	16	65536	Zamanlanmış etkinlikler	
`AUTO_MODERATION_CONFIGURATION`	20	1048576	Otomatik moderasyon kuralları	

Hangi olayın hangi intent'e bağlı olduğu Olaylar sayfasında, her olayın altında yazar.

Bazı olaylar intent istemez, her oturuma gider: `READY`, `RESUMED`, `RATE_LIMITED`, `GUILD_MEMBERS_CHUNK` ve botun kendi profilini anlatan `USER_UPDATE`. Botun kendi üyeliğiyle ilgili üye olayları da `GUILD_MEMBERS` olmadan gelir.

Değeri hesaplamak

İstediğin intent'lerin değerlerini topla ya da bitlerini `|` ile birleştir:

```
const GUILDS = 1 << 0;
const GUILD_MESSAGES = 1 << 9;
const MESSAGE_CONTENT = 1 << 15;

const intents = GUILDS | GUILD_MESSAGES | MESSAGE_CONTENT; // 33281
```

Değer sayı olarak gönderilir, metin olarak değil. Yoksa, sayı değilse ya da listede olmayan bir bit içeriyorsa bağlantı `4013` ile kapanır. `0` geçerlidir: bot o zaman yalnızca intent istemeyen olayları alır.

Ayrıcalıklı intent'ler

Üç intent hassas veri açar: `GUILD_MEMBERS`, `GUILD_PRESENCES` ve `MESSAGE_CONTENT`. Bunları istemeden önce portalda uygulama sayfasının **Ayrıcalıklı intent'ler** bölümünden açman gerekir. Portaldaki adları **Sunucu üyeleri**, **Presence** ve **Mesaj içeriği**'dir.

Açmadığın bir ayrıcalıklı intent'i identify'da istersen bağlantı `4014` ile kapanır.

REST'te üye listesi. `GET /guilds/{guild_id}/members` aynı kurala uyar; bir bağlantı olmadığı için portalın o anki haline bakar. `GUILD_MEMBERS` yoksa `403 member_list_intent_required` döner. Tek üye istemek, `GET /guilds/{guild_id}/members/{user_id}`, intent istemez.

100 sınırı. Bot 100'den az sunucudaysa bu seçim senindir, açman yeter. Bot 100 ya da daha çok sunucudaysa yalnızca onaylanmış ayrıcalıklı intent'ler geçerlidir: açık ama onaysız bir intent'i istersen yine `4014` gelir, onaysız yeni bir intent'i açmak ise `400 intent_approval_required` döner. Onay başvurusu ve gerekçe örnekleri Doğrulama sayfasındadır.

Açık bir oturum, identify ettiği intent'lerle sürer. Portaldaki değişiklik canlı oturumu şöyle etkiler:

- **Bir intent'i açarsan** çalışan oturum bunu kendiliğinden kazanmaz. Yeni intent'i kullanmak için programını yeniden başlat ve yeniden identify et; resume yetmez, çünkü resume aynı oturumu sürdürür.
- **Bir intent'i kapatırsan** botun bağlantıları hemen düşer ve oturumlar sürdürülemez. Kapattığın intent'i artık istemeden yeniden identify et, yoksa `4014` alırsın.

Mesaj içeriği kuralı

`GUILD_MESSAGES`, mesajların yazıldığını haber verir. Ne yazdığını okumak ise ayrı bir intent ister: `MESSAGE_CONTENT`. Bu intent olmayan bir bot `MESSAGE_CREATE` ve `MESSAGE_UPDATE` olaylarını yine alır, ama içeriği süzülmüş olarak.

İki istisnada mesaj tam gelir:

- Mesajı botun kendisi yazdıysa.
- Mesaj botu adıyla etiketliyorsa, yani bot mesajın `mentions` listesindeyse. Rol etiketi ve `@everyone` bu sayılmaz.

Süzülmüş bir mesajda `content` boş metin, `attachments` boş liste olur. Şu alanlar kalır, diğerlerinin hepsi çıkarılır:

`id`, `type`, `channel_id`, `guild_id`, `author`, `timestamp`, `edited_timestamp`, `mentions`, `mention_roles`, `mention_everyone`, `message_reference`, `pinned`, `nonce`, `webhook_id`, `reactions`, `activity`

Çıkarılanlar arasında çıkartmalar (`sticker_items`), anketler (`poll`), iletilen mesajlar (`forward`) ve mesajın başka tüm içerik alanları vardır. Boş `content` hata değildir: "içerik görme iznin yok" demektir.

> **İpucu:** Yalnızca komut dinleyen bir botun bu intent'e ihtiyacı olmayabilir. Kullanıcılar botu etiketleyerek yazarsa (`@bot !ping`) mesaj zaten tam gelir.

REST okumaları da aynı kurala uyar

Gateway'de kapalı olan içeriği REST'ten okuyamazsın. Sunucu kanallarında mesaj listesi, tek mesaj ve sabitlenenler aynı süzmeyi uygular, aynı iki istisnayla.

Tek fark şudur: REST isteğinde intent göndermezsin, bu yüzden uygulamanın portaldaki ayarına bakılır, oturumun identify ettiği intent'lere değil. Yani **Mesaj içeriği** açıksa (ve bot 100 ya da daha çok sunucudaysa onaylıysa) REST tam içerik döner; identify'da `MESSAGE_CONTENT` istememiş olsan bile.

İletme de aynı kurala uyar

Okuyamadığın mesajı iletemezsin, yoksa iletilen kopyayı okuyarak kuralı aşmış olurdun. **Mesaj içeriği** kapalı bir bot, kendisinin yazmadığı ve kendisini etiketlemeyen bir mesajı `forward` ile göndermeye çalışırsa `400 validation_failed` alır: `{"forward":"invalid"}`.

Etkileşim intent istemez

`INTERACTION_CREATE` hiçbir intent'e bağlı değildir: bir kişi botunun bileşenine bastıysa bot bunu `intents: 0` ile de alır. Olayın taşıdığı `message` botun kendi mesajı olduğu için `MESSAGE_CONTENT` kuralına da takılmaz, seçilen üye `GUILD_MEMBERS` olmadan da `resolved` içinde gelir. Ayrıntı Bileşenler sayfasında.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya olay dinleyen bir bot yazdırırken şunları belirt:
>
> - Yalnızca eğik çizgi komutu ve buton kullanan bir botun `intents: 0` göndermesi yeterlidir.
> - `GUILD_MEMBERS`, `GUILD_PRESENCES` ve `MESSAGE_CONTENT` kodda istenmeden önce portalda açılmış olmalıdır; aksi halde gateway `4014` ile kapanır.

Sırada ne var

- Bağlantının tamamı: Gateway

- Olayların alanları ve örnekleri: Olaylar
- Çok sunuculu botlar: Shard'lar

Shard'lar

Bot çok büyüdüğünde tek bir bağlantı tüm sunucularının olaylarını taşıyamaz. **Shard**, botun sunucularını birkaç bağlantıya bölmektir: her bağlantı yalnızca kendi payına düşen sunucuların olaylarını alır.

Küçük bir bot için bu sayfaya gerek yok. `GET /gateway/bot` yanıtında `shards` değeri `1` ise `shard` alanını hiç göndermeden tek bağlantı açarsın. 100 sunucuyu aşmak için Doğrulama gerekir; resmi JavaScript ve Python kütüphaneleri shard sayısını ve 5,1 saniyelik başlatma aralığını senin yerine yönetir.

Bir sunucu hangi shard'a düşer

Sunucunun ID'si ve toplam shard sayısı bunu belirler:

```
shard_id = (guild_id >> 22) % count
```

ID'ler 64 bitlik olduğu için JavaScript'te `BigInt` kullan:

```
const shardOf = (guildId, count) => Number((BigInt(guildId) >> 22n) % BigInt(count));
```

Bunu kendin hesaplamak zorunda değilsin: `READY` her oturuma kendi sunucularını listeler. Ama `op 8` isteğini doğru oturumdan göndermek için işine yarar, çünkü başka shard'daki bir sunucu için yalnızca boş bir parça gelir.

Sunucusu olmayan olaylar, yani DM'ler ve botun kendi profilini anlatan `USER_UPDATE`, **shard 0**'a gider. Kişilerin botunla açtığı DM'ler de, oradaki mesajlar ve etkileşimler de shard 0'a gelir.

Kaç shard açmalı

Sayıyı identify'daki `shard` alanıyla söylersin: `[id, count]`. `id`, 0'dan başlar ve `count`'tan küçük olmalıdır. Biçim yanlışsa bağlantı `4012` ile kapanır.

Üç sınır vardır ve üçü de botun bulunduğu sunucu sayısına bağlıdır:

- **Önerilen:** `GET /gateway/bot` yanıtındaki `shards`. 100 sunucuya kadar 1, sonrasında her 1000 sunucu için 1 ve en az 2.
- **Zorunlu:** Açabileceğin en küçük sayı. 100 sunucuya kadar 1, sonrasında her 2500 sunucu için 1 ve en az 2. Altında kalırsan bağlantı `4015` ile kapanır.
- **Üst sınır:** Önerilenin iki katı ve en az 16. Üstüne çıkarsan bağlantı `4012` ile kapanır.

Sunucu	Önerilen	Zorunlu	Üst sınır
100	1	1	16
101	2	2	16
2600	3	2	16
10000	10	4	20

Zorunlu sayı önerilenin altında durur, böylece bir sunucu daha eklendi diye bot kapı dışı kalmaz. Sen önerileni kullan ve program her başladığında yeniden sor.

Shard'ları başlatmak

Her shard ayrı bir bağlantı ve ayrı bir oturumdur, bu yüzden oturum başlatma sınırları geçerlidir:

- **5 saniyede 1 başlatma.** `session_start_limit.max_concurrency` bu sayıyı söyler ve bugün `1`'dir. Daha sık denersen bağlantı `4008` ile kapanır, bu yüzden shard'ları sırayla ve aralıklı başlat.
- **Günde 1000 başlatma.** Kalanı `session_start_limit.remaining`, sıfırlanmasına kalan milisaniyeyi `reset_after` söyler. Hak biterse bağlantı `4016` ile kapanır.

Resume başlatma sayılmaz. Bu yüzden yeniden başlatma döngüsüne giren bir bot, günlük hakkını hızla tüketir; önce resume dene.

Gateway sayfasındaki `gateway.mjs` ile:

```
import { connect } from "./gateway.mjs";

const res = await fetch("https://noluyo.net/api/v1/gateway/bot", {
  headers: { Authorization: `Bot ${process.env.BOT_TOKEN}` },
});
const { shards } = await res.json();

for (let id = 0; id < shards; id++) {
  await connect({ intents: 1, shard: [id, shards], onEvent: (t) => console.log(id, t) });
  await new Promise((done) => setTimeout(done, 5500));
}
```

Oturumlar birbirini nasıl etkiler

Bu kuralları bilmezsen botun kendi kendini düşürebilir:

- **Aynı `[id, count]` ile ikinci kez identify edersen** eski oturum `4011` ile kapanır. Botun iki kopyası çalışıyorsa birbirini böyle düşürür; bu kasıtlıdır, iki kopya aynı mesaja iki kez yanıt vermesin diye.
- **Toplamda en çok `2 * count` oturum açık kalabilir.** `count`, yeni identify'daki sayıdır. Fazlası, kopmuş olanlardan ve en eskilerden başlanarak `4011` ile kapanır.

Bu ikinci kural şunu da söyler: **Daha küçük bir `count` göndermek diğer shard'ların oturumlarını bitirir.** Örneğin 4 shard'la çalışan bir bot `[0, 1]` ile identify ederse toplam sınır 2 olur ve en eski üç oturum kapanır. Sayıyı küçültürken önce eski oturumları kendin kapat.

Sayıyı büyütürken eski oturumlar kendiliğinden kapanmaz. İkisi birlikte çalıştığı sürece aynı sunucunun olayları iki oturumdan da gelir, yani bir mesaja iki kez yanıt verebilirsin. Geçişi bitirince eski oturumları `ws.close(1000)` ile kapat.

Botun durumu

Botun durumu shard'lar arasında ortaktır. `op 3` hangi oturumdan gönderilirse gönderilsin botun tüm sunucularında görünür. Bir shard'ın oturumu kapanırsa bot, o shard'ın sunucularında çevrimdışı görünür.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya çok bağlantılı (sharded) bir bot yazdırırken şu kuralları hatırlat:
>
> - Shard sayısı sabit yazılmaz; her açılışta `GET /gateway/bot` yanıtındaki `shards` alanından okunur.
> - `IDENTIFY` çerçeveleri arasında en az 5 saniye beklenir (`4008` almamak için) ve kopmalarda yeni oturum açmak yerine önce `op 6` (`RESUME`) denenir.

Sırada ne var

- Gateway: WebSocket çerçeveleri, heartbeat ve `RESUME` akışı
- Doğrulama: 100 sunucu sınırını kaldırmak için başvuru koşulları
- JavaScript kütüphanesi ve Python kütüphanesi: otomatik shard yönetimi

Doğrulama

Yeni oluşturduğun bir bot ya da **N'oluyo ile giriş** uygulaması hemen çalışır; geliştirmek, arkadaşlarının sunucularında denemek ve ilk topluluğunu kurmak için hiçbir başvuru yapman gerekmez. Doğrulama yalnızca uygulaman büyüdüğünde devreye girer.

Doğrulama ne zaman gerekir

Üç sınırdan birine yaklaştığında doğrulama başvurusu yaparsın:

- **100 sunucu sınırı (botlar):** Doğrulanmamış bir bot en çok **100** sunucuya eklenebilir; 101. sunucu `/bot-ekle` ekranında `403 bot_guild_limit` alır.
- **100 sunucu üstünde ayrıcalıklı intent'ler:** Botun 100 sunucunun altındayken portaldan `GUILD_MEMBERS`, `GUILD_PRESENCES` ve `MESSAGE_CONTENT` kutularını serbestçe açabilirsin. 100 sunucuya ulaştığında bu üç ayrıcalıklı intent'in açık kalması için incelemede onaylanmış olmaları (`approved_intents`) gerekir; aksi halde gateway `4014` ile, REST çağrısı `403 intent_approval_required` ile kapanır.
- **100 kullanıcı sınırı (N'oluyo ile giriş):** Doğrulanmamış bir OAuth2 uygulamasına en çok **100** farklı kişi giriş yapabilir (`403 oauth_user_cap`).
- **Bot rehberinde listelenme:** `/kesfet` altındaki **Botlar** dizininde yer almak isteyen her bot önce doğrulanır (`want_listing: true`).

Başvuru öncesi kontrol listesi

`/gelistirici/uygulamalar` sayfasındaki **Doğrulama** bölümü (`POST /api/v1/applications/{application_id}/review`) başvuruyu göndermeden önce altı önkoşulu denetler. Eksik varsa `400 review_requirements` yanıtının `errors` nesnesinde eksik anahtarlar tek tek döner:

Anahtar	Koşul	Nasıl tamamlanır
`public`	Uygulama dışarıya açık olmalı	Bot için `public: true` açılmalı ya da OAuth2 yönlendirme adresi tanımlanmış olmalıdır.
`description`	Uygulama açıklaması dolu olmalı	Portalda uygulamanın ne yaptığını anlatan net bir açıklama yaz.
`avatar`	Uygulama görseli yüklenmiş olmalı	Uygulamayı tanıtan özgün bir avatar yükle.
`privacy_url`	`https://` ile başlayan gizlilik politikası adresi	Hangi verileri sakladığını, ne kadar tuttuğunu ve silme talebinin nasıl iletileceğini anlatan yayındaki sayfanın adresini gir.
`support_url`	`https://` ile başlayan destek adresi	Destek sunucunun davet bağlantısını, iletişim sayfanı ya da hata bildirim adresini ekle.
`owner_mfa`	Uygulama sahibinin iki aşamalı doğrulaması açık olmalı	Kendi N'oluyo hesabında **Ayarlar → Güvenlik** bölümünden iki aşamalı doğrulamayı (2FA) etkinleştir.

İyi bir başvuru metni nasıl yazılır

Başvuru formunda iki metin alanı (`purpose` ve `data_use`), ayrıcalıklı intent bit maskesi (`requested_intents`) ve rehber tercihi (`want_listing`) bulunur. İnceleyen kişi kodunu değil, bu açıklamaları ve uygulamanın gerçek davranışını okur.

İyi örnek:

- **Amaç (`purpose`):** *"Sunucu yöneticilerinin `/anket` komutuyla butonlu oylama başlatmasını ve `/hatirlat` komutuyla belirli bir kanala zamanlı duyuru bırakmasını sağlar."*
- **Veri kullanımı (`data_use`):** *"Yalnızca aktif anketlerin kanal kimliğini, mesaj kimliğini, seçenek metinlerini ve oy verenlerin kullanıcı kimliklerini (`user.id`) PostgreSQL tablosunda tutarız. Anket kapandıktan 7 gün sonra oy kayıtları otomatik silinir. Sunucu bottan çıkarıldığında (`GUILD_DELETE`) o sunucuya ait tüm satırlar hemen silinir. Mesaj içerikleri okunmaz ve üçüncü taraflarla paylaşılmaz."*

Zayıf örnek (reddedilir):

- *"Çok amaçlı moderasyon ve eğlence botu. Hiçbir veri toplamıyoruz, her şey güvenli."* (Veritabanında sunucu ayarı ya da sayaç tuttuğu halde "hiçbir veri toplamıyoruz" demek ya da ne yaptığını tek cümleyle geçiştirmek başvurunun geri dönmesine yol açar.)

Ayrıcalıklı intent gerekçesi

`requested_intents` alanında `GUILD_MEMBERS` (`1 << 1`), `GUILD_PRESENCES` (`1 << 8`) veya `MESSAGE_CONTENT` (`1 << 15`) istiyorsan her birinin neden normal komutlarla çözülemediğini açıkça yaz:

- **`MESSAGE_CONTENT`:** Yalnızca `/komut` veya `@bot` etiketleriyle çalışabilen bir bota `MESSAGE_CONTENT` onayı verilmez. Otomatik sözcük süzgeci ya da bağlantı denetimi gibi her mesajı görmek zorunda olan özellikler geçerli gerekçedir.
- **`GUILD_MEMBERS`:** Yeni katılan üyeye karşılama mesajı atmak, otomatik rol vermek ya da üye listesi senkronlamak geçerli gerekçedir.
- **`GUILD_PRESENCES`:** Çevrimiçi/çevrimdışı durum geçişi olmadan çalışmayan somut bir özellik göstermelisin; salt istatistik grafiği toplamak için onaylanmaz.

Onay çıktığında yalnızca uygun görülen intent'ler `approved_intents` alanına işlenir.

İnceleme süreci ve ret durumunda ne olur

- Aynı uygulama için aynı anda yalnızca **bir** bekleyen başvuru bulunabilir; bekleyen başvuru varken ikinci kez gönderirsen `400 review_pending` döner.
- Başvurular insan gözüyle tek tek incelenir; **sabit bir sonuçlanma süresi sözü verilmez**. Durumu `/gelistirici/uygulamalar` içindeki **Doğrulama** kartından (`GET /api/v1/applications/{application_id}/review`) takip edebilirsin; karar çıktığında hesabının dilinde e-posta da gönderilir.
- **Başvuru reddedilirse (`rejected`):** İnceleyen kişinin yazdığı açıklama (`note`) portalda **Doğrulama** kartında görünür. Eksikleri giderip dilediğin zaman yeni bir başvuru gönderebilirsin.

Doğrulanmış uygulamanın sorumlulukları ve acil durdurma

Doğrulanmış rozeti kalıcı bir ayrıcalık değil, süregelen bir güven sözüdür:

- `privacy_url` ve `support_url` adresleri yayında ve erişilebilir kalmalıdır.
- Beyan ettiğin `data_use` kapsamının dışına çıkacak yeni bir veri toplamaya başlarsan önce gizlilik metnini güncelle.
- **Acil durdurma:** Bir uygulamanın token'ı sızdırılırsa, kullanıcı gizliliğini ihlal ederse ya da platform sınırlarını kötüye kullanırsa uygulama acil olarak durdurulur. Durdurulan uygulamanın bot token'ı anında geçersiz kılınır (`401 invalid_bot_token` , gateway `4004`), açık tüm gateway ve ses bağlantıları kesilir, OAuth2 token'ları `401` döndürür, `/bot-ekle` kapanır ve bot rehberden düşer. Sorun giderilip uygulama yeniden açılsa bile eski bot token'ı kalıcı olarak geçersizdir; portaldan yeni bir token üretip `.env` dosyanı güncellemen gerekir.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Doğrulama metinlerini veya gizlilik politikanı hazırlarken yapay zekadan yardım alıyorsan kodunu birlikte verip şu soruları gerçekçi biçimde yanıtlat:
>
> - Bot veritabanında veya bellekte hangi snowflake kimliklerini ve alanları saklıyor?
> - Bot sunucudan çıkarıldığında (`GUILD_DELETE`) veya kullanıcı verisinin silinmesini istediğinde bu kayıtlar nasıl temizleniyor?
> - Bot `MESSAGE_CONTENT` , `GUILD_MEMBERS` veya `GUILD_PRESENCES` kullanıyorsa bunu hangi özellik için yapıyor; `/komut` ile yapılabilecek bir iş için gereksiz ayrıcalıklı intent açılmış mı?

Sırada ne var

- Rehbere girmek: doğrulanmış botunu `/kesfet` altındaki Botlar dizininde listeleme
- Intent'ler: normal ve ayrıcalıklı intent bitlerinin tam tablosu
- Güvenlik: token sızıntılarını önleme ve iki aşamalı doğrulama

Rehbere girmek

N'oluyo'nun `/kesfet` ekranındaki **Botlar** sekmesi, sunucu yöneticilerinin doğrulanmış botları kategoriye ve anahtar sözcüğe göre arayıp tek tıkla kendi sunucularına ekleyebildiği resmi dizindir. Web, masaüstü, Android ve iOS istemcilerinin tamamında aynı kartlarla görünür.

> **Not:** Rehber giriş yapmamış ziyaretçilere de açıktır (`GET /api/v1/discovery/bots?q=&category=&offset=` , sayfa başına 24 kayıt, en yüksek `offset` 960).

Listelenme koşulları

Bir botun rehberde görünmesi için üç koşulun aynı anda sağlanması gerekir:

1. Uygulama Doğrulama incelemesinden geçmiş olmalıdır (`verified: true`).
2. Doğrulama başvurusunda **Rehberde listelensin (`want_listing: true`)** kutusu işaretlenmiş ve onaylanmış olmalıdır (`listed_at` dolu).
3. Bot aktif ve dışarıya açık (`public: true`) durumda olmalıdır; durdurulan, silinen veya gizliye alınan botlar listeden anında düşer.

Kategoriler

`/gelistirici/uygulamalar` sayfasında (`PATCH /api/v1/applications/{application_id}`) botun ana işine en uygun tek kategoriyi (`listing_category`) seç:

Kategori anahtarı	Ne zaman seçilmeli
`moderation`	Otomatik sözcük süzgeci, zaman aşımı, denetim kaydı ve sunucu düzeni araçları
`music`	Ses kanallarında yerel ses veya müzik yayını yapan botlar
`games`	Sunucu içi mini oyunlar, soru-cevap, skor tablosu ve eşleştirme araçları
`utility`	Hatırlatıcı, anket, rol seçici, duyuru ve üretkenlik komutları
`fun`	Mizah, rastgele içerik ve hafif topluluk etkileşimleri
`social`	Karşılama kartları, topluluk etkinlikleri ve tanışma yardımcıları
`other`	Yukarıdaki altı gruba doğrudan girmeyen özel amaçlı botlar

Bot kartı nasıl görünür

Rehberdeki her kart kullanıcıya şu bilgileri gösterir:

- **Kimlik:** Uygulamanın avatarı, adı, doğrulanmış rozeti, seçilen kategori etiketi ve o an bulunduğu gerçek sunucu sayısı (`guild_count`).

- **Kısa tanıtım (`listing_text`):** Kartın gövdesinde görünen özet metin (`listing_text` boşsa uygulama açıklaması `description` kullanılır).
- **Sunucuya ekle düğmesi:** Yöneticinin doğrudan `/bot-ekle/<uygulama_id>?izin=<install_permissions>` ekranını açmasını sağlar.
- **Yardım ve şeffaflık bağlantıları:** `support_url`, varsa `website_url` ve `privacy_url`. Uygulama sahibinin kullanıcı kimliği, e-postası veya gizli anahtarları hiçbir zaman rehber yanıtında yer almaz.

İyi bir kısa tanıtım metni (`listing_text`) yazmak

Sunucu yöneticisi onlarca kart arasında göz gezdirirken ilk iki cümleye bakar:

- **İyi örnek:** *"`/anket` ile süreli buton oylamaları açar, `/hatirlat` ile kanallara zamanlı duyuru gönderir ve yeni üyelere seçtiğin karşılama rolünü verir."*
- **Kaçınılması gerekenler:** Tamamı büyük harfli sloganlar (*"EN İYİ TÜRKÇE BOT!!!"*), sahte sunucu sayıları (kart zaten gerçek `guild_count` sayısını otomatik gösterir) ve ilgisiz anahtar sözcük yığınları.

En az izin ilkesi (`install_permissions`)

Portalda belirlediğin `install_permissions` tam sayısı, rehberdeki **Sunucuya ekle** düğmesinin `?izin=` parametresini oluşturur. Yalnızca botun gerçekten çalışmak için ihtiyaç duyduğu izin bitlerini işaretle:

- Yalnızca eğik çizgi komutu yanıtlayan bir bot için `0` yeterlidir.
- Karşılama mesajı atan bir bot için `ViewChannel | SendMessages` yeterlidir; `Administrator` (`8`) istemek sunucu yöneticilerinin ekleme ekranından vazgeçmesinin bir numaralı sebebidir.
- İzin bitlerinin tam listesini ve değerlerini İzinler sayfasında bulabilirsin.

Destek ve web sitesi adresleri

- **`support_url` (`https://`):** Yöneticilerin soru sorabileceği ya da hata bildirebileceği N'oluyo destek sunucusu daveti veya destek sayfası. Doğrulanmış ve listelenmiş botlarda çalışır durumda kalması zorunludur.
- **`website_url` (`https://`, isteğe bağlı):** Komut listesini ya da kullanım kılavuzunu barındırdığın web sayfası.

Bilgileri güncelleme ve rehberden çıkma

- Rehber alanlarını (`listing_category`, `listing_text`, `install_permissions`, `support_url`, `website_url`) `/gelistirici/uygulamalar` üzerinden dilediğin an güncelleyebilirsin; yeni inceleme beklemeden karta yansır.
- Botunu rehberden kendin çekmek istersen portalda `public` anahtarını kapatabilirsin; mevcut sunuculardaki çalışması bozulmaz, yalnızca `/kesfet` listesinden ve yeni eklemelerden kalkar.
- Kurallara aykırı yanıltıcı tanıtım metni veya çalışmayan destek bağlantısı tespit edilirse bot rehberden çıkarılabilir (`listed_at` temizlenir); eksik giderildikten sonra yeniden başvuru yapılabilir.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Botunun rehber kartı ayarlarını yapay zekaya hazırlatırken şu üç kuralı hatırlat:
>
> - `listing_category` yalnızca `moderation`, `music`, `games`, `utility`, `fun`, `social` veya `other` değerlerinden biri olabilir.
> - `install_permissions` için `Administrator` (`8`) yerine yalnızca kodda çağrılan REST uçlarının gerektirdiği en küçük izin bitlerinin toplamı seçilmelidir.
> - `support_url`, `website_url` ve `privacy_url` adreslerinin tamamı `https://` ile başlamalıdır.

Sırada ne var

- Doğrulama: doğrulama önkoşulları ve inceleme başvurusu
- İzinler: `install_permissions` bitlerinin tam tablosu
- Bot uygulamaları: `/bot-ekle` kurulum akışı ve uygulama ayarları

Sık hatalar

Bot yazarken ya da çalıştırırken karşılaştığın belirtilerin büyük bölümü aynı birkaç nedenden çıkar. Önce aşağıdaki özet tabloya bak, ardından ilgili başlıktaki adımları uygula.

Hızlı teşhis tablosu

Belirti	En sık sebep	Hızlı çözüm	Ayrıntılı rehber
Bot üye listesinde çevrimdışı görünüyor	Program çalışmıyor ya da gateway'e hiç `identify` göndermedi	Terminalde `READY` satırını gör; yalnızca REST çağıran program botu çevrimiçi yapmaz	Gateway, Botu açık tut
`message.content` boş (`""`) geliyor	`MESSAGE_CONTENT` intent'i kapalı ya da kodda istenmemiş	Portalda **Mesaj içeriği**'ni aç, `intents` değerine ekle ve programı yeniden başlat	Intent'ler
Gateway `4014` ile kapanıyor	Portalda açılmamış (veya 100+ sunucuda onaylanmamış) ayrıcalıklı intent istendi	Portalda ilgili intent'i aç ya da `intents` değerinden çıkar	Intent'ler, Doğrulama
Gateway `4004` veya HTTP `401` dönüyor	Token yanlış, yenilenmiş ya da REST başlığında `Bot ` öneki eksik	`.env` içindeki token'ı kontrol et; gerekirse portaldan **Token'ı yenile** yap	Kimlik doğrulama
Gateway `4011` ile kapanıyor	Aynı bot iki yerde birden çalışıyor (örneğin bilgisayar + VPS)	Eski kopyayı durdur; iki süreç birbirini düşürmeye devam eder	Shard'lar, Botu açık tut
HTTP `403` (`missing_access` / `missing_permissions` / `role_hierarchy`)	Kanalı görme izni yok, işlem izni yok ya da botun rolü aşağıda	Kanalda `ViewChannel` ver, rolde izni aç ya da bot rolünü yukarı taşı	İzinler
HTTP `429` (`rate_limited` / `slowmode`)	Hız sınırı aşıldı veya kanalda yavaş mod açık	`retry_after` saniye bekle ya da resmi `noluyo-bot` kuyruğunu kullan	Hız sınırları
Eğik çizgi (`/`) komutu listede görünmüyor	Komut kaydedilmedi, kanalda `UseApplicationCommands` kapalı ya da 60 sn dolmadı	`commands.sync` çıktısını kontrol et, 1 dakika bekle, kanal izinlerine bak	Komutlar, Komut izinleri
Butona basınca "Bot yanıt vermedi." / etkileşim başarısız	Bot 3 saniye içinde `callback` dönmedi ya da uzun işi yanıttan önce yaptı	Önce `i.defer()` (`type: 5` / `6`) gönder, ağır işi ondan sonra çalıştır	Etkileşimler, Bileşenler
Ses kanalına girmiyor ya da ses çıkmıyor	`Connect` / `Speak` izni yok, `ffmpeg` kurulu değil ya da HTTP ses rotası çağrıldı	Gateway'den `op 4` gönder, `VOICE_JOIN_ERROR` koduna bak ve `ffmpeg -version` dene	Ses

Bot çevrimdışı görünüyor

Bir botun sunucu üye listesinde yeşil (çevrimiçi) görünmesi yalnızca aktif bir **gateway WebSocket oturumu** açıkken olur:

1. Terminal penceren açıksa programın çalışmaya devam ettiğini ve `READY` olayını aldığını doğrula.
2. Yalnızca `curl` ya da `fetch` ile mesaj gönderen bir betik botu çevrimiçi göstermez; mesaj yine gider ama durum çevrimdışı kalır.
3. Bilgisayarını kapattığında bot çevrimdışı düşüyorsa süreci bir sunucuya taşı; adımlar Botu açık tut sayfasında.

`message.content` boş geliyor

Hata mesajı almadığın halde `d.content === ""` ve `d.attachments === []` görüyorsan bu bir arıza değil, mesaj içeriği kuralıdır:

- `MESSAGE_CONTENT` ayrıcalıklı intent'i olmadan bot yalnızca **kendi yazdığı** ve **kendisini doğrudan etiketleyen** (`@bot`) mesajların metnini görür.
- Çözüm: Geliştirici portalında uygulamanı aç → **Ayrıcalıklı intent'ler** altında **Mesaj içeriği**'ni etkinleştir → kodunda `Intents.MESSAGE_CONTENT` (`1 << 15`) bitini ekle → **programını yeniden başlat** (açık oturum yeni intent'i kendiliğinden almaz).

Gateway `4014` veya `4004` ile kapanıyor

Bu iki kapanış kodunda **yeniden bağlanmayı deneme**; kodu ya da portal ayarını düzeltmeden her deneme aynı hatayla biter:

- **`4014 DISALLOWED_INTENTS`:** Kodunda `GUILD_MEMBERS` (`1 << 1`), `GUILD_PRESENCES` (`1 << 8`) veya `MESSAGE_CONTENT` (`1 << 15`) istedin ama portalda o kutu kapalı; ya da botun 100+ sunucuda ve o intent henüz doğrulama sırasında onaylanmadı.
- **`4004 AUTHENTICATION_FAILED`:** Token yanlış, başında/sonunda görünmez boşluk var, portalda yenilenmiş ya da uygulama durdurulmuş. Ayrıca yanlışlıkla kişilere ait `/gateway` adresine bağlanırsan da `4004` alırsın; adresi her zaman `GET /api/v1/gateway/bot` rotasından al.

HTTP `401` ve üç farklı `403` hatası

- **`401 unauthorized`:** REST isteğinde `Authorization: Bot <token>` başlığı eksik ya da `Bot` sözcüğü unutulmuş (`Bearer` değil, bot rotalarında `Bot` yazılır).
- **`403 missing_access`:** Botun o kanalı görme (`ViewChannel` , `1024`) izni yok. Kanal özel bir kanalsa kanal izinlerinden botun rolüne `ViewChannel` ekle.
- **`403 missing_permissions`:** Bot kanalı görüyor ama yapmak istediği işin izni (`SendMessages` , `ManageMessages` , `ManageRoles` gibi) rolünde yok ya da bot zaman aşımında.
- **`403 role_hierarchy`:** İzin var ama botun rütbesi yetmiyor. Yeni eklenen botun rolü `@everyone` rolünün hemen üstünde başlar; sunucu yöneticisi **Sunucu ayarları → Roller** ekranında botun rolünü hedef üyenin veya verilecek rolün **üstüne** sürüklemelidir. Ayrıntılar İzinler sayfasında.

HTTP `429` hız sınırı ve geçici engel

`429` aldığında yanıt gövdesindeki `retry_after` saniye kadar beklemeden aynı isteği tekrarlama. `X-RateLimit-Scope: shared` geliyorsa tüm platformun ortak bot tavanı dolmuştur; `X-RateLimit-Scope: user` veya `bot` geliyorsa kendi kotan dolmuştur.

Reddedilen (`401` , `403` , `429`) istekleri saniyede yüzlerce kez tekrarlayan bozuk bir döngü 10 dakikada 10.000 hataya ulaşırsa IP adresin geçici olarak `403 temporarily_banned` alır. Resmi JavaScript ve Python kütüphaneleri bucket kuyruğunu ve `429` beklemelerini senin yerine otomatik yönetir.

Eğik çizgi (`/`) komutu görünmüyor

1. Komutu kaydettiğinden emin ol (`await bot.commands.sync([...])` veya `PUT /applications/{id}/commands`).
2. İstemciler komut listesini en çok **60 saniyede bir** yeniler; kaydettikten sonra bir dakika bekle ya da başka kanala geçip dön.
3. Kişinin o kanalda **Uygulama komutlarını kullan** (`UseApplicationCommands`) izni kapalıysa ya da komutun `default_member_permissions` bitleri kişide yoksa komut o kişiye gizlenir.
4. Botla açılan DM'de deniyorsan komutun global olduğundan, `contexts` içinde `1` bulunduğundan ve `default_member_permissions` değerinin `null` olduğundan emin ol; bkz. Komutlar.

Buton ya da komut "Bot yanıt vermedi." diyor

N'oluyo her `INTERACTION_CREATE` olayına **3 saniye içinde** ilk yanıtın (`POST /interactions/{id}/{token}/callback`) gelmesini bekler:

- Yanıttan önce veritabanı sorgusu, dış API isteği ya da çoklu REST çağrısı yapıyorsan süre dolar ve `400 interaction_timeout` alırsın.
- Çözüm: İşleyici başlar başlamaz `await i.defer({ ephemeral: true })` (ham API'de `type: 5` veya bileşende `type: 6`) gönder; 3 saniyelik süreyi 15 dakikaya uzatmış olursun, işin bitince `await i.editReply(...)` ile sonucu yaz. Ayrıntılar Etkileşimler sayfasında.
- HTTP ile teslim kullanıyorsan portalda **Teslim günlüğü** tablosuna bak: imza doğrulaması (`401`) ya da TLS hatası orada açıkça yazar.

Ses kanalına girmiyor ya da ses çıkmıyor

1. Ses kanalına girmek için `POST /channels/{id}/voice` çağırma (`403 bot_forbidden` döner); açık gateway bağlantından `op 4` gönder veya SDK'da `bot.voice.join(...)` kullan.
2. `VOICE_JOIN_ERROR` olayı geliyorsa `code` alanını oku: `missing_permissions` ise ses kanalında `ViewChannel` , `Connect` ve `Speak` (`izin=3146752`) izinlerini ver; `channel_bot_limit` ise kanalda zaten 2 bot vardır.
3. Bot odaya giriyor ama ses gelmiyorsa sunucunda `ffmpeg -version` komutunun çalıştığını, sesin 48 kHz 2 kanal `s16le` PCM olarak çözüldüğünü ve `TrackSource.SOURCE_MICROPHONE` ile yayınlandığını doğrula. Ayrıntılar Ses sayfasında.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Botun hata verdiğinde terminal çıktısını (içinden `.env` ve token'ı çıkararak!) bu sayfayla birlikte yapay zekaya yapıştır. Özellikle `missing_permissions` ile `role_hierarchy` farkını ve `4014` intent kuralını kontrol etmesini iste.

Sırada ne var

- Hatalar: API'deki tüm makinece okunur hata kodlarının tam referans tablosu
- SSS: izinler, mesajlar, DM kuralları ve ephemeral mesajlar hakkında sık sorulan sorular
- İzinler: izin bitleri ve rol hiyerarşisi hesaplaması

Sözlük

Rehberlerde karşılaştığın teknik terimlerin her biri burada iki cümleyle açıklanır: ilk cümle günlük dille ne işe yaradığını, ikinci cümle ise API'deki tam teknik karşılığını söyler. Kavramların birbirine nasıl bağlandığını hikâye akışıyla okumak istersen Temel kavramlar sayfasına bak.

Token

Token, bot programının N'oluyo'ya kendisini tanıtmak için kullandığı gizli şifredir ve kimseyle paylaşılmaz. Teknik olarak REST isteklerinde `Authorization: Bot <token>` başlığında, gateway bağlantısında ise `op 2` (`IDENTIFY`) çerçevesinde gönderilen kimlik dizgisidir; OAuth2 kullanıcı girişinde ise `Bearer <access_token>` biçiminde ayrı erişim token'ları kullanılır.

REST API

REST API, botunun mesaj göndermek, rol vermek ya da kanal listelemek gibi bir işlem yapmak istediğinde N'oluyo'ya soru-cevap şeklinde istek yolladığı kapıdır. Teknik olarak `https://noluyo.net/api/v1` taban adresi altındaki standart HTTP (`GET` , `POST` , `PUT` , `PATCH` , `DELETE`) uçlarıdır; JSON ya da `multipart/form-data` gövdesi alır ve `X-RateLimit-*` hız sınırı başlıklarıyla JSON yanıtı döndürür.

Gateway

Gateway, sunucuda yeni bir mesaj yazıldığında ya da bir üye katıldığında N'oluyo'nun botuna anında haber uçurduğu sürekli açık telefon hattıdır. Teknik olarak `GET /api/v1/gateway/bot` ucundan öğrenilen adrese açılan sıkıştırmasız JSON `WebSocket` bağlantısıdır; `op 1` (`HEARTBEAT`) ile canlı tutulur ve koptuğunda `op 6` (`RESUME`) ile kaçan olayları kaybetmeden sürdürülür.

Intent

Intent, botunun sunuculardaki her gürültüyü değil yalnızca ilgilendiği haber başlıklarını dinlemek için yaptığı abonelik seçimidir. Teknik olarak `op 2` (`IDENTIFY`) çerçevesinde gönderilen bir tam sayı bit maskesidir; `GUILD_MEMBERS` (`1 << 1`), `GUILD_PRESENCES` (`1 << 8`) ve `MESSAGE_CONTENT` (`1 << 15`) ayrıcalıklıdır ve önce geliştirici portalında açılmamışsa bağlantıyı `4014` koduyla kapatır.

Shard

Shard, yüzlerce sunucuda bulunan büyük bir botun tek bir kablodan taşmamak için sunucularını birkaç paralel bağlantıya bölmesidir. Teknik olarak her sunucuyu `(BigInt(guild_id) >> 22n) % BigInt(count)` formülüyle `[id, count]` oturumlarından birine dağıtan gateway bölümleme düzenidir; 100 sunucunun üstünde zorunlu hale gelir ve sunucusuz DM olayları her zaman shard `0` oturumuna düşer.

Guild (Sunucu)

Sunucu, insanların metin ve ses kanallarında bir araya geldiği topluluk alanıdır; günlük konuşmada "sunucu" dediğimiz yere API "guild" adını verir. Teknik olarak kanalları, sıralı rolleri ve üyeleri barındıran, rotalarda `/guilds/{guild_id}` ve olaylarda `guild_id` alanıyla taşınan üst düzey topluluk nesnesidir.

Snowflake (ID)

Snowflake, N'oluyo'daki her kullanıcıya, sunucuya, kanala, mesaja ve role verilen, üretildiği anın zaman damgasını da içinde taşıyan benzersiz kimlik numarasıdır. Teknik olarak 64 bitlik işaretsiz bir tam sayıdır;

JavaScript'in 53 bitlik `Number` sınırında hassasiyet kaybı yaşanmaması için JSON gövdelerinde her zaman tırnak içinde ondalık metin (`"215318790144000001"`) olarak taşınır.

Embed

Embed, botun düz yazı mesajının altında beliren, renkli kenar şeridi, başlığı, tablo alanları ve resmi olan düzenli bilgi kartıdır. Teknik olarak mesaj gövdesinin `embeds` dizisinde (mesaj başına en çok 10 adet, toplam 6000 karakter) gönderilen yapılandırılmış JSON nesnesidir ve dış izleme bağlantısı sızıntısını önlemek için görsel olarak yalnızca aynı istekle yüklenen `attachment://dosya.png` eklerini kabul eder.

Ephemeral (Yalnızca sana görünür)

Ephemeral mesaj, bir butona basan ya da komut çalıştıran kişiye kanaldaki diğer kimseyi rahatsız etmeden gösterilen ve sayfa yenilenince kaybolan özel yanıttır. Teknik olarak etkileşim yanıtlarında ve takip mesajlarında `flags: 64` bitiyle işaretlenen, veritabanına yazılmayan ve dosya eki taşıyamayan geçici teslim biçimidir.

Etkileşim (Interaction)

Etkileşim, bir kullanıcının botunun eğik çizgi komutunu çalıştırması, butonuna basması, menüsünden seçim yapması ya da açtığı formu göndermesiyle başlayan karşılıklı konuşma adımıdır. Teknik olarak bota `INTERACTION_CREATE` olayı (veya imzalı HTTP isteği) olarak ulaşan, **3 saniye içinde** ilk `callback` yanıtını zorunlu kılan ve `token` değeriyle 15 dakika boyunca mesaj düzenleme ya da takip mesajı izni veren nesnedir.

Webhook / HTTP teslim

Webhook ve HTTP teslim, açık bir WebSocket bağlantısı tutmak yerine mesaj göndermeyi ya da etkileşim almayı doğrudan HTTPS adresleri üzerinden yürüten yöntemdir. Teknik olarak etkileşim tarafında N'oluyo'nun botun kayıtlı HTTPS adresine `X-Signature-Ed25519` ve `X-Signature-Timestamp` başlıklarıyla imzalı `POST` isteği yollaması (HTTP ile teslim), kanal tarafında ise bir webhook adresine JSON gövdesi göndererek kanala mesaj bırakılmasıdır.

Kapsam (OAuth2 Scope)

Kapsam, kendi web sitene "N'oluyo ile giriş yap" koyduğunda kullanıcıdan hesabının tam olarak hangi bilgisini okumak için izin istediğini belirten yetki etiketidir. Teknik olarak `/oauth2/authorize` sorgusunda boşlukla ayrılarak gönderilen (`identify` , `email` , `guilds`) dizgilerdir; kullanıcı şifresini, özel mesajlarını ya da mesaj yazma yetkisini hiçbir kapsam açmaz.

PKCE

PKCE, kullanıcının giriş onayından dönen tek kullanımlık yetki kodunun yolda başka bir uygulama tarafından çalınıp kullanılmasını engelleyen kriptografik güvenlik kilididir. Teknik olarak istemcinin rastgele ürettiği `code_verifier` dizgisinin SHA-256 `base64url` özetini (`code_challenge` , `S256`) başta göndermesi ve `/api/v1/oauth2/token` adımında asıl `code_verifier` değerini ibraz ederek kanıtlamasıdır; N'oluyo'da tüm OAuth2 istemcileri için zorunludur.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Bu sözlük N'oluyo'ya özgü kesin sınırları (64 bit metin Snowflake ID'ler, 3 saniyelik etkileşim yanıtı, yalnızca `attachment://` kabul eden embed görselleri ve zorunlu `S256` PKCE) özetler; yapay zeka terimleri başka platformlarla karıştırdığında bu sayfayı bağlama ekleyebilirsin.

Sırada ne var

- Temel kavramlar: bu terimlerin bir isteğin yolculuğunda nasıl bir araya geldiği
- Başka platformdan gelenler: tanıdık kavramlar ile N'oluyo'nun bilinçli farkları
- Hızlı başlangıç: ilk botunu birkaç dakikada ayağa kaldırmak

Başka platformdan gelenler

Daha önce başka bir sohbet platformunda bot geliştirdiysen N'oluyo'nun kavramlarının büyük bölümü sana ilk dakikadan tanıdık gelecektir. Ancak N'oluyo başka bir platformun birebir kopyası veya uyumluluk katmanı değildir: gizlilik, güvenlik ve öngörülebilirlik için bilinçli olarak farklı tasarlanmış kuralları vardır.

> **Uyarı: Başka platformun kütüphanesiyle bağlanmaya çalışma.** Yabancı bir kütüphanenin taban adresini değiştirip N'oluyo'ya yönlendirmeye kalktığında sıkıştırma, opcode farkları, görsel kuralları ve ses protokolü yüzünden bağlantın kırılır. Sıfır bağımlılıklı resmi JavaScript veya Python `noluyo-bot` kütüphanesini ya da doğrudan standart `fetch` + `WebSocket` çağrılarını kullan.

Tanıdık gelenler

Şu yapı taşları bildiğin zihinsel modelle aynı şekilde çalışır:

- **Snowflake kimlikler:** 64 bitlik zaman sıralı kimlikler JSON içinde her zaman ondalık metin (`"21531879014400001"`) olarak taşınır; sağ kaydırma (`id >> 22`) mantığı aynıdır.
- **Bot kimlik başlığı:** REST rotalarında `Authorization: Bot <token>`, gateway `op 2` (`IDENTIFY`) çerçevesinde ise öneksiz düz `token` gönderilir.
- **Intent bit maskesi ve ayrıcalıklı üçlü:** Hangi olayların geleceğini `intents` sayısı belirler; `GUILD_MEMBERS` (`1 << 1`), `GUILD_PRESENCES` (`1 << 8`) ve `MESSAGE_CONTENT` (`1 << 15`) ayrıcalıklıdır, portaldan açılır ve 100 sunucudan itibaren onay ister.
- **Shard mantığı ve oturum sürdürme:** `[shard_id, shard_count]` dizisi, `op 1` (`HEARTBEAT`) / `op 11` (`HEARTBEAT_ACK`), `READY` ardından tek tek akan `GUILD_CREATE` olayları ve kopunca `op 6` (`RESUME`) ile devam etme akışı tanıdıktır.
- **Zengin mesajlar ve etkileşimler:** `embeds`, eylem satırlı (`type: 1`) buton ve seçim menüleri (`components`), eğik çizgi (`/`) ve sağ tık komutları, açılır formlar (`modal`), 3 saniyelik ilk yanıt kuralı, `flags: 64` ephemeral yanıtlar ve Ed25519 imzalı HTTP etkileşim teslimi aynı kavramlarla çalışır.
- **Hız sınırı başlıkları:** Yanıtlarda `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset-After`, `X-RateLimit-Bucket` ve `429` durumunda `Retry-After` ile `X-RateLimit-Scope` döner.

Bilinçli olarak farklı olanlar

Başka bir platformdaki botunu N'oluyo'ya taşırken takılmamak için aşağıdaki fark tablosunu baştan kontrol et:

Konu	N'oluyo'daki kural	Neden böyle ve ne yapmalısın
Taban adres ve sürüm	`https://noluyo.net/api/v1` ve bot gateway adresi için `GET /api/v1/gateway/bot` (`wss://noluyo.net/bot-gateway`)	`/api/v9` veya `/api/v10` yoktur; insan istemcilerinin kullandığı `/gateway` yolu bot token'ını `4004` ile reddeder.
Gateway opcode `30` ve `31`	`op 30` (`PHONE_CAMERA`) ve `op 31` (`ACTIVE`) yalnızca insan uygulamalarına aittir; gönderen bot **`4001 UNKNOWN_OPCODE`** ile kapatılır	Sunucu abonelikleri için istemci tarafı özel opcode gönderilmez; sunucu olayları `op 2` içindeki `intents` bitleriyle gelir.
Gateway sıkıştırması yok	Bot gateway bağlantısında `zlib-stream` veya `zstd` sıkıştırması sunulmaz; `?compress=` parametresi yok sayılır	Her çerçeve her iki yönde de düz UTF-8 JSON metnidir; ek sıkıştırma çözücüsü kurma.
Embed görselleri yalnızca `attachment://`	`embeds[].image.url` , `thumbnail.url` , `author.icon_url` ve `footer.icon_url` **yalnızca** aynı mesajla yüklenen `attachment://dosya.png` değerini kabul eder	Dış `https://` görsel adresi kullanıcıların IP adresini üçüncü taraf sunuculara sızdırabileceği için `400 validation_failed` ile reddedilir. Görseli `multipart/form-data` ile ekle (Embed'ler).
Seste sıfır dinleme (yalnız yayın)	Botlar seste `op 4` ve **LiveKit** (`wss://noluyo.net/rtc`) üzerinden yalnızca tek bir mikrofon ses izi yayınlar; `canSubscribe: false` zorunludur	Özel UDP ses şifrelemesi yerine standart LiveKit WebRTC kullanılır ve medya sunucusu bota hiçbir ses karesi göndermez (Ses).
Sıkı DM gizliliği ve 24 saat penceresi	`POST /users/@me/channels` botlara kapalıdır (`403 bot_forbidden`); bot asla ilk DM'yi atamaz	DM kanalını yalnızca kişi açar. Kişinin son mesajı, komutu veya buton basmasından sonra botun 24 saat yanıt penceresi vardır; sonrasında `403 dm_window_closed` döner (Özel mesajlar).
Katı JSON ve alan doğrulaması	Kişi/uygulama rotalarında bilinmeyen JSON alanları `400 invalid_body` ile reddedilir; bot rotalarında sayısal ID'ler ve şema dışı değerler reddedilir	ID ve izin bitlerini her zaman tırnak içinde metin (`"68608"`) olarak gönder; yalnızca REST referansında belgelenen alanları kullan.
Ekleme bağlantısı biçimi	Bot ekleme adresi `/oauth2/authorize?scope=bot` değil, doğrudan `https://noluyo.net/bot-ekle/<application_id>?izin=<deger>` şeklindedir	`/oauth2/authorize` yalnızca N'oluyo ile giriş yap (kullanıcı kimlik doğrulaması) içindir ve `S256` PKCE zorunludur.
Ortak platform bütçesi	Botların yoğun yazma ve büyük embed yayılma istekleri ortak koruma tavanına tabidir (`X-RateLimit-Scope: shared` ve `fanout_too_large`)	`429` aldığında `X-RateLimit-Scope: shared` gelebilir; `retry_after` süresine uy (Hız sınırları).

Taşıma kontrol listesi

Var olan bir botun iş mantığını N'oluyo'ya taşırken şu sırayı izle:

1. Eski kütüphane bağımlılığını kaldırıp `noluyo-bot` paketini (JS veya Python) kur.
2. Tüm dış URL'li embed görsellerini yerel dosya veya bellek tamponu yüklemesiyle `attachment://...` biçimine çevir.
3. Üyelere kendiliğinden özel mesaj (`DM`) atan akışları kanal içi ephemeral (`flags: 64`) yanıtlara dönüştür; üye bota kendisi DM açarsa 24 saat kuralına ve `dm_closed` hatasına göre davran.
4. Ses çalan kodun varsa özel ses soketi paketlerini çıkarıp `bot.voice.join(...)` ve `@livekit/rtc-node` / `livekit` akışına geçir; ses dinleme/kayıt özelliği varsa kaldır.
5. Sunucu ekleme bağlantısını `https://noluyo.net/bot-ekle/<app_id>?izin=...` olarak güncelle.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Başka bir platform için yazılmış eski bot kodunu N'oluyo'ya çevirtirken eski kodun yanına bu sayfayı ve `/gelistirici/llms-kisa.txt` dosyasını ekle. Yapay zekaya eski kütüphaneyi tamamen silip `noluyo-bot` kullanmasını, dış embed resimlerini `attachment://` yapmasını ve otomatik DM gönderen satırları ephemeral yanıta çevirmesini açıkça söyle.

Sırada ne var

- JavaScript kütüphanesi ve Python kütüphanesi: resmi `noluyo-bot` SDK'ları ile temiz başlangıç
- Embed'ler: `attachment://` görsel kuralı ve kart sınırları
- Özel mesajlar ve Ses: DM penceresi ve LiveKit tabanlı yayın modeli

Güvenlik

Bot, sunucuda gerçek işler yapan bir hesaptır: mesaj siler, üye çıkarır, rol verir. Bu sayfa botunu ve ekli olduğu sunucuları nasıl koruyacağını anlatır.

Token'ı güvende tut

Token, botun şifresidir. Onu bilen herkes senin botun adına, onun tüm izinleriyle iş yapabilir.

> **Uyarı:** Token'ı koda yazma, depoya gönderme, ekran görüntüsünde ve paylaşılan ekranda gösterme, hiç kimseyle paylaşma.

- **Ortam değişkeninde tut.** Kod `process.env.BOT_TOKEN` okusun; değer kodun dışında kalsın.
- `.env` **dosyasını depo dışında bırak.** Token'ı dosyada tutuyorsan bu dosyayı `.gitignore`'a ekle:

```
echo ".env" >> .gitignore
```

- **Tarayıcıya koyma.** Sayfa kodu, herkesin okuyabildiği koddur. Token yalnızca senin kontrol ettiğin bir sunucuda veya bilgisayarda çalışan programda dursun.
- **Günlüklere yazma.** İstekleri kaydediyorsan `Authorization` başlığını kaydetme. Gateway çerçevelerini kaydediyorsan `op 2` ve `op 6` çerçevelerini de: token onların içinde.
- **Her bot için ayrı uygulama aç.** Biri sızarsa diğerleri etkilenmez.

Token'la bir kişinin oturum token'ını da karıştırma. Portal rotaları kişi oturumu ister ve o oturum hesabının tamamına erişir; bir botun kodunda veya sunucusunda asla durmamalıdır.

Token sızarsa

Token'ın başkasının eline geçtiğinden şüpheleniyorsan bekleme:

1. Portalda uygulamanı aç ve **Token'ı yenile**'ye bas. Eski token o anda çalışmayı bırakır.
2. Yeni token'ı programına koyup programı yeniden başlat.
3. Botun bulunduğu sunuculardaki denetim kaydına bak. Bot adına yapılmış, senin yapmadığın işlemleri sunucu yöneticilerine bildir.
4. Token bir depoya gittiyse dosyadan silmek yetmez, depo geçmişinde kalır. Bu yüzden önce token'ı yenile; asıl koruma odur.

Uygulamayı tamamen kaldırmak istersen uygulama sayfasında **Uygulamayı sil** var. Bot tüm sunuculardan çıkar ve token çalışmayı bırakır; gönderdiği mesajlar kalır.

En az izinle çalış

Bota ihtiyacından fazla izin vermek, token sızdığında zararı büyütür. Bir hata da aynı şekilde daha çok şeyi bozabilir.

- **Yalnızca kullanacağı izinleri iste.** Ekleme bağlantısında yalnızca gereken kutuları işaretle. Her örnekte hangi iznin gerektiği Örnekler sayfasında yazıyor, her rotanınki de REST referansında.
- `Administrator` **isteme.** Bu izin her şeye izin verir ve kanal ayarlarından etkilenmez. Neredeyse hiçbir bot buna gerçekten ihtiyaç duymaz; ekleme sayfası da bunu kullanıcıya ayrıca uyarı olarak gösterir.
- **Botu yalnızca gereken kanallarda çalıştır.** Sunucu yöneticisi, kanal ayarlarıyla botun rolünün kanal iznini kaldırabilir.
- **Rolü gerektiği kadar yukarı taşı.** Bot, rolünün altındaki üyeler üzerinde işlem yapabilir. Rolü en üste koymak gereksiz yetki demektir.
- **Herkese açık seçeneğini gerekince aç.** Uygulama başlangıçta özeldir: botu yalnızca sen eklersin.

Kullanıcı girdisine güvenme

Bot okuduğu her mesajı dışarıdan gelen veri olarak görmelidir.

- Mesajdaki metni kabuk komutu, SQL veya dosya adı içinde doğrudan kullanma.
- Kullanıcının yazdığını olduğu gibi tekrar eden bir botta dikkatli ol. Botun `MentionEveryone` izni varsa, tekrar ettiği `@everyone` gerçekten herkese bildirim gönderir. Bu izni yalnızca gerekirse ver.
- Yönetim işlemi yapan komutlarda, komutu yazan kişinin yetkisini de kontrol et. Üyeyi getir rotası üyenin rollerini söyler; yetkisi olmayan birinin yazdığı `!ban` komutunu botun çalıştırmamalıdır.

Basan kişinin yetkisini denetle

Sunucu, `INTERACTION_CREATE` ile gelen `custom_id`'nin mesajda gerçekten durduğunu doğrular; yani kimse botuna uydurma bir değer yollayamaz. Ama basan kişinin yetkisini denetlemez: kanalı gören herkes basabilir. Yönetim işi yapan bir butonda işi yapmadan önce `member.permissions`'a bak. Olaydaki `token` da gizlidir, log'a yazma. `custom_id`'ye ve bileşen yazılarına gizli bilgi koyma: kanalı gören herkesin istemcisine giderler. Örnek Bileşenler sayfasında.

Formlar

Formda şifre, doğrulama kodu, kart bilgisi, kimlik numarası ya da token isteme; geliştirici koşulları bunu yasaklar ve böyle bir bot kapatılır. Kişi her formun altında "Bu form {bot} botuna gider. Şifreni ya da doğrulama kodunu asla yazma." satırını görür; bu satırı gizleyemez, değiştiremezsin. Yalnızca işin için gereken alanı sor ve gelen içeriği işin bitince saklama. Gönderilen formun alanlarını da kullanıcı girdisi say: sunucu uzunluğu ve menü değerlerini denetler, ama metnin ne dediğini denetlemez.

Etkileşim adresi

Etkileşimleri HTTP ile alıyorsan adresine gelen her isteğin imzasını, JSON'a çevirmeden önce ham gövde üzerinde doğrula ve tutmazsa `401` dön. 5 dakikadan eski zaman damgasını reddet ve aynı `id`'yi ikinci kez işleme. Adresi bilen herkes oraya istek yollayabilir; yalnızca imza N'oluyo'dan geldiğini kanıtlar. Adresin sorgu dizesi yalnızca sana görünür, ama gizli bir değeri yalnızca ona emanet etme.

Hız sınırlarına saygı göster

Sınırların nasıl sayıldığı Hız sınırları sayfasında. İyi davranan bir bot şunları yapar:

- `429` **gelince bekler.** Yanıttaki `retry_after` kadar bekleyip yeniden dener. Beklemeden denemek işe yaramaz ve bir sonraki maddeye götürür.
- `401` **gelince durur.** Token geçersizse tekrar denemek düzeltmez; programı durdurup token'a bak.
- `403`**'ü tekrar etmez.** İzin eksikse aynı istek yine reddedilir.
- **Kanalı boşuna okumaz.** Yeni mesajları gateway anında haber verir, kanalı aralıklarla okumaya gerek yoktur. Geçmişi okurken tek istek 100 mesaj döndürebildiği için `after` ile yalnızca gerekenleri iste.
- `4004` **gelince yeniden bağlanmaz.** Kapanış kodu yeniden bağlanmanın işe yaramayacağını söylüyorsa döngüye girme; tablo Gateway sayfasında.
- **Yükü yayar.** Çok kanala aynı anda mesaj göndereceksen araya küçük beklemeler koy. Tüm rotalarda bot başına saniyede 50 istek sınırı vardır.

> **Uyarı:** Bir IP adresinden gelen, `Bot` şemasını kullanan reddedilmiş istekler (`401` , `403` , `429`) sayılır. 10 dakikada 10000'e ulaşırsa o adresten gelen tüm bot istekleri, geçerli token'lar dahil, süre dolana kadar `403 temporarily_banned` ile reddedilir. Hata döngüsüne giren bir bot buna kolayca yol açar.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Yapay zekaya kod yazdırırken veya hata ayıklatırken `.env` dosyanı, bot token'ını ya da OAuth2 `client_secret` değerini sohbete yapıştırma; kodun her gizli anahtarı ortam değişkeninden okumasını ve `.gitignore` içine `.env` eklemesini iste.

Sırada ne var

- Kimlik doğrulama: `Authorization: Bot <token>` ve başlık kuralları
- HTTP ile teslim: Ed25519 istek imzası doğrulama
- Botu açık tut: Linux sunucuda `.env` izinlerini (`chmod 600`) ve `systemd` servisini yapılandırma

Hatalar

Başarısız her istek aynı biçimde bir JSON gövdesi döndürür:

```
{
  "code": "validation_failed",
  "message": "One or more fields are invalid.",
  "errors": { "content": "required" }
}
```

Alan	Anlamı
`code`	Değişmeyen, makinece okunur kod. Kararını buna göre ver.
`message`	Geliştiriciler için açıklama. Değişebilir, kullanıcıya olduğu gibi gösterme.
`errors`	Yalnızca `validation_failed` ile: alan başına bir hata kodu.
`retry_after`	Yalnızca hız sınırı ve geçici engel hatalarında: beklenecek saniye.

`message` metinleri İngilizcedir. Tek istisna `maintenance`'tır: `Accept-Language` başlığı İngilizceyi öne çıkarıyorsa İngilizce, aksi halde Türkçe gelir.

Alan hata kodları

`validation_failed` ile gelen `errors` içinde görülür:

Kod	Anlamı
`required`	Alan zorunlu ve boş
`invalid`	Değer geçersiz
`too_short`	Çok kısa
`too_long`	Çok uzun
`taken`	Bu değer alınmış, örneğin kullanıcı adı

Listeler ve sorgu parametreleri için de aynı biçim kullanılır: `limit` aralık dışındaysa `{"limit":"invalid"}` döner.

Kimlik ve erişim

HTTP	Kod	Anlamı	Ne yapmalı
401	`unauthorized`	Token eksik, yanlış veya artık geçersiz.	Tekrar deneme. Token'ı ve `Bot` önekini kontrol et; gerekirse portaldan yenile.
403	`bot_forbidden`	Bu rota botlara kapalı.	Rotayı kullanma. Bota açık rotalar REST referansında.
403	`temporarily_banned`	Bu IP adresinden çok fazla reddedilmiş bot isteği geldi.	`retry_after` kadar bekle, reddedilen istekleri tekrarlamayı bırak. Bkz. Hız sınırları.
403	`missing_access`	Kanalı görme izni yok.	Bota `ViewChannel` verilmeli.
403	`missing_permissions`	Gereken izin yok ya da bot susturulmuş.	Rotanın izinleri REST referansında yazar.
403	`role_hierarchy`	Hedef veya rol, botun rütbesinden aşağıda değil.	Botun rolü yukarı taşınmalı. Bkz. İzinler.

Bulunamayanlar

Üyesi olmadığın bir sunucu `unknown_guild`, görmediğin bir sunucunun kanalı `unknown_channel` döner; böylece var olup olmadıkları açığa çıkmaz.

HTTP	Kod	Anlamı
404	`not_found`	Böyle bir rota yok. Yöntem yanlışsa da bu döner.
404	`unknown_guild`	Sunucu yok ya da üyesi değilsin.
404	`unknown_channel`	Kanal yok ya da erişimin yok.
404	`unknown_role`	Rol yok.
404	`unknown_member`	Kişi bu sunucunun üyesi değil.
404	`unknown_user`	Kullanıcı yok.
404	`unknown_message`	Mesaj bu kanalda yok.
404	`unknown_ban`	Bu kullanıcı yasaklı değil.
404	`unknown_application`	Uygulama yok ya da senin değil.
404	`unknown_command`	Komut yok ya da bu kapsamda değil: sunucu komutu global rotadan, global komut sunucu rotasından bulunmaz.

İstek hataları

HTTP	Kod	Anlamı
400	`invalid_body`	Gövde tek bir geçerli JSON nesnesi değil, bir alanın türü yanlış (örneğin metin yerine sayı olarak ID) ya da multipart parçası tanınmıyor.
400	`validation_failed`	Bir veya daha fazla alan geçersiz; ayrıntı `errors` içinde.
413	`body_too_large`	Gövde çok büyük.
413	`file_too_large`	Dosya yükleme sınırını aşıyor.
400	`invalid_channel_type`	İşlem bu kanal türünde yapılamaz, örneğin ses kanalına mesaj.
400	`unknown_emoji`	Emoji geçersiz, bulunamadı veya bot kullanamıyor.
400	`invalid_image`	Görsel PNG, JPEG, WebP veya GIF değil.
413	`image_too_large`	Görsel boyut sınırını aşıyor.

Mesajlar, roller ve moderasyon

HTTP	Kod	Anlamı
403	`not_author`	Yalnızca kendi mesajını düzenleyebilirsin.
403	`system_message`	Sistem mesajı üzerinde bu işlem yapılamaz.
400	`ephemeral_forbidden`	Kanala giden mesajda `flags: 64` (ephemeral) gönderildi; kanal mesajının gösterileceği tek bir kişi yoktur. `flags` alanını çıkar.
400	`fanout_too_large`	Mesajın embed ve bileşenleri, sunucunun üye sayısıyla çarpılınca yayılma bütçesinin 30 saniyesinden fazla tutuyor. `max_bytes` o kanalın alabileceği en büyük boyuttur; embed'leri küçült.
400	`automod_blocked`	Sunucunun otomatik moderasyonu mesajı engelledi. `message`, sunucunun yazdığı metin olabilir. Kural ayrıca göndereni susturabilir. Sunucu sahibi ve `ManageGuild` izni olanlar muaftır.
400	`messages_too_old`	Toplu silmede 14 günden eski mesaj var.
400	`max_pins`	Kanal sabitleme sınırında.
400	`too_many_reactions`	Mesaj farklı emoji sınırında.
400	`everyone_role`	`@everyone` silinemez, verilemez, adı veya sırası değiştirilemez.
400	`managed_role`	Yönetilen rol silinemez, verilemez, alınamaz, otomatik atanamaz.
400	`max_roles`	Sunucu rol sınırında.
400	`max_channels`	Sunucu kanal sınırında.
400	`max_overwrites`	Kanal, izin geçersiz kılma sınırında.
400	`cannot_target_self`	Bu işlem kendine uygulanamaz.
403	`cannot_timeout`	Sahip ve yöneticiler susturulamaz.
400	`not_in_voice`	Kişi seste değil.
400	`owner_cannot_leave`	Sunucunun sahibi sunucudan ayrılamaz.

Uygulamalar ve bot ekleme

HTTP	Kod	Anlamı
403	`bots_closed`	Yalnızca platform anahtarı kapalıyken döner; açık platformda üretilmez.
403	`bot_developer_required`	Yalnızca kapalı dönemde üretilir; açık platformda her doğrulanmış hesap uygulama oluşturabilir.
403	`developer_terms_required`	Hesap geliştirici sözleşmesini kabul etmedi; portaldan kabul et.
403	`email_not_verified`	E-posta doğrulanmamış.
403	`guest_not_allowed`	Misafir hesabı uygulama oluşturamaz.
400	`max_applications`	Hesap, uygulama sınırında.
403	`bot_private`	Bu botu yalnızca sahibi ekleyebilir.
403	`banned`	Bot bu sunucudan yasaklı.
400	`bot_in_guild`	Bot zaten sunucuda.
400	`bot_guild_limit`	Bot, sunucu sınırında.
403	`member_list_intent_required`	Üye listesi `GUILD_MEMBERS` ister: portalda açık, 100 sunucudan itibaren de onaylı olmalı.
400	`intent_approval_required`	Bot 100 ya da daha çok sunucuda ve ayrıcalıklı intent henüz onaylanmadı.
400	`review_requirements`	Doğrulama önkoşulları eksik (`errors` içinde `public` , `description` , `avatar` , `privacy_url` , `support_url` , `owner_mfa`). Doğrulama
400	`review_pending`	Bu uygulama için zaten bekleyen bir doğrulama başvurusu var. Doğrulama
409	`review_decided`	Bu doğrulama başvurusu zaten sonuçlandırılmış. Doğrulama
400	`max_commands`	Bu kapsam bu türden komut sınırında: 100 eğik çizgi komutu, her menü türünden 15. Komutlar
403	`command_permission_escalation`	Kişiye döner, bota değil: komut izinlerini düzenleyen kişi, komutun `default_member_permissions` bitlerine kendisi sahip değil. Sahip ve `Administrator` bunu almaz. Komut izinleri

Bileşene basma

Bunlar botuna değil, basan kişiye döner. Ayrıntı Bileşenler sayfasında.

HTTP	Kod	Anlamı
404	`component_not_found`	Mesajda artık bu `custom_id` yok: bot butonu kaldırmış ya da mesajı bir bot göndermemiş.
400	`component_disabled`	Bileşen kapalı.
409	`application_unavailable`	Mesajı gönderen uygulama silinmiş, kapatılmış ya da botu artık burada değil.
429	`interaction_busy`	Basmalar uygulamanın ya da tüm botların sınırına takıldı; HTTP ile teslimde adresine aynı anda 16 teslim gidiyor ya da adresinin yanıtı tüm botların tavanına takıldı. `retry_after` kadar bekle.
504	`application_no_response`	Uygulamanın etkileşim adresi 3 saniyede kullanılabilir bir yanıt vermedi, yanıtı reddedildi ya da adres art arda hata verdiği için şu an aranmıyor. Sebebi teslim günlüğü söyler.
403	`cannot_dm`	Kişi DM'de bu botu engellemiş; engeli kaldırana dek basamaz.

Komut çalıştırma

Bunlar da botuna değil, komutu çalıştıran kişiye döner; olay sana hiç gelmez. Kişi bir komutun boşa gittiğini söylüyorsa sebep çoğunlukla buradadır. Komut çalıştırma ve form gönderme basmalarla aynı sınırları paylaşır, `interaction_busy` burada da gelebilir. Otomatik tamamlama da aynı kodları döner, ama kendi sınırları vardır. Ayrıntı Komutlar sayfasında.

HTTP	Kod	Anlamı
409	`command_outdated`	Komutu, kişinin ekranı onu yükledikten sonra değiştirdin. Uygulama yeni tanımı alır, uyan değerleri korur ve kişi yeniden gönderir.
403	`command_forbidden`	Kişi komutu bu kanalda kullanamaz: `UseApplicationCommands` yok, zaman aşımında, bir komut izni kuralı engelliyor ya da `default_member_permissions` bitleri eksik.
400	`invalid_target`	Seçenekteki ya da menüdeki kullanıcı, rol, kanal veya mesaj kişinin burada gösterebileceği biri değil: üye değil, başka sunucunun rolü, görmediği kanal, `channel_types` dışında bir kanal, başka kanalın mesajı, silinmiş ya da ephemeral mesaj, ya da kişinin `ReadMessageHistory` izni yok.
400	`command_channel`	Bu kanalda komut çalışmaz. Komutlar sunucunun yazı kanallarında ve kişinin botla açtığı DM'de çalışır; iki kişinin DM'sinde ve grup DM'de değil.
404	`unknown_command`	Komut silinmiş ya da başka sunucunun; ya da bot sunucudan çıkarılmış veya uygulaması kapalı. DM'de: komut başka botun, sunucu komutu, `contexts` içinde `1` yok ya da `default_member_permissions` dolu.
403	`missing_access`	DM'de: kişi botla DM'yi kapatmış; yeniden açınca komut çalışır.
403	`cannot_dm`	DM'de: kişi botu engellemiş; engeli kaldırana dek komut çalıştıramaz.
400	`modal_expired`	Form gönderiminde: form 1 saatten eski, üç kez gönderildi ya da başka bir kişinin veya kanalın. Kişi formu yeniden açar.

Etkileşim yanıtları

Bunlar botuna döner. Ne yapacağın Etkileşimler sayfasında.

HTTP	Kod	Anlamı
400	`interaction_token_invalid`	Token bozuk, adresteki kimlikle uyuşmuyor ya da başka bir botun.
400	`interaction_timeout`	İlk yanıt 3 saniyeyi geçti.
400	`interaction_already_acknowledged`	Bu etkileşime zaten yanıt verdin.
400	`interaction_not_acknowledged`	Takip ya da düzenleme ilk yanıttan önce geldi.
400	`interaction_response_invalid`	`type` bu etkileşime uymuyor, `type: 7` 'de `data` yok, `choices` `type: 8` dışında geldi, `type: 9` otomatik tamamlamaya ya da gönderilen forma gitti veya yanında mesaj alanı var, ya da otomatik tamamlamanın token'ıyla takip mesajı veya düzenleme denendi.
400	`interaction_expired`	Token'ın 15 dakikası doldu. Kişi ephemeral mesajın 15 dakikası dolmuş butonuna bastığında da bunu alır.
400	`ephemeral_files`	Ephemeral mesajda dosya ya da `attachment://` var; bu mesajın dosya bağlanacak satırı yok.
404	`unknown_channel`	Ephemeral mesajda: basan kişi artık kanalı görmüyor, ona bir şey gitmedi.
403	`dm_closed`	DM'de: kişi DM'yi kapattı ya da botu engelledi. Yanıt da gitmez.

Özel mesajlar

Bir kişinin botunla açtığı DM'de kendiliğinden yazarken (mesaj, tepki, sabitleme, yazıyor) dönenler. Ayrıntılar Özel mesajlar sayfasında.

HTTP	Kod	Anlamı
403	`dm_closed`	Kişi DM'yi kapattı ya da botu engelledi; ikisi aynı görünür. O kişiye yazmayı bırak, yeniden deneme.
403	`dm_window_closed`	Kişinin son mesajı, komutu ya da basmasından beri 24 saat geçti. Kişi yeniden yazana dek bekle.
429	`rate_limited`	`X-RateLimit-Scope: bot` ile: tüm DM'lerindeki serbest yazma sınırı (`dm_send`) doldu. `retry_after` kadar bekle.

Etkileşim adresi

Portalda adres kaydedilirken dönenler. Yalnızca hatanın türü söylenir. Ayrıntılar HTTP ile teslim sayfasında.

HTTP	Kod	Anlamı
400	`endpoint_not_allowed`	Adres https, 443 ve genel bir ad değil, ya da N'oluyo'nun kendi adresi.
400	`endpoint_unreachable`	Adrese ulaşılamadı ya da 3 saniyede yanıt gelmedi.
400	`endpoint_bad_response`	Doğru imzalı `PING`'e `200 {"type":1}` dönmedi.
400	`endpoint_no_signature_check`	Yanlış imzalı isteğe `401` dönmedi: imzayı doğrula.

Hız sınırları

HTTP	Kod	Anlamı
429	`rate_limited`	Hız sınırı aşıldı. `retry_after` kadar bekle.
429	`slowmode`	Kanalda yavaş mod açık. `retry_after` kadar bekle.
429	`upload_quota`	Kısa sürede çok fazla dosya yüklendi. `retry_after` kadar bekle.

Ayrıntılar Hız sınırları sayfasında.

N'oluyo ile giriş (OAuth2)

`/oauth2/authorize` ve `/oauth2/@me/*` rotaları standart `{code, message}` döndürür; `/oauth2/token` ve `/oauth2/revoke` ise OAuth2 standardına uygun `{error, error_description}` gövdesi döndürür:

HTTP	Kod / `error`	Anlamı
403	`oauth_closed`	Yalnızca platform anahtarı kapalıyken döner; açık platformda üretilmez.
400	`oauth_unknown_client`	`client_id` geçersiz, bulunamadı ya da uygulaması devre dışı.
400	`oauth_bad_redirect`	`redirect_uri` bu uygulama için kayıtlı değil; sayfa o adrese yönlendirilmez.
400	`oauth_bad_request`	`response_type=code`, `state`, `code_challenge_method=S256` ya da `scope` geçersiz.
403	`oauth_user_cap`	Doğrulanmamış uygulama 100 kullanıcı yetki sınırına ulaştı.
403	`terms_not_accepted`	Kişi güncel kullanım koşullarını henüz kabul etmedi.
403	`oauth_scope_required`	Erişim token'ı bu rotanın istediği kapsama (`guilds` gibi) sahip değil.
400	`invalid_request`	`/oauth2/token` ya da `/oauth2/revoke` form gövdesi eksik veya bozuk.
401	`invalid_client`	`client_id` ya da `client_secret` yanlış veya eksik.
400	`invalid_grant`	Yetki kodu ya da yenileme token'ı geçersiz, süresi dolmuş veya yeniden kullanılmış.
400	`unsupported_grant_type`	Yalnızca `authorization_code` ve `refresh_token` desteklenir.
429 / 503	`temporarily_unavailable`	Token hız sınırı aşıldı (`Retry-After` kadar bekle) ya da hizmet bakımda.

Ses (VOICE_JOIN_ERROR)

Gateway üzerinden `op 4` gönderdiğinde bağlantı kapanmaz; bir engel varsa `VOICE_JOIN_ERROR` olayı içinde şu kodlardan biri gelir. Ayrıntılar Ses sayfasında.

Kod	Anlamı
`unknown_guild`	Sunucu yok ya da bot üye değil.
`unknown_channel`	Kanal bu sunucuda yok.
`invalid_channel_type`	Kanal bir ses kanalı (`type: 2`) değil.
`missing_access`	Botun bu kanalı görme (`ViewChannel`) izni yok.
`missing_permissions`	Botun kanalda `Connect` izni yok ya da zaman aşımında.
`channel_bot_limit`	Bu ses kanalında en çok sayıda bot var.
`guild_bot_limit`	Bu sunucuda en çok sayıda bot seste.
`app_voice_limit`	Uygulama eşzamanlı ses bağlantısı sınırında.
`bot_voice_full`	Platform genelindeki bot ses kapasitesi dolu.
`voice_unavailable`	Ses hizmeti o an kullanılamıyor.

Gateway kapanış kodları

Bu sayfadaki kodlar REST içindir. Gateway bağlantısı `4000` ile başlayan kapanış kodlarıyla kapanır: neyin yeniden bağlanmayı gerektirdiği Gateway sayfasındaki tabloda, adları Olaylar sayfasında.

Sunucu tarafı

HTTP	Kod	Anlamı	Ne yapmalı
503	`maintenance`	N'oluyo bakımda; bot ve uygulama rotaları bu yanıtı verir.	Bir süre bekleyip yeniden dene. `GET /api/v1/maintenance` kimlik istemez ve `enabled` alanında bakımın sürüp sürmediğini söyler.
503	`voice_unavailable`	Ses hizmeti bu sunucuda kullanılamıyor.	Daha sonra yeniden dene.
500	`internal`	Beklenmeyen bir hata oldu.	Biraz bekleyip yeniden dene; sürerse bize bildir.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Hata yönetimi yazdırırken yapay zekaya `res.status` yanında gövdedeki `code` ve `errors` alanlarını da okumasını, `401` ve `403` hatalarını döngüde tekrar denememesini, `429` yanıtlarında ise `retry_after` süresini beklemesini söyle.

Sırada ne var

- Sık hatalar: en çok karşılaşılan belirtilerin adım adım çözümü
- Hız sınırları: `429` yanıtları, kovalar ve geçersiz istek sınırı
- Gateway: `4000` serisi WebSocket kapanış kodları ve yeniden bağlanma kuralları

Sık sorulan sorular

Takıldığın yeri burada bulabilirsin. Her hatanın kısa anlamı için Hatalar sayfasına da bakabilirsin.

Başlarken

`403 developer_terms_required` alıyorum

Portaldan geliştirici sözleşmesini kabul et, sonra aynı hesapla yeniden dene.

`403 email_not_verified` veya `403 guest_not_allowed` alıyorum

Uygulama açmak için e-posta adresin doğrulanmış olmalı: gelen kutundaki bağlantıya tıkla. Misafir hesapla uygulama açılamaz; önce ayarlardan hesabına e-posta ve şifre bağla.

Ekleme sayfasında sunucum listede yok

Liste yalnızca `ManageGuild` iznin olan, yani yönettiğin sunucuları gösterir. Sunucu sahibinden sana bu izni vermesini ya da botu kendisinin eklemesini iste.

`403 bot_private` alıyorum

Bu botu yalnızca sahibi ekleyebilir. Bot seninse bu hesapla giriş yap; başkası da ekleyebilsin istiyorsan portalda uygulamanın **Herkese açık** seçeneğini aç.

Botu eklerken `400 bot_guild_limit` alıyorum

Doğrulanmamış bir bot en çok 100 sunucuda olabilir. Sınırı kaldırmak için Doğrulama başvurusu yapabilirsin.

Token ve kimlik

Her istekte `401 unauthorized` alıyorum

Token eksik, yanlış veya artık geçersiz. Sırayla bak:

1. Başlık tam olarak `Authorization: Bot <token>` mi? `Bot` sözcüğü ve ardından tek boşluk gerekli.
2. Token'ı yeniledikten sonra eski değeri mi kullanıyorsun? Yenileme eskisini anında geçersiz kılar.
3. Uygulama silinmiş ya da botun veya sahibinin hesabı kapatılmış olabilir.

Hâlâ çözülmediyse portalda token'ı yenile. `401` aldığın isteği tekrar etme; botu durdurup bak.

Token'ımı kaybettim

Kayıtlı token bir daha gösterilmez. Portalda uygulama sayfasındaki **Token'ı yenile** ile yenisini al ve programına koy.

`403 bot_forbidden` alıyorum

Çağırdığın rota botlara kapalı. Bu çoğunlukla kişi hesabına özel bir şeydir: özel mesaj açmak, davet kullanmak, profil değiştirmek ya da portal rotaları. Botların kullanabildiği her şey REST referansında.

İzinler

Her şey doğru görünüyor ama `403 missing_permissions` alıyorum

Botun bu işlem için gereken izni yok. Rotaya gereken izin REST referansında her rotanın altında yazar. Şunlara bak:

1. **Botun rolünde bu izin var mı?** Sunucu ayarlarındaki Roller listesinde botun adını taşıyan rolü aç. İzin yoksa ekle; bot bunu kendisi ekleyemez.
2. **Kanal ayarları izni kaldırıyor mu?** Sunucuda izni olsa bile kanalın izin ayarı onu alabilir.
3. **Bot zaman aşımında mı?** Zaman aşımındaki bir üye yalnızca `ViewChannel` ve `ReadMessageHistory` izinlerini korur.
4. **Sahip olmadığı bir izni mi veriyor?** Bot sahip olmadığı bir izni bir role veya kanal ayarına veremez; bu da `missing_permissions` döner. Çözüm, önce botun rolüne o izni vermek.

Rol yeterince yüksek değil: `403 role_hierarchy`

Bot, işlem yaptığı üyenin en yüksek rolünden veya düzenlediği rolden daha yüksek bir role sahip olmalı. Eşit olmak yetmez. Yeni botun rolü `@everyone`'ın hemen üstünde başlar, bu yüzden başta yalnızca rolü olmayan üyeler üzerinde işlem yapabilir.

Çözüm: Bir sunucu yöneticisi, **Sunucu ayarları** içindeki **Roller** listesinde botun rolünü işlem yapacağı rollerin üstüne sürüklesin. Sunucu sahibi üzerinde hiçbir bot işlem yapamaz.

> **Not:** İki hata kolayca karışır. `missing_permissions`, izin **yok** demektir; rolü yukarı taşımak düzeltmez. `role_hierarchy`, izin var ama rütbe **yetmiyor** demektir; izin eklemek düzeltmez.

`403 missing_access` alıyorum

Botun bu kanalı görme izni (`ViewChannel`) yok. Sunucuda veya kanal ayarlarında bu izni ver.

Mesaj listesi hep boş geliyor, hata da yok

Botun **Mesaj geçmişini oku** (`ReadMessageHistory`) izni yok. Liste rotaları bu durumda hata vermez, boş liste döndürür. Rolüne bu izni ekle.

Botun rolünü silemiyorum veya başkasına veremiyorum: `400 managed_role`

Botun rolü ona aittir. Silinemez, başkasına verilemez, bottan alınamaz. Adını, rengini, izinlerini ve sırasını ise normal kurallarla değiştirebilirsin. Bot sunucudan çıkınca rol de silinir.

Hız sınırları

`429 rate_limited` alıyorum

Sınırı aştın. Yanıttaki `retry_after` kadar saniye bekleyip yeniden dene:

```
if (res.status === 429) {
  const { retry_after } = await res.json();
  await new Promise((done) => setTimeout(done, retry_after * 1000));
  // sonra isteği yeniden gönder
}
```

Sürekli alıyorsan isteklerini azalt. Sınırlar Hız sınırları sayfasında.

`429 slowmode` alıyorum

Kanalda yavaş mod açık. `retry_after` kadar bekle. Bota `ManageMessages` veya `ManageChannels` izni verilirse yavaş moddan etkilenmez.

`429 upload_quota` alıyorum

Kısa sürede çok dosya yükledin: saatte 1 GiB, günde 3 GiB sınırı var. `retry_after` kadar bekle.

`403 temporarily_banned` alıyorum

Bu IP adresinden çok fazla reddedilmiş bot isteği geldi. `retry_after` kadar bekle ve programını düzelt: `401`'de dur, `403`'ü tekrar etme, `429`'da bekle. Ayrıntılar Hız sınırları sayfasında.

Mesajlar

Botum `!ping` yazılınca bir şey yapmıyor

Sırayla şunlara bak:

1. Program çalışıyor mu ve terminalde `READY` görünüyor mu? Görünmüyorsa bağlantının kapanış koduna bak, tablosu Gateway sayfasında.
2. Identify'daki `intents` değerinde `GUILD_MESSAGES` var mı? Yoksa mesaj olayları hiç gelmez.
3. `MESSAGE_CONTENT` var mı? Yoksa `content` boş gelir ve `!ping` eşleşmez. Aşağıdaki soruya bak.
4. Bot o kanalı görebiliyor mu? **Kanalları görüntüle** izni olmayan kanalların olayları gelmez.
5. Yanıt göndermek için **Mesaj gönder** ve **Mesaj geçmişini oku** izni var mı?

Çalışan bir örnek Örnekler sayfasında.

Mesajların `content` alanı boş geliyor

Bot, `MESSAGE_CONTENT` intent'i olmadan yalnızca kendi mesajlarının ve kendisini etiketleyen mesajların metnini görür. Portalda **Mesaj içeriği**'ni aç ve identify'da bu intent'i de iste. Portalda yeni açtıysan programı yeniden başlat: açık oturum, identify ettiği intent'lerle sürer. Kural Intent'ler sayfasında.

Bağlantı `4014` ile kapanıyor

Identify'da portalda açmadığın ayrıcalıklı bir intent istiyorsun: `GUILD_MEMBERS`, `GUILD_PRESENCES` ya da `MESSAGE_CONTENT`. Portaldan aç ya da `intents` değerinden çıkar. `op 8` de oturumda olmayan bir intent gerektirirse aynı kod gelir.

Bağlantı `4011` ile kapanıyor

Botun için aynı shard'la daha yeni bir oturum açıldı. Çoğunlukla programın iki kopyası çalışıyordur: biri eski bir terminalde ya da başka bir sunucuda kalmış olabilir. Birini durdur. Yeniden bağlanma döngüsüne girme, iki kopya birbirini düşürmeye devam eder. Daha küçük bir shard sayısıyla bağlanmak da aynı kodu verir, bkz. Shard'lar.

`400 invalid_body` alıyorum

İstek gövdesi tek bir geçerli JSON nesnesi değil ya da bir alanın türü yanlış. En sık nedeni ID'yi veya izin değerini metin yerine sayı olarak göndermektir: `"111600000000000000"` doğru, tırnaksız yazmak yanlış. Gövde isteyen bir rotaya boş gövde göndermek de bu hatayı verir; değişiklik yoksa `{}` gönder.

`400 validation_failed` alıyorum

Bir alan kurala uymuyor. Yanıttaki `errors`, hangi alanın neden reddedildiğini söyler: `{"content":"required"}`. Örneğin dosya, anket, iletme veya çıkartma yoksa `content` zorunludur ve en çok 2000 karakterdir.

`400 automod_blocked` alıyorum

Sunucunun otomatik moderasyonu mesajı engelledi. Yanıttaki `message`, sunucunun kendi metni olabilir. Kural göndereni zaman aşımına da alabilir. Sunucu sahibi ve `ManageGuild` izni olanlar bundan muaftır.

`403 not_author` alıyorum

Bir bot yalnızca kendi mesajlarını düzenleyebilir. Başkasının mesajını düzenleyemez ama `ManageMessages` izniyle silebilir.

`404 unknown_guild` veya `404 unknown_channel` alıyorum

Bot o sunucunun üyesi değil, kanal yok ya da ID yanlış. Botun gördüğü sunucuları `GET /users/@me/guilds`, kanalları `GET /guilds/{guild_id}/channels` ile listele.

Yanıtım `201` değil `200` geldi

Doğru: mesaj göndermek `200` döner. Her rotanın başarılı yanıtı REST referansında yazar.

Botum kişiye neden ilk yazamıyor?

Çünkü N'oluyo'da bot kimseye kendiliğinden yazamaz: istenmeyen mesajı baştan önlemek için DM'yi yalnızca kişi açar. Kişi botunun profilinden ya da üye listesinden "Mesaj gönder" dediğinde DM açılır; kişi yazdıktan, komut çalıştırdıktan ya da butona bastıktan sonra 24 saat serbestçe yazabilirsin. Bu süre dışında `403 dm_window_closed`, kişi DM'yi kapattıysa ya da botu engellediyse `403 dm_closed` alırsın. Ayrıntılar Özel mesajlar sayfasında.

Embed'ler ve butonlar

Butona basınca "Bot yanıt vermedi." görünüyor

Bota basma ulaştı ama 3 saniye içinde geçerli bir yanıt gelmedi. Sırayla bak:

1. Bot çalışıyor ve gateway'e bağlı mı? Çevrimdışı bota basma sonradan gönderilmez.
2. `INTERACTION_CREATE` gelince ilk iş yanıt mı? Rol vermek ya da başka bir servise sormak gibi işleri yanıttan sonra yap; iş uzunsa önce `type: 5` ya da `type: 6` gönder.

3. `callback` isteği hata döndürüyor mu? Programının log'una bak: `validation_failed` gibi bir hata yanıtı geçersiz kılar. Reddedilen yanıt hakkını harcamaz, süre içinde düzeltip yeniden gönderebilirsin.

Ayrıntılar Etkileşimler sayfasında.

`400 interaction_timeout` alıyorum

İlk yanıt 3 saniyeden sonra geldi. O etkileşime artık yanıt verilemez. Kodunda yanıttan önce çalışan uzun işi yanıtın arkasına taşı.

Kişiler "Bu bileşen artık geçerli değil." görüyor

Bot mesajı düzenleyip butonu kaldırmış ya da `custom_id`'sini değiştirmiş; ekranı henüz güncellenmemiş biri eski butona basıyor. Basma bota ulaşmaz. Bu doğru davranıştır.

Embed'deki görsel `invalid` diye reddediliyor

Embed görseli yalnızca aynı istekle yüklenen bir dosya olabilir: `attachment://dosya.png` yaz ve dosyayı `multipart/form-data` ile `files[0]` olarak gönder. Başka bir siteden adres kabul edilmez. Ayrıntılar Embed'ler sayfasında.

Yalnızca basan kişinin gördüğü mesajlar

Ephemeral mesajım sayfa yenilenince kayboluyor

Doğru: N'oluyo bu mesajı saklamaz, yenilemede ve uygulama yeniden açılınca gider. Kalıcı olması gereken bir sonucu kişinin bulabileceği başka bir yere de yaz. Ayrıntısı Etkileşimler sayfasında.

`400 ephemeral_files` alıyorum

Ephemeral mesaj dosya taşıyamaz: `attachment://` ve `multipart/form-data` ikisi de reddedilir. Görseli herkese açık bir mesajla gönder.

Ephemeral mesajı düzenledim, metni boşaldı

Ephemeral mesajda birleştirilecek bir satır yoktur, göndermediğin alan boş sayılır. `PATCH` ve `type: 7` içinde mesajın tamamını gönder.

Ses

Bot ses kanalındaki konuşmaları duyabilir mi?

Hayır. Botların aldığı medya token'ında `canSubscribe: false` yazılıdır ve medya sunucusu botun hiçbir ses, görüntü ya da veri izine abone olmasına izin vermez; kodunda abone olmayı denesen bile tek bir ses karesi gelmez. Bot seste yalnızca ses yayınlar. Odadayken yalnızca kimin bağlı olduğunu, susturma durumunu ve kimin o an konuştuğunu (ses içeriği olmadan) görür. Ayrıntılar Ses sayfasında.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Aldığın bir hatayı yapay zekaya çözdürürken hata metnini yapıştırırken `.env` dosyanı veya `BOT_TOKEN` değerini kesinlikle paylaşma; yanında bu sayfayı ya da Sık hatalar tablosunu ver.

Sırada ne var

- Sık hatalar: belirtiye göre sebep ve çözüm tablosu
- Hatalar: tüm REST ve etkileşim hata kodlarının tam listesi
- Sözlük: teknik terimlerin iki cümlelik sade açıklamaları

Değişiklikler

API'deki değişiklikler burada, en yenisi üstte olacak şekilde yazılır.

6. aşama: N'oluyo ile giriş, doğrulama, bot rehberi ve resmi kütüphaneler

Kullanıcıların kendi hesaplarıyla üçüncü taraf sitelere giriş yapması, büyüyen uygulamalar için doğrulama ve rehber süreci, iki dilde resmi `noluyo-bot` kütüphaneleri ve yapay zeka uyumlu doküman paketleri eklendi.

Eklenenler

- N'oluyo ile giriş: PKCE (`S256`) ve `state` zorunluluğuyla OAuth2 yetkilendirme akışı (`identify`, `email`, `guilds` kapsamları, yenileme rotasyonu, açık istemci desteği ve **Ayarlar → Bağlı uygulamalar** yönetimi).
- Doğrulama ve Rehbere girmek: 100 sunucu ve 100 OAuth2 kullanıcısı sınırını aşmak için portal başvurusu, ayrıcalıklı intent onayı ve `/kesfet` altındaki **Botlar** dizininde kategoriye göre listelenme.
- Resmi kütüphaneler: Node.js 22+ için sıfır bağımlılıklı JavaScript kütüphanesi ve Python 3.11+ için `asyncio` tabanlı Python kütüphanesi (`noluyo-bot`), ayrıca moderasyon, anket, ses ve OAuth2 örnek projeleri.
- Yapay zeka ve çevrimdışı doküman çıktıları: her rehber için ham `.md`, tarayıcıdan `.html` / `.pdf` kaydı, 30 KB altı `llms-kisa.txt` özeti ve tam `llms-full.txt` referansı.
- Yeni başlayanlar ve geçiş yapanlar için 13 yeni rehber: Nereden başlamalı, Yapay zekaya yazdır, Kurulum, Hazır tarifler, Botu açık tut, Sık hatalar, Sözlük ve Başka platformdan gelenler.

Var olan botlara etkisi

- Mevcut REST rotaları, WebSocket gateway çerçeveleri ve ses akışı geriye dönük uyumludur; çalışan botlarda kod değişikliği gerekmez.
- 100 sunucunun altındaki botlar eskisi gibi doğrulama başvurusu yapmadan çalışmayı sürdürür. Uygulama ilk kez oluşturulurken geliştirici sözleşmesi onayı (`developer_terms_required`) istenir; var olan uygulamalar kesintisiz çalışmaya devam eder.
- Acil durdurulan bir uygulamanın bot token'ı güvenlik gereği kalıcı olarak iptal edilir; uygulama yeniden açıldığında portaldan yeni token üretilir.

5. aşama: seste bot

Botlar artık ses kanallarına girip müzik, duyuru ya da ses efekti yayınlayabilir; ses kanallarındaki kimseyi ise hiçbir koşulda duyamaz.

Eklenenler

- Ses: gateway `op 4` ile ses kanalına girme ve çıkma, `VOICE_SERVER_UPDATE` ve `VOICE_JOIN_ERROR` olayları, LiveKit üzerinden `ffmpeg` ile çalışan örnek (`cal.mjs`) ve bot ses sınırları.
- Bir bot farklı sunucularda aynı anda birden çok ses kanalında bulunabilir; web, masaüstü (oyun içi arayüz dahil) ve Android istemcilerinde **BOT** etiketiyle gösterilir ve kanalın kişi sınırına sayılmaz.
- Gateway sayfasındaki `connect()` yardımcısı artık `{ send(op, d) }` döndürür; var olan örnekler dönüş değerini kullanmadığı için kırılmaz.

Hâlâ olmayanlar

- Resmi kütüphaneler

4. aşama: komutlar, formlar, özel mesajlar ve herkese açık beta

Bot platformu herkese açık beta olarak açıldı: e-postası doğrulanmış her hesap geliştirici sözleşmesini kabul edip uygulama oluşturabilir ve bot ekleyebilir. Davet ve `bots_closed` / `bot_developer_required` hataları kalktı.

Eklenenler

- Komutlar: `/` listesinde ve kişi ya da mesaja sağ tıklayınca menüde görünen komutlar.
- Otomatik tamamlama: komut seçeneğine yazarken en çok 25 öneri.
- Formlar: komuta ya da butona yanıt olarak açılan, en çok beş alanlı pencere.
- Özel mesajlar: kişi botla DM açar; komutlar, butonlar ve formlar orada da çalışır. Bot asla ilk yazan olmaz.
- Komut izinleri: sunucu yöneticileri komutların kimde ve hangi kanalda çalışacağını belirler.
- HTTP ile teslim: etkileşimleri gateway yerine imzalı bir https adresine aldır.
- Roller artık arayüzde ID'si ile kopyalanabilir; bkz. İzinler.

Hâlâ olmayanlar

- Botların ses kanalına girmesi
- Resmi kütüphaneler

3. aşama: embed, bileşen ve etkileşimler

Botların mesajları artık kart, buton ve menü taşıyabilir; basmalar bota gelir. Erişim o dönemde hâlâ davetleydi.

Eklenenler

- Embed'ler: başlık, açıklama, alanlar, renk, zaman ve aynı mesajla yüklenen görseller (`attachment://`). Mesajda en çok 10 embed.
- Bileşenler: beş stilde buton, bağlantı butonu, metin, kullanıcı, rol ve kanal menüsü.
- Etkileşimler: `INTERACTION_CREATE` olayı ve dört yanıt türü (`type` 4, 5, 6, 7), `@original` düzenlemesi, takip mesajları.
- Yalnızca basan kişinin gördüğü mesajlar (`flags: 64`).
- Botlar için tam `allowed_mentions` : `parse` , `users` , `roles` , `replied_user` .
- Webhook mesajlarında embed.
- Yeni hız sınırı sınıfı `interaction` ve büyük embed'ler için tüm botların ortak bütçesi; ayrıntı Hız sınırları sayfasında.
- Çalışan üç yeni örnek: anket, rol seçici ve sayfalı liste.

2. aşamaya göre değişenler

Bunlar var olan botları etkileyebilir:

- **`embeds` , `components` ve `flags` artık okunuyor.** Önceden botun isteğindeki bu alanlar bilinmeyen alan olarak yok sayılırdı. Şimdi denetlenir: kurala uymayan bir embed ya da bileşen `400 validation_failed` , kanal mesajında `flags: 64` `400 ephemeral_forbidden` , başka bir `flags` değeri `{"flags":"invalid"}` döner. Bu alanları gönderip sonucun yok sayılmasına güvenen bir bot artık hata alabilir.
- **`allowed_mentions` tam uygulanıyor.** Önceden yalnızca `replied_user` okunurdu. Şimdi `"parse": []` gönderen bir botun mesajındaki etiketler bildirim göndermez; eskiden gönderirdi. `parse` ile aynı türün

listesini birlikte göndermek ya da `parse` içinde bilinmeyen bir değer `400 validation_failed` döner.
- **Mesaj düzenleme (`PATCH`) artık kısmi.** Önceden `content` zorunluydu. Şimdi `content` , `embeds` ve `components` ayrı ayrı gönderilir, göndermediğin alan olduğu gibi kalır. `content` göndermeden yapılan bir `PATCH` artık kabul edilir; `"content": ""` mesajda metin, embed ya da dosya bırakmıyorsa yine `required` döner.
- **Webhook'un `embeds` 'i denetleniyor.** Önceden alınıp atılırdı. Şimdi liste olmayan `embeds` `400 invalid_body` , sınırı aşan bir alan `400 validation_failed` döner. Yalnızca embed içeren istek eskiden boş içerik diye `400` alırdı, şimdi çalışır. Görsel alanları sessizce düşer.
- **Yeni bir olay.** `INTERACTION_CREATE` , hangi intent'leri istersen iste bağlantına gelir. Bilmediğin olayları yok say.
- **Mesajda yeni alanlar.** Mesaj nesnesinde `embeds` , `components` , `flags` , `application_id` ve `interaction` gelebilir. **Mesaj içeriği** olmayan bota başkalarının `embeds` ve `components` 'i gelmez.

Henüz olmayanlar

Bunlar ileride gelecek:

- Botların ses kanalına girmesi
- Resmi kütüphaneler

2. aşama: gateway

Botlar artık olayları canlı dinleyebilir. Erişim o dönemde davetleydi.

Eklenenler

- Bot gateway'i: WebSocket bağlantısı, heartbeat, identify, resume ve kapanış kodları. Adresi `GET /gateway/bot` söyler.
- Olaylar: mesajlar, tepkiler, üyeler, roller, kanallar, ses, durum ve diğerleri; her biri alanları ve örneğiyle.
- Intent'ler: bot istediği olayları seçer. Üç ayrıcalıklı intent portaldan açılır.
- Shard'lar: çok sunuculu botlar için birden çok bağlantı, önerilen ve zorunlu sayı.
- `op 3` ile botun durumunu değiştirme, `op 8` ile üye listesi ve arama.
- Tüm botların ortak yazma tavanı: `X-RateLimit-Scope: shared` .

1. aşamaya göre değişenler

Bunlar var olan botları etkileyebilir:

- **REST okumalarında içerik kuralı.** Sunucu kanallarında mesaj listesi, tek mesaj ve sabitlenenler artık **Mesaj içeriği** açık olmayan bir bota metni, ekleri ve diğer içeriği süzülmüş olarak döner; botun kendi mesajları ve kendisini etiketleyen mesajlar hariç. Kanalı okuyan eski botlar `content` alanını boş görür: portaldan **Mesaj içeriği**'ni aç.
- **İletmede içerik kuralı.** Bot okuyamadığı bir mesajı iletemez: `400 validation_failed` , `{"forward":"invalid"}` .
- **Üye listesi intent ister.** `GET /guilds/{guild_id}/members` , **Sunucu üyeleri** açık olmayan bota artık `403 member_list_intent_required` döner. Tek üye istemek açık kalır.
- **Token yenileme.** Eski token'la açık gateway bağlantıları `4004` ile hemen kapanır.

1. aşama: kapalı dönem

Bot API'nin ilk sürümü. Erişim davetle açılıyor.

Eklenenler

- Geliştirici portalı: uygulama oluşturma, token, ekleme bağlantısı.
- Uygulamalar ve bot hesapları; `Authorization: Bot <token>` ile kimlik doğrulama.
- Botu sunucuya ekleme ve bota ait yönetilen rol.
- Bot rotaları: kullanıcı, sunucu, kanal, rol, üye, yasak, mesaj, sabitleme ve tepki işlemleri.
- `X-RateLimit-*` başlıklarıyla hız sınırları ve geçersiz istekler için geçici engel.

Bu sayfayı yapay zekaya verirsen

> **İpucu:** Eski bir bota yeni özellik ekletirken yapay zekaya her zaman 6. aşama kurallarını ve güncel `llms-kisa.txt` ya da `llms-full.txt` dosyasını ver ki eski aşamalardaki kısıtlamaları hâlâ geçerli sanmasın.

Sırada ne var

- Nereden başlamalı: sana uygun geliştirme yolunu seç
- JavaScript kütüphanesi ve Python kütüphanesi: resmi `noluyo-bot` paketleri
- N'oluyo ile giriş: web sitene veya uygulamana kullanıcı girişi ekle

API'ye genel bakış

Son güncelleme: 2026-11-11 (11 Kasım 2026) · Temel adres: `https://noluyo.net/api/v1`

Tüm REST rotaları bu adresin altındadır, ID alanları metindir (`Snowflake`) ve herkese açık ya da OAuth2 Bearer olarak işaretlenmeyen rotalar `Authorization: Bot <BOT_TOKEN>` başlığı ister.

Hız sınırı sınıfları

Sınıf	İstek	Pencere (sn)
`read`	30	5
`write`	10	10
`message`	5	5
`reaction`	4	1
`bulk`	1	1
`interaction`	50	1
`command`	5	20
`oauth_read`	30	5

Rota grupları

Kullanıcılar

- `GET /users/@me` — Botun kendisini getir (`getCurrentBot`)
- `GET /users/@me/guilds` — Botun bulunduğu sunucuları listele (`listCurrentGuilds`)
- `DELETE /users/@me/guilds/{guild_id}` — Sunucudan ayrıl (`leaveGuild`)

Sunucular

- `GET /guilds/{guild_id}` — Sunucuyu getir (`getGuild`)
- `GET /guilds/{guild_id}/channels` — Sunucunun kanallarını listele (`listGuildChannels`)
- `POST /guilds/{guild_id}/channels` — Kanal oluştur (`createChannel`)

Roller

- `GET /guilds/{guild_id}/roles` — Sunucunun rollerini listele (`listRoles`)
- `POST /guilds/{guild_id}/roles` — Rol oluştur (`createRole`)
- `PATCH /guilds/{guild_id}/roles/{role_id}` — Rolü güncelle (`updateRole`)
- `DELETE /guilds/{guild_id}/roles/{role_id}` — Rolü sil (`deleteRole`)

Üyeler

- `GET /guilds/{guild_id}/members` — Üyeleri listele (`listMembers`)
- `GET /guilds/{guild_id}/members/{user_id}` — Üyeyi getir (`getMember`)
- `PATCH /guilds/{guild_id}/members/{user_id}` — Üyeyi güncelle (`updateMember`)
- `DELETE /guilds/{guild_id}/members/{user_id}` — Üyeyi çıkar (`kickMember`)
- `PUT /guilds/{guild_id}/members/{user_id}/roles/{role_id}` — Üyeye rol ver (`addMemberRole`)
- `DELETE /guilds/{guild_id}/members/{user_id}/roles/{role_id}` — Üyeden rol al (`removeMemberRole`)

Yasaklar

- `GET /guilds/{guild_id}/bans` — Yasakları listele (`listBans`)
- `PUT /guilds/{guild_id}/bans/{user_id}` — Kullanıcıyı yasakla (`createBan`)
- `DELETE /guilds/{guild_id}/bans/{user_id}` — Yasağı kaldır (`deleteBan`)

Kanallar

- `GET /channels/{channel_id}` — Kanalı getir (`getChannel`)
- `PATCH /channels/{channel_id}` — Kanalı güncelle (`updateChannel`)
- `DELETE /channels/{channel_id}` — Kanalı sil (`deleteChannel`)
- `POST /channels/{channel_id}/typing` — Yazıyor göstergesini göster (`sendTyping`)

Mesajlar

- `GET /channels/{channel_id}/messages` — Mesajları listele (`listMessages`)
- `POST /channels/{channel_id}/messages` — Mesaj gönder (`createMessage`)
- `GET /channels/{channel_id}/messages/{message_id}` — Mesajı getir (`getMessage`)
- `PATCH /channels/{channel_id}/messages/{message_id}` — Mesajı düzenle (`editMessage`)
- `DELETE /channels/{channel_id}/messages/{message_id}` — Mesajı sil (`deleteMessage`)
- `POST /channels/{channel_id}/messages/bulk-delete` — Birden çok mesajı sil (`bulkDeleteMessages`)

Sabitlenen mesajlar

- `GET /channels/{channel_id}/pins` — Sabitlenen mesajları listele (`listPins`)
- `PUT /channels/{channel_id}/pins/{message_id}` — Mesajı sabitle (`pinMessage`)
- `DELETE /channels/{channel_id}/pins/{message_id}` — Mesajın sabitlemesini kaldır (`unpinMessage`)

Tepkiler

- `PUT /channels/{channel_id}/messages/{message_id}/reactions/{emoji}/@me` — Mesaja tepki ver (`addReaction`)
- `DELETE /channels/{channel_id}/messages/{message_id}/reactions/{emoji}/{user_id}` — Tepkiyi kaldır (`removeReaction`)
- `GET /channels/{channel_id}/messages/{message_id}/reactions/{emoji}` — Tepki verenleri listele (`listReactors`)
- `DELETE /channels/{channel_id}/messages/{message_id}/reactions/{emoji}` — Bir emojinin tepkilerini kaldır (`clearEmojiReactions`)
- `DELETE /channels/{channel_id}/messages/{message_id}/reactions` — Tüm tepkileri kaldır (`clearReactions`)

Uygulamalar

- `GET /applications` — Uygulamalarını listele (`listApplications`)

- `POST /applications` — Uygulama oluştur (`createApplication`)
- `POST /applications/developer-terms` — Geliştirici sözleşmesini kabul et (`acceptDeveloperTerms`)
- `GET /applications/{application_id}` — Uygulamanı getir (`getApplication`)
- `PATCH /applications/{application_id}` — Uygulamanı güncelle (`updateApplication`)
- `DELETE /applications/{application_id}` — Uygulamanı sil (`deleteApplication`)
- `POST /applications/{application_id}/token` — Bot token'ını yenile (`resetApplicationToken`)
- `GET /applications/{application_id}/endpoint/deliveries` — Adresinin teslim günlüğünü oku (`getEndpointDeliveries`)
- `GET /applications/{application_id}/public` — Eklemeden önce uygulamayı gör (`getPublicApplication`)
- `POST /applications/{application_id}/authorize` — Botu sunucuya ekle (`authorizeApplication`)
- `POST /applications/{application_id}/avatar` — Botun resmini ayarla (`setApplicationAvatar`)
- `DELETE /applications/{application_id}/avatar` — Botun resmini kaldır (`deleteApplicationAvatar`)
- `POST /applications/{application_id}/oauth-secret` — OAuth2 istemci gizli anahtarını yenile (`resetOAuthSecret`)
- `GET /applications/{application_id}/review` — Son doğrulama başvurusunu getir (`getApplicationReview`)
- `POST /applications/{application_id}/review` — Doğrulama başvurusu gönder (`submitApplicationReview`)

Gateway

- `GET /gateway/bot` — Bot gateway bilgisini getir (`getGatewayBot`)

Etkileşimler

- `POST /interactions/{interaction_id}/{interaction_token}/callback` — Etkileşime yanıt ver (`interactionCallback`)
- `POST /interactions/{interaction_id}/{interaction_token}/messages` — Takip mesajı gönder (`createInteractionFollowup`)
- `PATCH /interactions/{interaction_id}/{interaction_token}/messages/{message_id}` — Etkileşimin mesajını düzenle (`editInteractionMessage`)
- `DELETE /interactions/{interaction_id}/{interaction_token}/messages/{message_id}` — Etkileşimin mesajını sil (`deleteInteractionMessage`)

Komutlar

- `GET /applications/{application_id}/commands` — Global komutları listele (`listGlobalCommands`)
- `PUT /applications/{application_id}/commands` — Global komutların hepsini değiştir (`replaceGlobalCommands`)
- `POST /applications/{application_id}/commands` — Global komutu oluştur (`createGlobalCommand`)
- `GET /applications/{application_id}/commands/{command_id}` — Global komutu getir (`getGlobalCommand`)
- `PATCH /applications/{application_id}/commands/{command_id}` — Global komutu düzenle (`editGlobalCommand`)
- `DELETE /applications/{application_id}/commands/{command_id}` — Global komutu sil (`deleteGlobalCommand`)
- `GET /applications/{application_id}/guilds/{guild_id}/commands` — Sunucu komutlarını listele (`listGuildCommands`)
- `PUT /applications/{application_id}/guilds/{guild_id}/commands` — Sunucu komutlarının hepsini değiştir (`replaceGuildCommands`)
- `POST /applications/{application_id}/guilds/{guild_id}/commands` — Sunucu komutu oluştur (`createGuildCommand`)
- `GET /applications/{application_id}/guilds/{guild_id}/commands/{command_id}` — Sunucu komutunu getir (`getGuildCommand`)

- `PATCH /applications/{application_id}/guilds/{guild_id}/commands/{command_id}` — Sunucu komutunu düzenle (`editGuildCommand`)
- `DELETE /applications/{application_id}/guilds/{guild_id}/commands/{command_id}` — Sunucu komutunu sil (`deleteGuildCommand`)
- `GET /applications/{application_id}/guilds/{guild_id}/commands/permissions` — Sunucudaki komut kurallarını listele (`listCommandPermissions`)
- `GET /applications/{application_id}/guilds/{guild_id}/commands/{command_id}/permissions` — Komutun kurallarını getir (`getCommandPermissions`)

OAuth2 (N'oluyo ile giriş)

- `POST /oauth2/token` — Yetki kodunu ya da yenileme token'ını değiştir (`exchangeOAuthToken`)
- `POST /oauth2/revoke` — Erişim ya da yenileme token'ını iptal et (`revokeOAuthToken`)
- `GET /oauth2/@me` — Giriş yapan kişinin kimliğini oku (`getOAuthMe`)
- `GET /oauth2/@me/guilds` — Giriş yapan kişinin sunucularını listele (`listOAuthGuilds`)

Keşfet (Bot rehberi)

- `GET /discovery/bots` — Rehberdeki doğrulanmış botları listele (`listDirectoryBots`)

Kullanıcılar

Kullanıcılar için REST API rotaları.

Botun kendisini getir (`getCurrentBot`)

`GET https://noluyo.net/api/v1/users/@me` (`GET /users/@me`)

Token'ın ait olduğu bot hesabını döndürür. Bot, herkese açık kullanıcı nesnesini alır; e-posta adresi yer almaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/users/@me" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getCurrentBot();
```

```
# Python (noluyo_bot)
await bot.rest.get_current_bot()
```

Yanıt (200)

```
{
  "id": "111540000000000001",
  "username": "ornek_bot",
  "display_name": "Örnek Bot",
  "avatar": null,
  "cosmetics": null,
  "bot": true
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Botun bulunduğu sunucuları listele (`listCurrentGuilds`)

`GET https://noluyo.net/api/v1/users/@me/guilds` (`GET /users/@me/guilds`)

Çağıranın üyesi olduğu her sunucu için kısa bir nesne döndürür. Listede sayfalama yoktur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/users/@me/guilds" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listCurrentGuilds();
```

```
# Python (noluyo_bot)
await bot.rest.list_current_guilds()
```

Yanıt (200)

```
[
  {
    "id": "111500000000000000",
    "name": "Örnek Sunucu",
    "icon": null,
    "owner": false,
    "features": []
  }
]
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucudan ayrıl (`leaveGuild`)

`DELETE https://noluyo.net/api/v1/users/@me/guilds/{guild_id}` (`DELETE /users/@me/guilds/{guild_id}`)

Çağıranı sunucudan çıkarır. Bot için, kendine ait yönetilen rol üyelikle birlikte silinir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)

- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/users/@me/guilds/111500000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.leaveGuild("111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.leave_guild("111500000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `owner_cannot_leave`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucular

Sunucular; API'de sunucuya guild denir.

Sunucuyu getir (`getGuild`)

`GET https://noluyo.net/api/v1/guilds/{guild_id}` (`GET /guilds/{guild_id}`)

Sunucuyu tüm rolleriyle döndürür. Üye olmayan çağıran unknown_guild alır; böylece sunucunun var olup olmadığı açığa çıkmaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/guilds/111500000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getGuild("111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.get_guild("111500000000000000")
```

Yanıt (200)


```
{
  "id": "111500000000000000",
  "name": "Örnek Sunucu",
  "icon": null,
  "banner": null,
  "description": null,
  "owner_id": "111400000000000500",
  "created_at": "2026-11-04T16:21:11.569Z",
  "features": [],
  "roles": [
    {
      "id": "111500000000000000",
      "name": "@everyone",
      "color": 0,
      "hoist": false,
      "mentionable": false,
      "position": 0,
      "permissions": "1071698529857",
      "auto_assign": false
    },
    {
      "id": "111550000000000000",
      "name": "Örnek Bot",
      "color": 0,
      "hoist": false,
      "mentionable": false,
      "position": 1,
      "permissions": "68672",
      "auto_assign": false,
      "application_id": "111540000000000000"
    }
  ],
  "discoverable": false,
  "discovery_category": null,
  "join_channel_id": null,
  "leave_channel_id": null
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucunun kanallarını listele (`listGuildChannels`)

`GET https://noluyo.net/api/v1/guilds/{guild_id}/channels` (`GET /guilds/{guild_id}/channels`)

Çağıranın ViewChannel iznine sahip olduğu kanal ve kategorileri döndürür. Diğerleri hata vermeden listeden çıkarılır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/guilds/111500000000000000/channels" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listGuildChannels("111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.list_guild_channels("111500000000000000")
```

Yanıt (200)

```
[
  {
    "id": "111500000000000100",
    "type": 0,
    "guild_id": "111500000000000000",
    "name": "genel",
    "topic": null,
    "position": 1,
    "parent_id": null,
    "nsfw": false,
    "last_message_id": "111600000000000000",
    "permission_overwrites": []
  }
]
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Kanal oluştur (`createChannel`)

`POST https://noluyo.net/api/v1/guilds/{guild_id}/channels` (`POST /guilds/{guild_id}/channels`)

Metin kanalı (tür 0, varsayılan), ses kanalı (2) veya kategori (4) oluşturur. topic ve rate_limit_per_user metin kanalları, bitrate ve user_limit ses kanalları içindir. permission_overwrites verilmezse kategoriye konan kanal kategorinin izin geçersiz kılmalarıyla başlar. Bir geçersiz kılma, yalnızca çağıranın sahip olduğu izinlere dokunabilir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageChannels`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`name`	`string`	evet	(1–100)
`type`	`integer`	-	(0, 2, 4 · = 0)
`topic`	`string | null`	-	(<= 1024)
`position`	`integer`	-	
`parent_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`nsfw`	`boolean`	-	
`bitrate`	`integer`	-	(8000–96000 · = 64000)
`user_limit`	`integer`	-	(0–99 · = 0)
`rate_limit_per_user`	`integer`	-	(0–21600)
`permission_overwrites`	`Overwrite[]`	-	(<= 100)

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/guilds/111500000000000000/channels" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"name":"duyurular","type":0}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createChannel("111500000000000000", { name: "duyurular", type: 0 });
```

```
# Python (noluyo_bot)
await bot.rest.create_channel("111500000000000000", {"name": "duyurular", "type": 0})
```

Yanıt (201)

```
{
  "id": "111500000000000100",
  "type": 0,
  "guild_id": "111500000000000000",
  "name": "duyurular",
  "topic": "Önemli haberler",
  "position": 1,
  "parent_id": null,
  "nsfw": false,
  "last_message_id": null,
  "permission_overwrites": [],
  "rate_limit_per_user": 30
}
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_channels`, `max_overwrites`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Roller

Roller için REST API rotaları.

Sunucunun rollerini listele (`listRoles`)

`GET https://noluyo.net/api/v1/guilds/{guild_id}/roles` (`GET /guilds/{guild_id}/roles`)

Sunucunun, @everyone ve botların yönetilen rolleri dâhil tüm rollerini döndürür. @everyone rolünün ID'si sunucunun ID'siyle aynıdır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/guilds/111500000000000000/roles" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listRoles("111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.list_roles("111500000000000000")
```

Yanıt (200)


```
[
  {
    "id": "111500000000000000",
    "name": "@everyone",
    "color": 0,
    "hoist": false,
    "mentionable": false,
    "position": 0,
    "permissions": "1071698529857",
    "auto_assign": false
  },
  {
    "id": "111550000000000000",
    "name": "Örnek Bot",
    "color": 0,
    "hoist": false,
    "mentionable": false,
    "position": 1,
    "permissions": "68672",
    "auto_assign": false,
    "application_id": "111540000000000000"
  }
]
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Rol oluştur (`createRole`)

`POST https://noluyo.net/api/v1/guilds/{guild_id}/roles` (`POST /guilds/{guild_id}/roles`)

Doğrudan @everyone'ın üstünde bir rol oluşturur. Gövde zorunludur; yalnızca {} bile olabilir. position burada yok sayılır; ad verilmezse rolün adı "yeni rol" olur. Kimse sahip olmadığı bir izni veremez.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageRoles`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`name`	`string`	-	(1–100)
`color`	`integer`	-	(0–16777215)
`hoist`	`boolean`	-	
`mentionable`	`boolean`	-	
`position`	`integer`	-	Yalnızca güncellemede.
`permissions`	`permissions`	-	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.
`auto_assign`	`boolean`	-	

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/guilds/111500000000000000/roles" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"name":"Moderatör","color":3447003}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createRole("111500000000000000", { name: "Moderatör", color: 3447003 });
```

```
# Python (noluyo_bot)
await bot.rest.create_role("111500000000000000", {"name": "Moderatör", "color": 3447003})
```

Yanıt (201)

```
{
  "id": "1115500000000000200",
  "name": "Moderatör",
  "color": 3447003,
  "hoist": true,
  "mentionable": true,
  "position": 1,
  "permissions": "8192",
  "auto_assign": false
}
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_roles`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Rolü güncelle (`updateRole`)

`PATCH https://noluyo.net/api/v1/guilds/{guild_id}/roles/{role_id}` (`PATCH /guilds/{guild_id}/roles/{role_id}`)

Yalnızca gönderilen alanları değiştirir. Çağıran rolün üstünde olmalı, position da çağıranın en yüksek rolünün altında kalmalıdır. Yeni izin bitleri çağıranda bulunmalıdır. @everyone için name, position ve auto_assign, yönetilen rol için auto_assign true reddedilir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageRoles`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`role_id`	`path`	`string`	evet	Rol ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`name`	`string`	-	(1–100)
`color`	`integer`	-	(0–16777215)
`hoist`	`boolean`	-	
`mentionable`	`boolean`	-	
`position`	`integer`	-	Yalnızca güncellemede.
`permissions`	`permissions`	-	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.
`auto_assign`	`boolean`	-	

Örnek çağrılar

```
curl -X PATCH "https://noluyo.net/api/v1/guilds/111500000000000000/roles/111550000000000000" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"color":15158332}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.updateRole("111500000000000000", "111550000000000000", { color: 15158332 });
```

```
# Python (noluyo_bot)
await bot.rest.update_role("111500000000000000", "111550000000000000", {"color": 15158332})
```

Yanıt (200)

```
{
  "id": "111550000000000000",
  "name": "Örnek Bot",
  "color": 15158332,
  "hoist": false,
  "mentionable": true,
  "position": 1,
  "permissions": "68672",
  "auto_assign": false,
  "application_id": "111540000000000000"
}
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `everyone_role`, `managed_role`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_role`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Rolü sil (`deleteRole`)

`DELETE https://noluyo.net/api/v1/guilds/{guild_id}/roles/{role_id}` (`DELETE /guilds/{guild_id}/roles/{role_id}`)

Çağıranın üstünde olduğu bir rolü siler. @everyone ve yönetilen roller silinemez.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageRoles`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`role_id`	`path`	`string`	evet	Rol ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/guilds/111500000000000000/roles/111550000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteRole("111500000000000000", "111550000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.delete_role("111500000000000000", "111550000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `everyone_role`, `managed_role`

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_role`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Üyeler

Üyeler için REST API rotaları.

Üyeleri listele (`listMembers`)

`GET https://noluyo.net/api/v1/guilds/{guild_id}/members` (`GET /guilds/{guild_id}/members`)

Üyeleri kullanıcı ID'sine göre sıralı döndürür. Sayfalamak için after'a son aldığın kullanıcı ID'sini ver. limit 1 ile 1000 arasındadır, varsayılanı 100'dür. Bot için GUILD_MEMBERS intent'i gerekir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`limit`	`query`	`integer`	-	Kaç üye döneceği. (1–1000 · = 100)
`after`	`query`	`ID`	-	Kullanıcı ID'si bundan büyük üyeleri döndürür.

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/guilds/111500000000000000/members?limit=100" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listMembers("111500000000000000", { limit: 100 });
```

```
# Python (noluyo_bot)
await bot.rest.list_members("111500000000000000", {"limit": 100})
```

Yanıt (200)

```
[
  {
    "user": {
      "id": "1114000000000000000",
      "username": "ayse",
      "display_name": "Ayşe",
      "avatar": null,
      "cosmetics": null
    },
    "nick": null,
    "roles": [],
    "joined_at": "2026-11-04T16:21:11.569Z",
    "mute": false,
    "deaf": false,
    "communication_disabled_until": null
  }
]
```

Hata kodları

- **HTTP 400:** `validation_failed`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`, `member_list_intent_required`
- **HTTP 404:** `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Üyeyi getir (`getMember`)

`GET https://noluyo.net/api/v1/guilds/{guild_id}/members/{user_id}` (`GET /guilds/{guild_id}/members/{user_id}`)

Tek bir üyeyi döndürür. Çağıranın kendisi için user_id @me olabilir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Üyenin kullanıcı ID'si ya da çağıran için @me.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/guilds/111500000000000000/members/@me" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getMember("111500000000000000", "@me");
```

```
# Python (noluyo_bot)
await bot.rest.get_member("111500000000000000", "@me")
```

Yanıt (200)

```
{
  "user": {
    "id": "114400000000000000",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  },
  "nick": null,
  "roles": [],
  "joined_at": "2026-11-04T16:21:11.569Z",
  "mute": false,
  "deaf": false,
  "communication_disabled_until": null
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_member`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Üyeyi güncelle (`updateMember`)

`PATCH https://noluyo.net/api/v1/guilds/{guild_id}/members/{user_id}` (`PATCH /guilds/{guild_id}/members/{user_id}`)

Yalnızca gönderilen alanları değiştirir. nick, kişinin kendisi için ChangeNickname, başkası için ManageNicknames ve daha yüksek rütbe ister. roles üyenin tüm rol listesini değiştirir ve ManageRoles ister; eklenen veya çıkarılan her rol çağıranın altında olmalı, yönetilen rol ya da @everyone olmamalıdır. mute MuteMembers, deaf

DeafenMembers ister. communication_disabled_until, ModerateMembers ve daha yüksek rütbe ister: en çok 28 gün sonrası bir zaman üyeyi susturur, null ya da geçmiş bir zaman susturmayı bitirir; sahip ve yöneticiler susturulamaz. channel_id, MoveMembers ister ve seste olan üyeyi başka bir ses kanalına taşır, null ise sesten düşürür.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ChangeNickname` , `ManageNicknames` , `ManageRoles` , `MuteMembers` , `DeafenMembers` , `ModerateMembers` , `MoveMembers`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Üyenin kullanıcı ID'si ya da çağıran için @me.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`nick`	`string | null`	-	(<= 32)
`roles`	`ID[]`	-	
`mute`	`boolean`	-	
`deaf`	`boolean`	-	
`communication_disabled_until`	`string | null`	-	
`channel_id`	`ID | null`	-	

Örnek çağrılar

```
curl -X PATCH "https://noluyo.net/api/v1/guilds/111500000000000000/members/111400000000000000" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"nick":"Ayşe (mod)"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.updateMember("111500000000000000", "111400000000000000", { nick: "Ayşe (mod)" });
```

```
# Python (noluyo_bot)
await bot.rest.update_member("111500000000000000", "111400000000000000", {"nick": "Ayşe (mod)"})
```

Yanıt (200)

```
{
  "user": {
    "id": "111400000000000000",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  },
  "nick": "Ayşe (mod)",
  "roles": [],
  "joined_at": "2026-11-04T16:21:11.569Z",
  "mute": false,
  "deaf": false,
  "communication_disabled_until": "2026-11-05T12:00:00Z"
}
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `managed_role`, `cannot_target_self`, `not_in_voice`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `cannot_timeout`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_member`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `voice_unavailable`, `maintenance`

Üyeyi çıkar (`kickMember`)

`DELETE https://noluyo.net/api/v1/guilds/{guild_id}/members/{user_id}` (`DELETE /guilds/{guild_id}/members/{user_id}`)

Çağıranın üstünde olduğu bir üyeyi sunucudan çıkarır. Kendin ayrılmak için DELETE /users/@me/guilds/{guild_id} kullan. Bot çıkarılırsa yönetilen rolü de silinir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `KickMembers`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Üyenin kullanıcı ID'si ya da çağıran için @me.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/guilds/111500000000000000/members/111400000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.kickMember("111500000000000000", "111400000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.kick_member("111500000000000000", "111400000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `cannot_target_self`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_member`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Üyeye rol ver (`addMemberRole`)

`PUT https://noluyo.net/api/v1/guilds/{guild_id}/members/{user_id}/roles/{role_id}` (`PUT /guilds/{guild_id}/members/{user_id}/roles/{role_id}`)

Çağıranın altındaki tek bir rolü verir. Yönetilen roller ve @everyone verilemez. İstek gövdesi yoktur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageRoles`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Üyenin kullanıcı ID'si ya da çağıran için @me.
`role_id`	`path`	`string`	evet	Rol ID'si.

Örnek çağrılar

```
curl -X PUT
"https://noluyo.net/api/v1/guilds/111500000000000000/members/111400000000000000/roles/111550000000000200" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.addMemberRole("111500000000000000", "111400000000000000", "111550000000000200");
```

```
# Python (noluyo_bot)
await bot.rest.add_member_role("111500000000000000", "111400000000000000", "111550000000000200")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `everyone_role`, `managed_role`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_member`, `unknown_role`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Üyeden rol al (`removeMemberRole`)

`DELETE https://noluyo.net/api/v1/guilds/{guild_id}/members/{user_id}/roles/{role_id}` (`DELETE /guilds/{guild_id}/members/{user_id}/roles/{role_id}`)

Çağıranın altındaki tek bir rolü alır. Yönetilen roller ve @everyone alınamaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageRoles`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Üyenin kullanıcı ID'si ya da çağıran için @me.
`role_id`	`path`	`string`	evet	Rol ID'si.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/guilds/111500000000000000/members/111400000000000000/roles/111550000000000200" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.removeMemberRole("111500000000000000", "111400000000000000",
"111550000000000200");
```

```
# Python (noluyo_bot)
await bot.rest.remove_member_role("111500000000000000", "111400000000000000",
"111550000000000200")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `everyone_role`, `managed_role`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_member`, `unknown_role`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Yasaklar

Yasaklar için REST API rotaları.

Yasakları listele (`listBans`)

`GET https://noluyo.net/api/v1/guilds/{guild_id}/bans` (`GET /guilds/{guild_id}/bans`)

Sunucunun yasaklarını nedenleriyle döndürür. Listede sayfalama yoktur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `BanMembers`
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/guilds/111500000000000000/bans" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listBans("111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.list_bans("111500000000000000")
```

Yanıt (200)

```
[
  {
    "user": {
      "id": "111400000000000000",
      "username": "ayse",
      "display_name": "Ayşe",
      "avatar": null,
      "cosmetics": null
    },
    "reason": "Spam"
  }
]
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Kullanıcıyı yasakla (`createBan`)

`PUT https://noluyo.net/api/v1/guilds/{guild_id}/bans/{user_id}` (`PUT /guilds/{guild_id}/bans/{user_id}`)

Kullanıcıyı yasaklar, üyeyse sunucudan çıkarır; üye, çağıranın altında olmalıdır. Üye olmayan biri önceden yasaklanabilir. Gövde isteğe bağlıdır. delete_message_seconds, kişinin o kadar saniye öncesine kadarki mesajlarını da siler; en fazla 604800 (yedi gün). reason en çok 512 karakterdir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `BanMembers`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Kullanıcı ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`reason`	`string | null`	-	(<= 512)
`delete_message_seconds`	`integer`	-	(0–604800)

Örnek çağrılar

```
curl -X PUT "https://noluyo.net/api/v1/guilds/111500000000000000/bans/111400000000000000" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"reason":"Spam","delete_message_seconds":3600}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createBan("111500000000000000", "111400000000000000", { reason: "Spam",
delete_message_seconds: 3600 });
```

```
# Python (noluyo_bot)
await bot.rest.create_ban("111500000000000000", "111400000000000000", {"reason": "Spam",
"delete_message_seconds": 3600})
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `cannot_target_self`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `role_hierarchy`, `temporarily_banned`
- **HTTP 404:** `unknown_guild`, `unknown_user`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Yasağı kaldır (`deleteBan`)

`DELETE https://noluyo.net/api/v1/guilds/{guild_id}/bans/{user_id}` (`DELETE /guilds/{guild_id}/bans/{user_id}`)

Yasağı kaldırır. Yasaklı olmayan kullanıcı için unknown_ban döner.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `BanMembers`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`user_id`	`path`	`string`	evet	Kullanıcı ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/guilds/111500000000000000/bans/111400000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteBan("111500000000000000", "111400000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.delete_ban("111500000000000000", "111400000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_ban`, `unknown_guild`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Kanallar

Kanallar için REST API rotaları.

Kanalı getir (`getChannel`)

`GET https://noluyo.net/api/v1/channels/{channel_id}` (`GET /channels/{channel_id}`)

Çağıranın görebildiği bir kanalı döndürür. ViewChannel izni olmayan üye missing_access, sunucunun dışındaki biri unknown_channel alır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/channels/111500000000000100" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getChannel("111500000000000100");
```

```
# Python (noluyo_bot)
await bot.rest.get_channel("111500000000000100")
```

Yanıt (200)


```
{
  "id": "111500000000000100",
  "type": 0,
  "guild_id": "111500000000000000",
  "name": "genel",
  "topic": null,
  "position": 1,
  "parent_id": null,
  "nsfw": false,
  "last_message_id": "111600000000000000",
  "permission_overwrites": []
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Kanalı güncelle (`updateChannel`)

`PATCH https://noluyo.net/api/v1/channels/{channel_id}` (`PATCH /channels/{channel_id}`)

Yalnızca gönderilen alanları değiştirir. topic ve rate_limit_per_user metin, bitrate ve user_limit ses kanalları içindir; parent_id aynı sunucudaki bir kategori olmalıdır, null kanalı kategorisinden çıkarır, kategorinin üst kategorisi olamaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageChannels`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`name`	`string`	-	(1–100)
`topic`	`string | null`	-	(<= 1024)
`position`	`integer`	-	
`parent_id`	`ID | null`	-	
`nsfw`	`boolean`	-	
`bitrate`	`integer`	-	(8000–96000)
`user_limit`	`integer`	-	(0–99)
`rate_limit_per_user`	`integer`	-	(0–21600)

Örnek çağrılar

```
curl -X PATCH "https://noluyo.net/api/v1/channels/111500000000000100" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"topic":"Hoş geldiniz"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.updateChannel("111500000000000100", { topic: "Hoş geldiniz" });
```

```
# Python (noluyo_bot)
await bot.rest.update_channel("111500000000000100", {"topic": "Hoş geldiniz"})
```

Yanıt (200)

```
{
  "id": "111500000000000100",
  "type": 0,
  "guild_id": "111500000000000000",
  "name": "genel",
  "topic": "Hoş geldiniz",
  "position": 1,
  "parent_id": null,
  "nsfw": false,
  "last_message_id": "111600000000000000",
  "permission_overwrites": [],
  "rate_limit_per_user": 5
}
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Kanalı sil (`deleteChannel`)

`DELETE https://noluyo.net/api/v1/channels/{channel_id}` (`DELETE /channels/{channel_id}`)

Kanalı siler ve silinen kanalın nesnesiyle yanıt verir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageChannels`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/channels/111500000000000100" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteChannel("111500000000000100");
```

```
# Python (noluyo_bot)
await bot.rest.delete_channel("111500000000000100")
```

Yanıt (200)

```
{
  "id": "111500000000000100",
  "type": 0,
  "guild_id": "111500000000000000",
  "name": "genel",
  "topic": null,
  "position": 1,
  "parent_id": null,
  "nsfw": false,
  "last_message_id": "111600000000000000",
  "permission_overwrites": []
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Yazıyor göstergesini göster (`sendTyping`)

`POST https://noluyo.net/api/v1/channels/{channel_id}/typing` (`POST /channels/{channel_id}/typing`)

Kanaldaki kişilere botun yazdığını gösterir. Yalnızca metin kanallarında çalışır. İstek gövdesi yoktur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `SendMessages`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/channels/111500000000000100/typing" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.sendTyping("111500000000000100");
```

```
# Python (noluyo_bot)
await bot.rest.send_typing("111500000000000100")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`, `dm_window_closed`
- **HTTP 404:** `unknown_channel`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Mesajlar

Mesajlar için REST API rotaları.

Mesajları listele (`listMessages`)

`GET https://noluyo.net/api/v1/channels/{channel_id}/messages` (`GET /channels/{channel_id}/messages`)

Mesajları en yeniden eskiye döndürür: son sayfayı ya da bir mesajın öncesini, sonrasını veya çevresini. before ile after birlikte, around da bunlardan biriyle kullanılamaz. ReadMessageHistory yoksa yanıt hata değil, boş listedir. Çağıranın tepkileri reactions[].me ile işaretlenir. Sunucu kanalında, içerik intent'i olmayan bir bot başkalarının mesajlarını içeriği çıkarılmış olarak alır; botu etiketleyenler hariç.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ReadMessageHistory`
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`limit`	`query`	`integer`	-	Kaç mesaj döneceği. (1–100 · = 50)
`before`	`query`	`ID`	-	Bu ID'den eski mesajlar.
`after`	`query`	`ID`	-	Bu ID'den yeni mesajlar. Ona en yakın olanlar döner, sıra yine en yeniden eskiyedir.
`around`	`query`	`ID`	-	Bu ID'nin çevresindeki mesajlar: kendisi ve yenileri sayfanın büyük yarısını doldurur.

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/channels/111500000000000100/messages?limit=50" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listMessages("111500000000000100", { limit: 50 });
```

```
# Python (noluyo_bot)
await bot.rest.list_messages("111500000000000100", {"limit": 50})
```

Yanıt (200)

```
[
  {
    "id": "111600000000000000",
    "type": 0,
    "channel_id": "111500000000000100",
    "guild_id": "111500000000000000",
    "author": {
      "id": "111540000000000001",
      "username": "ornek_bot",
      "display_name": "Örnek Bot",
      "avatar": null,
      "cosmetics": null,
      "bot": true
    },
    "content": "Merhaba!",
    "timestamp": "2026-11-04T22:58:33.427Z",
    "edited_timestamp": null,
    "attachments": [],
    "mentions": [],
    "mention_roles": [],
    "mention_everyone": false,
    "pinned": false
  }
]
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Mesaj gönder (`createMessage`)

`POST https://noluyo.net/api/v1/channels/{channel_id}/messages` (`POST /channels/{channel_id}/messages`)

Metin kanalına mesaj gönderir ve mesajla birlikte 200 döner. content en çok 2000 karakterdir; mesajda dosya, iletme, anket veya çıkartma yoksa zorunludur. Dosya göndermek için multipart/form-data kullan: JSON payload_json adlı parçada, en çok on dosya files[0], files[1] gibi adlandırılmış parçalarda olur; bunun için AttachFiles gerekir. message_reference mesajı yanıta çevirir (ReadMessageHistory); allowed_mentions.replied_user false, yanıtın kişiyi etiketlemesini önler. En çok 64 karakterlik nonce yalnızca bu yanıtta geri gelir. Bir mesajda anket, iletme ve çıkartmadan yalnızca biri olabilir; hiçbiri dosyayla birlikte gönderilemez. Sunucunun otomatik moderasyonu mesajı automod_blocked, kanalın yavaş modu slowmode ile reddedebilir. Botlar mesaja embed (en çok 10 kart) ve bileşen (buton ve menü, en çok 5 satır) ekleyebilir; nasıl

yapıldığını embed rehberi anlatır. Embed görselleri mesajla gönderilen dosyalardır, başka bir adres olamaz. flags 64 (ephemeral) ephemeral_forbidden döner. Embed ve bileşenler ayrıca, boyutları sunucunun üye sayısıyla çarpılarak x-rate-limits içindeki fanout tavanına sayılır: X-RateLimit-Scope bot ile gelen 429 sahibinin payı ya da bekleme süresidir, shared ile gelen 429 tüm botların ortak tavanıdır ve botun suçu değildir; max_bytes ile gelen 400 fanout_too_large, mesajın o kanal için fazla büyük olduğunu söyler. Bir kişinin botunla açtığı DM'de kendiliğinden ancak kişinin son mesajı, komutu ya da basmasından sonraki 24 saat içinde yazabilirsin (dm_window_closed); kişi DM'yi kapattıysa ya da botu engellediyse hiç yazamazsın (dm_closed); tüm DM'lerinde toplam dm_send sınırı geçerlidir. Ayrıntılar özel mesajlar rehberinde.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `SendMessages` , `AttachFiles` , `ReadMessageHistory` , `MentionEveryone` , `UseExternalStickers`
- **Hız sınırı:** `message` (5 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`content`	`string`	-	(<= 2000)
`nonce`	`string`	-	Daha uzunu hata vermeden atılır. (<= 64)
`message_reference`	`object`	-	
`message_reference.message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`allowed_mentions`	`object`	-	
`allowed_mentions.replied_user`	`boolean`	-	
`allowed_mentions.parse`	`string[]`	-	Yalnızca botlar için. Metindeki hangi etiket türlerinin bildirim göndereceği. parse, users veya roles gönderilirse yalnızca listelenenler bildirim gönderir; varsayılan için üçünü de yazma.
`allowed_mentions.users`	`ID[]`	-	Yalnızca botlar için. parse içinde users yokken de bildirim gönderen kullanıcılar. parse içindeki users ile birlikte yazılamaz. En çok 100. (<= 100)
`allowed_mentions.roles`	`ID[]`	-	Yalnızca botlar için. parse içinde roles yokken de bildirim gönderen roller. parse içindeki roles ile birlikte yazılamaz. En çok 100. (<= 100)
`forward`	`object`	-	
`forward.channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`poll`	`object`	-	
`poll.question`	`string`	evet	(1–300)
`poll.answers`	`string[]`	evet	(2–10)

Alan	Tür	Zorunlu	Açıklama
`poll.allow_multiselect`	`boolean`	-	
`poll.duration`	`integer`	-	Saat. (1, 4, 8, 24, 72, 168 · = 24)
`sticker_ids`	`ID[]`	-	(<= 1)
`embeds`	`Embed[]`	-	Yalnızca botlar için; kişi gönderirse validation_failed alır. Embed'li mesajda content gerekmez. Görseller aynı mesajla gönderilen dosyalardan gelir (attachment://); başka bir adres reddedilir. (<= 10)
`components`	`MessageComponent[]`	-	Yalnızca botlar için. Yalnızca bileşenli mesaj reddedilir: content, embed veya dosya gerekir. (<= 5)
`flags`	`integer`	-	Yalnızca botlar için. Boş bırak: 64 (ephemeral) ephemeral_forbidden döner çünkü kanal mesajının gösterileceği tek bir kişi yoktur; başka her bit invalid olur.

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/channels/111500000000000100/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"content":"Merhaba!"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createMessage("111500000000000100", { content: "Merhaba!" });
```

```
# Python (noluyo_bot)
await bot.rest.create_message("111500000000000100", {"content": "Merhaba!"})
```

Yanıt (200)


```
{
  "id": "111600000000000000",
  "type": 0,
  "channel_id": "111500000000000100",
  "guild_id": "111500000000000000",
  "author": {
    "id": "111540000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "content": "Merhaba!",
  "timestamp": "2026-11-04T22:58:33.427Z",
  "edited_timestamp": null,
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false,
  "nonce": "istek-1"
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `invalid_channel_type`, `automod_blocked`, `ephemeral_forbidden`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`, `dm_window_closed`
- **HTTP 404:** `unknown_channel`
- **HTTP 413:** `body_too_large`, `file_too_large`
- **HTTP 429:** `rate_limited`, `slowmode`, `upload_quota`
- **HTTP 503:** `maintenance`

Mesajı getir (`getMessage`)

`GET https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}` (`GET /channels/{channel_id}/messages/{message_id}`)

Tek bir mesajı döndürür. Listeden farklı olarak, ReadMessageHistory yoksa missing_permissions döner. Sunucu kanalında, içerik intent'i olmayan bir bot başkalarının mesajlarını içeriği çıkarılmış olarak alır; botu etiketleyenler hariç.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ReadMessageHistory`
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getMessage("111500000000000100", "111600000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.get_message("111500000000000100", "111600000000000000")
```

Yanıt (200)


```
{
  "id": "111600000000000000",
  "type": 0,
  "channel_id": "111500000000000100",
  "guild_id": "111500000000000000",
  "author": {
    "id": "111540000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "content": "Merhaba!",
  "timestamp": "2026-11-04T22:58:33.427Z",
  "edited_timestamp": null,
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false
}
```

Hata kodları

- **HTTP 400:** `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Mesajı düzenle (`editMessage`)

`PATCH https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}` (`PATCH /channels/{channel_id}/messages/{message_id}`)

Çağıranın kendi mesajını değiştirir. Kişi yalnızca content değiştirir. Bot content, embeds ve components alanlarını ayrı ayrı değiştirebilir: göndermediği alan kalır, boş liste embed veya bileşenleri kaldırır, yalnızca bileşen değişirse mesaj düzenlendi sayılmaz. Embed içindeki attachment:// yalnızca mesajın var olan dosyalarına bakar; yeni dosyalı istek reddedilir. Düzenleme, etiketlenenleri günceller ama kimseye yeniden bildirim göndermez. Dosya, iletme, anket, çıkartma veya embed içeren mesajın metni boşaltılabilir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `SendMessages`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`content`	`string`	-	(<= 2000)
`embeds`	`Embed[]`	-	Yalnızca botlar için. Embed'lerin yerine geçer. attachment:// yalnızca mesajın var olan dosyalarına bakar; yeni dosya yeni mesajla gelir. (<= 10)
`components`	`MessageComponent[]`	-	Yalnızca botlar için. Satırların yerine geçer. Yalnızca bu değişirse mesaj düzenlendi sayılmaz. (<= 5)
`allowed_mentions`	`object`	-	
`allowed_mentions.replied_user`	`boolean`	-	
`allowed_mentions.parse`	`string[]`	-	Yalnızca botlar için. Metindeki hangi etiket türlerinin bildirim göndereceği. parse, users veya roles gönderilirse yalnızca listelenenler bildirim gönderir; varsayılan için üçünü de yazma.
`allowed_mentions.users`	`ID[]`	-	Yalnızca botlar için. parse içinde users yokken de bildirim gönderen kullanıcılar. parse içindeki users ile birlikte yazılamaz. En çok 100. (<= 100)
`allowed_mentions.roles`	`ID[]`	-	Yalnızca botlar için. parse içinde roles yokken de bildirim gönderen roller. parse içindeki roles ile birlikte yazılamaz. En çok 100. (<= 100)

Örnek çağrılar

```
curl -X PATCH "https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000"
\
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"content":"Merhaba, düzenlendi!"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.editMessage("111500000000000100", "111600000000000000", { content: "Merhaba,
düzenlendi!" });
```

```
# Python (noluyo_bot)
await bot.rest.edit_message("111500000000000100", "111600000000000000", {"content": "Merhaba, düzenlendi!"})
```

Yanıt (200)

```
{
  "id": "111600000000000000",
  "type": 0,
  "channel_id": "111500000000000100",
  "guild_id": "111500000000000000",
  "author": {
    "id": "111540000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "content": "Merhaba, düzenlendi!",
  "timestamp": "2026-11-04T22:58:33.427Z",
  "edited_timestamp": "2026-11-05T09:00:00Z",
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false
}
```

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `invalid_channel_type`, `automod_blocked`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `not_author`, `system_message`, `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Mesajı sil (`deleteMessage`)

`DELETE https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}` (`DELETE /channels/{channel_id}/messages/{message_id}`)

Mesajı siler. Çağıranın kendi mesajı izin istemez; başkasının mesajı veya sunucudaki sistem mesajı ManageMessages ister.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageMessages`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteMessage("111500000000000100", "111600000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.delete_message("111500000000000100", "111600000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access` , `missing_permissions` , `temporarily_banned`
- **HTTP 404:** `unknown_channel` , `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Birden çok mesajı sil (`bulkDeleteMessages`)

`POST https://noluyo.net/api/v1/channels/{channel_id}/messages/bulk-delete` (`POST /channels/{channel_id}/messages/bulk-delete`)

Sunucudaki bir metin kanalının 2 ile 100 arası mesajını siler; hiçbiri 14 günden eski olamaz. Var olmayan mesajların ID'leri yok sayılır, tekrar edenler bir kez sayılır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageMessages`

- **Hız sınırı:** `bulk` (1 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`messages`	`ID[]`	evet	(2–100)

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/channels/111500000000000100/messages/bulk-delete" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"messages":["111600000000000000","111600000000000100"]}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.bulkDeleteMessages("111500000000000100", { messages: ["111600000000000000",
"111600000000000100"] });
```

```
# Python (noluyo_bot)
await bot.rest.bulk_delete_messages("111500000000000100", {"messages": ["111600000000000000",
"111600000000000100"]})
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `invalid_channel_type`, `messages_too_old`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sabitlenen mesajlar

Sabitlenen mesajlar için REST API rotaları.

Sabitlenen mesajları listele (`listPins`)

`GET https://noluyo.net/api/v1/channels/{channel_id}/pins` (`GET /channels/{channel_id}/pins`)

Sabitlenen mesajları, en son sabitlenenden başlayarak döndürür. limit 1 ile 50 arasındadır, varsayılanı 50'dir. Sayfalamak için has_more true olduğu sürece before'a son öğenin pinned_at değerini ver. ReadMessageHistory yoksa liste boştur. Sunucu kanalında, içerik intent'i olmayan bir bot başkalarının mesajlarını içeriği çıkarılmış olarak alır; botu etiketleyenler hariç.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ReadMessageHistory`
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`limit`	`query`	`integer`	-	Kaç sabitleme döneceği. (1–50 · = 50)
`before`	`query`	`datetime`	-	Bu zamandan önce yapılan sabitlemeler (RFC 3339).

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/channels/111500000000000100/pins?limit=50" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listPins("111500000000000100", { limit: 50 });
```

```
# Python (noluyo_bot)
await bot.rest.list_pins("111500000000000100", {"limit": 50})
```

Yanıt (200)


```
{
  "items": [
    {
      "pinned_at": "2026-11-05T09:00:00Z",
      "message": {
        "id": "111600000000000000",
        "type": 0,
        "channel_id": "111500000000000100",
        "guild_id": "111500000000000000",
        "author": {
          "id": "111540000000000001",
          "username": "ornek_bot",
          "display_name": "Örnek Bot",
          "avatar": null,
          "cosmetics": null,
          "bot": true
        },
        "content": "Merhaba!",
        "timestamp": "2026-11-04T22:58:33.427Z",
        "edited_timestamp": null,
        "attachments": [],
        "mentions": [],
        "mention_roles": [],
        "mention_everyone": false,
        "pinned": true
      }
    }
  ],
  "has_more": false
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Mesajı sabitle (`pinMessage`)

`PUT https://noluyo.net/api/v1/channels/{channel_id}/pins/{message_id}` (`PUT /channels/{channel_id}/pins/{message_id}`)

Mesajı sabitler ve kanala bir sabitleme bildirimi (mesaj türü 6) gönderir. Zaten sabit olan mesajı sabitlemek bir şey değiştirmez. Bir kanalda en çok 250 sabit mesaj olur. Sistem mesajları sabitlenemez.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageMessages` , `ReadMessageHistory`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.

Örnek çağrılar

```
curl -X PUT "https://noluyo.net/api/v1/channels/111500000000000100/pins/111600000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.pinMessage("111500000000000100", "111600000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.pin_message("111500000000000100", "111600000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `max_pins` , `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `system_message` , `missing_access` , `missing_permissions` , `temporarily_banned` , `dm_closed` , `dm_window_closed`
- **HTTP 404:** `unknown_channel` , `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Mesajın sabitlemesini kaldır (`unpinMessage`)

`DELETE https://noluyo.net/api/v1/channels/{channel_id}/pins/{message_id}` (`DELETE /channels/{channel_id}/pins/{message_id}`)

Mesajın sabitlemesini kaldırır. Kanala bir şey gönderilmez.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageMessages` , `ReadMessageHistory`
- **Hız sınırı:** `write` (10 / 10s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/channels/111500000000000100/pins/111600000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.unpinMessage("111500000000000100", "111600000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.unpin_message("111500000000000100", "111600000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `system_message` , `missing_access` , `missing_permissions` , `temporarily_banned` , `dm_closed` , `dm_window_closed`
- **HTTP 404:** `unknown_channel` , `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Tepkiler

Tepkiler için REST API rotaları.

Mesaja tepki ver (`addReaction`)

`PUT https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}/reactions/{emoji}/@me` (`PUT /channels/{channel_id}/messages/{message_id}/reactions/{emoji}/@me`)

Botun tepkisini ekler. Mesajda olmayan bir emoji AddReactions ister; zaten olan emojiye katılmak istemez, ama bot susturulmuşsa buna da izin verilmez. Bir mesajda en çok 20 farklı emoji olur. İki kez tepki vermek bir şey değiştirmez.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ReadMessageHistory` , `AddReactions`
- **Hız sınırı:** `reaction` (4 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.
`emoji`	`path`	`string`	evet	URL kodlanmış Unicode emoji; ya da sunucu emojisi için ad:id, hareketliyse a:ad:id veya yalnızca ID.

Örnek çağrılar

```
curl -X PUT
"https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000/reactions/%F0%
9F%91%8B/@me" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.addReaction("111500000000000100", "111600000000000000", "□");
```

```
# Python (noluyo_bot)
await bot.rest.add_reaction("111500000000000100", "111600000000000000", "□")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `unknown_emoji`, `too_many_reactions`, `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`, `dm_window_closed`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Tepkiyi kaldır (`removeReaction`)

`DELETE https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}/reactions/{emoji}/{user_id}`
(`DELETE /channels/{channel_id}/messages/{message_id}/reactions/{emoji}/{user_id}`)

Bir kişinin tepkisini kaldırır. Botun kendi tepkisi için user_id @me olabilir; başkasınınki ManageMessages ister.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ReadMessageHistory`, `ManageMessages`
- **Hız sınırı:** `reaction` (4 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.
`emoji`	`path`	`string`	evet	URL kodlanmış Unicode emoji; ya da sunucu emojisi için ad:id, hareketliyse a:ad:id veya yalnızca ID.
`user_id`	`path`	`string`	evet	Kullanıcı ID'si ya da botun kendisi için @me.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000/reactions/%F0%
9F%91%8B/@me" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.removeReaction("111500000000000100", "111600000000000000", "👋", "@me");
```

```
# Python (noluyo_bot)
await bot.rest.remove_reaction("111500000000000100", "111600000000000000", "👋", "@me")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `unknown_emoji`, `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`, `unknown_message`, `unknown_user`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Tepki verenleri listele (`listReactors`)

`GET https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}/reactions/{emoji}` (`GET /channels/{channel_id}/messages/{message_id}/reactions/{emoji}`)

Bir emojiyle tepki veren kullanıcıları döndürür. limit 1 ile 100 arasındadır, varsayılanı 25'tir; sayfalamak için after'a bir kullanıcı ID'si ver.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ReadMessageHistory`
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.
`emoji`	`path`	`string`	evet	URL kodlanmış Unicode emoji; ya da sunucu emojisi için ad:id, hareketliyse a:ad:id veya yalnızca ID.
`limit`	`query`	`integer`	-	Kaç kullanıcı döneceği. (1–100 · = 25)
`after`	`query`	`ID`	-	Bu kullanıcı ID'sinden sonrasını getirir.

Örnek çağrılar

```
curl
"https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000/reactions/%F0%9F%91%8B?limit=25" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listReactors("111500000000000100", "111600000000000000", "", { limit: 25 });
```

```
# Python (noluyo_bot)
await bot.rest.list_reactors("111500000000000100", "111600000000000000", "□", {"limit": 25})
```

Yanıt (200)

```
[
  {
    "id": "111400000000000000",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  }
]
```

Hata kodları

- **HTTP 400:** `unknown_emoji`, `validation_failed`, `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Bir emojinin tepkilerini kaldır (`clearEmojiReactions`)

`DELETE https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}/reactions/{emoji}` (`DELETE /channels/{channel_id}/messages/{message_id}/reactions/{emoji}`)

Mesajdan bu emojiyle verilmiş tüm tepkileri kaldırır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageMessages`
- **Hız sınırı:** `reaction` (4 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.
`emoji`	`path`	`string`	evet	URL kodlanmış Unicode emoji; ya da sunucu emojisi için ad:id, hareketliyse a:ad:id veya yalnızca ID.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000/reactions/%F0%9F%91%8B" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.clearEmojiReactions("111500000000000100", "111600000000000000", "□");
```

```
# Python (noluyo_bot)
await bot.rest.clear_emoji_reactions("111500000000000100", "111600000000000000", "□")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `unknown_emoji`, `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Tüm tepkileri kaldır (`clearReactions`)

`DELETE https://noluyo.net/api/v1/channels/{channel_id}/messages/{message_id}/reactions` (`DELETE /channels/{channel_id}/messages/{message_id}/reactions`)

Mesajdaki tüm tepkileri kaldırır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `ManageMessages`
- **Hız sınırı:** `reaction` (4 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`channel_id`	`path`	`string`	evet	Kanal ID'si.
`message_id`	`path`	`string`	evet	Mesaj ID'si.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/channels/111500000000000100/messages/111600000000000000/reactions" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.clearReactions("111500000000000100", "111600000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.clear_reactions("111500000000000100", "111600000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `invalid_channel_type`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `missing_permissions`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Uygulamalar

Geliştirici portalının kullandığı rotalar; bot token'ıyla değil, bir kişinin oturumuyla çağrılır.

Uygulamalarını listele (`listApplications`)

`GET https://noluyo.net/api/v1/applications` (`GET /applications`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Hesabın sahip olduğu uygulamaları döndürür. Token'lar asla yer almaz.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/applications" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listApplications({ accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.list_applications(access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
[
  {
    "id": "111540000000000000",
    "owner_id": "111400000000000500",
    "name": "Örnek Bot",
    "description": "Sunucuya hoş geldin diyen bot",
    "public": false,
    "verified": false,
    "bot": {
      "id": "111540000000000001",
      "username": "ornek_bot",
      "display_name": "Örnek Bot",
      "avatar": null,
      "cosmetics": null,
      "bot": true
    },
    "created_at": "2026-11-04T19:00:08.312Z",
    "privileged_intents": 0,
    "approved_intents": 0,
    "guild_count": 0,
    "interactions_endpoint_url": null,
    "public_key": "3b6a27bcceb6a42d62a3a8d02a6f0d73653215771de243a63ac048a18b59da29",
    "redirect_uris": [],
    "oauth_public": false,
    "privacy_url": null,
    "terms_url": null,
    "has_oauth_secret": false,
    "listed": false,
    "listing_category": null,
    "listing_text": null,
    "support_url": null,
    "website_url": null,
    "install_permissions": "0"
  }
]
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Uygulama oluştur (`createApplication`)

`POST https://noluyo.net/api/v1/applications` (`POST /applications`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Bir uygulama ve botunu oluşturur, bot token'ıyla yanıt verir. Token'ın gösterildiği yalnızca iki yerden biri budur; hemen sakla. Şimdilik yalnızca kapalı döneme davet edilen hesaplar uygulama oluşturabilir ve e-postaları doğrulanmış olmalıdır. username (a-z, 0-9, _ ve . içinden 4 ile 32 karakter; art arda iki nokta olamaz) küçük harfe çevrilir ve sonradan değiştirilemez. name en çok 32, description en çok 400 karakterdir. Bir hesap en çok 25 uygulamaya sahip olabilir. Platform açıldıktan sonra hesabın önce geliştirici sözleşmesini kabul etmesi gerekir (POST /applications/developer-terms), aksi halde developer_terms_required.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`name`	`string`	evet	(1–32)
`username`	`string`	evet	(4–32)
`description`	`string | null`	-	(<= 400)

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications" \
  -H "Authorization: Bearer $SESSION_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"name":"Örnek Bot","username":"ornek_bot","description":"Sunucuya hoş geldin diyen bot"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createApplication({ name: "Örnek Bot", username: "ornek_bot", description:
"Sunucuya hoş geldin diyen bot" }, { accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.create_application({"name": "Örnek Bot", "username": "ornek_bot", "description":
"Sunucuya hoş geldin diyen bot"}, access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (201)


```
{
  "application": {
    "id": "111540000000000000",
    "owner_id": "111400000000000500",
    "name": "Örnek Bot",
    "description": "Sunucuya hoş geldin diyen bot",
    "public": false,
    "verified": false,
    "bot": {
      "id": "111540000000000001",
      "username": "ornek_bot",
      "display_name": "Örnek Bot",
      "avatar": null,
      "cosmetics": null,
      "bot": true
    },
    "created_at": "2026-11-04T19:00:08.312Z",
    "privileged_intents": 0,
    "approved_intents": 0,
    "guild_count": 0,
    "interactions_endpoint_url": null,
    "public_key": "3b6a27bcceb6a42d62a3a8d02a6f0d73653215771de243a63ac048a18b59da29",
    "redirect_uris": [],
    "oauth_public": false,
    "privacy_url": null,
    "terms_url": null,
    "has_oauth_secret": false,
    "listed": false,
    "listing_category": null,
    "listing_text": null,
    "support_url": null,
    "website_url": null,
    "install_permissions": "0"
  },
  "token": "<BOT_TOKEN>"
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_applications`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `guest_not_allowed`, `email_not_verified`, `bot_developer_required`, `developer_terms_required`, `bot_forbidden`, `temporarily_banned`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Geliştirici sözleşmesini kabul et (`acceptDeveloperTerms`)

`POST https://noluyo.net/api/v1/applications/developer-terms` (`POST /applications/developer-terms`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Hesabın /yasal/gelistirici-sozlesmesi adresindeki geliştirici sözleşmesini kabul ettiğini kaydeder. Platform açıldıktan sonra hesabın ilk uygulamasını oluşturmadan önce buna ihtiyacı vardır. version sözleşmenin güncel sürümü, 1 olmalıdır; yeniden kabul etmek bir şey değiştirmez.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`version`	`integer`	evet	Sözleşmenin güncel sürümü.

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications/developer-terms" \
  -H "Authorization: Bearer $SESSION_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"version":1}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.acceptDeveloperTerms({ version: 1 }, { accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.accept_developer_terms({"version": 1}, access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Uygulamanı getir (`getApplication`)

`GET https://noluyo.net/api/v1/applications/{application_id}` (`GET /applications/{application_id}`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Hesabın uygulamalarından birini döndürür. Başkasınınki, herkese açık olsa bile unknown_application verir.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/applications/111540000000000000" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getApplication("111540000000000000", { accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.get_application("111540000000000000", access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
{
  "id": "1115400000000000000",
  "owner_id": "1114000000000000500",
  "name": "Örnek Bot",
  "description": "Sunucuya hoş geldin diyen bot",
  "public": false,
  "verified": false,
  "bot": {
    "id": "1115400000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "created_at": "2026-11-04T19:00:08.312Z",
  "privileged_intents": 0,
  "approved_intents": 0,
  "guild_count": 0,
  "interactions_endpoint_url": null,
  "public_key": "3b6a27bcceb6a42d62a3a8d02a6f0d73653215771de243a63ac048a18b59da29",
  "redirect_uris": [],
  "oauth_public": false,
  "privacy_url": null,
  "terms_url": null,
  "has_oauth_secret": false,
  "listed": false,
  "listing_category": null,
  "listing_text": null,
  "support_url": null,
  "website_url": null,
  "install_permissions": "0"
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Uygulamanı güncelle (`updateApplication`)

`PATCH https://noluyo.net/api/v1/applications/{application_id}` (`PATCH /applications/{application_id}`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Yalnızca gönderilen alanları değiştirir. Yeni ad, botun görünen adı da olur. public true, başkalarının botu kendi sunucularına eklemesine izin verir. description null açıklamayı siler. interactions_endpoint_url her şeyden önce test edilir; test geçmezse istekteki hiçbir şey kaydedilmez.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`name`	`string`	-	(1–32)
`description`	`string | null`	-	(<= 400)
`public`	`boolean`	-	
`privileged_intents`	`integer`	-	Açılacak ayrıcalıklı intent'ler: 2, 256 ve 32768'in herhangi bir toplamı. 100 sunucudan sonra henüz onaylanmamış bir bit intent_approval_required ile reddedilir; kapatmak her zaman çalışır.
`interactions_endpoint_url`	`string | null`	-	Botunun etkileşimler için https adresi. Kaydetmeden önce N'oluyo iki PING isteği yollar: doğru imzalıya 200 {"type":1}, yanlış imzalıya 401 dönmeli. İstek bunu bekler, en çok 6 saniye kadar; dakikada en çok 5 test. null gateway'e döner. Nasıl yapıldığını HTTP teslim rehberi anlatır. (<= 2048)
`redirect_uris`	`string[]`	-	Kayıtlı OAuth2 yönlendirme adresleri. (<= 10)
`oauth_public`	`boolean`	-	İstemci gizli anahtarı olmadan PKCE token değişimine izin ver.
`privacy_url`	`string | null`	-	HTTPS gizlilik metni adresi.
`terms_url`	`string | null`	-	HTTPS kullanım koşulları adresi.
`listing_category`	`string | null`	-	Bot rehberi kategorisi. (moderation, music, games, utility, fun, social, other,)
`listing_text`	`string | null`	-	Kısa rehber kartı özeti. (<= 160)
`support_url`	`string | null`	-	HTTPS destek adresi.
`website_url`	`string | null`	-	HTTPS web sitesi adresi.
`install_permissions`	`string`	-	Önerilen kurulum izin bitleri (ondalık metin).

Örnek çağrılar

```
curl -X PATCH "https://noluyo.net/api/v1/applications/1115400000000000000" \
  -H "Authorization: Bearer $SESSION_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"public":true}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.updateApplication("1115400000000000000", { public: true }, { accessToken:
process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.update_application("1115400000000000000", {"public": True},
access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
{
  "id": "1115400000000000000",
  "owner_id": "1114000000000000500",
  "name": "Yeni Ad",
  "description": "Sunucuya hoş geldin diyen bot",
  "public": true,
  "verified": false,
  "bot": {
    "id": "1115400000000000001",
    "username": "ornek_bot",
    "display_name": "Yeni Ad",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "created_at": "2026-11-04T19:00:08.312Z",
  "privileged_intents": 0,
  "approved_intents": 0,
  "guild_count": 0,
  "interactions_endpoint_url": null,
  "public_key": "3b6a27bcceb6a42d62a3a8d02a6f0d73653215771de243a63ac048a18b59da29",
  "redirect_uris": [],
  "oauth_public": false,
  "privacy_url": null,
  "terms_url": null,
  "has_oauth_secret": false,
  "listed": false,
  "listing_category": null,
  "listing_text": null,
  "support_url": null,
  "website_url": null,
  "install_permissions": "0"
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `intent_approval_required`, `endpoint_not_allowed`, `endpoint_unreachable`, `endpoint_bad_response`, `endpoint_no_signature_check`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Uygulamanı sil (`deleteApplication`)

`DELETE https://noluyo.net/api/v1/applications/{application_id}` (`DELETE /applications/{application_id}`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Uygulamayı ve bot hesabını siler. Token hemen çalışmaz olur.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/applications/111540000000000000" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteApplication("111540000000000000", { accessToken: process.env.SESSION_TOKEN
});
```

```
# Python (noluyo_bot)
await bot.rest.delete_application("111540000000000000", access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden` , `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Bot token'ını yenile (`resetApplicationToken`)

`POST https://noluyo.net/api/v1/applications/{application_id}/token` (`POST /applications/{application_id}/token`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Bot token'ını yeniler: eskisi hemen çalışmaz olur, yenisi yalnızca bu yanıtta gösterilir. İstek gövdesi yoktur.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications/111540000000000000/token" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.resetApplicationToken("111540000000000000", { accessToken:
process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.reset_application_token("111540000000000000",
access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)

```
{
  "token": "<BOT_TOKEN>"
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Adresinin teslim günlüğünü oku (`getEndpointDeliveries`)

`GET https://noluyo.net/api/v1/applications/{application_id}/endpoint/deliveries` (`GET /applications/{application_id}/endpoint/deliveries`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Etkileşim adresine yapılan son teslimleri ve çok hatadan sonra N'oluyo'nun onu aramayı bırakıp bırakmadığını gösterir. İçinde kişi ya da içerik yoktur.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/applications/111540000000000000/endpoint/deliveries" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getEndpointDeliveries("111540000000000000", { accessToken:
process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.get_endpoint_deliveries("111540000000000000",
access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)

```
{
  "deliveries": [
    {
      "at": "2026-11-08T14:02:11.204Z",
      "type": 2,
      "status": 0,
      "took_ms": 3000,
      "result": "timeout"
    },
    {
      "at": "2026-11-08T14:01:40.917Z",
      "type": 3,
      "status": 200,
      "took_ms": 84,
      "result": "ok"
    }
  ],
  "open": false
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`

- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Eklemeden önce uygulamayı gör (`getPublicApplication`)

`GET https://noluyo.net/api/v1/applications/{application_id}/public` (`GET /applications/{application_id}/public`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Sunucuya ekleme ekranında görünen bilgiler; sahibi yer almaz. Özel uygulamayı yalnızca sahibi görür; başkaları için, botu veya sahibi kapatılmış ya da silinmiş uygulama gibi unknown_application döner. icon, botun avatar hash'idir.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/applications/111540000000000000/public" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getPublicApplication("111540000000000000", { accessToken:
process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.get_public_application("111540000000000000",
access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
{
  "id": "1115400000000000000",
  "name": "Örnek Bot",
  "description": "Sunucuya hoş geldin diyen bot",
  "icon": null,
  "bot": {
    "id": "1115400000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "public": true,
  "verified": false
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Botu sunucuya ekle (`authorizeApplication`)

`POST https://noluyo.net/api/v1/applications/{application_id}/authorize` (`POST /applications/{application_id}/authorize`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Botu, çağıranın ManageGuild iznine sahip olduğu sunucuya ekler. Bot, doğrudan @everyone'ın üstünde, kendi adını taşıyan ve istenen izinlere sahip bir yönetilen rol alır; bu izinler çağıranda bulunmalıdır. Özel botu yalnızca sahibi ekleyebilir. Şimdilik sunucunun sahibi de kapalı döneme davet edilmiş bir hesap olmalıdır (aksi halde bots_closed). Henüz doğrulanmamış bir bot en çok 100 sunucuda olabilir.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)
- **İzinler:** `ManageGuild`

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`permissions`	`permissions`	-	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications/111540000000000000/authorize" \
  -H "Authorization: Bearer $SESSION_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"guild_id":"111500000000000000","permissions":"68672"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.authorizeApplication("111540000000000000", { guild_id: "111500000000000000",
permissions: "68672" }, { accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.authorize_application("111540000000000000", {"guild_id": "111500000000000000",
"permissions": "68672"}, access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `invalid_body`, `bot_in_guild`, `bot_guild_limit`, `max_roles`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_private`, `bots_closed`, `banned`, `missing_permissions`, `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_guild`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Botun resmini ayarla (`setApplicationAvatar`)

`POST https://noluyo.net/api/v1/applications/{application_id}/avatar` (`POST /applications/{application_id}/avatar`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Botun avatarını multipart/form-data ile yükler; görsel file adlı parçada olur: PNG, JPEG, WebP veya GIF, en çok 5 MB (hareketli GIF 10 MB) ve 4096×4096 piksel. Uygulamayla yanıt verir.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`file`	`string`	evet	

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications/111540000000000000/avatar" \
  -H "Authorization: Bearer $SESSION_TOKEN" \
  -F "file=@avatar.png"
```

```
// JavaScript (noluyo-bot)
await bot.rest.setApplicationAvatar("111540000000000000", { name: "avatar.png", data: new Uint8Array(), contentType: "image/png" }, { accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.set_application_avatar("111540000000000000", file={"name": "avatar.png", "data": b"", "content_type": "image/png"}, access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
{
  "id": "1115400000000000000",
  "owner_id": "1114000000000000500",
  "name": "Örnek Bot",
  "description": "Sunucuya hoş geldin diyen bot",
  "public": false,
  "verified": false,
  "bot": {
    "id": "1115400000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": "6f1c0a9d2b3e4f5a6b7c8d9e0f1a2b3c",
    "cosmetics": null,
    "bot": true
  },
  "created_at": "2026-11-04T19:00:08.312Z",
  "privileged_intents": 0,
  "approved_intents": 0,
  "guild_count": 0,
  "interactions_endpoint_url": null,
  "public_key": "3b6a27bcceb6a42d62a3a8d02a6f0d73653215771de243a63ac048a18b59da29"
}
```


Hata kodları

- **HTTP 400:** `invalid_image`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 413:** `image_too_large`
- **HTTP 429:** `upload_quota`, `rate_limited`
- **HTTP 503:** `maintenance`

Botun resmini kaldır (`deleteApplicationAvatar`)

`DELETE https://noluyo.net/api/v1/applications/{application_id}/avatar` (`DELETE /applications/{application_id}/avatar`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Kendin yalnızca otomasyon için çağır. Botun avatarını kaldırır ve uygulamayla yanıt verir.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X DELETE "https://noluyo.net/api/v1/applications/111540000000000000/avatar" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteApplicationAvatar("111540000000000000", { accessToken:
process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.delete_application_avatar("111540000000000000",
access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
{
  "id": "111540000000000000",
  "owner_id": "111400000000000500",
  "name": "Örnek Bot",
  "description": "Sunucuya hoş geldin diyen bot",
  "public": false,
  "verified": false,
  "bot": {
    "id": "111540000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "created_at": "2026-11-04T19:00:08.312Z",
  "privileged_intents": 0,
  "approved_intents": 0,
  "guild_count": 0,
  "interactions_endpoint_url": null,
  "public_key": "3b6a27bcceb6a42d62a3a8d02a6f0d73653215771de243a63ac048a18b59da29"
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

OAuth2 istemci gizli anahtarını yenile (`resetOAuthSecret`)

`POST https://noluyo.net/api/v1/applications/{application_id}/oauth-secret` (`POST /applications/{application_id}/oauth-secret`)

Geliştirici portalı bu rotayı, bot token'ıyla değil, bir kişinin oturumuyla (Authorization: Bearer) kullanır. Gizli istemcili uygulamalar için yeni bir OAuth2 istemci gizli anahtarı üretir; önceki anahtar hemen geçersiz olur, düz değer yalnızca bu yanıtta döner.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications/111540000000000000/oauth-secret" \
  -H "Authorization: Bearer $SESSION_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.resetOAuthSecret("111540000000000000", { accessToken: process.env.SESSION_TOKEN
});
```

```
# Python (noluyo_bot)
await bot.rest.reset_oauth_secret("111540000000000000", access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)

```
{
  "client_secret": "<CLIENT_SECRET>"
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Son doğrulama başvurusunu getir (`getApplicationReview`)

`GET https://noluyo.net/api/v1/applications/{application_id}/review` (`GET /applications/{application_id}/review`)

Bu uygulama için gönderilmiş son doğrulama başvurusunu döndürür; henüz başvuru yoksa `{review: null}` döner.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl https://noluyo.net/api/v1/applications/21879167385600004/review \
  -H "Authorization: Bearer $USER_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getApplicationReview("21879167385600004", { accessToken: process.env.SESSION_TOKEN
});
```

```
# Python (noluyo_bot)
await bot.rest.get_application_review("21879167385600004",
access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (200)


```
{
  "review": {
    "id": "22104891204198401",
    "application_id": "21879167385600004",
    "status": "pending",
    "purpose": "Moderates spam and runs server polls.",
    "data_use": "Stores per-guild filter words and active poll tallies until closed.",
    "requested_intents": 32768,
    "want_listing": true,
    "note": null,
    "created_at": "2026-11-11T00:00:00Z",
    "decided_at": null
  }
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 503:** `maintenance`

Doğrulama başvurusu gönder (`submitApplicationReview`)

`POST https://noluyo.net/api/v1/applications/{application_id}/review` (`POST /applications/{application_id}/review`)

Uygulamayı doğrulama incelemesine gönderir (100 sunucu üstü, ölçekte ayrıcalıklı intent onayı ve bot rehberinde yer almak için gerekir). Herkese açık bot, açıklama, avatar, https gizlilik adresi, https destek adresi ve sahip hesabında açık iki adımlı doğrulama ister.

- **Kimlik:** Yalnızca kullanıcı oturumu (bot token'ı reddedilir)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`purpose`	`string`	evet	(10–2000)
`data_use`	`string`	evet	(10–2000)
`requested_intents`	`integer`	-	(= 0)
`want_listing`	`boolean`	-	(= false)

Örnek çağrılar

```
curl -X POST https://noluyo.net/api/v1/applications/21879167385600004/review \
  -H "Authorization: Bearer $USER_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"purpose":"Moderates spam and runs server polls.","data_use":"Stores per-guild filter words and active poll tallies.","requested_intents":32768,"want_listing":true}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.submitApplicationReview("21879167385600004", { purpose: "Moderates spam and runs server polls.", data_use: "Stores per-guild filter words and active poll tallies.", requested_intents: 32768, want_listing: true }, { accessToken: process.env.SESSION_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.submit_application_review("21879167385600004", {"purpose": "Moderates spam and runs server polls.", "data_use": "Stores per-guild filter words and active poll tallies.", "requested_intents": 32768, "want_listing": True}, access_token=os.environ["SESSION_TOKEN"])
```

Yanıt (201)


```
{
  "id": "22104891204198401",
  "application_id": "21879167385600004",
  "status": "pending",
  "purpose": "Moderates spam and runs server polls.",
  "data_use": "Stores per-guild filter words and active poll tallies until closed.",
  "requested_intents": 32768,
  "want_listing": true,
  "note": null,
  "created_at": "2026-11-11T00:00:00Z",
  "decided_at": null
}
```

Hata kodları

- **HTTP 400:** `review_requirements`, `review_pending`, `validation_failed`, `invalid_body`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Gateway

Botların nereye ve nasıl bağlanacağı.

Bot gateway bilgisini getir (`getGatewayBot`)

`GET https://noluyo.net/api/v1/gateway/bot` (`GET /gateway/bot`)

Botun nereye bağlanacağını ve kaç shard açacağını söyler. `shards`, botun bulunduğu sunucu sayısına göre önerilen sayıdır. `session_start_limit` botun o günkü oturum başlatmalarını gösterir: `total`, `remaining`, `reset_after` (günlük pencerenin sıfırlanmasına kalan milisaniye) ve `max_concurrency` (5 saniyede başlatılabilecek oturum sayısı). Okumak oturum başlatma hakkı harcamaz. Yalnızca bot token'ları.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/gateway/bot" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getGatewayBot();
```

```
# Python (noluyo_bot)
await bot.rest.get_gateway_bot()
```

Yanıt (200)

```
{
  "url": "wss://noluyo.net/bot-gateway",
  "shards": 1,
  "session_start_limit": {
    "total": 1000,
    "remaining": 999,
    "reset_after": 14400000,
    "max_concurrency": 1
  }
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`

- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Etkileşimler

Botun, bileşenine basılmasına verdiği yanıt. Bu rotalar etkileşim token'ının yanında botun token'ını da ister.

Etkileşime yanıt ver (`interactionCallback`)

`POST https://noluyo.net/api/v1/interactions/{interaction_id}/{interaction_token}/callback` (`POST /interactions/{interaction_id}/{interaction_token}/callback`)

Basmaya ya da komut çalıştırmaya verilen ilk yanıt, 3 saniye içinde gelmelidir; bir etkileşim bir yanıt alır. Mesaj yanıtlarından komut çalıştırma yalnızca type 4 ve 5 alır: güncellenecek mesajı yoktur. type 4 yeni mesaj yazar, 5 PATCH @original ile doldurulacak bir "düşünüyor" mesajı (flags 128) yazar, 6 yalnızca onaylar ve izin istemez, 7 basılan mesajı günceller, göndermediğin alan kalır. Reddedilen yanıt hakkı harcamaz. Basan kişiye başarı bildirilir. 4, 5 ve 7 {"message": ...} ile 200, 6 ise 204 döner. Yavaş mod işlemez; otomatik moderasyon ve ortak tavanlar işler. type 5, data içinden yalnızca flags'i okur; type 6 data'yı yok sayar. type 4 ya da 5 ile flags 64, yanıtı yalnızca basan kişiye gösterir ve kimliği etkileşimin kimliğidir; ephemeral mesaja basıldığında type 7 o mesajı bütünüyle değiştirir. Otomatik tamamlama (type 4) yalnızca data.choices ile type 8 alır; 204 döner, seçenekler yalnızca yazan kişiye gider ve token'ı takip mesajı ya da düzenleme almaz. Type 8 ve ephemeral yanıtlar ortak yazma tavanına sayılmaz: yalnızca herkesin gördüğü 4, 5 ve 7 sayılır. Komut çalıştırma ya da basma (otomatik tamamlama ve gönderilen form değil) bunun yerine data {custom_id, title, components} ile type 9 alabilir: kişiye form açılır, 204 döner, hiçbir şey yazmaz ve yazma tavanına sayılmaz. Formu açmak tek yanıttır; önce 5 verip sonra açamazsın. Kişinin gönderdiği yeni bir etkileşim (type 5) olarak gelir; o 4 ya da 5 alır, 6 ya da 7'yi yalnızca form bir mesajın butonundan açıldıysa alır. Etkileşim adresi olan uygulama etkileşimi orada alır ve aynı yanıtı o isteğin 200 gövdesi olarak verebilir; nasıl yapıldığını HTTP teslim rehberi anlatır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `SendMessages`
- **Hız sınırı:** `interaction` (50 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`interaction_id`	`path`	`string`	evet	INTERACTION_CREATE olayının id'si.
`interaction_token`	`path`	`string`	evet	Aynı olayın token'ı. Gizlidir, 15 dakika geçerlidir.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`type`	`integer`	evet	4 yeni mesaj, 5 sonradan doldurulacak "düşünüyor" mesajı, 6 sessiz onay, 7 basılan mesajı güncelleme. 8 otomatik tamamlamanın seçenekleri; otomatik tamamlama yalnızca 8 alır. 9 form açma; komut çalıştırmaya ya da basmaya ilk yanıt olarak. (4, 5, 6, 7, 8, 9)
`data`	`object`	-	Yanıtın gösterdiği şey. Yeni mesajda metin ya da embed zorunludur.
`data.content`	`string`	-	(<= 2000)
`data.embeds`	`Embed[]`	-	Yalnızca botlar için; kişi gönderirse validation_failed alır. Embed'li mesajda content gerekmez. Görseller aynı mesajla gönderilen dosyalardan gelir (attachment://); başka bir adres reddedilir. (<= 10)
`data.components`	`MessageComponent[]`	-	Yalnızca botlar için. Yalnızca bileşenli mesaj reddedilir: content, embed veya dosya gerekir. type 9'da formun alanları: 1-5 eylem satırı, her biri bir metin girişi (4) ya da etiketli bir menü taşır. (<= 5)
`data.allowed_mentions`	`object`	-	
`data.allowed_mentions.replied_user`	`boolean`	-	
`data.allowed_mentions.parse`	`string[]`	-	Yalnızca botlar için. Metindeki hangi etiket türlerinin bildirim göndereceği. parse, users veya roles gönderilirse yalnızca listelenenler bildirim gönderir; varsayılan için üçünü de yazma.
`data.allowed_mentions.users`	`ID[]`	-	Yalnızca botlar için. parse içinde users yokken de bildirim gönderen kullanıcılar. parse

Alan	Tür	Zorunlu	Açıklama
			içindeki users ile birlikte yazılamaz. En çok 100. (<= 100)
`data.allowed_mentions.roles`	`ID[]`	-	Yalnızca botlar için. parse içinde roles yokken de bildirim gönderen roller. parse içindeki roles ile birlikte yazılamaz. En çok 100. (<= 100)
`data.flags`	`integer`	-	0 ya da yalnızca basan kişinin göreceği ephemeral yanıt için 64. Başka değer reddedilir.
`data.choices`	`OptionChoice[]`	-	Yalnızca type 8'de, orada zorunludur ve yanında mesajın başka alanı olmaz: en çok 25 öneri, her değer odaktaki seçeneğin türündedir. Boş liste hiçbir şeyin eşleşmediğini söyler. Kişi onları kendi dilinde görür (name_localizations). (<= 25)
`data.custom_id`	`string`	-	Yalnızca type 9'da, orada zorunludur: formun kimliği, kişinin gönderdiğiyle geri gelir. (<= 100)
`data.title`	`string`	-	Yalnızca type 9'da, orada zorunludur: formun başlığı. Otomatik moderasyon onu, etiketleri ve ön doldurulan değerleri okur. Kişi her formun altında kaldıramayacağın bir satır görür: form botuna gider, içine asla şifre ya da doğrulama kodu yazmamalıdır. (1–45)

Örnek çağrılar

```
curl -X POST
"https://noluyo.net/api/v1/interactions/111600000000000400/$INTERACTION_TOKEN/callback" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"type":4,"data":{"content":"Kaydın alındı."}}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.interactionCallback("111600000000000400", process.env.INTERACTION_TOKEN, { type: 4, data: { content: "Kaydın alındı." } });
```

```
# Python (noluyo_bot)
await bot.rest.interaction_callback("111600000000000400", os.environ["INTERACTION_TOKEN"], {"type": 4, "data": {"content": "Kaydın alındı."}})
```

Yanıt (200)


```
{
  "message": {
    "id": "111600000000000000",
    "type": 0,
    "channel_id": "111500000000000100",
    "guild_id": "111500000000000000",
    "author": {
      "id": "111540000000000001",
      "username": "ornek_bot",
      "display_name": "Örnek Bot",
      "avatar": null,
      "cosmetics": null,
      "bot": true
    },
    "content": "Kaydın alındı.",
    "timestamp": "2026-11-04T22:58:33.427Z",
    "edited_timestamp": null,
    "attachments": [],
    "mentions": [],
    "mention_roles": [],
    "mention_everyone": false,
    "pinned": false,
    "nonce": "istek-1",
    "application_id": "111540000000000000",
    "interaction": {
      "id": "111600000000000400",
      "type": 3,
      "user": {
        "id": "111540000000000009",
        "username": "ayse",
        "display_name": null,
        "avatar": null,
        "cosmetics": null
      }
    }
  }
}
```


Hata kodları

- **HTTP 400:** `interaction_token_invalid`, `interaction_expired`, `interaction_timeout`, `interaction_already_acknowledged`, `interaction_response_invalid`, `validation_failed`, `invalid_body`, `automod_blocked`, `ephemeral_files`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`

- **HTTP 503:** `maintenance`

Takip mesajı gönder (`createInteractionFollowup`)

`POST https://noluyo.net/api/v1/interactions/{interaction_id}/{interaction_token}/messages` (`POST /interactions/{interaction_id}/{interaction_token}/messages`)

Yanıtlanmış bir etkileşim için bir mesaj daha yazar; basmadan sonra 15 dakika boyunca, etkileşim başına en çok 5 tane. Dosyalar, mesaj göndermedeki gibi multipart/form-data ve AttachFiles ile gider. flags 64 onu yalnızca basan kişiye gösterir; dosya almaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `SendMessages` , `AttachFiles`
- **Hız sınırı:** `interaction` (50 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`interaction_id`	`path`	`string`	evet	INTERACTION_CREATE olayının id'si.
`interaction_token`	`path`	`string`	evet	Aynı olayın token'ı. Gizlidir, 15 dakika geçerlidir.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`content`	`string`	-	(<= 2000)
`embeds`	`Embed[]`	-	Yalnızca botlar için; kişi gönderirse validation_failed alır. Embed'li mesajda content gerekmez. Görseller aynı mesajla gönderilen dosyalardan gelir (attachment://); başka bir adres reddedilir. (<= 10)
`components`	`MessageComponent[]`	-	Yalnızca botlar için. Yalnızca bileşenli mesaj reddedilir: content, embed veya dosya gerekir. type 9'da formun alanları: 1-5 eylem satırı, her biri bir metin girişi (4) ya da etiketli bir menü taşır. (<= 5)
`allowed_mentions`	`object`	-	
`allowed_mentions.replied_user`	`boolean`	-	
`allowed_mentions.parse`	`string[]`	-	Yalnızca botlar için. Metindeki hangi etiket türlerinin bildirim göndereceği. parse, users veya roles gönderilirse yalnızca listelenenler bildirim gönderir; varsayılan için üçünü de yazma.
`allowed_mentions.users`	`ID[]`	-	Yalnızca botlar için. parse içinde users yokken de bildirim gönderen kullanıcılar. parse içindeki users ile birlikte yazılamaz. En çok 100. (<= 100)
`allowed_mentions.roles`	`ID[]`	-	Yalnızca botlar için. parse içinde roles yokken de bildirim gönderen roller. parse içindeki roles ile birlikte yazılamaz. En çok 100. (<= 100)
`flags`	`integer`	-	0 ya da yalnızca basan kişinin göreceği ephemeral yanıt için 64. Başka değer reddedilir.
`choices`	`OptionChoice[]`	-	Yalnızca type 8'de, orada zorunludur ve yanında mesajın başka alanı olmaz: en çok 25 öneri, her değer odaktaki seçeneğin türündedir. Boş liste hiçbir şeyin eşleşmediğini söyler. Kişi onları

Alan	Tür	Zorunlu	Açıklama
			kendi dilinde görür (name_localizations). (<= 25)
`custom_id`	`string`	-	Yalnızca type 9'da, orada zorunludur: formun kimliği, kişinin gönderdiğiyle geri gelir. (<= 100)
`title`	`string`	-	Yalnızca type 9'da, orada zorunludur: formun başlığı. Otomatik moderasyon onu, etiketleri ve ön doldurulan değerleri okur. Kişi her formun altında kaldıramayacağın bir satır görür: form botuna gider, içine asla şifre ya da doğrulama kodu yazmamalıdır. (1–45)

Örnek çağrılar

```
curl -X POST
"https://noluyo.net/api/v1/interactions/111600000000000400/$INTERACTION_TOKEN/messages" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"content":"Bir şey daha"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createInteractionFollowup("111600000000000400", process.env.INTERACTION_TOKEN, {
content: "Bir şey daha" });
```

```
# Python (noluyo_bot)
await bot.rest.create_interaction_followup("111600000000000400", os.environ["INTERACTION_TOKEN"],
{"content": "Bir şey daha"})
```

Yanıt (200)


```
{
  "id": "1116000000000000000",
  "type": 0,
  "channel_id": "1115000000000000100",
  "guild_id": "1115000000000000000",
  "author": {
    "id": "1115400000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "content": "Kaydın alındı.",
  "timestamp": "2026-11-04T22:58:33.427Z",
  "edited_timestamp": null,
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false,
  "nonce": "istek-1",
  "application_id": "1115400000000000000",
  "interaction": {
    "id": "1116000000000000400",
    "type": 3,
    "user": {
      "id": "1115400000000000009",
      "username": "ayse",
      "display_name": null,
      "avatar": null,
      "cosmetics": null
    }
  }
}
```


Hata kodları

- **HTTP 400:** `interaction_token_invalid`, `interaction_expired`, `interaction_not_acknowledged`, `validation_failed`, `invalid_body`, `automod_blocked`, `ephemeral_files`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`
- **HTTP 404:** `unknown_channel`
- **HTTP 413:** `body_too_large`, `file_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Etkileşimin mesajını düzenle (`editInteractionMessage`)

`PATCH https://noluyo.net/api/v1/interactions/{interaction_id}/{interaction_token}/messages/{message_id}`
(`PATCH /interactions/{interaction_id}/{interaction_token}/messages/{message_id}`)

Bu etkileşimin bir mesajını düzenler: @original type 4 veya 5 yanıtıdır, takip mesajı kendi kimliğiyle bulunur. Göndermediğin alan kalır. "Düşünüyor" mesajını doldurmak flags 128'i siler, düzenlendi işareti koymaz, içindeki etiket de bildirim göndermez. Satırı olmayan kimlik ephemeral mesaj sayılır, yanıtın satırı yoksa @original da: 200 döner, yalnızca basan kişiye gider, göndermediğin alan gider. Hiç gönderilmemiş kimliği tutan istemci olmadığından type 6 ya da 7 sonrası ekranda bir şey değişmez. Ephemeral düzenleme etkileşim başına 5 saniyede en çok 5 kezdir.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **İzinler:** `SendMessages`
- **Hız sınırı:** `interaction` (50 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`interaction_id`	`path`	`string`	evet	INTERACTION_CREATE olayının id'si.
`interaction_token`	`path`	`string`	evet	Aynı olayın token'ı. Gizlidir, 15 dakika geçerlidir.
`message_id`	`path`	`string`	evet	Bu etkileşimin bir mesajı ya da type 4 veya 5 yanıtı için @original.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`content`	`string`	-	(<= 2000)
`embeds`	`Embed[]`	-	Yalnızca botlar için. Embed'lerin yerine geçer. attachment:// yalnızca mesajın var olan dosyalarına bakar; yeni dosya yeni mesajla gelir. (<= 10)
`components`	`MessageComponent[]`	-	Yalnızca botlar için. Satırların yerine geçer. Yalnızca bu değişirse mesaj düzenlendi sayılmaz. (<= 5)
`allowed_mentions`	`object`	-	
`allowed_mentions.replied_user`	`boolean`	-	
`allowed_mentions.parse`	`string[]`	-	Yalnızca botlar için. Metindeki hangi etiket türlerinin bildirim göndereceği. parse, users veya roles gönderilirse yalnızca listelenenler bildirim gönderir; varsayılan için üçünü de yazma.
`allowed_mentions.users`	`ID[]`	-	Yalnızca botlar için. parse içinde users yokken de bildirim gönderen kullanıcılar. parse içindeki users ile birlikte yazılamaz. En çok 100. (<= 100)
`allowed_mentions.roles`	`ID[]`	-	Yalnızca botlar için. parse içinde roles yokken de bildirim gönderen roller. parse içindeki roles ile birlikte yazılamaz. En çok 100. (<= 100)

Örnek çağrılar

```
curl -X PATCH
"https://noluyo.net/api/v1/interactions/111600000000000400/$INTERACTION_TOKEN/messages/@original"
\
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"content":"Rapor hazır."}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.editInteractionMessage("111600000000000400", process.env.INTERACTION_TOKEN,
"@original", { content: "Rapor hazır." });
```

```
# Python (noluyo_bot)
await bot.rest.edit_interaction_message("111600000000000400", os.environ["INTERACTION_TOKEN"],
"@original", {"content": "Rapor hazır."})
```

Yanıt (200)

```
{
  "id": "111600000000000000",
  "type": 0,
  "channel_id": "111500000000000100",
  "guild_id": "111500000000000000",
  "author": {
    "id": "111540000000000001",
    "username": "ornek_bot",
    "display_name": "Örnek Bot",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "content": "Kaydın alındı.",
  "timestamp": "2026-11-04T22:58:33.427Z",
  "edited_timestamp": null,
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false,
  "nonce": "istek-1",
  "application_id": "111540000000000000",
  "interaction": {
    "id": "111600000000000400",
    "type": 3,
    "user": {
      "id": "111540000000000009",
      "username": "ayse",
      "display_name": null,
      "avatar": null,
      "cosmetics": null
    }
  }
}
```

Hata kodları

- **HTTP 400:** `interaction_token_invalid`, `interaction_expired`, `interaction_not_acknowledged`, `validation_failed`, `invalid_body`, `automod_blocked`, `ephemeral_files`
- **HTTP 401:** `unauthorized`

- **HTTP 403:** `bot_forbidden`, `missing_access`, `missing_permissions`, `temporarily_banned`, `dm_closed`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Etkileşimin mesajını sil (`deleteInteractionMessage`)

`DELETE https://noluyo.net/api/v1/interactions/{interaction_id}/{interaction_token}/messages/{message_id}`
(`DELETE /interactions/{interaction_id}/{interaction_token}/messages/{message_id}`)

Bu etkileşimin bir mesajını siler; PATCH hangilerine ulaşıyorsa onlara. Kendi mesajların gibi izin istemez. Ephemeral mesaj basan kişinin ekranlarından gider.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `interaction` (50 / 1s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`interaction_id`	`path`	`string`	evet	INTERACTION_CREATE olayının id'si.
`interaction_token`	`path`	`string`	evet	Aynı olayın token'ı. Gizlidir, 15 dakika geçerlidir.
`message_id`	`path`	`string`	evet	Bu etkileşimin bir mesajı ya da type 4 veya 5 yanıtı için @original.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/interactions/111600000000000400/$INTERACTION_TOKEN/messages/@original"
\
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteInteractionMessage("111600000000000400", process.env.INTERACTION_TOKEN,
"@original");
```

```
# Python (noluyo_bot)
await bot.rest.delete_interaction_message("111600000000000400", os.environ["INTERACTION_TOKEN"],
"@original")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 400:** `interaction_token_invalid`, `interaction_expired`, `interaction_not_acknowledged`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `bot_forbidden`, `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_channel`, `unknown_message`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Komutlar

Botun sunduğu komutlar: eğik çizgi komutları, kullanıcı ve mesaj menüleri; global ya da bir sunucuya özel. Kişinin oturumu, sahibi olduğu uygulamalar için aynı rotalara ulaşır.

Global komutları listele (`listGlobalCommands`)

`GET https://noluyo.net/api/v1/applications/{application_id}/commands` (`GET /applications/{application_id}/commands`)

Uygulamanın global komutlarını seçenekleriyle listeler. Global komutlar botun bulunduğu her sunucuda çalışır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/applications/111540000000000000/commands" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listGlobalCommands("111540000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.list_global_commands("111540000000000000")
```

Yanıt (200)


```
[
  {
    "id": "111560000000000000",
    "application_id": "1115400000000000000",
    "version": "1115600000000000001",
    "guild_id": null,
    "type": 1,
    "name": "selam",
    "name_localizations": {
      "en": "hello"
    },
    "description": "Birini selamlar",
    "description_localizations": {
      "en": "Greets someone"
    },
    "options": [
      {
        "type": 6,
        "name": "kisi",
        "name_localizations": {
          "en": "person"
        },
        "description": "Selamlanacak kişi",
        "description_localizations": {
          "en": "The person to greet"
        },
        "required": true
      }
    ],
    "default_member_permissions": null,
    "contexts": [
      0,
      1
    ]
  }
]
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Global komutların hepsini değiştir (`replaceGlobalCommands`)

`PUT https://noluyo.net/api/v1/applications/{application_id}/commands` (`PUT /applications/{application_id}/commands`)

Listeyi global komutlarının tamamı yapar, tek adımda. Türü ve adı eşleşen komutun kimliği kalır; listede olmayanlar silinir; boş liste hepsini siler. Bir tanım geçersizse hiçbir şey değişmez. Günlük sınıra yalnız yeni komutlar sayılır; bunu bot açılırken gönder. Global komutlar botun bulunduğu her sunucuda çalışır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Örnek çağrılar

```
curl -X PUT "https://noluyo.net/api/v1/applications/111540000000000000/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '[{"name":"selam","description":"Birini selamlar","options":
[{"type":6,"name":"kisi","description":"Selamlanacak kişi","required":true}]},
{"type":2,"name":"Profili aç"}]'
```

```
// JavaScript (noluyo-bot)
await bot.rest.replaceGlobalCommands("111540000000000000", [{ name: "selam", description: "Birini
selamlar", options: [{ type: 6, name: "kisi", description: "Selamlanacak kişi", required: true }]
}, { type: 2, name: "Profili aç" }]);
```

```
# Python (noluyo_bot)
await bot.rest.replace_global_commands("111540000000000000", [{"name": "selam", "description":
"Birini selamlar", "options": [{"type": 6, "name": "kisi", "description": "Selamlanacak kişi",
"required": True}]}, {"type": 2, "name": "Profili aç"}])
```

Yanıt (200)


```
[
  {
    "id": "111560000000000000",
    "application_id": "111540000000000000",
    "version": "111560000000000001",
    "guild_id": null,
    "type": 1,
    "name": "selam",
    "name_localizations": {
      "en": "hello"
    },
    "description": "Birini selamlar",
    "description_localizations": {
      "en": "Greets someone"
    },
    "options": [
      {
        "type": 6,
        "name": "kisi",
        "name_localizations": {
          "en": "person"
        },
        "description": "Selamlanacak kişi",
        "description_localizations": {
          "en": "The person to greet"
        },
        "required": true
      }
    ],
    "default_member_permissions": null,
    "contexts": [
      0,
      1
    ]
  }
]
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_commands`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Global komutu oluştur (`createGlobalCommand`)

`POST https://noluyo.net/api/v1/applications/{application_id}/commands` (`POST /applications/{application_id}/commands`)

Komut ekler. Aynı tür ve adda komut varsa tanımı değişir, kimliği kalır: 201 yerine 200 döner. Yeni komut, kapsam başına günde 200 yeni komuta sayılır; güncelleme sayılmaz. Kişiler değişikliği en geç 60 saniyede görür. Global komutlar botun bulunduğu her sunucuda çalışır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`type`	`integer`	-	1 eğik çizgi komutu (varsayılan), 2 kullanıcı menüsü, 3 mesaj menüsü. (1, 2, 3)
`name`	`string`	evet	Eğik çizgi komutu: küçük harf, rakam, - ve _. Menü: en çok 32 karakter, boşluk ve büyük harf olabilir. (1–32)
`name_localizations`	`object`	-	Adın tr ve en karşılığı.
`description`	`string`	-	Eğik çizgi komutunda zorunlu, menüde boş. (<= 100)
`description_localizations`	`object`	-	Açıklamanın tr ve en karşılığı.
`options`	`CommandOption[]`	-	Yalnız eğik çizgi komutunda. (<= 25)
`default_member_permissions`	`permissions | null`	-	Sunucu kural koyana dek kimin kullanacağı: null herkes, "0" yalnız sahip ve yöneticiler, değilse bu bitlerin hepsine sahip üyeler.
`contexts`	`integer[]`	-	Nerede çalıştığı: 0 sunucu kanalı, 1 kişinin botunla açtığı DM. Gönderilmezse [0, 1]. Her biri en çok bir kez. default_member_permissions dolu komut bu ne derse desin DM'de çalışmaz: orada kimsenin izin biti yok. (1–2)

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/applications/1115400000000000000/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"name":"selam","description":"Birini selamlar","options":
[{"type":6,"name":"kisi","description":"Selamlanacak kişi","required":true}]}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createGlobalCommand("1115400000000000000", { name: "selam", description: "Birini
selamlar", options: [{ type: 6, name: "kisi", description: "Selamlanacak kişi", required: true }]
});
```

```
# Python (noluyo_bot)
await bot.rest.create_global_command("1115400000000000000", {"name": "selam", "description":
"Birini selamlar", "options": [{"type": 6, "name": "kisi", "description": "Selamlanacak kişi",
"required": True}]})
```

Yanıt (200)


```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": null,
  "type": 1,
  "name": "selam",
  "name_localizations": {
    "en": "hello"
  },
  "description": "Birini selamlar",
  "description_localizations": {
    "en": "Greets someone"
  },
  "options": [
    {
      "type": 6,
      "name": "kisi",
      "name_localizations": {
        "en": "person"
      },
      "description": "Selamlanacak kişi",
      "description_localizations": {
        "en": "The person to greet"
      },
      "required": true
    }
  ],
  "default_member_permissions": null,
  "contexts": [
    0,
    1
  ]
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_commands`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Global komutu getir (`getGlobalCommand`)

`GET https://noluyo.net/api/v1/applications/{application_id}/commands/{command_id}` (`GET /applications/{application_id}/commands/{command_id}`)

Global komutlarından birini döner. Global komutlar botun bulunduğu her sunucuda çalışır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`command_id`	`path`	`string`	evet	Komut ID'si.

Örnek çağrılar

```
curl -X GET
"https://noluyo.net/api/v1/applications/111540000000000000/commands/111560000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getGlobalCommand("111540000000000000", "111560000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.get_global_command("111540000000000000", "111560000000000000")
```

Yanıt (200)


```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": null,
  "type": 1,
  "name": "selam",
  "name_localizations": {
    "en": "hello"
  },
  "description": "Birini selamlar",
  "description_localizations": {
    "en": "Greets someone"
  },
  "options": [
    {
      "type": 6,
      "name": "kisi",
      "name_localizations": {
        "en": "person"
      },
      "description": "Selamlanacak kişi",
      "description_localizations": {
        "en": "The person to greet"
      },
      "required": true
    }
  ],
  "default_member_permissions": null,
  "contexts": [
    0,
    1
  ]
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Global komutu düzenle (`editGlobalCommand`)

`PATCH https://noluyo.net/api/v1/applications/{application_id}/commands/{command_id}` (`PATCH /applications/{application_id}/commands/{command_id}`)

Gönderilen alanları değiştirir, diğerleri kalır. type değişmez: komutu silip yeniden oluştur. version değişir. Global komutlar botun bulunduğu her sunucuda çalışır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`command_id`	`path`	`string`	evet	Komut ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`type`	`integer`	-	1 eğik çizgi komutu (varsayılan), 2 kullanıcı menüsü, 3 mesaj menüsü. (1, 2, 3)
`name`	`string`	evet	Eğik çizgi komutu: küçük harf, rakam, - ve _. Menü: en çok 32 karakter, boşluk ve büyük harf olabilir. (1–32)
`name_localizations`	`object`	-	Adın tr ve en karşılığı.
`description`	`string`	-	Eğik çizgi komutunda zorunlu, menüde boş. (<= 100)
`description_localizations`	`object`	-	Açıklamanın tr ve en karşılığı.
`options`	`CommandOption[]`	-	Yalnız eğik çizgi komutunda. (<= 25)
`default_member_permissions`	`permissions | null`	-	Sunucu kural koyana dek kimin kullanacağı: null herkes, "0" yalnız sahip ve yöneticiler, değilse bu bitlerin hepsine sahip üyeler.
`contexts`	`integer[]`	-	Nerede çalıştığı: 0 sunucu kanalı, 1 kişinin botunla açtığı DM. Gönderilmezse [0, 1]. Her biri en çok bir kez. default_member_permissions dolu komut bu ne derse desin DM'de çalışmaz: orada kimsenin izin biti yok. (1–2)

Örnek çağrılar

```
curl -X PATCH
"https://noluyo.net/api/v1/applications/111540000000000000/commands/111560000000000000" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"description":"Birine merhaba der"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.editGlobalCommand("111540000000000000", "111560000000000000", { description:
"Birine merhaba der" });
```

```
# Python (noluyo_bot)
await bot.rest.edit_global_command("111540000000000000", "111560000000000000", {"description":
"Birine merhaba der"})
```

Yanıt (200)


```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": null,
  "type": 1,
  "name": "selam",
  "name_localizations": {
    "en": "hello"
  },
  "description": "Birini selamlar",
  "description_localizations": {
    "en": "Greets someone"
  },
  "options": [
    {
      "type": 6,
      "name": "kisi",
      "name_localizations": {
        "en": "person"
      },
      "description": "Selamlanacak kişi",
      "description_localizations": {
        "en": "The person to greet"
      },
      "required": true
    }
  ],
  "default_member_permissions": null,
  "contexts": [
    0,
    1
  ]
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Global komutu sil (`deleteGlobalCommand`)

`DELETE https://noluyo.net/api/v1/applications/{application_id}/commands/{command_id}` (`DELETE /applications/{application_id}/commands/{command_id}`)

Global komutlarından birini izin kurallarıyla siler. Global komutlar botun bulunduğu her sunucuda çalışır.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`command_id`	`path`	`string`	evet	Komut ID'si.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/applications/111540000000000000/commands/111560000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteGlobalCommand("111540000000000000", "111560000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.delete_global_command("111540000000000000", "111560000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucu komutlarını listele (`listGuildCommands`)

`GET https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands` (`GET /applications/{application_id}/guilds/{guild_id}/commands`)

Uygulamanın bu sunucunun komutlarını seçenekleriyle listeler. Botun bir sunucuya özel komutları. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listGuildCommands("111540000000000000", "111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.list_guild_commands("111540000000000000", "111500000000000000")
```

Yanıt (200)


```
[
  {
    "id": "111560000000000000",
    "application_id": "111540000000000000",
    "version": "111560000000000001",
    "guild_id": "111500000000000000",
    "type": 1,
    "name": "selam",
    "name_localizations": {
      "en": "hello"
    },
    "description": "Birini selamlar",
    "description_localizations": {
      "en": "Greets someone"
    },
    "options": [
      {
        "type": 6,
        "name": "kisi",
        "name_localizations": {
          "en": "person"
        },
        "description": "Selamlanacak kişi",
        "description_localizations": {
          "en": "The person to greet"
        },
        "required": true
      }
    ],
    "default_member_permissions": null,
    "contexts": [
      0,
      1
    ]
  }
]
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucu komutlarının hepsini değiştir (`replaceGuildCommands`)

`PUT https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands` (`PUT /applications/{application_id}/guilds/{guild_id}/commands`)

Listeyi bu sunucunun komutlarının tamamı yapar, tek adımda. Türü ve adı eşleşen komutun kimliği kalır; listede olmayanlar silinir; boş liste hepsini siler. Bir tanım geçersizse hiçbir şey değişmez. Günlük sınıra yalnız yeni komutlar sayılır; bunu bot açılırken gönder. Botun bir sunucuya özel komutları. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X PUT
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '[{"name":"selam","description":"Birini selamlar","options":
[{"type":6,"name":"kisi","description":"Selamlanacak kişi","required":true}]},
{"type":2,"name":"Profili aç"}]'
```

```
// JavaScript (noluyo-bot)
await bot.rest.replaceGuildCommands("111540000000000000", "111500000000000000", [{ name: "selam",
description: "Birini selamlar", options: [{ type: 6, name: "kisi", description: "Selamlanacak
kişi", required: true }] }, { type: 2, name: "Profili aç" }]);
```

```
# Python (noluyo_bot)
await bot.rest.replace_guild_commands("111540000000000000", "111500000000000000", [{"name":
"selam", "description": "Birini selamlar", "options": [{"type": 6, "name": "kisi", "description":
"Selamlanacak kişi", "required": True}]}, {"type": 2, "name": "Profili aç"}])
```

Yanıt (200)


```
[
  {
    "id": "111560000000000000",
    "application_id": "111540000000000000",
    "version": "111560000000000001",
    "guild_id": "111500000000000000",
    "type": 1,
    "name": "selam",
    "name_localizations": {
      "en": "hello"
    },
    "description": "Birini selamlar",
    "description_localizations": {
      "en": "Greets someone"
    },
    "options": [
      {
        "type": 6,
        "name": "kisi",
        "name_localizations": {
          "en": "person"
        },
        "description": "Selamlanacak kişi",
        "description_localizations": {
          "en": "The person to greet"
        },
        "required": true
      }
    ],
    "default_member_permissions": null,
    "contexts": [
      0,
      1
    ]
  }
]
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_commands`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucu komutu oluştur (`createGuildCommand`)

`POST https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands` (`POST /applications/{application_id}/guilds/{guild_id}/commands`)

Komut ekler. Aynı tür ve adda komut varsa tanımı değişir, kimliği kalır: 201 yerine 200 döner. Yeni komut, kapsam başına günde 200 yeni komuta sayılır; güncelleme sayılmaz. Kişiler değişikliği en geç 60 saniyede görür. Botun bir sunucuya özel komutları. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`type`	`integer`	-	1 eğik çizgi komutu (varsayılan), 2 kullanıcı menüsü, 3 mesaj menüsü. (1, 2, 3)
`name`	`string`	evet	Eğik çizgi komutu: küçük harf, rakam, - ve _. Menü: en çok 32 karakter, boşluk ve büyük harf olabilir. (1–32)
`name_localizations`	`object`	-	Adın tr ve en karşılığı.
`description`	`string`	-	Eğik çizgi komutunda zorunlu, menüde boş. (<= 100)
`description_localizations`	`object`	-	Açıklamanın tr ve en karşılığı.
`options`	`CommandOption[]`	-	Yalnız eğik çizgi komutunda. (<= 25)
`default_member_permissions`	`permissions | null`	-	Sunucu kural koyana dek kimin kullanacağı: null herkes, "0" yalnız sahip ve yöneticiler, değilse bu bitlerin hepsine sahip üyeler.
`contexts`	`integer[]`	-	Nerede çalıştığı: 0 sunucu kanalı, 1 kişinin botunla açtığı DM. Gönderilmezse [0, 1]. Her biri en çok bir kez. default_member_permissions dolu komut bu ne derse desin DM'de çalışmaz: orada kimsenin izin biti yok. (1–2)

Örnek çağrılar

```
curl -X POST
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"name":"selam","description":"Birini selamlar","options":
[{"type":6,"name":"kisi","description":"Selamlanacak kişi","required":true}]}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.createGuildCommand("111540000000000000", "111500000000000000", { name: "selam",
description: "Birini selamlar", options: [{ type: 6, name: "kisi", description: "Selamlanacak
kişi", required: true }] });
```

```
# Python (noluyo_bot)
await bot.rest.create_guild_command("111540000000000000", "111500000000000000", {"name": "selam",
"description": "Birini selamlar", "options": [{"type": 6, "name": "kisi", "description":
"Selamlanacak kişi", "required": True}]})
```

Yanıt (200)


```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": "1115000000000000000",
  "type": 1,
  "name": "selam",
  "name_localizations": {
    "en": "hello"
  },
  "description": "Birini selamlar",
  "description_localizations": {
    "en": "Greets someone"
  },
  "options": [
    {
      "type": 6,
      "name": "kisi",
      "name_localizations": {
        "en": "person"
      },
      "description": "Selamlanacak kişi",
      "description_localizations": {
        "en": "The person to greet"
      },
      "required": true
    }
  ],
  "default_member_permissions": null,
  "contexts": [
    0,
    1
  ]
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `max_commands`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucu komutunu getir (`getGuildCommand`)

`GET https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands/{command_id}` (`GET /applications/{application_id}/guilds/{guild_id}/commands/{command_id}`)

Bu sunucunun komutlarından birini döner. Botun bir sunucuya özel komutları. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`command_id`	`path`	`string`	evet	Komut ID'si.

Örnek çağrılar

```
curl -X GET "https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands/111560000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getGuildCommand("111540000000000000", "111500000000000000", "111560000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.get_guild_command("111540000000000000", "111500000000000000", "111560000000000000")
```

Yanıt (200)


```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": "1115000000000000000",
  "type": 1,
  "name": "selam",
  "name_localizations": {
    "en": "hello"
  },
  "description": "Birini selamlar",
  "description_localizations": {
    "en": "Greets someone"
  },
  "options": [
    {
      "type": 6,
      "name": "kisi",
      "name_localizations": {
        "en": "person"
      },
      "description": "Selamlanacak kişi",
      "description_localizations": {
        "en": "The person to greet"
      },
      "required": true
    }
  ],
  "default_member_permissions": null,
  "contexts": [
    0,
    1
  ]
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucu komutunu düzenle (`editGuildCommand`)

`PATCH https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands/{command_id}`
(`PATCH /applications/{application_id}/guilds/{guild_id}/commands/{command_id}`)

Gönderilen alanları değiştirir, diğerleri kalır. type değişmez: komutu silip yeniden oluştur. version değişir. Botun bir sunucuya özel komutları. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`command_id`	`path`	`string`	evet	Komut ID'si.

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`type`	`integer`	-	1 eğik çizgi komutu (varsayılan), 2 kullanıcı menüsü, 3 mesaj menüsü. (1, 2, 3)
`name`	`string`	evet	Eğik çizgi komutu: küçük harf, rakam, - ve _. Menü: en çok 32 karakter, boşluk ve büyük harf olabilir. (1–32)
`name_localizations`	`object`	-	Adın tr ve en karşılığı.
`description`	`string`	-	Eğik çizgi komutunda zorunlu, menüde boş. (<= 100)
`description_localizations`	`object`	-	Açıklamanın tr ve en karşılığı.
`options`	`CommandOption[]`	-	Yalnız eğik çizgi komutunda. (<= 25)
`default_member_permissions`	`permissions | null`	-	Sunucu kural koyana dek kimin kullanacağı: null herkes, "0" yalnız sahip ve yöneticiler, değilse bu bitlerin hepsine sahip üyeler.
`contexts`	`integer[]`	-	Nerede çalıştığı: 0 sunucu kanalı, 1 kişinin botunla açtığı DM. Gönderilmezse [0, 1]. Her biri en çok bir kez. default_member_permissions dolu komut bu ne derse desin DM'de çalışmaz: orada kimsenin izin biti yok. (1–2)

Örnek çağrılar

```
curl -X PATCH
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands/111
560000000000000" \
  -H "Authorization: Bot $BOT_TOKEN" \
  -H 'Content-Type: application/json' \
  -d '{"description":"Birine merhaba der"}'
```

```
// JavaScript (noluyo-bot)
await bot.rest.editGuildCommand("111540000000000000", "111500000000000000", "111560000000000000",
{ description: "Birine merhaba der" });
```

```
# Python (noluyo_bot)
await bot.rest.edit_guild_command("111540000000000000", "111500000000000000",
"111560000000000000", {"description": "Birine merhaba der"})
```

Yanıt (200)


```
{
  "id": "1115600000000000000",
  "application_id": "1115400000000000000",
  "version": "1115600000000000001",
  "guild_id": "1115000000000000000",
  "type": 1,
  "name": "selam",
  "name_localizations": {
    "en": "hello"
  },
  "description": "Birini selamlar",
  "description_localizations": {
    "en": "Greets someone"
  },
  "options": [
    {
      "type": 6,
      "name": "kisi",
      "name_localizations": {
        "en": "person"
      },
      "description": "Selamlanacak kişi",
      "description_localizations": {
        "en": "The person to greet"
      },
      "required": true
    }
  ],
  "default_member_permissions": null,
  "contexts": [
    0,
    1
  ]
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`
- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 413:** `body_too_large`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucu komutunu sil (`deleteGuildCommand`)

`DELETE https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands/{command_id}`
(`DELETE /applications/{application_id}/guilds/{guild_id}/commands/{command_id}`)

Bu sunucunun komutlarından birini izin kurallarıyla siler. Botun bir sunucuya özel komutları. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `command` (5 / 20s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.
`command_id`	`path`	`string`	evet	Komut ID'si.

Örnek çağrılar

```
curl -X DELETE
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands/111560000000000000" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.deleteGuildCommand("111540000000000000", "111500000000000000",
"111560000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.delete_guild_command("111540000000000000", "111500000000000000",
"111560000000000000")
```

Yanıt (204)

Yanıt gövdesi yok.

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Sunucudaki komut kurallarını listele (`listCommandPermissions`)

`GET https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands/permissions` (`GET /applications/{application_id}/guilds/{guild_id}/commands/permissions`)

Bu botun komutları için bir sunucuda konmuş her kural: kuralı olan her komut için bir kayıt, tüm komutlarının kuralları için uygulamanın ID'siyle bir kayıt. Kuralı olmayan komut listede yer almaz. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur. Kuralları yalnızca sunucu yöneticileri N'oluyo'da yazar; bot okur ve değişince APPLICATION_COMMAND_PERMISSIONS_UPDATE alır. Sunucu kuralları zaten uygular: kişinin kullanamayacağı komut ona gösterilmez ve çalışmaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands/permissions" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listCommandPermissions("111540000000000000", "111500000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.list_command_permissions("111540000000000000", "111500000000000000")
```

Yanıt (200)


```
[
  {
    "id": "111540000000000000",
    "application_id": "111540000000000000",
    "guild_id": "111500000000000000",
    "permissions": [
      {
        "type": 1,
        "id": "111500000000000000",
        "allow": false
      }
    ]
  },
  {
    "id": "111560000000000000",
    "application_id": "111540000000000000",
    "guild_id": "111500000000000000",
    "permissions": [
      {
        "type": 1,
        "id": "111520000000000000",
        "allow": true
      },
      {
        "type": 3,
        "id": "0",
        "allow": false
      },
      {
        "type": 3,
        "id": "111510000000000000",
        "allow": true
      }
    ]
  }
]
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Komutun kurallarını getir (`getCommandPermissions`)

`GET`
`https://noluyo.net/api/v1/applications/{application_id}/guilds/{guild_id}/commands/{command_id}/permissions`
(`GET /applications/{application_id}/guilds/{guild_id}/commands/{command_id}/permissions`)

Bir komutun bir sunucudaki kuralları: global komut ya da bu sunucunun komutu. Kural yoksa liste boştur. Tüm komutların kuralları (uygulamanın ID'siyle olan kayıt) burada yer almaz; onları yukarıdaki rotayla listele. Bot o sunucuda üye olmalı; değilse yanıt missing_access olur. Kuralları yalnızca sunucu yöneticileri N'oluyo'da yazar; bot okur ve değişince APPLICATION_COMMAND_PERMISSIONS_UPDATE alır. Sunucu kuralları zaten uygular: kişinin kullanamayacağı komut ona gösterilmez ve çalışmaz.

- **Kimlik:** Bot token'ı (`Authorization: Bot <BOT_TOKEN>`)
- **Hız sınırı:** `read` (30 / 5s)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`application_id`	`path`	`string`	evet	Uygulama ID'si.
`guild_id`	`path`	`string`	evet	Sunucu ID'si.

Örnek çağrılar

```
curl -X GET
"https://noluyo.net/api/v1/applications/111540000000000000/guilds/111500000000000000/commands/111560000000000000/permissions" \
  -H "Authorization: Bot $BOT_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getCommandPermissions("111540000000000000", "111500000000000000",
"111560000000000000");
```

```
# Python (noluyo_bot)
await bot.rest.get_command_permissions("111540000000000000", "111500000000000000",
"111560000000000000")
```

Yanıt (200)


```
{
  "id": "111560000000000000",
  "application_id": "111540000000000000",
  "guild_id": "111500000000000000",
  "permissions": [
    {
      "type": 1,
      "id": "111520000000000000",
      "allow": true
    },
    {
      "type": 3,
      "id": "0",
      "allow": false
    },
    {
      "type": 3,
      "id": "111510000000000000",
      "allow": true
    }
  ]
}
```


Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `missing_access`, `temporarily_banned`
- **HTTP 404:** `unknown_application`, `unknown_command`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

OAuth2 (N'oluyo ile giriş)

Kişileri kendi onaylarıyla sitende ya da uygulamanda N'oluyo hesabıyla giriş yaptır. /oauth2/token ve /oauth2/revoke istekleri application/x-www-form-urlencoded gövde alır ve hataları {error, error_description} olarak döndürür; /oauth2/@me ve /oauth2/@me/guilds ise Authorization: Bearer <access_token> ister.

Yetki kodunu ya da yenileme token'ını değiştir (`exchangeOAuthToken`)

`POST https://noluyo.net/api/v1/oauth2/token` (`POST /oauth2/token`)

Tek kullanımlık yetki kodunu (PKCE code_verifier ile) ya da yenileme token'ını 1 saatlik yeni erişim token'ı ve 30 günlük yenileme token'ı ile değiştirir. En çok 8 KiB boyutunda application/x-www-form-urlencoded gövde alır. Gizli istemciler client_secret değerini formda ya da HTTP Basic ile gönderir; açık istemciler gizli anahtar olmadan client_id ve PKCE gönderir. Hatalar standart OAuth2 {error, error_description} nesnesiyle döner.

- **Kimlik:** Herkese açık (Authorization başlığı gerekmez)

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`grant_type`	`string`	evet	(authorization_code, refresh_token)
`client_id`	`string`	-	
`client_secret`	`string`	-	
`code`	`string`	-	
`redirect_uri`	`string`	-	
`code_verifier`	`string`	-	
`refresh_token`	`string`	-	

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/oauth2/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=authorization_code&client_id=111540000000000000&client_secret=$CLIENT_SECRET&code=$CODE&redirect_uri=https%3A%2F%2Fornek.com%2Fcb&code_verifier=$CODE_VERIFIER"
```

```
// JavaScript (noluyo-bot)
await bot.rest.exchangeOAuthToken({ grant_type: "authorization_code", client_id:
"111540000000000000", client_secret: process.env.CLIENT_SECRET, code: process.env.CODE,
redirect_uri: "https://ornek.com/cb", code_verifier: process.env.CODE_VERIFIER });
```

```
# Python (noluyo_bot)
await bot.rest.exchange_oauth_token({"grant_type": "authorization_code", "client_id":
"111540000000000000", "client_secret": os.environ["CLIENT_SECRET"], "code": os.environ["CODE"],
"redirect_uri": "https://ornek.com/cb", "code_verifier": os.environ["CODE_VERIFIER"]})
```

Yanıt (200)

```
{
  "access_token": "<ACCESS_TOKEN>",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "<REFRESH_TOKEN>",
  "scope": "identify email"
}
```

Hata kodları

- **HTTP 400:** `invalid_request`, `invalid_grant`, `unsupported_grant_type`
- **HTTP 401:** `invalid_client`
- **HTTP 429:** `temporarily_unavailable`
- **HTTP 503:** `temporarily_unavailable`

Erişim ya da yenileme token'ını iptal et (`revokeOAuthToken`)

`POST https://noluyo.net/api/v1/oauth2/revoke` (`POST /oauth2/revoke`)

Verilen erişim ya da yenileme token'ının ait olduğu yetkinin tüm token'larını geçersiz kılar. Bilinmeyen ya da başka uygulamaya ait token'lar da 200 döner.

- **Kimlik:** Herkese açık (Authorization başlığı gerekmez)

Gövde alanları

Alan	Tür	Zorunlu	Açıklama
`token`	`string`	evet	
`client_id`	`string`	-	
`client_secret`	`string`	-	

Örnek çağrılar

```
curl -X POST "https://noluyo.net/api/v1/oauth2/revoke" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "client_id=111540000000000000&client_secret=$CLIENT_SECRET&token=$ACCESS_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.revokeOAuthToken({ client_id: "111540000000000000", client_secret:
process.env.CLIENT_SECRET, token: process.env.ACCESS_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.revoke_oauth_token({"client_id": "111540000000000000", "client_secret":
os.environ["CLIENT_SECRET"], "token": os.environ["ACCESS_TOKEN"]})
```

Yanıt (200)

```
{}
```

Hata kodları

- **HTTP 400:** `invalid_request`
- **HTTP 401:** `invalid_client`
- **HTTP 429:** `temporarily_unavailable`
- **HTTP 503:** `temporarily_unavailable`

Giriş yapan kişinin kimliğini oku (`getOAuthMe`)

`GET https://noluyo.net/api/v1/oauth2/@me` (`GET /oauth2/@me`)

Bir OAuth2 erişim token'ı (`identify` kapsamı) için uygulamayı, onaylı kapsamları, token bitiş zamanını ve kişinin kimliğini döndürür. `email` kapsamı da varsa `email` (doğrulanmamışsa null) ve `email_verified` alanları eklenir.

- **Kimlik:** OAuth2 erişim token'ı (`Authorization: Bearer <ACCESS_TOKEN>`)
- **Hız sınırı:** `oauth_read` (30 / 5s)

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/oauth2/@me" \
  -H "Authorization: Bearer $ACCESS_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.getOAuthMe({ accessToken: process.env.ACCESS_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.get_oauth_me(access_token=os.environ["ACCESS_TOKEN"])
```

Yanıt (200)

```
{
  "application": {
    "id": "1115400000000000000",
    "name": "Örnek Site"
  },
  "scopes": [
    "identify",
    "email"
  ],
  "expires_at": "2026-11-11T13:00:00Z",
  "user": {
    "id": "1114000000000000500",
    "username": "deniz",
    "display_name": "Deniz",
    "avatar": "a1b2c3d4",
    "email": "deniz@ornek.com",
    "email_verified": true
  }
}
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Giriş yapan kişinin sunucularını listele (`listOAuthGuilds`)

`GET https://noluyo.net/api/v1/oauth2/@me/guilds` (`GET /oauth2/@me/guilds`)

Kişinin üyesi olduğu sunucuları, her birinin sahibi olup olmadığı ve oradaki etkin izin bitleriyle listeler. `guilds` kapsamını ister.

- **Kimlik:** OAuth2 erişim token'ı (`Authorization: Bearer <ACCESS_TOKEN>`)
- **Hız sınırı:** `oauth_read` (30 / 5s)

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/oauth2/@me/guilds" \
  -H "Authorization: Bearer $ACCESS_TOKEN"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listOAuthGuilds({ accessToken: process.env.ACCESS_TOKEN });
```

```
# Python (noluyo_bot)
await bot.rest.list_oauth_guilds(access_token=os.environ["ACCESS_TOKEN"])
```

Yanıt (200)

```
[
  {
    "id": "111500000000000000",
    "name": "Topluluk",
    "icon": null,
    "owner": true,
    "permissions": "9007199254740991"
  }
]
```

Hata kodları

- **HTTP 401:** `unauthorized`
- **HTTP 403:** `oauth_scope_required`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Keşfet (Bot rehberi)

Doğrulanmış ve listelenmiş botların herkese açık rehberi.

Rehberdeki doğrulanmış botları listele (`listDirectoryBots`)

`GET https://noluyo.net/api/v1/discovery/bots` (`GET /discovery/bots`)

Bot rehberindeki doğrulanmış herkese açık botları sunucu sayısına göre sıralayarak listeler (sayfa başına `24`, `offset` en çok `960`). Bot platformu kapalıyken boş liste döner.

- **Kimlik:** Herkese açık (Authorization başlığı gerekmez)

Parametreler

Ad	Konum	Tür	Zorunlu	Açıklama
`category`	`query`	`string`	-	Bot kategorisine göre süz. (moderation, music, games, utility, fun, social, other)
`q`	`query`	`string`	-	Bot adına ya da açıklamasına göre ara. (<= 64)
`offset`	`query`	`integer`	-	Sonuç kaydırma değeri (0..960). (0–960 · = 0)

Örnek çağrılar

```
curl "https://noluyo.net/api/v1/discovery/bots?category=moderation"
```

```
// JavaScript (noluyo-bot)
await bot.rest.listDirectoryBots({ category: "moderation" });
```

```
# Python (noluyo_bot)
await bot.rest.list_directory_bots({"category": "moderation"})
```

Yanıt (200)


```
{
  "bots": [
    {
      "id": "21879167385600004",
      "bot_user_id": "21879171579904005",
      "name": "Kalkan Bot",
      "description": "Sunucu moderasyonu ve hızlı anket botu.",
      "listing_text": "Küfür süzgeci, zaman aşımı ve butonlu anketler.",
      "avatar": "a1b2c3d4",
      "category": "moderation",
      "guild_count": 142,
      "verified": true,
      "install_permissions": "68608",
      "support_url": "https://ornek.com/destek",
      "website_url": "https://ornek.com",
      "privacy_url": "https://ornek.com/gizlilik"
    }
  ],
  "total": 1,
  "has_more": false
}
```


Hata kodları

- **HTTP 400:** `validation_failed`, `invalid_body`, `intent_approval_required`, `endpoint_not_allowed`, `endpoint_unreachable`, `endpoint_bad_response`, `endpoint_no_signature_check`
- **HTTP 429:** `rate_limited`
- **HTTP 503:** `maintenance`

Gateway olayları, opcode'lar, intent'ler ve kapanış kodları

Son güncelleme: 2026-11-11 (11 Kasım 2026) · N'oluyo bot gateway'inde her çerçeve `op` alanı taşıyan bir JSON nesnesidir.

Opcode'lar

`op: 0` — DISPATCH (Bot alır)

Bir olay. `t` adını, `s` sıra numarasını, `d` içeriğini verir. Son `s` saklanır: resume için gerekir.

```
{
  "op": 0,
  "d": {
    "id": "21531879014400001"
  },
  "s": 42,
  "t": "GUILD_DELETE"
}
```

`op: 1` — HEARTBEAT (Bot gönderir)

Bağlantıyı canlı tutar; her `heartbeat_interval` süresinde gönderilir. Sunucu op 11 ile yanıtlar, `d` alanını okumaz. Heartbeat gelmezse bağlantı kapanış kodu olmadan kopar, oturum sürdürülebilir kalır.

```
{
  "op": 1,
  "d": 42
}
```

`op: 2` — IDENTIFY (Bot gönderir)

Oturum başlatır. `token` ve `intents` zorunludur. `shard` `[id, count]` biçimindedir, verilmezse `[0, 1]` olur. `presence` isteğe bağlıdır. Uygulamanın oturum başlatma haklarından birini harcar.

```
{
  "op": 2,
  "d": {
    "token": "<token>",
    "intents": 513,
    "shard": [
      0,
      1
    ],
    "presence": {
      "status": "online",
      "activities": []
    }
  }
}
```

`op: 3` — PRESENCE_UPDATE (Bot gönderir)

Botun durumunu değiştirir: `online`, `idle`, `dnd` ya da `invisible`. Bir oturum dakikada 5 kez gönderebilir; fazlasında `RATE_LIMITED` gelir ve güncelleme yok sayılır.

```
{
  "op": 3,
  "d": {
    "status": "idle",
    "activities": []
  }
}
```

`op: 4` — VOICE_STATE_UPDATE (Bot gönderir)

Bir sunucunun ses kanalına katılmayı (`channel_id` dolu) ya da o sunucuda sesten ayrılmayı (`channel_id` null) ister. Katılma `VOICE_SERVER_UPDATE` ya da `VOICE_JOIN_ERROR` ile yanıtlanır; ayrılma ikisini de göndermez. Intent gerektirmez, kanalda `ViewChannel`, `Connect` ve `Speak` izinleri gerekir. Bot sunucu başına 20 saniyede 5 kez gönderebilir.

```
{
  "op": 4,
  "d": {
    "guild_id": "21531879014400001",
    "channel_id": "21531879014400005",
    "self_mute": false
  }
}
```

`op: 6` — RESUME (Bot gönderir)

Kopan bağlantıdan sonra oturumu sürdürür, oturum başlatma hakkı harcamaz. `seq` sonrasında kaçırılan olaylar yeniden gönderilir, ardından `RESUMED` gelir. Oturum artık yoksa op 9 gelir.

```
{
  "op": 6,
  "d": {
    "token": "<token>",
    "session_id": "6f1c0e2ab7d94c3f8e5a1d0b9c7e4a26",
    "seq": 42
  }
}
```

`op: 7` — RECONNECT (Bot alır)

Sunucu kapanıyor. Yeniden bağlanıp resume yapılır.

```
{
  "op": 7,
  "d": null,
  "s": null,
  "t": null
}
```

`op: 8` — REQUEST_GUILD_MEMBERS (Bot gönderir)

Bir sunucunun üyelerini ister; yanıt `GUILD_MEMBERS_CHUNK` olarak gelir. `query` ve `user_ids` alanlarından tam olarak biri gönderilir. Boş `query` tam listedir ve `GUILD_MEMBERS` ister; `presences: true` `GUILD_PRESENCES` ister. Aynı anda tek istek olur.

```
{
  "op": 8,
  "d": {
    "guild_id": "21531879014400001",
    "query": "ay",
    "limit": 10,
    "presences": false,
    "nonce": "arama-1"
  }
}
```

`op: 9` — INVALID_SESSION (Bot alır)

Oturum sürdürülemez. `d` her zaman false'tur: op 2 ile yeni oturum açılır.

```
{
  "op": 9,
  "d": false,
  "s": null,
  "t": null
}
```

`op: 10` — HELLO (Bot alır)

Her bağlantının ilk çerçevesi. `heartbeat_interval` milisaniyedir.

```
{
  "op": 10,
  "d": {
    "heartbeat_interval": 41250
  },
  "s": null,
  "t": null
}
```

`op: 11` — HEARTBEAT_ACK (Bot alır)

Op 1'in yanıtı. Gelmezse bağlantı ölmüştür: yeniden bağlanıp resume yapılır.

```
{
  "op": 11,
  "d": null,
  "s": null,
  "t": null
}
```

`op: 30` — PHONE_CAMERA (Bot gönderir)

Yalnızca kişilerin uygulamaları içindir. Gönderen bot 4001 ile kapatılır.

```
{
  "op": 30,
  "d": {}
}
```

`op: 31` — ACTIVE (Bot gönderir)

Yalnızca kişilerin uygulamaları içindir. Gönderen bot 4001 ile kapatılır.

```
{
  "op": 31,
  "d": null
}
```

Kapanış kodları

Kod	Ad	Yeniden bağlan	Açıklama
`4000`	`UNKNOWN_ERROR`	`reconnect: true` (Evet)	Sunucuda bir şeyler ters gitti. Yeniden bağlanılır.
`4001`	`UNKNOWN_OPCODE`	`reconnect: false` (Hayır)	Bot, olmayan ya da botlara açık olmayan bir opcode gönderdi.
`4002`	`DECODE_ERROR`	`reconnect: false` (Hayır)	Bir çerçeve ya da içerik okunamadı veya değerleri geçersiz.
`4003`	`NOT_AUTHENTICATED`	`reconnect: false` (Hayır)	Bot, identify'dan önce op 1, op 2 ya da op 6 dışında bir şey gönderdi veya 10 saniye içinde identify etmedi.
`4004`	`AUTHENTICATION_FAILED`	`reconnect: false` (Hayır)	Token yanlış ya da sıfırlandı, veya bot ya da sahibi devre dışı bırakıldı ya da silindi. Önce yeni token gerekir.
`4005`	`ALREADY_AUTHENTICATED`	`reconnect: false` (Hayır)	Bot, oturumu olan bir bağlantıda op 2 ya da op 6 gönderdi.
`4008`	`RATE_LIMITED`	`reconnect: true` (Evet)	Çok fazla çerçeve (dakikada 120) ya da kısa sürede çok fazla oturum başlatma. Beklenir, sonra yeniden bağlanılır.
`4010`	`MAINTENANCE`	`reconnect: true` (Evet)	Platform bakımda. Daha sonra yeniden denenir.
`4011`	`TOO_MANY_SESSIONS`	`reconnect: false` (Hayır)	Daha yeni bir oturum bunun yerini aldı: aynı shard yeniden identify edildi ya da botun oturumu çok fazla. Bir botun iki kopyası birbirini böyle düşürür.
`4012`	`INVALID_SHARD`	`reconnect: false` (Hayır)	`shard` biçimi ya da aralığı yanlış, veya izin verilenden çok shard istiyor.
`4013`	`INVALID_INTENTS`	`reconnect: false` (Hayır)	`intents` yok, sayı değil ya da tanımsız bitler içeriyor.
`4014`	`DISALLOWED_INTENTS`	`reconnect: false` (Hayır)	Bot izni olmayan ayrıcalıklı bir intent istedi ya da `op 8` oturumda olmayan bir intent gerektiriyor.
`4015`	`SHARDING_REQUIRED`	`reconnect: false` (Hayır)	Bot bu shard sayısı için çok fazla sunucuda. Yalnızca daha çok shard ile başarılı olur.
`4016`	`SESSION_START_LIMIT`	`reconnect: true` (Evet)	Günlük oturum başlatma hakları bitti. Yeniden bağlanmak için `GET /gateway/bot` yanıtındaki `reset_after` süresi beklenir.

Intent'ler

Intent	Bit	Değer	Ayrıcalıklı	Açıklama
`GUILDS`	`1 << 0`	`1`	Hayır	Sunucular, rolleri, kanalları ve sabitlemeler.
`GUILD_MEMBERS`	`1 << 1`	`2`	Evet	Üyelerin katılması, değişmesi ve ayrılması; op 8 ile tam liste.
`GUILD_MODERATION`	`1 << 2`	`4`	Hayır	Yasaklar ve kaldırılmaları.
`GUILD_EXPRESSIONS`	`1 << 3`	`8`	Hayır	Emojiler, çıkartmalar ve ses paneli.
`GUILD_WEBHOOKS`	`1 << 5`	`32`	Hayır	Bir kanalın webhook'larındaki değişiklikler.
`GUILD_INVITES`	`1 << 6`	`64`	Hayır	Oluşturulan ve silinen davetler.
`GUILD_VOICE_STATES`	`1 << 7`	`128`	Hayır	Seste kimlerin olduğu ve orada çalınan sesler.
`GUILD_PRESENCES`	`1 << 8`	`256`	Evet	Üyelerin çevrimiçi durumu ve etkinlikleri; op 8 ile de.
`GUILD_MESSAGES`	`1 << 9`	`512`	Hayır	Sunucu kanallarındaki mesajlar.
`GUILD_MESSAGE_REACTIONS`	`1 << 10`	`1024`	Hayır	Sunucu kanallarındaki tepkiler.
`GUILD_MESSAGE_TYPING`	`1 << 11`	`2048`	Hayır	Sunucu kanallarında yazıyor bildirimi.
`DIRECT_MESSAGES`	`1 << 12`	`4096`	Hayır	Sunucu dışındaki mesajlar.

Intent	Bit	Değer	Ayrıcalıklı	Açıklama
`DIRECT_MESSAGE_REACTIONS`	`1 << 13`	`8192`	Hayır	Sunucu dışındaki tepkiler.
`DIRECT_MESSAGE_TYPING`	`1 << 14`	`16384`	Hayır	Sunucu dışında yazıyor bildirimi.
`MESSAGE_CONTENT`	`1 << 15`	`32768`	Evet	Mesajların içeriği. Kendi başına bir olay açmaz: bu intent olmadan mesajlar içeriği süzülmüş gelir.
`GUILD_SCHEDULED_EVENTS`	`1 << 16`	`65536`	Hayır	Sunucudaki etkinlikler.
`AUTO_MODERATION_CONFIGURATION`	`1 << 20`	`1048576`	Hayır	Otomatik moderasyon kurallarındaki değişiklikler.

Tüm olaylar

`READY` — Oturum başladı

Op 2'den sonraki ilk olay. Botun kim olduğunu söyler ve bu shard'ın sunucularını `unavailable` olarak listeler; her biri ardından `GUILD_CREATE` olarak gelir, başka olaylar ondan önce de gelebilir. `session_id` resume için saklanır. Resume'da yeniden gönderilmez.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`v`	`integer`	evet	Gateway sürümü, şimdilik 1.
`user`	`object`	evet	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`session_id`	`string`	evet	
`application`	`object`	evet	
`application.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`shard`	`integer[]`	evet	`[id, count]`.
`guilds`	`object[]`	evet	

```
{
  "v": 1,
  "user": {
    "id": "21879171579904005",
    "username": "selam_botu",
    "display_name": "Selam Botu",
    "avatar": null,
    "cosmetics": null,
    "bot": true
  },
  "session_id": "6f1c0e2ab7d94c3f8e5a1d0b9c7e4a26",
  "application": {
    "id": "21879167385600004"
  },
  "shard": [
    0,
    1
  ],
  "guilds": [
    {
      "id": "21531879014400001",
      "unavailable": true
    }
  ]
}
```

`RESUMED` — Oturum sürdürüldü

Op 6'yı doğrular. `seq` sonrasında kaçırılan olaylar bundan önce gelir. Botun onaylamadığı `GUILD_CREATE`'ler yeniden gönderilir; `GUILD_MEMBERS_CHUNK` ve `RATE_LIMITED` gönderilmez, yeniden istenir.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

```
{}
```

`USER_UPDATE` — Botun profili değişti

Değişen profilin ardından botun kendi herkese açık profili. Bot başkasınınkini duymaz. Yalnızca shard 0'a gider.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`username`	`string`	evet	
`display_name`	`string | null`	evet	
`avatar`	`string | null`	evet	Avatar hash'i.
`cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.

```
{
  "id": "21879171579904005",
  "username": "selam_botu",
  "display_name": "Merhaba Botu",
  "avatar": null,
  "cosmetics": null,
  "bot": true
}
```

`VOICE_SERVER_UPDATE` — Ses bağlantı token'ı hazır

Op 4 isteğini (veya bir yöneticinin kanal taşımasını) medya sunucusu adresi, bağlanmak için 10 dakika geçerli salt-yayın token'ı ve kanalın saniyedeki bit cinsinden `bitrate` değeriyle yanıtlar. Bot bir LiveKit istemcisiyle doğrudan `endpoint` adresine bağlanır ve yalnızca tek bir mikrofon ses izi yayınlayabilir; kimsenin sesine abone olamaz.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`endpoint`	`string`	evet	Her zaman doludur (`wss://` ya da `ws://`).
`token`	`string`	evet	Salt-yayın LiveKit JWT (`canSubscribe: false`). Gizli tutulur.
`bitrate`	`integer`	evet	Kanalın saniyedeki bit cinsinden hedef ses bit hızı.

```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531879014400005",
  "endpoint": "wss://noluyo.net",
  "token": "<livekit-jwt>",
  "bitrate": 64000
}
```

`VOICE_JOIN_ERROR` — Ses kanalına katılma reddedildi

Op 4 reddedildiğinde ya da medya sunucusuna bağlanırken bot ses sınırı dolduğunda `VOICE_SERVER_UPDATE` yerine gönderilir. Gateway oturumu açık kalır.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID | null`	evet	
`code`	`string`	evet	(unknown_guild, unknown_channel, invalid_channel_type, missing_permissions, voice_unavailable, channel_bot_limit, guild_bot_limit, app_voice_limit, bot_voice_full)
`message`	`string`	evet	

```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531879014400005",
  "code": "channel_bot_limit",
  "message": "This voice channel already has the maximum number of bots."
}
```

`RATE_LIMITED` — Bir op çok hızlı olduğu için reddedildi

Hangi op'un ne zaman yeniden gönderileceğini söyler. Op 3, op 4 ve op 8 için gelir. Resume'da yeniden gönderilmez.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`opcode`	`integer`	evet	
`retry_after`	`number`	evet	Beklenecek saniye.
`meta`	`object`	-	Op 4 ve op 8 için: reddedilen isteğin sunucusu (ve op 8'de nonce). Op 3'te yer almaz.
`meta.guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`meta.nonce`	`string`	-	

```
{
  "opcode": 8,
  "retry_after": 12.5,
  "meta": {
    "guild_id": "21531879014400001",
    "nonce": "arama-1"
  }
}
```

`GUILD_MEMBERS_CHUNK` — Op 8 yanıtının bir parçası

Bir parça en çok 1000 üye taşır. `chunk_index + 1`, `chunk_count` değerine eşit olduğunda yanıt biter. Botun içinde olmadığı ya da başka shard'daki bir sunucu için tek bir boş parça gelir.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`members`	`Member[]`	evet	
`chunk_index`	`integer`	evet	
`chunk_count`	`integer`	evet	
`not_found`	`ID[]`	-	İstenen `user_ids` içinde üye olmayanlar.
`presences`	`Presence[]`	-	`presences: true` ile: parçadaki üyelerden çevrimiçi olanlar.
`nonce`	`string`	-	İsteğin `nonce` değeri.

```
{
  "guild_id": "21531879014400001",
  "members": [
    {
      "user": {
        "id": "14631410073600003",
        "username": "ayse",
        "display_name": "Ayşe",
        "avatar": null,
        "cosmetics": null
      },
      "nick": "Ayşe K.",
      "roles": [
        "22649241600000010"
      ],
      "joined_at": "2026-03-05T18:20:00Z",
      "mute": false,
      "deaf": false,
      "communication_disabled_until": null
    }
  ],
  "chunk_index": 0,
  "chunk_count": 1,
  "nonce": "arama-1"
}
```

`INTERACTION_CREATE` — Biri butona bastı, menüden seçti, komut çalıştırdı ya da form gönderdi

Yalnızca basılan mesajın ya da komutun botuna, sunucusunun shard'ına gider; kişinin botla açtığı DM'den gelirse shard 0'a, member yerine user ile gider. Intent istemez. Basmada (type 3) sunucu `data.custom_id` ve `data.values` değerlerini mesajın şimdiki haline göre denetler, ama kişinin o işe yetkili olup olmadığını

denetlemez: `member.permissions`'a bak. Komut çalıştırmada (type 2) seçenekleri tanımına göre ve komutu kimin kullanabileceğini denetler. Otomatik tamamlamada (type 4) yazılanı tanımına göre denetler; 3 saniye içinde callback type 8 ile yanıtlarsın, kişi yazdıkça 300 ms'de bir gelebilir. Gönderilen formda (type 5) her alanı açtığın forma göre, kişinin kanala, bota ve komuta erişimini yeniden denetler; her gönderim kendi token'ı olan yeni bir etkileşimdir. `context` sunucuyu (0) botunla açılmış DM'den (1) ayırır; DM'de yalnız `contexts` alanında 1 olan ve `default_member_permissions` koymayan global komutların çalışır.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`application_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`type`	`integer`	evet	2 komut çalıştırmadır (eğik çizgi, kullanıcı menüsü ya da mesaj menüsü), 3 bir bileşene basmadır, 4 otomatik tamamlamadır: kişi seçeneklerinden birine yazıyor. 5 gönderilen formdur. (2, 3, 4, 5)
`token`	`string`	evet	Gizlidir, 15 dakika geçerlidir. Log'a yazma.
`version`	`integer`	evet	(1)
`context`	`integer`	evet	Nerede olduğu: 0 sunucu kanalı, 1 kişinin botunla açtığı DM. Her zaman vardır. DM'de member yerine user gelir, guild_id yoktur, app_permissions DM'de herkesin sahip olduğu izinlerdir. (0, 1)
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`member`	`object`	-	
`member.user`	`object`	evet	
`member.user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`member.user.username`	`string`	evet	

Alan	Tür	Zorunlu	Açıklama
`member.user.display_name`	`string | null`	evet	
`member.user.avatar`	`string | null`	evet	Avatar hash'i.
`member.user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`member.user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`member.nick`	`string | null`	evet	
`member.roles`	`ID[]`	evet	@everyone hariç rol ID'leri.
`member.joined_at`	`datetime`	evet	
`member.mute`	`boolean`	evet	
`member.deaf`	`boolean`	evet	
`member.communication_disabled_until`	`string | null`	evet	Susturma sürerken doludur.
`member.permissions`	`permissions`	-	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.
`user`	`object`	-	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`app_permissions`	`permissions`	evet	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.
`locale`	`string`	evet	Basan ya da çalıştıran kişinin dili.

Alan	Tür	Zorunlu	Açıklama
`message`	`Message | PressedEphemeral`	-	Basılan mesaj. Botun kendi mesajı, intent'lerinden bağımsız eksiksiz gelir; ephemeral mesajda yalnızca id, channel_id, flags, author ve components. Yalnızca basmada; gönderilen form taşımaz.
`data`	`object | object | object`	evet	

```
{
  "id": "1118042062848000 99",
  "application_id": "14631410073600008",
  "type": 3,
  "token": "eyJpZCI6IjExMTgwNDIwNjI4NDgwMDA5OSJ9.bWFj",
  "version": 1,
  "context": 0,
  "guild_id": "21531879014400001",
  "channel_id": "21531887403008007",
  "member": {
    "user": {
      "id": "14631410073600003",
      "username": "ayse",
      "display_name": "Ayşe",
      "avatar": null,
      "cosmetics": null
    },
    "nick": null,
    "roles": [],
    "joined_at": "2026-11-01T10:00:00Z",
    "mute": false,
    "deaf": false,
    "communication_disabled_until": null,
    "permissions": "68672"
  },
  "app_permissions": "268504128",
  "locale": "tr",
  "message": {
    "id": "111804206284800011",
    "type": 0,
    "channel_id": "21531887403008007",
    "guild_id": "21531879014400001",
    "author": {
      "id": "14631410073600009",
      "username": "rolbotu",
      "display_name": null,
      "avatar": null,
      "cosmetics": null,
      "bot": true
    },
    "content": "Rolü almak için bas.",
    "timestamp": "2026-11-05T12:30:00Z",
    "edited_timestamp": null,
    "attachments": [],
    "mentions": [],
    "mention_roles": [],
    "mention_everyone": false,
    "pinned": false,
```

```
    "components": [
      {
        "type": 1,
        "components": [
          {
            "type": 2,
            "style": 1,
            "label": "Oyuncu",
            "custom_id": "rol:21531879014400009"
          }
        ]
      }
    ],
    "application_id": "14631410073600008"
  },
  "data": {
    "custom_id": "rol:21531879014400009",
    "component_type": 2
  }
}
```

`APPLICATION_COMMAND_PERMISSIONS_UPDATE` — Bir komutu kimin kullanabileceği değişti

Bir sunucu yöneticisi bu botun bir komutunun bir sunucudaki kurallarını ya da tüm komutlarının kurallarını değiştirdi (o zaman id uygulamanın ID'sidir). permissions yeni kümenin tamamıdır; boş liste kuralların kaldırıldığını söyler. Yalnızca bota gider ve intent istemez. Kuralları sunucu zaten uygular; botun uygulaması gerekmez.

- **Gereken intent'ler:** Gerekmez (her oturuma gider)

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`application_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`permissions`	`object[]`	evet	

```
{
  "id": "111560000000000000",
  "application_id": "111540000000000000",
  "guild_id": "111500000000000000",
  "permissions": [
    {
      "type": 1,
      "id": "111520000000000000",
      "allow": true
    },
    {
      "type": 3,
      "id": "0",
      "allow": false
    },
    {
      "type": 3,
      "id": "111510000000000000",
      "allow": true
    }
  ]
}
```

`GUILD_CREATE` — Bir sunucu kullanılabilir oldu

`READY` içindeki her sunucu için ve bot bir sunucuya katıldığında gelir. Botun görebildiği kanalları ve üye olarak yalnızca botu listeler: diğerleri op 8 ile istenir. `GUILD_VOICE_STATES` yoksa `voice_states` boştur.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`name`	`string`	evet	
`icon`	`string | null`	evet	
`banner`	`string | null`	evet	
`description`	`string | null`	evet	
`owner_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`created_at`	`datetime`	evet	
`features`	`string[]`	evet	
`roles`	`Role[]`	evet	
`discoverable`	`boolean`	evet	
`discovery_category`	`string | null`	evet	
`join_channel_id`	`ID | null`	evet	
`leave_channel_id`	`ID | null`	evet	
`channels`	`Channel[]`	evet	Botun görebildiği kanallar.
`members`	`Member[]`	evet	Yalnızca botun kendisi. Diğerleri op 8 ile alınır.
`member_count`	`integer`	evet	
`large`	`boolean`	evet	Sunucuda 250'den çok üye var.
`voice_states`	`VoiceState[]`	evet	Botun görebildiği ses kanallarında kimlerin olduğu.
`stickers`	`Sticker[]`	evet	
`emojis`	`Emoji[]`	evet	
`tag`	`string | null`	-	Üyelerin takabildiği kısa yazı; sunucuda yoksa null.
`tag_icon`	`string | null`	-	Etiketin yanında görünen simge.

Alan	Tür	Zorunlu	Açıklama
`tag_style`	`object | null`	-	Etiketin nasıl çizildiği.

```
{
  "id": "21531879014400001",
  "name": "Oyun Gecesi",
  "icon": null,
  "banner": null,
  "description": null,
  "owner_id": "11369919283200002",
  "created_at": "2026-03-01T10:00:00Z",
  "features": [],
  "roles": [
    {
      "id": "21531879014400001",
      "name": "@everyone",
      "color": 0,
      "hoist": false,
      "mentionable": false,
      "position": 0,
      "permissions": "68672",
      "auto_assign": false
    },
    {
      "id": "22286853734400009",
      "name": "Selam Botu",
      "color": 0,
      "hoist": false,
      "mentionable": false,
      "position": 1,
      "permissions": "68672",
      "auto_assign": false,
      "application_id": "21879167385600004"
    }
  ],
  "discoverable": false,
  "discovery_category": null,
  "join_channel_id": null,
  "leave_channel_id": null,
  "channels": [
    {
      "id": "21531883208704006",
      "type": 4,
      "guild_id": "21531879014400001",
      "name": "Metin kanalları",
      "topic": null,
      "position": 0,
      "parent_id": null,
      "nsfw": false,
      "last_message_id": null
    },
```

```
    {
      "id": "21531887403008007",
      "type": 0,
      "guild_id": "21531879014400001",
      "name": "genel",
      "topic": null,
      "position": 1,
      "parent_id": "21531883208704006",
      "nsfw": false,
      "last_message_id": "111804206284800011"
    },
    {
      "id": "21531891597312008",
      "type": 2,
      "guild_id": "21531879014400001",
      "name": "Sohbet",
      "topic": null,
      "position": 3,
      "parent_id": null,
      "nsfw": false,
      "bitrate": 64000,
      "user_limit": 0,
      "last_message_id": null
    }
  ],
  "members": [
    {
      "user": {
        "id": "21879171579904005",
        "username": "selam_botu",
        "display_name": "Selam Botu",
        "avatar": null,
        "cosmetics": null,
        "bot": true
      },
      "nick": null,
      "roles": [
        "22286853734400009"
      ],
      "joined_at": "2026-03-03T12:00:00Z",
      "mute": false,
      "deaf": false,
      "communication_disabled_until": null
    }
  ],
  "member_count": 3,
  "large": false,
  "voice_states": [],
  "stickers": [],
```

```
  "emojis": []
}
```

`GUILD_UPDATE` — Bir sunucu değişti

Değişiklikten sonra sunucu ve rolleri. Kanal ve üye içermez.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`name`	`string`	evet	
`icon`	`string | null`	evet	
`banner`	`string | null`	evet	
`description`	`string | null`	evet	
`owner_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`created_at`	`datetime`	evet	
`features`	`string[]`	evet	
`roles`	`Role[]`	evet	
`discoverable`	`boolean`	evet	
`discovery_category`	`string | null`	evet	
`join_channel_id`	`ID | null`	evet	
`leave_channel_id`	`ID | null`	evet	
`tag`	`string | null`	-	Üyelerin takabildiği kısa yazı; sunucuda yoksa null.
`tag_icon`	`string | null`	-	Etiketin yanında görünen simge.
`tag_style`	`object | null`	-	Etiketin nasıl çizildiği.

```
{
  "id": "21531879014400001",
  "name": "Turnuva Gecesi",
  "icon": null,
  "banner": null,
  "description": null,
  "owner_id": "11369919283200002",
  "created_at": "2026-03-01T10:00:00Z",
  "features": [],
  "roles": [
    {
      "id": "21531879014400001",
      "name": "@everyone",
      "color": 0,
      "hoist": false,
      "mentionable": false,
      "position": 0,
      "permissions": "68672",
      "auto_assign": false
    },
    {
      "id": "22286853734400009",
      "name": "Selam Botu",
      "color": 0,
      "hoist": false,
      "mentionable": false,
      "position": 1,
      "permissions": "68672",
      "auto_assign": false,
      "application_id": "21879167385600004"
    }
  ],
  "discoverable": false,
  "discovery_category": null,
  "join_channel_id": null,
  "leave_channel_id": null
}
```


`GUILD_DELETE` — Sunucu bot için yok oldu

Sunucu silindi ya da bot sunucudan çıkarıldı.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "id": "21531879014400001"
}
```

`GUILD_ROLE_CREATE` — Bir rol oluşturuldu

Sunucuya bot eklendiğinde, botun kendi rolü için de gelir.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`role`	`object`	evet	
`role.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`role.name`	`string`	evet	
`role.color`	`integer`	evet	Tam sayı olarak RGB, 0 ile 16777215 arası.
`role.hoist`	`boolean`	evet	
`role.mentionable`	`boolean`	evet	
`role.position`	`integer`	evet	Büyük olan daha üsttedir. @everyone en alttadır.
`role.permissions`	`permissions`	evet	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.
`role.auto_assign`	`boolean`	evet	
`role.application_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "guild_id": "21531879014400001",
  "role": {
    "id": "22649241600000010",
    "name": "Moderatör",
    "color": 3447003,
    "hoist": true,
    "mentionable": true,
    "position": 2,
    "permissions": "68672",
    "auto_assign": false
  }
}
```

`GUILD_ROLE_UPDATE` — Bir rol değişti

Değişiklikten sonraki rol. Bir rolü taşımak başkalarının sırasını da değiştirebilir; her birinin kendi olayı gelir.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`role`	`object`	evet	
`role.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`role.name`	`string`	evet	
`role.color`	`integer`	evet	Tam sayı olarak RGB, 0 ile 16777215 arası.
`role.hoist`	`boolean`	evet	
`role.mentionable`	`boolean`	evet	
`role.position`	`integer`	evet	Büyük olan daha üsttedir. @everyone en alttadır.
`role.permissions`	`permissions`	evet	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.
`role.auto_assign`	`boolean`	evet	
`role.application_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "guild_id": "21531879014400001",
  "role": {
    "id": "22649241600000010",
    "name": "Yönetici",
    "color": 3447003,
    "hoist": true,
    "mentionable": true,
    "position": 2,
    "permissions": "68672",
    "auto_assign": false
  }
}
```


`GUILD_ROLE_DELETE` — Bir rol silindi

Yalnızca ID'ler: rol artık yok.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`role_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.


```
{
  "guild_id": "21531879014400001",
  "role_id": "22649241600000010"
}
```


`CHANNEL_CREATE` — Bir kanal oluşturuldu

Kanalı görebilen botlara gelir.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`type`	`integer`	evet	0 metin, 2 ses, 4 kategori. (0, 2, 4)
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`name`	`string`	-	
`topic`	`string | null`	evet	
`position`	`integer`	evet	
`parent_id`	`ID | null`	evet	
`nsfw`	`boolean`	evet	
`bitrate`	`integer`	-	Ses kanalları.
`user_limit`	`integer`	-	Ses kanalları; 0 sınırsızdır.
`rate_limit_per_user`	`integer`	-	Saniye olarak yavaş mod; kapalıyken yer almaz.
`last_message_id`	`ID | null`	evet	
`permission_overwrites`	`Overwrite[]`	-	
`voice_status`	`string`	-	
`voice_since`	`datetime`	-	


```
{
  "id": "21531887403008007",
  "type": 0,
  "guild_id": "21531879014400001",
  "name": "genel",
  "topic": null,
  "position": 1,
  "parent_id": "21531883208704006",
  "nsfw": false,
  "last_message_id": "111804206284800011"
}
```

`CHANNEL_UPDATE` — Bir kanal değişti

Değişiklikten sonraki kanal. Değişiklik kanalı bottan gizlerse yerine `CHANNEL_DELETE` gelir.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`type`	`integer`	evet	0 metin, 2 ses, 4 kategori. (0, 2, 4)
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`name`	`string`	-	
`topic`	`string | null`	evet	
`position`	`integer`	evet	
`parent_id`	`ID | null`	evet	
`nsfw`	`boolean`	evet	
`bitrate`	`integer`	-	Ses kanalları.
`user_limit`	`integer`	-	Ses kanalları; 0 sınırsızdır.
`rate_limit_per_user`	`integer`	-	Saniye olarak yavaş mod; kapalıyken yer almaz.
`last_message_id`	`ID | null`	evet	
`permission_overwrites`	`Overwrite[]`	-	
`voice_status`	`string`	-	
`voice_since`	`datetime`	-	

```
{
  "id": "21531887403008007",
  "type": 0,
  "guild_id": "21531879014400001",
  "name": "genel",
  "topic": "Genel sohbet",
  "position": 1,
  "parent_id": "21531883208704006",
  "nsfw": false,
  "last_message_id": "111804206284800011"
}
```


`CHANNEL_DELETE` — Kanal bot için yok oldu

Kanal silindi ya da bot artık kanalı göremiyor.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`type`	`integer`	evet	0 metin, 2 ses, 4 kategori. (0, 2, 4)
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`name`	`string`	-	
`topic`	`string | null`	evet	
`position`	`integer`	evet	
`parent_id`	`ID | null`	evet	
`nsfw`	`boolean`	evet	
`bitrate`	`integer`	-	Ses kanalları.
`user_limit`	`integer`	-	Ses kanalları; 0 sınırsızdır.
`rate_limit_per_user`	`integer`	-	Saniye olarak yavaş mod; kapalıyken yer almaz.
`last_message_id`	`ID | null`	evet	
`permission_overwrites`	`Overwrite[]`	-	
`voice_status`	`string`	-	
`voice_since`	`datetime`	-	


```
{
  "id": "21531887403008007",
  "type": 0,
  "guild_id": "21531879014400001",
  "name": "genel",
  "topic": null,
  "position": 1,
  "parent_id": "21531883208704006",
  "nsfw": false,
  "last_message_id": "111804206284800011"
}
```

`CHANNEL_PINS_UPDATE` — Bir kanalın sabitlenenleri değişti

Sabitlenen mesaj kalmayınca `last_pin_timestamp` null olur.

- **Gereken intent'ler:** `GUILDS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`last_pin_timestamp`	`string | null`	evet	

```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531887403008007",
  "last_pin_timestamp": "2026-11-05T12:30:00Z"
}
```

`GUILD_MEMBER_ADD` — Biri sunucuya katıldı

Üye; alanlarının yanında sunucunun ID'si.

- **Gereken intent'ler:** `GUILD_MEMBERS`

> **Not:** `GUILD_MEMBERS` ayrıcalıklıdır: sahibi portaldan açar, 100 sunucudan itibaren onay gerekir. Bot kendisiyle ilgili olayları bu intent olmadan da alır.

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user`	`object`	evet	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`nick`	`string | null`	evet	
`roles`	`ID[]`	evet	@everyone hariç rol ID'leri.
`joined_at`	`datetime`	evet	
`mute`	`boolean`	evet	
`deaf`	`boolean`	evet	
`communication_disabled_until`	`string | null`	evet	Susturma sürerken doludur.

```
{
  "guild_id": "21531879014400001",
  "user": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  },
  "nick": "Ayşe K.",
  "roles": [],
  "joined_at": "2026-03-05T18:20:00Z",
  "mute": false,
  "deaf": false,
  "communication_disabled_until": null
}
```


`GUILD_MEMBER_UPDATE` — Bir üye değişti

Değişiklikten sonraki üye: takma ad, roller, susturma ya da zaman aşımı.

- **Gereken intent'ler:** `GUILD_MEMBERS`

> **Not:** `GUILD_MEMBERS` ayrıcalıklıdır: sahibi portaldan açar, 100 sunucudan itibaren onay gerekir. Bot kendisiyle ilgili olayları bu intent olmadan da alır.

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user`	`object`	evet	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`nick`	`string | null`	evet	
`roles`	`ID[]`	evet	@everyone hariç rol ID'leri.
`joined_at`	`datetime`	evet	
`mute`	`boolean`	evet	
`deaf`	`boolean`	evet	
`communication_disabled_until`	`string | null`	evet	Susturma sürerken doludur.

```
{
  "guild_id": "21531879014400001",
  "user": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  },
  "nick": "Ayşe K.",
  "roles": [
    "22649241600000010"
  ],
  "joined_at": "2026-03-05T18:20:00Z",
  "mute": false,
  "deaf": false,
  "communication_disabled_until": null
}
```


`GUILD_MEMBER_REMOVE` — Biri sunucudan ayrıldı

Kendisi ayrıldı, ya da atıldı veya yasaklandı.

- **Gereken intent'ler:** `GUILD_MEMBERS`

> **Not:** `GUILD_MEMBERS` ayrıcalıklıdır: sahibi portaldan açar, 100 sunucudan itibaren onay gerekir. Bot kendisiyle ilgili olayları bu intent olmadan da alır.

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user`	`object`	evet	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.


```
{
  "guild_id": "21531879014400001",
  "user": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  }
}
```


`GUILD_BAN_ADD` — Biri yasaklandı

Yalnızca botun sunucuda `BanMembers` izni varsa gelir.

- **Gereken intent'ler:** `GUILD_MODERATION`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user`	`object`	evet	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.


```
{
  "guild_id": "21531879014400001",
  "user": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  }
}
```


`GUILD_BAN_REMOVE` — Bir yasak kaldırıldı

Yalnızca botun sunucuda `BanMembers` izni varsa gelir.

- **Gereken intent'ler:** `GUILD_MODERATION`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user`	`object`	evet	
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user.username`	`string`	evet	
`user.display_name`	`string | null`	evet	
`user.avatar`	`string | null`	evet	Avatar hash'i.
`user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.


```
{
  "guild_id": "21531879014400001",
  "user": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  }
}
```


`GUILD_EMOJIS_UPDATE` — Sunucunun emojileri değişti

Yalnızca değişeni değil, listenin tamamını taşır.

- **Gereken intent'ler:** `GUILD_EXPRESSIONS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`emojis`	`Emoji[]`	evet	

```
{
  "guild_id": "21531879014400001",
  "emojis": [
    {
      "id": "32735703859200014",
      "guild_id": "21531879014400001",
      "name": "selam",
      "animated": false,
      "creator_id": "11369919283200002"
    }
  ]
}
```


`GUILD_STICKERS_UPDATE` — Sunucunun çıkartmaları değişti

Yalnızca değişeni değil, listenin tamamını taşır.

- **Gereken intent'ler:** `GUILD_EXPRESSIONS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`stickers`	`Sticker[]`	evet	


```
{
  "guild_id": "21531879014400001",
  "stickers": [
    {
      "id": "33098091724800015",
      "guild_id": "21531879014400001",
      "name": "kedi",
      "tags": "kedi",
      "description": null,
      "hash": "9f3c1a7e",
      "width": 320,
      "height": 320,
      "creator_id": "11369919283200002"
    }
  ]
}
```


`GUILD_SOUNDBOARD_SOUNDS_UPDATE` — Sunucunun ses paneli değişti

Yalnızca değişeni değil, listenin tamamını taşır.

- **Gereken intent'ler:** `GUILD_EXPRESSIONS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`soundboard_sounds`	`Sound[]`	evet	


```
{
  "guild_id": "21531879014400001",
  "soundboard_sounds": [
    {
      "id": "33460479590400016",
      "guild_id": "21531879014400001",
      "name": "alkış",
      "emoji": "",
      "volume": 1,
      "creator_id": "11369919283200002"
    }
  ]
}
```


`WEBHOOKS_UPDATE` — Bir kanalın webhook'ları değişti

Yalnızca nerede olduğunu söyler. Kanalı görebilen botlara gelir.

- **Gereken intent'ler:** `GUILD_WEBHOOKS`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.


```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531887403008007"
}
```


`INVITE_CREATE` — Bir davet oluşturuldu

Yalnızca botun sunucuda `ManageGuild` izni varsa gelir.

- **Gereken intent'ler:** `GUILD_INVITES`

Alan	Tür	Zorunlu	Açıklama
`code`	`string`	evet	
`guild`	`object`	evet	
`guild.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild.name`	`string`	evet	
`guild.icon`	`string | null`	evet	
`guild.banner`	`string | null`	evet	
`guild.description`	`string | null`	evet	
`guild.features`	`string[]`	evet	
`channel`	`object`	evet	
`channel.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel.name`	`string`	evet	
`channel.type`	`integer`	evet	
`inviter_id`	`ID | null`	evet	
`max_age`	`integer`	evet	Saniye; 0 süresizdir.
`max_uses`	`integer`	evet	0 sınırsızdır.
`uses`	`integer`	evet	
`created_at`	`datetime`	evet	
`expires_at`	`string | null`	evet	

```
{
  "code": "aB3xY9kQ",
  "guild": {
    "id": "21531879014400001",
    "name": "Oyun Gecesi",
    "icon": null,
    "banner": null,
    "description": null,
    "features": []
  },
  "channel": {
    "id": "21531887403008007",
    "name": "genel",
    "type": 0
  },
  "inviter_id": "11369919283200002",
  "max_age": 86400,
  "max_uses": 0,
  "uses": 0,
  "created_at": "2026-11-05T12:00:00Z",
  "expires_at": "2026-11-06T12:00:00Z"
}
```


`INVITE_DELETE` — Bir davet silindi

Yalnızca botun sunucuda `ManageGuild` izni varsa gelir.

- **Gereken intent'ler:** `GUILD_INVITES`

Alan	Tür	Zorunlu	Açıklama
`code`	`string`	evet	
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.


```
{
  "code": "aB3xY9kQ",
  "guild_id": "21531879014400001",
  "channel_id": "21531887403008007"
}
```


`VOICE_STATE_UPDATE` — Birinin ses durumu değişti

Katıldı, taşındı, sesini kapattı ya da ayrıldı; ayrılınca `channel_id` null olur. Kanalı görebilen botlara gelir.

- **Gereken intent'ler:** `GUILD_VOICE_STATES`

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID | null`	evet	Kullanıcı kanaldan ayrıldığında null olur.
`user_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`member`	`object`	evet	
`member.user`	`object`	evet	
`member.user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`member.user.username`	`string`	evet	
`member.user.display_name`	`string | null`	evet	
`member.user.avatar`	`string | null`	evet	Avatar hash'i.
`member.user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`member.user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`member.nick`	`string | null`	evet	
`member.roles`	`ID[]`	evet	@everyone hariç rol ID'leri.
`member.joined_at`	`datetime`	evet	
`member.mute`	`boolean`	evet	
`member.deaf`	`boolean`	evet	
`member.communication_disabled_until`	`string | null`	evet	Susturma sürerken doludur.
`member.permissions`	`permissions`	-	Ondalık metin olarak izin bit alanı. Metin yerine sayı veya tanımlı olmayan bitler invalid_body verir.

Alan	Tür	Zorunlu	Açıklama
`session_id`	`string`	evet	
`mute`	`boolean`	evet	
`deaf`	`boolean`	evet	
`self_mute`	`boolean`	evet	
`self_deaf`	`boolean`	evet	
`self_stream`	`boolean`	evet	
`self_video`	`boolean`	evet	


```
{
  "guild_id": "21531879014400001",
  "channel_id": "21531891597312008",
  "user_id": "14631410073600003",
  "member": {
    "user": {
      "id": "14631410073600003",
      "username": "ayse",
      "display_name": "Ayşe",
      "avatar": null,
      "cosmetics": null
    },
    "nick": "Ayşe K.",
    "roles": [
      "22649241600000010"
    ],
    "joined_at": "2026-03-05T18:20:00Z",
    "mute": false,
    "deaf": false,
    "communication_disabled_until": null
  },
  "session_id": "b41d7e90",
  "mute": false,
  "deaf": false,
  "self_mute": false,
  "self_deaf": false,
  "self_stream": false,
  "self_video": false
}
```


`VOICE_CHANNEL_EFFECT_SEND` — Seste bir ses çalındı

Biri bir ses kanalında ses panelinden bir ses çaldı.

- **Gereken intent'ler:** `GUILD_VOICE_STATES`

Alan	Tür	Zorunlu	Açıklama
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`sound_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`sound_volume`	`number`	evet	
`emoji`	`string | null`	evet	


```
{
  "channel_id": "21531891597312008",
  "guild_id": "21531879014400001",
  "user_id": "14631410073600003",
  "sound_id": "33460483784704017",
  "sound_volume": 1,
  "emoji": "□"
}
```


`VOICE_CHANNEL_EFFECT_STOP` — Çalınan ses durduruldu

Biri çaldığı sesi durdurdu.

- **Gereken intent'ler:** `GUILD_VOICE_STATES`

Alan	Tür	Zorunlu	Açıklama
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "channel_id": "21531891597312008",
  "guild_id": "21531879014400001",
  "user_id": "14631410073600003"
}
```

`PRESENCE_UPDATE` — Bir üyenin durumu değişti

Durumu ya da etkinlikleri değişti. Çok sayıda bağlı bot ve kişi olan sunucuda bunlar yerine `PRESENCE_UPDATE_BATCH` olarak toplu gelir.

- **Gereken intent'ler:** `GUILD_PRESENCES`

> **Not:** `GUILD_PRESENCES` ayrıcalıklıdır: sahibi portaldan açar, 100 sunucudan itibaren onay gerekir.

Alan	Tür	Zorunlu	Açıklama
`user`	`object`	evet	Yalnızca kullanıcının ID'si.
`user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`status`	`string`	evet	Çevrimdışı görünen kişi offline olarak gelir. (online, idle, dnd, offline)
`activities`	`Activity[]`	evet	Çevrimdışıyken boştur.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "user": {
    "id": "14631410073600003"
  },
  "status": "online",
  "activities": [
    {
      "type": 0,
      "name": "Minecraft",
      "timestamps": {
        "start": 1793878200000
      }
    }
  ],
  "guild_id": "21531879014400001"
}
```

`PRESENCE_UPDATE_BATCH` — Bir sunucunun birkaç durumu

Yoğun bir sunucunun `PRESENCE_UPDATE`'leri, toplanıp birlikte gönderilir. Her biri tek başına gönderileceği haliyledir.

- **Gereken intent'ler:** `GUILD_PRESENCES`

> **Not:** `GUILD_PRESENCES` ayrıcalıklıdır: sahibi portaldan açar, 100 sunucudan itibaren onay gerekir.

Alan	Tür	Zorunlu	Açıklama
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`presences`	`GuildPresence[]`	evet	

```
{
  "guild_id": "21531879014400001",
  "presences": [
    {
      "user": {
        "id": "14631410073600003"
      },
      "status": "online",
      "activities": [
        {
          "type": 0,
          "name": "Minecraft",
          "timestamps": {
            "start": 1793878200000
          }
        }
      ],
      "guild_id": "21531879014400001"
    }
  ]
}
```

`MESSAGE_CREATE` — Bir mesaj gönderildi

Botun görebildiği kanallardaki mesajlar için, kendininkiler için de gelir. Sunucu dışında `guild_id` yer almaz. Ephemeral mesaj buraya hiç gelmez: yalnızca basan kişiye gider. Bir kişinin botla açtığı DM'deki mesaj DIRECT_MESSAGES ister, guild_id taşımaz ve shard 0'a gider.

- **Gereken intent'ler:** `GUILD_MESSAGES`, `DIRECT_MESSAGES`

Not: Ayrıcalıklı `MESSAGE_CONTENT` intent'i olmadan `content` boş, `attachments` boş liste gelir; `sticker_items`, `poll`, `forward`, `hidden_embeds` ve `data` yer almaz. Mesajı bot yazdıysa ya da mesaj botu adıyla etiketliyorsa hepsi gelir. Botla DM'de bu kural uygulanmaz: kişinin orada bota yazdığı her şey tam gelir.

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`type`	`integer`	evet	0 normal, 19 yanıt, 6 sabitleme bildirimi, 18 başlık açılışı, 24 otomatik moderasyon uyarısı, 1 ile 5 arası grup ve arama bildirimleri. Yenileri eklenebilir.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`author`	`object`	evet	
`author.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`author.username`	`string`	evet	
`author.display_name`	`string | null`	evet	
`author.avatar`	`string | null`	evet	Avatar hash'i.
`author.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`author.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`content`	`string`	evet	
`timestamp`	`datetime`	evet	
`edited_timestamp`	`string | null`	evet	
`attachments`	`Attachment[]`	evet	
`mentions`	`User[]`	evet	
`mention_roles`	`ID[]`	evet	
`mention_everyone`	`boolean`	evet	

Alan	Tür	Zorunlu	Açıklama
`message_reference`	`object`	-	
`message_reference.message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_reference.channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`pinned`	`boolean`	evet	
`data`	`object`	-	Sistem mesajının gösterilmesi için gerekenler.
`forward`	`object`	-	
`forward.channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.author`	`object`	evet	
`forward.author.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.author.username`	`string`	evet	
`forward.author.display_name`	`string | null`	evet	
`forward.author.avatar`	`string | null`	evet	Avatar hash'i.
`forward.author.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`forward.author.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`forward.content`	`string`	evet	
`forward.timestamp`	`datetime`	evet	

Alan	Tür	Zorunlu	Açıklama
`forward.mentions`	`User[]`	evet	
`forward.sticker_items`	`StickerItem[]`	-	
`forward.webhook_name`	`string`	-	
`poll`	`object`	-	
`poll.question`	`string`	evet	
`poll.answers`	`object[]`	evet	
`poll.allow_multiselect`	`boolean`	evet	
`poll.expiry`	`datetime`	evet	
`poll.results`	`object`	-	
`poll.results.finalized`	`boolean`	evet	
`poll.results.voters`	`integer`	evet	
`poll.results.answers`	`object[]`	evet	
`sticker_items`	`StickerItem[]`	-	
`nonce`	`string`	-	Gönderenin mesaja verdiği kendi kimliği; oluşturulurken aynen döner.
`webhook_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`activity`	`object`	-	Başkalarını yazarın etkinliğine katılmaya çağırır.
`activity.type`	`integer`	evet	
`activity.party_id`	`string`	evet	
`hidden_embeds`	`string[]`	-	Yazarın kaldırdığı bağlantı önizlemeleri ve davet kartları: yazıldığı gibi bir bağlantı ya da küçük harfle bir davet kodu.
`embeds`	`Embed[]`	-	Botun eklediği kartlar.
`components`	`MessageComponent[]`	-	Botun eklediği butonları ve menüleri taşıyan satırlar.

Alan	Tür	Zorunlu	Açıklama
`flags`	`integer`	-	Bit bayrakları. 0 ise yoktur. 64: ephemeral, yalnızca basan kişi görür. 128: henüz hazırlanan yanıt.
`application_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`interaction`	`object`	-	Bir etkileşime verilen yanıtta: hangisi ve kimin bastığı ya da çalıştırdığı.
`interaction.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`interaction.type`	`integer`	evet	2 komut çalıştırma, 3 bileşene basmadır.
`interaction.name`	`string`	-	Çalıştırılan komut, "X, /ad kullandı" için. Basmada yer almaz.
`interaction.user`	`object`	evet	
`interaction.user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`interaction.user.username`	`string`	evet	
`interaction.user.display_name`	`string | null`	evet	
`interaction.user.avatar`	`string | null`	evet	Avatar hash'i.
`interaction.user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`interaction.user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`ephemeral_state`	`string`	-	Bileşenli ephemeral mesajda: mesajın mühürlü bileşenleri. Yalnızca mesajın sahibine gider, istemcisi basarken onu geri gönderir.

```
{
  "id": "1118042062848000 11",
  "type": 0,
  "channel_id": "21531887403008007",
  "guild_id": "21531879014400001",
  "author": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  },
  "content": "Bu akşam kim var?",
  "timestamp": "2026-11-05T12:30:00Z",
  "edited_timestamp": null,
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false
}
```


`MESSAGE_UPDATE` — Bir mesaj değişti

Düzenleme, sabitleme ya da önizleme değişikliğinden sonra mesajın tamamı. Ephemeral güncelleme ya da silme yalnızca basan kişiye gider; istemci onu yalnızca elinde flags 64 ile tuttuğu ve application_id'si aynı olan mesaja uygular.

- **Gereken intent'ler:** `GUILD_MESSAGES`, `DIRECT_MESSAGES`

> **Not:** Ayrıcalıklı `MESSAGE_CONTENT` intent'i olmadan `content` boş, `attachments` boş liste gelir; `sticker_items`, `poll`, `forward`, `hidden_embeds` ve `data` yer almaz. Mesajı bot yazdıysa ya da mesaj botu adıyla etiketliyorsa hepsi gelir.

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`type`	`integer`	evet	0 normal, 19 yanıt, 6 sabitleme bildirimi, 18 başlık açılışı, 24 otomatik moderasyon uyarısı, 1 ile 5 arası grup ve arama bildirimleri. Yenileri eklenebilir.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`author`	`object`	evet	
`author.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`author.username`	`string`	evet	
`author.display_name`	`string | null`	evet	
`author.avatar`	`string | null`	evet	Avatar hash'i.
`author.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`author.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`content`	`string`	evet	
`timestamp`	`datetime`	evet	
`edited_timestamp`	`string | null`	evet	
`attachments`	`Attachment[]`	evet	
`mentions`	`User[]`	evet	
`mention_roles`	`ID[]`	evet	
`mention_everyone`	`boolean`	evet	

Alan	Tür	Zorunlu	Açıklama
`message_reference`	`object`	-	
`message_reference.message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_reference.channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`pinned`	`boolean`	evet	
`data`	`object`	-	Sistem mesajının gösterilmesi için gerekenler.
`forward`	`object`	-	
`forward.channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.author`	`object`	evet	
`forward.author.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`forward.author.username`	`string`	evet	
`forward.author.display_name`	`string | null`	evet	
`forward.author.avatar`	`string | null`	evet	Avatar hash'i.
`forward.author.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`forward.author.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`forward.content`	`string`	evet	
`forward.timestamp`	`datetime`	evet	

Alan	Tür	Zorunlu	Açıklama
`forward.mentions`	`User[]`	evet	
`forward.sticker_items`	`StickerItem[]`	-	
`forward.webhook_name`	`string`	-	
`poll`	`object`	-	
`poll.question`	`string`	evet	
`poll.answers`	`object[]`	evet	
`poll.allow_multiselect`	`boolean`	evet	
`poll.expiry`	`datetime`	evet	
`poll.results`	`object`	-	
`poll.results.finalized`	`boolean`	evet	
`poll.results.voters`	`integer`	evet	
`poll.results.answers`	`object[]`	evet	
`sticker_items`	`StickerItem[]`	-	
`nonce`	`string`	-	Gönderenin mesaja verdiği kendi kimliği; oluşturulurken aynen döner.
`webhook_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`activity`	`object`	-	Başkalarını yazarın etkinliğine katılmaya çağırır.
`activity.type`	`integer`	evet	
`activity.party_id`	`string`	evet	
`hidden_embeds`	`string[]`	-	Yazarın kaldırdığı bağlantı önizlemeleri ve davet kartları: yazıldığı gibi bir bağlantı ya da küçük harfle bir davet kodu.
`embeds`	`Embed[]`	-	Botun eklediği kartlar.
`components`	`MessageComponent[]`	-	Botun eklediği butonları ve menüleri taşıyan satırlar.

Alan	Tür	Zorunlu	Açıklama
`flags`	`integer`	-	Bit bayrakları. 0 ise yoktur. 64: ephemeral, yalnızca basan kişi görür. 128: henüz hazırlanan yanıt.
`application_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`interaction`	`object`	-	Bir etkileşime verilen yanıtta: hangisi ve kimin bastığı ya da çalıştırdığı.
`interaction.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`interaction.type`	`integer`	evet	2 komut çalıştırma, 3 bileşene basmadır.
`interaction.name`	`string`	-	Çalıştırılan komut, "X, /ad kullandı" için. Basmada yer almaz.
`interaction.user`	`object`	evet	
`interaction.user.id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`interaction.user.username`	`string`	evet	
`interaction.user.display_name`	`string | null`	evet	
`interaction.user.avatar`	`string | null`	evet	Avatar hash'i.
`interaction.user.cosmetics`	`object | null`	evet	Varsa profil süslemeleri.
`interaction.user.bot`	`boolean`	-	Bot hesaplarında ve webhook'ların yazdığı hesaplarda bulunur ve true'dur; diğerlerinde yer almaz.
`ephemeral_state`	`string`	-	Bileşenli ephemeral mesajda: mesajın mühürlü bileşenleri. Yalnızca mesajın sahibine gider, istemcisi basarken onu geri gönderir.

```
{
  "id": "111804206284800011",
  "type": 0,
  "channel_id": "21531887403008007",
  "guild_id": "21531879014400001",
  "author": {
    "id": "14631410073600003",
    "username": "ayse",
    "display_name": "Ayşe",
    "avatar": null,
    "cosmetics": null
  },
  "content": "Bu akşam kim var? 21.00",
  "timestamp": "2026-11-05T12:30:00Z",
  "edited_timestamp": "2026-11-05T12:31:00Z",
  "attachments": [],
  "mentions": [],
  "mention_roles": [],
  "mention_everyone": false,
  "pinned": false
}
```


`MESSAGE_DELETE` — Bir mesaj silindi

Yalnızca ID'ler: mesaj artık yok. Ephemeral güncelleme ya da silme yalnızca basan kişiye gider; istemci onu yalnızca elinde flags 64 ile tuttuğu ve application_id'si aynı olan mesaja uygular.

- **Gereken intent'ler:** `GUILD_MESSAGES`, `DIRECT_MESSAGES`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`ephemeral`	`boolean`	-	Ephemeral mesajda true; yalnızca basan kişiye gider.
`application_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "id": "111804206284800011",
  "channel_id": "21531887403008007",
  "guild_id": "21531879014400001"
}
```

`MESSAGE_DELETE_BULK` — Birkaç mesaj silindi

Toplu silmede ve yasak üyenin son mesajlarını da sildiğinde gelir.

- **Gereken intent'ler:** `GUILD_MESSAGES`, `DIRECT_MESSAGES`

Alan	Tür	Zorunlu	Açıklama
`ids`	`ID[]`	evet	
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.

```
{
  "ids": [
    "111804206284800011"
  ],
  "channel_id": "21531887403008007",
  "guild_id": "21531879014400001"
}
```

`MESSAGE_POLL_UPDATE` — Bir anketin sonuçları değişti

Oy verilince ya da anket bitince anketin yeni sayıları.

- **Gereken intent'ler:** `GUILD_MESSAGES`, `DIRECT_MESSAGES`

Alan	Tür	Zorunlu	Açıklama
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`expiry`	`datetime`	evet	
`results`	`object`	evet	
`results.finalized`	`boolean`	evet	
`results.voters`	`integer`	evet	
`results.answers`	`object[]`	evet	

```
{
  "channel_id": "21531887403008007",
  "guild_id": "21531879014400001",
  "message_id": "111804206284800011",
  "expiry": "2026-11-06T12:30:00Z",
  "results": {
    "finalized": false,
    "voters": 2,
    "answers": [
      {
        "id": 1,
        "count": 2
      },
      {
        "id": 2,
        "count": 0
      }
    ]
  }
}
```

`MESSAGE_REACTION_ADD` — Bir tepki eklendi

Standart emojide `emoji.id` null olur.

- **Gereken intent'ler:** `GUILD_MESSAGE_REACTIONS`, `DIRECT_MESSAGE_REACTIONS`

Alan	Tür	Zorunlu	Açıklama
`user_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`emoji`	`object`	evet	
`emoji.id`	`ID | null`	evet	Sunucu emojisinde doludur.
`emoji.name`	`string`	evet	
`emoji.animated`	`boolean`	-	

```
{
  "user_id": "14631410073600003",
  "channel_id": "21531887403008007",
  "message_id": "111804206284800011",
  "guild_id": "21531879014400001",
  "emoji": {
    "id": null,
    "name": "□"
  }
}
```

`MESSAGE_REACTION_REMOVE` — Bir tepki kaldırıldı

`user_id` tepkisi kalkan kişidir, kaldıran değil.

- **Gereken intent'ler:** `GUILD_MESSAGE_REACTIONS`, `DIRECT_MESSAGE_REACTIONS`

Alan	Tür	Zorunlu	Açıklama
`user_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`emoji`	`object`	evet	
`emoji.id`	`ID | null`	evet	Sunucu emojisinde doludur.
`emoji.name`	`string`	evet	
`emoji.animated`	`boolean`	-	

```
{
  "user_id": "14631410073600003",
  "channel_id": "21531887403008007",
  "message_id": "111804206284800011",
  "guild_id": "21531879014400001",
  "emoji": {
    "id": null,
    "name": "□"
  }
}
```

`MESSAGE_REACTION_REMOVE_ALL` — Mesajın tüm tepkileri kaldırıldı

`ManageMessages` izni olan biri mesajı temizledi.

- **Gereken intent'ler:** `GUILD_MESSAGE_REACTIONS`, `DIRECT_MESSAGE_REACTIONS`

Alan	Tür	Zorunlu	Açıklama
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.


```
{
  "channel_id": "21531887403008007",
  "message_id": "111804206284800011",
  "guild_id": "21531879014400001"
}
```


`MESSAGE_REACTION_REMOVE_EMOJI` — Bir emojinin tüm tepkileri kaldırıldı

`ManageMessages` izni olan biri mesajdan bir emojiyi temizledi.

- **Gereken intent'ler:** `GUILD_MESSAGE_REACTIONS`, `DIRECT_MESSAGE_REACTIONS`

Alan	Tür	Zorunlu	Açıklama
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`message_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`emoji`	`object`	evet	
`emoji.id`	`ID | null`	evet	Sunucu emojisinde doludur.
`emoji.name`	`string`	evet	
`emoji.animated`	`boolean`	-	

```
{
  "channel_id": "21531887403008007",
  "message_id": "111804206284800011",
  "guild_id": "21531879014400001",
  "emoji": {
    "id": null,
    "name": "□"
  }
}
```


`TYPING_START` — Biri yazmaya başladı

`timestamp` Unix saniyedir.

- **Gereken intent'ler:** `GUILD_MESSAGE_TYPING`, `DIRECT_MESSAGE_TYPING`

Alan	Tür	Zorunlu	Açıklama
`channel_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	-	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`user_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`timestamp`	`integer`	evet	


```
{
  "channel_id": "21531887403008007",
  "guild_id": "21531879014400001",
  "user_id": "14631410073600003",
  "timestamp": 1793881800
}
```


`GUILD_SCHEDULED_EVENT_CREATE` — Bir etkinlik oluşturuldu

Ses kanalındaki etkinlik yalnızca kanalı görebilen botlara gelir.

- **Gereken intent'ler:** `GUILD_SCHEDULED_EVENTS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID | null`	evet	
`location`	`string | null`	evet	
`name`	`string`	evet	
`description`	`string | null`	evet	
`scheduled_start_time`	`datetime`	evet	
`scheduled_end_time`	`string | null`	evet	
`status`	`integer`	evet	1 planlandı, 2 sürüyor, 3 bitti, 4 iptal edildi. (1, 2, 3, 4)
`creator_id`	`ID | null`	evet	
`user_count`	`integer`	evet	


```
{
  "id": "105938052710400013",
  "guild_id": "21531879014400001",
  "channel_id": "21531891597312008",
  "location": null,
  "name": "Turnuva gecesi",
  "description": null,
  "scheduled_start_time": "2026-11-20T19:00:00Z",
  "scheduled_end_time": null,
  "status": 1,
  "creator_id": "113699192832000002",
  "user_count": 0
}
```


`GUILD_SCHEDULED_EVENT_UPDATE` — Bir etkinlik değişti

Başladığında, bittiğinde ya da `user_count` değiştiğinde de gelir.

- **Gereken intent'ler:** `GUILD_SCHEDULED_EVENTS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID | null`	evet	
`location`	`string | null`	evet	
`name`	`string`	evet	
`description`	`string | null`	evet	
`scheduled_start_time`	`datetime`	evet	
`scheduled_end_time`	`string | null`	evet	
`status`	`integer`	evet	1 planlandı, 2 sürüyor, 3 bitti, 4 iptal edildi. (1, 2, 3, 4)
`creator_id`	`ID | null`	evet	
`user_count`	`integer`	evet	


```
{
  "id": "105938052710400013",
  "guild_id": "21531879014400001",
  "channel_id": "21531891597312008",
  "location": null,
  "name": "Turnuva gecesi",
  "description": null,
  "scheduled_start_time": "2026-11-20T19:00:00Z",
  "scheduled_end_time": null,
  "status": 2,
  "creator_id": "113699192832000002",
  "user_count": 0
}
```


`GUILD_SCHEDULED_EVENT_DELETE` — Bir etkinlik silindi

Etkinliğin son hali. Etkinlik başka kanala taşındığında eski kanalı görebilen botlara önce bu gelir, yeni kanalı da görüyorlarsa `GUILD_SCHEDULED_EVENT_UPDATE` gelir.

- **Gereken intent'ler:** `GUILD_SCHEDULED_EVENTS`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`channel_id`	`ID | null`	evet	
`location`	`string | null`	evet	
`name`	`string`	evet	
`description`	`string | null`	evet	
`scheduled_start_time`	`datetime`	evet	
`scheduled_end_time`	`string | null`	evet	
`status`	`integer`	evet	1 planlandı, 2 sürüyor, 3 bitti, 4 iptal edildi. (1, 2, 3, 4)
`creator_id`	`ID | null`	evet	
`user_count`	`integer`	evet	


```
{
  "id": "105938052710400013",
  "guild_id": "21531879014400001",
  "channel_id": "21531891597312008",
  "location": null,
  "name": "Turnuva gecesi",
  "description": null,
  "scheduled_start_time": "2026-11-20T19:00:00Z",
  "scheduled_end_time": null,
  "status": 1,
  "creator_id": "11369919283200002",
  "user_count": 0
}
```


`AUTO_MODERATION_RULE_CREATE` — Bir otomatik moderasyon kuralı oluşturuldu

Yalnızca botun sunucuda `ManageGuild` izni varsa gelir.

- **Gereken intent'ler:** `AUTO_MODERATION_CONFIGURATION`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`creator_id`	`ID | null`	evet	
`name`	`string`	evet	
`enabled`	`boolean`	evet	
`trigger_type`	`integer`	evet	1 sözcük, 4 hazır sözcük listesi, 5 toplu etiketleme. (1, 4, 5)
`trigger_metadata`	`object`	evet	
`trigger_metadata.keyword_filter`	`string[]`	-	
`trigger_metadata.allow_list`	`string[]`	-	
`trigger_metadata.presets`	`integer[]`	-	1 küfür, 2 davetler.
`trigger_metadata.mention_total_limit`	`integer`	-	
`actions`	`object[]`	evet	
`exempt_roles`	`ID[]`	evet	
`exempt_channels`	`ID[]`	evet	

```
{
  "id": "54841363660800012",
  "guild_id": "21531879014400001",
  "creator_id": "11369919283200002",
  "name": "Yasaklı sözcükler",
  "enabled": true,
  "trigger_type": 1,
  "trigger_metadata": {
    "keyword_filter": [
      "spam"
    ]
  },
  "actions": [
    {
      "type": 1
    }
  ],
  "exempt_roles": [],
  "exempt_channels": []
}
```

`AUTO_MODERATION_RULE_UPDATE` — Bir otomatik moderasyon kuralı değişti

Yalnızca botun sunucuda `ManageGuild` izni varsa gelir.

- **Gereken intent'ler:** `AUTO_MODERATION_CONFIGURATION`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`creator_id`	`ID | null`	evet	
`name`	`string`	evet	
`enabled`	`boolean`	evet	
`trigger_type`	`integer`	evet	1 sözcük, 4 hazır sözcük listesi, 5 toplu etiketleme. (1, 4, 5)
`trigger_metadata`	`object`	evet	
`trigger_metadata.keyword_filter`	`string[]`	-	
`trigger_metadata.allow_list`	`string[]`	-	
`trigger_metadata.presets`	`integer[]`	-	1 küfür, 2 davetler.
`trigger_metadata.mention_total_limit`	`integer`	-	
`actions`	`object[]`	evet	
`exempt_roles`	`ID[]`	evet	
`exempt_channels`	`ID[]`	evet	

```
{
  "id": "54841363660800012",
  "guild_id": "21531879014400001",
  "creator_id": "11369919283200002",
  "name": "Yasaklı sözcükler",
  "enabled": false,
  "trigger_type": 1,
  "trigger_metadata": {
    "keyword_filter": [
      "spam"
    ]
  },
  "actions": [
    {
      "type": 1
    }
  ],
  "exempt_roles": [],
  "exempt_channels": []
}
```


`AUTO_MODERATION_RULE_DELETE` — Bir otomatik moderasyon kuralı silindi

Yalnızca botun sunucuda `ManageGuild` izni varsa gelir.

- **Gereken intent'ler:** `AUTO_MODERATION_CONFIGURATION`

Alan	Tür	Zorunlu	Açıklama
`id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`guild_id`	`ID`	evet	64 bitlik ID; yanıtta da istekte de her zaman metin olarak yazılır. ID'ler oluşturulma zamanına göre sıralanır.
`creator_id`	`ID | null`	evet	
`name`	`string`	evet	
`enabled`	`boolean`	evet	
`trigger_type`	`integer`	evet	1 sözcük, 4 hazır sözcük listesi, 5 toplu etiketleme. (1, 4, 5)
`trigger_metadata`	`object`	evet	
`trigger_metadata.keyword_filter`	`string[]`	-	
`trigger_metadata.allow_list`	`string[]`	-	
`trigger_metadata.presets`	`integer[]`	-	1 küfür, 2 davetler.
`trigger_metadata.mention_total_limit`	`integer`	-	
`actions`	`object[]`	evet	
`exempt_roles`	`ID[]`	evet	
`exempt_channels`	`ID[]`	evet	

```
{
  "id": "54841363660800012",
  "guild_id": "21531879014400001",
  "creator_id": "11369919283200002",
  "name": "Yasaklı sözcükler",
  "enabled": true,
  "trigger_type": 1,
  "trigger_metadata": {
    "keyword_filter": [
      "spam"
    ]
  },
  "actions": [
    {
      "type": 1
    }
  ],
  "exempt_roles": [],
  "exempt_channels": []
}
```