

CLAUDE CODE · KOD KALİTESİ · YAPAY ZEKÂ

Claude'un yazdığı gereksiz kodu Ponytail ile bul

Ponytail, yapay zekâ ajanının kod yazmadan önce "buna gerçekten gerek var mı?" diye sormasını sağlayan açık kaynak bir eklenti. Claude Code'a kurulum, mevcut projeyi tarama, bulguları ayıklama, güvenle silme ve yeni kodda açık tutma.

SÜRE

20 dk

SEVİYE

Orta

ADIM

5

PROMPT

1

Claude Code gibi ajanlar hızlı kod yazıyor. Bazen gereğinden fazlasını da yazıyor: hiç kullanılmayan bir bileşen, tarayıcının zaten yaptığı bir iş için kurulan paket, tek yerde çağrılan bir soyutlama. Bu rehberde bu fazlalığı Ponytail ile nasıl bulacağını ve güvenle sileceğini anlattım. Ben bunu kendi sitemde, yani şu an okuduğun dec5'te denedim. Komutların ve prompt'ların sağ üstündeki **Kopyala** butonuna bas, yapıştır, devam et.

Ne yapacağız

Projedeki gereksiz kodu bulup sileceğiz, sonra yeni kodda aynı fazlalığın birikmesini önleyeceğiz. Önce neden.

Gereksiz kodun üç bedeli var:

1. **Okuması zor.** Projeyi açan herkes (sen de, Claude da) önce o kodun ne işe yaradığını anlamaya çalışıyor. Hiçbir işe yaramıyorsa bu zaman boşa gidiyor.
2. **Bakım yükü.** Kullanılmayan kod da güncelleme, tip hatası ve build süresi olarak sana dönüyor.
3. **Paketler güvenlik yüzeyi ve boyut demek.** Kurulu her paket, bir gün güvenlik açığı çıkabilecek bir bağımlılık. Bir kısmı da sitenin yüklediği JavaScript'i büyütüyor.

Ponytail ne yapıyor? [Ponytail](#) açık kaynak (MIT lisanslı) tek bir prompt. Ajan kod yazmadan önce bir merdivenden geçiyor ve tutan ilk basamakta duruyor:

1. Buna gerçekten gerek var mı?
2. Projede zaten var mı?
3. Standart kütüphane bunu yapıyor mu?
4. Platformun kendi özelliği var mı?
5. Kurulu bir paket bunu yapıyor mu?
6. Tek satır olur mu?
7. Ancak o zaman: çalışan en az kod.

Merdiven, problemi anlamanın yerine geçmiyor. Ajan önce kodu okuyup akışı anlıyor, sonra basamak seçiyor. Girdi doğrulama, veri kaybını önleyen hata yönetimi, güvenlik ve erişilebilirlik hiçbir seviyede kesilmiyor.

Örnek: Tarih seçici istiyorsun. Ponytail olmadan ajan bir paket kurup üstüne sarmalayıcı bileşen yazabiliyor. Ponytail ile tarayıcının kendi alanını kullanıyor:

HTML

```
<input type="date">
```

Rakamlar hakkında dürüst olalım. Ponytail'in sayfası "~%54 daha az kod, ~%20 daha ucuz, ~%27 daha hızlı" diyor. Bu ölçüm tek bir gerçek FastAPI + React projesinde, 12 görevde, Haiku 4.5 modeliyle, her görev 4 kez yapılarak alınmış; aynı ajan Ponytail'li ve Ponytail'siz karşılaştırılmış. Etki, gerçek bir "fazla inşa" tuzağı olan görevlerde büyük (tarih seçici 404 satırdan 23 satıra), zaten sade kodda sıfıra yakın. Yazar, daha önce yayımladığı "%80-94" tek seferlik ölçümün kısmen yanıltıcı olduğunu kendisi kabul etmiş. GPT-5.5 gibi uzun düşünen modellerde maliyetin tersine dönebildiğini de yazıyor. Bunlar repo sahibinin ölçümü ve küçük bir örneklem. Kendi projende `/ponytail-audit` ile kendin ölç.

Sonuçta elinde üç şey olacak:

1. Projedeki gereksiz kodun sıralı bir listesi
2. O listeden güvenle silinenleri içeren ayrı bir dal (PR)
3. Yeni kod yazarken açık duran bir Ponytail

MALZEMELER

- [Claude Code](#)
- Git ile takip edilen bir proje. Silme işini ayrı bir dalda yapacağız; git yoksa geri dönüş yolun da yok.
- Projenin build (ve varsa test) komutu. Sildikten sonra her şeyin çalıştığını bununla doğrulayacağız.

ARAÇLAR

Claude Code

Ponytail

Git

01 Ponytail'i kur

Claude Code'da iki komutu **ayrı ayrı** gönder. İkisini tek mesajda yazarsan kurulum çalışmıyor.

TEXT

```
/plugin marketplace add DietrichGebert/ponytail
```

TEXT

```
/plugin install ponytail@ponytail
```

Kurulumdan sonra Ponytail her oturumda varsayılan **full** seviyesinde açık geliyor.

Yalnız resmi kaynaktan kur. Ponytail'in sahibi iki kaynak gösteriyor:

GitHub'da `DietrichGebert/ponytail`, npm'de

`@dietrichgebert/ponytail`. Ponytail hiçbir zaman `.exe` ya da `.dll` dosyası içermiyor; içeren bir kopya sahte.

Codex, Cursor, Copilot ya da başka bir ajan kullanıyorsan kurulum adımları Ponytail'in [INSTALL.md](#) sayfasında.

02 Mevcut projeyi tara

Önce temiz bir yerden başla. Kaydedilmemiş değişikliğin varsa commit'le, sonra tarama için yeni bir dal aç:

TERMINAL

```
git checkout -b temizlik/ponytail
```

Sonra Claude Code'da:

TEXT

```
/ponytail-audit
```

`/ponytail-audit` bütün projeyi tarıyor ve bulguları en büyük kazançtan başlayarak numaralı bir liste olarak veriyor. Her satırın başında bir etiket var:

- `delete:` ölü kod, kullanılmayan esneklik, spekülative özellik
- `native:` platformun zaten yaptığı işi yapan kod ya da paket
- `stdlib:` standart kütüphanede olan bir şeyin elle yazılmışı
- `reuse:` projede zaten var olan bir yardımcının kopyası
- `yagni:` tek uygulaması olan soyutlama, kimsenin ayarlamadığı ayar
- `shrink:` aynı mantık, daha az satır

Liste `net: -N satır, -M paket` ile bitiyor. **Audit hiçbir şeyi değiştirmiyor**, yalnız rapor veriyor.

Bende ne çıktı? dec5'te tarama yaklaşık 3,5 dakika sürdü. Yaklaşık 1.400 satır ölü kod ve 7 kullanılmayan paket buldu. En büyük üç bulgu:

1. Anasayfada artık gösterilmeyen bölümler (Contact, Services, Work, Process, CreatorTopics ve diğerleri), yaklaşık 781 satır.
2. Hiç import edilmeyen paketler: gsap, @react-three/fiber, @react-three/drei, react-hook-form, rehype-raw ve diğerleri.

3. Çalışmayan bir kaydırma geri yükleme kodu (ScrollRestore ve smooth-scroll), yaklaşık 35 satır.

Claude "yüzde kaç?" soruma yaklaşık %15 dedi. Daha karmaşık kendi projelerimde bu oran bende %35'e kadar çıktı. Bu ölçülmüş bir istatistik değil, kendi gözlemim; senin projende farklı çıkabilir.

03 Bulguları oku ve ayır

Listeyi olduğu gibi uygulama. Önce iki gruba ayır:

- **Güvenle silinebilir:** Hiçbir yerden import edilmeyen dosya, `package.json`'da duran ama kodda geçmeyen paket, hiç çağrılmayan fonksiyon.
- **Sorulması gereken:** Elle tetiklenen bir script, bir dosya adıyla ya da string olarak çağrılan modül, başka bir projenin kullandığı ortak kod. Bunları sen bilirsin, ajan bilmeyebilir.

Claude'dan dokunmayacaklarının listesini de iste. dec5'te Claude, aktif kullanılan lenis, three, next-intl ve react-markdown'ı ayrı tuttu. Emin olmadığı bir modülü silmeden önce bana sordu. Bu listeye bak; orada olması gerekip olmayan bir şey varsa söyle.

Bu ayıklamayı Claude Code'a da yaptırabilirsin:

PROMPT

/ponytail-audit sonucunu birlikte uygulayacağız. Henüz hiçbir şeyi silme.

1. Bulguları iki gruba ayır: "güvenle silinebilir" ve "bana sorman gereken". Bir dosya dinamik import, string ile yol, script ya da CI ayarı üzerinden kullanılıyor olabilirse ikinci gruba koy.
2. Her "güvenle silinebilir" madde için projenin tamamında (testler, ayar dosyaları ve scriptler dahil) ara ve hiçbir yerden kullanılmadığını göster.
3. Dokunmayacağın paket ve dosyaları gerekçesiyle ayrı bir liste olarak yaz.
4. Planı bana göster, onay vermeden silme.

04 Sil ve doğrula

Onay verdiğin maddeleri sildir. Silme işi ayrı bir dalda ve ayrı bir PR'da olsun; o dala başka iş karıştırma. Bir şey ters giderse tek hamlede geri alırsın.

Sildikten sonra sırayla:

1. Paket sildiysen bağımlılıkları yeniden kur (ör. `npm install`). Lock dosyası da güncellenmeli.
2. Build al. Geçmiyorsa sorun büyük ihtimalle ölü sanılan ama aslında kullanılan bir dosya.
3. Testlerin ve lint'in varsa onları da çalıştır.
4. Siteyi ya da uygulamayı aç ve elle gez. Build'in yakalamadığı şeyler olabiliyor: bir sayfa açılıyor ama bir bölüm boş geliyor gibi.

TERMINAL

```
npm install  
npm run build  
npm run lint
```

Komut adları projene göre değişebilir; kendi `package.json`'una bak.

dec5'teki temizlik bu şekilde ayrı bir PR'a gitti: 1.060 satır ve 7 paket siliniyor. Taramanın bulduğu 1.400 satırın hepsi silinmedi; ör. elle tetiklenebilen bir script'e bilerek dokunmadık. Bu rehberi yazarken o PR hâlâ açık, henüz yayına alınmadı.

05 Yeni kodda açık tut

Temizlik bir kerelik. Asıl kazanç, fazlalığın tekrar birikmemesi. Ponytail kurulduktan sonra her oturumda açık; seviyesini değiştirebilirsin:

TEXT

```
/ponytail lite  
/ponytail full  
/ponytail ultra  
/ponytail off
```

- **lite:** İsteneni yapıyor, daha sade bir yol varsa tek satırla söylüyor. Seçim senin.
- **full:** Varsayılan. Merdiven zorunlu: önce standart kütüphane ve platform özellikleri, en kısa değişiklik.

- **ultra:** En agresif seviye. Eklemeden önce silmeyi düşünüyor, isteğin kendisini de sorguluyor.
- **off:** Kapatıyor.

Argümentsiz `/ponytail` kapalıysa açıyor, açıksa şu anki seviyeyi söylüyor.

Her PR'dan önce de değişikliği taratabilirsin:

TEXT

```
/ponytail-review
```

`/ponytail-review` yalnız mevcut değişikliğe (diff) bakıyor ve "bunlar silinebilir" listesi veriyor. Audit gibi o da bir şey değiştirmiyor.

Diğer komutlar:

- `/ponytail-debt` : Kodda `ponytail:` yorumuyla bıraktığın bilinçli kısayolları tek listede topluyor. "Sonra bakarım" dediklerin unutulmasın diye.
- `/ponytail-gain` : Ponytail'in kendi benchmark ortalamalarını gösteriyor. Senin projen için hesaplanmış bir rakam değil; kendi projen için `/ponytail-audit` 'e bak.
- `/ponytail-help` : Komutların kısa özeti.

PÜF NOKTALARI

Kullanılan bir dosya "ölü" görünüyor. Dinamik import, string olarak verilen dosya yolu, bir script ya da CI ayarı üzerinden kullanılan dosyalar, kodda

import edilmediği için ölü sanılabiliyor. Her silmeden sonra build al. Emin değilsen o maddeyi "sorulması gereken" grubuna koy.

Komutlar arada hata veriyor. Ajanın çalıştırdığı terminal komutlarında tırnak ya da kaçış karakteri hataları çıkabiliyor (ör. zsh'de). Bu Ponytail'in bulgusu değil, komutun hatası. Hata mesajını ajana yapıştır, komutu düzeltip tekrar denesin. Hatalı komut yüzünden yarım kalan bir aramayı "bulunamadı" diye okuma.

ultra fazla agresif geliyor. `ultra` eklemek yerine silmeyi tercih ediyor ve isteğin kendisini de sorguluyor. Temizlik yaparken `full` 'da kal. `ultra` 'yı kullanırsan her önerisini tek tek oku.

Bulguyu uygulayınca bir şey bozuldu. Ayrı dalda çalıştığın için o dalı silip baştan başlayabilirsin. Bu yüzden 2. adımdaki temiz dal önemli.

BİTTİ Mİ?

- ☐ Ponytail'i iki ayrı komutla, yalnız resmi kaynaktan kurdum
- ☐ Çalışmamı commit'leyip temizlik için ayrı bir dal açtım
- ☐ `/ponytail-audit` çalıştırdım, listeyi okudum
- ☐ Bulguları "güvenle silinebilir" ve "sorulması gereken" diye ayırdım
- ☐ Claude'un dokunmayacaklarım listesini kontrol ettim
- ☐ Sildikten sonra build, test ve lint geçti
- ☐ Siteyi ya da uygulamayı açıp elle gezdim
- ☐ Temizliği ayrı bir PR olarak açtım
- ☐ Yeni kodda Ponytail açık; PR öncesi `/ponytail-review` çalıştırıyorum

Takıldığın yeri Instagram'da yaz, rehberine eklerim.