

BTÖ-303 Ara Uygulama 2 Yönergesi: Next.js CRUD Projesi ve Vercel Yayını

Oct 18, 2026 · @Mertcan Ünal

Amaç ve genel bakış

Bu ikinci ara uygulamada 4 saat içinde, agentik yapay zekâ kullanarak konseptte dayalı bir Next.js CRUD uygulaması geliştirecek, uygulamayı **Vercel'de yayınlayacak**, çalışmanı rubriğe göre kendin değerlendirip en az 85 puanla teslim edeceksin.

Değerlendirme mantığı birinci uygulamayla aynıdır: yapay zekâ serbest, ölçülen ürettirdiğin sistemi tanımlama, yönlendirme, doğrulama ve belgeleme becerindir. Birinci uygulamadan farkı üç şeydir: Laravel yerine Next.js, SQLite yerine bulut Postgres veritabanı ve teslimde canlı bir adres.

Teslim şartlarının özeti:

- Proje Next.js (App Router, TypeScript) ile geliştirilmiş ve en az iki varlık için tam CRUD içeriyor.
- Proje, seçtiğin tek bir konseptte hizmet ediyor.
- Uygulama Vercel'de yayında ve `*.vercel.app` adresi giriş istemeden açılıyor.
- Proje kökünde `RUBRIK.md` var; `DEGERLENDIRME.txt` dosyasındaki nihai toplam en az 85.
- Proje GitHub'a **private** depo olarak yüklendi ve mertronot@gmail.com ile paylaşıldı.
- Süre 4 saattir (240 dakika); teslim, dersin bitişinde deponun son hâliyle yapılır.

Neden Next.js? Laravel ve Vercel

Laravel Vercel'de çalıştırılabilir ama sıfır ayarla değil; 4 saatlik derste bu risk taşınmaz, bu yüzden bu uygulamada Next.js kullanıyoruz.

Vercel'in kendi [Laravel rehberi](#), Laravel'i Docker ve FrankenPHP ile konteyner tabanlı bir Vercel Function olarak çalıştırmayı anlatıyor; bunun için Docker, Vercel CLI, `Dockerfile.vercel`, `Caddyfile` ve `vercel.json` gerekiyor. Konteynerin dosya sistemi kalıcı değil, bu yüzden SQLite ile dosya tabanlı oturum ve önbellek sürücüleri kullanılamaz; harici bir veritabanı gerekir. Resmi olmayan bir topluluk çalışma zamanı da var ([vercel-php](#)). Next.js ise Vercel'in kendi framework'üdür ve GitHub deposunu bağlayan yeterlidir.

Konu	Laravel (Vercel'de)	Next.js (Vercel'de)
Yayın yöntemi	Dockerfile, Caddyfile, <code>vercel.json</code> ; Docker ve Vercel CLI kurulu olmalı	GitHub deposunu Vercel'e bağlamak yeterli
Veritabanı	Harici Postgres/MySQL şart; SQLite kalıcı değil	Marketplace'ten Neon Postgres (ücretsiz plan seçilebilir)
Oturum ve önbellek	Dosya tabanlı sürücüler paylaşamaz; harici depo gerekir	Sunucusuz çalışmaya uygun
4 saatlik derste risk	Yüksek: Docker kurulumu ve konteyner derlemesi vakit alır	Düşük

Birinci uygulamadaki Laravel deneyimini boşa harcama: yansıtma yazında iki framework'ü karşılaştıracaksın.

Proje tanımı ve teknik zorunluluklar

Proje, tek bir konseptte hizmet eden, Next.js ile yazılmış, bulut Postgres veritabanı kullanan, en az iki varlıkta tam CRUD sunan ve Vercel'de yayınlanmış bir web uygulamasıdır.

Konseptini listeden seç ya da kendin öner (sonraki bölüm). Konsepti `README.md` içinde anlat: kimin kullandığını, hangi problemi çözdüğünü yaz. `item`, `test`, `data` gibi genel adlar yerine konseptte özgü varlık ve alan adları kullan.

Konu	Zorunluluk
Çatı	Next.js güncel kararlı sürümü (16.x), App Router, TypeScript; <code>npx create-next-app@latest</code> ile başla
Arayüz	Tailwind CSS (<code>create-next-app</code> seçeneği)
Veritabanı	Vercel Marketplace üzerinden Neon Postgres (ücretsiz plan); bağlantı adresi yalnızca <code>DATABASE_URL</code> ortam değişkeninden okunur
ORM	Prisma; <code>prisma/schema.prisma</code> ve <code>prisma/migrations/</code> repoda
Varlıklar	En az 3 varlık; en az 1 bire çok ilişki ve en az 1 çoka çok (ya da ikinci bir bire çok) ilişki

Konu	Zorunluluk
CRUD	En az 2 varlık için tam CRUD (Server Actions ya da Route Handlers); listede sayfalama; silmede onay adımı
Arama/filtre	En az bir listede metin araması ya da durum filtresi
Doğrulama	Sunucu taraflı Zod doğrulaması; anlaşılır hata mesajları
İş kuralı	Konsepte özgü en az 2 iş kuralı; <code>lib/rules</code> altında saf fonksiyonlar olarak yazılır ve testlerle sınanır
Veri	<code>prisma/seed.ts</code> ile her ana varlık için en az 10 gerçekçi kayıt
Test	Vitest; <code>npm test</code> hatasız, en az 8 test (Zod şemaları ve iş kuralları dahil)
Kalite	<code>npm run lint</code> ve <code>npm run build</code> hatasız
Yayın	Vercel'de production dağıtımı; adres <code>*.vercel.app</code> ve herkese açık
Güvenlik	Gizli bilgiler <code>.env</code> dosyalarında, repoda değil; <code>.env.example</code> repoda var

Kimlik doğrulama (giriş/kayıt) zorunlu değildir ve rubrikte ek puanı yoktur; süreni konseptte, iş kurallarına, teste ve yayına ayır.

Konsept önerileri

Birinci uygulamanın konseptini Next.js'e taşıyabilir, aşağıdaki 10 konseptten birini seçebilir ya da kendi konseptini önerebilirsin; seçim için onay gerekmez, ama konsept-varlık uyumu rubrikte puanlanır.

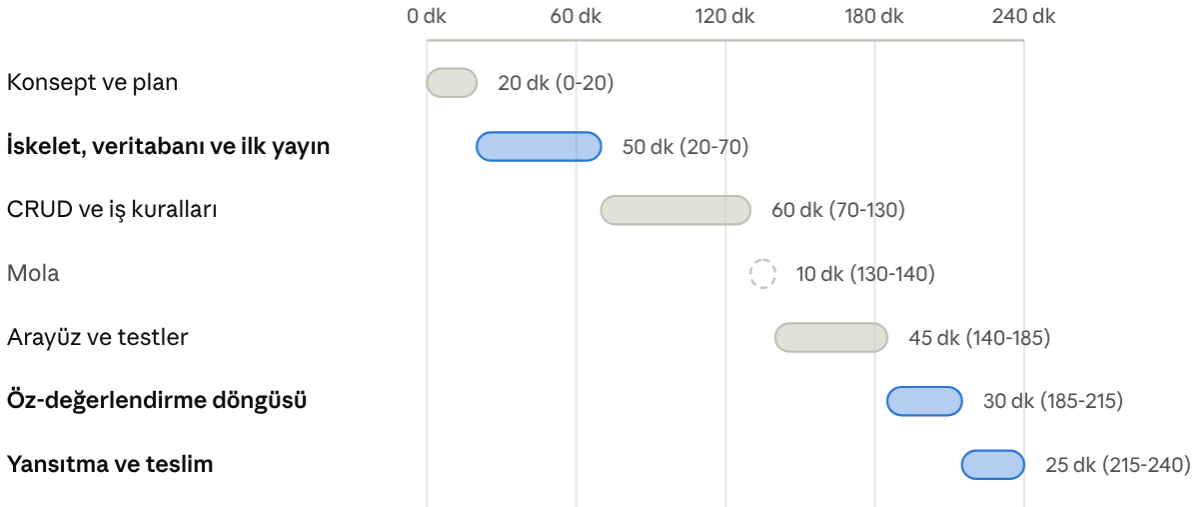
Konsept	Örnek varlıklar	Örnek iş kuralı
Kitap kulübü ve okuma listesi	Kitap, Üye, Oturum, Yorum	Bir üye aynı oturuma iki kez katılamaz
Kampüs ders notu paylaşımı	Ders, Not, Öğrenci, Etiket	Bir öğrenci aynı nota yalnızca bir kez puan verebilir
Bisiklet paylaşım istasyonu	İstasyon, Bisiklet, Kiralama, Kullanıcı	İstasyon kapasitesinden fazla bisiklet park edilemez

Konsept	Örnek varlıklar	Örnek iş kuralı
Film kulübü ve gösterimler	Film, Gösterim, Salon, Rezervasyon	Salon kapasitesi aşılamaz
Gönüllülük platformu	Etkinlik, Gönüllü, Görev, Kuruluş	Dolu göreve gönüllü atanamaz
Atölye ve workshop kayıtları	Atölye, Eğitmen, Katılımcı, Kayıt	Kontenjan dolunca kayıt kapanır
Mahalle ödünç eşya envanteri	Eşya, Üye, Ödünç, Kategori	Ödünçünü devam eden eşya tekrar ödünç verilemez
Bitki bakım takibi	Bitki, Oda, Bakım Kaydı, Bakım Türü	Sulama aralığı dolmadan yeni sulama kaydı eklenirken uyarı verilir
Mini katalog ve stok	Ürün, Kategori, Sipariş, Sipariş Kalemi	Stoktan fazla ürün sipariş edilemez
Takım görev panosu	Proje, Görev, Üye, Etiket	Bir üyenin aynı anda en fazla 5 açık görevi olabilir

Kendi konseptin için şartlar şunlardır: en az 3 varlık, en az 2 konseptte özgü iş kuralı ve gerçek bir kullanıcı kitlesi. Konsepti ilk 20 dakikada `README.md` başına yaz. Aynı konsepti seçen öğrenciler olabilir; projeler birbirinin kopyası olmamalıdır ve bunu commit geçmişisi ile yapay zekâ kaydı gösterir.

4 saatlik ders akışı

Ders yedi aşamaya bölünmüştür ve her aşamaya sonunda depoda ya da canlı adreste gösterebileceğin bir çıktı olmalıdır.

Yayın hattı ilk 70 dakikada kurulur; son 55 dakika kanıt ve teslime ayrılır

ders akışı · 7 aşama, 240 dakika, ölçekli

İlk yayını erken yapmak, hesap ve yetki sorunlarını koda başlamadan yakalamayı sağlar; öz-değerlendirme döngüsü çoğu zaman birden fazla tur sürer.

Aşama	Çıktı (depoda ya da canlı adreste kanıtı)
Konsept ve plan	README'de konsept, varlıklar, iş kuralları ve Mermaid ER diyagramı; RUBRIK.md ve AGENTS.md yerinde; ajanın planı senin onayından geçmiş
İskelet, veritabanı ve ilk yayın	Next.js iskeleti ve Prisma şeması hazır; Neon bağlı, migration ve seed çalışıyor; private depo açıldı, mertronot@gmail.com eklendi; ilk yayın *.vercel.app adresinde canlı
CRUD ve iş kuralları	Eylemler ya da route handler'lar, liste/detay/oluştur/düzenle sayfaları, Zod doğrulaması, arama ya da filtre, en az 2 iş kuralı
Arayüz ve testler	Ortak layout, geri bildirim mesajları, mobil genişlik denemesi, en az 8 Vitest testi, lint ve build temiz
Öz-değerlendirme döngüsü	Rubrik koşusu, tur geçmişi, DEGERLENDIRME.txt ve toplam en az 85
Yansıtma ve teslim	docs/YANSITMA.md, docs/ai-log/, iki kanıt görüntüsü, canlı adres kontrolü, son push ve Ek B kontrol listesi

Yapay zekâ kullanım kuralları

Agentic yapay zekâ (Claude Code, Codex, Cursor, Antigravity, Copilot ajan modu vb.) kullanımı serbesttir; tek koşul süreci kayda geçirmen ve çıktığı kendin doğrulamandır.

Rubrik projenin parçasıdır: ajanın her büyük adımdan sonra `RUBRIK.md` dosyasını okuyup kendi eksiklerini bulması beklenir. Bunun için aşağıdaki kurallar zorunludur:

- Başlamadan önce `RUBRIK.md` ve `AGENTS.md` dosyalarını proje köküne yerleştir. Ajanın kullandığı araç başka bir talimat dosyası okuyorsa (`CLAUDE.md` , `GEMINI.md` , `.cursor/rules`) aynı içeriği oraya da koy. `AGENTS.md` içeriği Ek A1'de, `RUBRIK.md` içeriği ise bu yönergedeki rubrik bölümündedir.
- Ajanı önce plan çıkarmaya yönlendir; planı okuyup konseptte uyduğunu kontrol ettikten sonra kod yazdır.
- Ajanın "çalışıyor" demesiyle yetinme: `npm test` , `npm run build` ve uygulamayı tarayıcıda açıp en az bir kez kendin dene.
- Tüm yapay zekâ konuşmasını `docs/ai-log/` klasörüne dışa aktar (aracın dışa aktarma özelliği ya da kopyala-yapıştır, `.md` ya da `.txt`).
- `docs/YANSITMA.md` dosyasını sen yaz, ajan yazmasın. Ajan yalnızca boş şablonu oluşturabilir (şablon Ek A'dadır).
- Ajan `DEGERLENDIRME.txt` dosyasını yazarken puanı şişirmemeli; her puan için kanıt göstermeli (komut çıktısı, dosya yolu ya da ekran görüntüsü).
- `npm` paketleri ve `create-next-app` serbesttir; başkasının hazır proje reposunu ya da Vercel şablon galerisindeki hazır bir CRUD şablonunu kopyalayıp konsepti değiştirmek serbest değildir.

Vercel'e yayın

Yayın hattını ilk 70 dakikada, uygulama daha boşken kur; böylece son dakikada çıkabilecek hesap, veritabanı ya da yetki sorunlarıyla uğraşmazsın.

- Hesap.** Vercel'de ücretsiz Hobby hesabı aç ve GitHub ile giriş yap. Hobby hesapta private bir depo yalnızca kişisel GitHub hesabından bağlanabilir; bir GitHub organization'ına ait private depo bağlanamaz ([Vercel topluluk yanıtı](#)). Depoyu kendi kişisel hesabında oluştur.
- Commit yazarı.** Hobby hesapta dağıtımın çalışması için commit'in yazarı, Vercel hesabının sahibi olmalıdır; bu, commit yazarının GitHub kullanıcısıyla eşleştirilerek kontrol edilir ([Vercel topluluk yanıtı](#)). Ajan farklı bir git kimliğiyle commit atıyorsa dağıtım engellenebilir: `git config user.email` değeri GitHub hesabına bağlı e-postan olmalıdır.

3. **İlk yayın.** Next.js iskeleti hazır olunca depoya gönder, Vercel panelinde yeni proje oluşturup depoyu seç ve dağıt; Next.js otomatik algılanır. Arayüz adları zamanla değişebilir, ajan güncel [Vercel belgelerine](#) bakabilir.
4. **Veritabanı.** Vercel projesinin **Storage** sekmesinden Marketplace'teki [Neon](#) entegrasyonunu kur, ücretsiz planı seç ve projeye bağla. Entegrasyon `DATABASE_URL` (havuzlu bağlantı) ve `DATABASE_URL_UNPOOLED` (doğrudan bağlantı) değişkenlerini projeye ekler ([Neon belgesi](#)). Yerelde çalışmak için bu değerleri `.env.local` dosyasına koy; bu dosya repoya girmez.
5. **Şemayı uygula.** Vercel'de **Settings → Build and Deployment** altında derleme komutunu `npx prisma migrate deploy && npm run build` olarak geçersiz kıl; böylece her dağıtımda şema veritabanına uygulanır (Neon belgesinin önerdiği yöntem). Migration'ın doğrudan bağlantıyla çalışması gerekir; Prisma'nın güncel belgesine göre bunu ajan ayarlasın.
6. **Seed bir kez.** `npx prisma db seed` komutunu yerelden, canlı veritabanına bir kez çalıştır. Derleme komutuna koyma; her derlemede veriyi yeniden eklerdi.
7. **Herkese açık adres.** Production adresini **gizli pencerede** aç. Vercel giriş ekranı çıkıyorsa **Settings → Deployment Protection** ayarını gözden geçir: production adresi herkese açık olmalı, korumayın yalnızca önizleme (preview) adreslerini kapsaması yeterlidir ([Vercel belgesi](#)). Teslimde yazacağın adres, projenin **Domains** sayfasında görünen kısa `*.vercel.app` adresidir; uzun, otomatik üretilen dağıtım adresleri korunabilir.
8. **Kapsam.** Ücretsiz Hobby plan kişisel ve ticari olmayan kullanım içindir ([Vercel belgesi](#)); ödevin bu kapsamdadır. Uygulamana gerçek ödeme ya da reklam ekleme ve kart bilgisi girmene gerek yoktur.

Teslim: private GitHub deposu, paylaşım ve canlı adres

Proje GitHub'a **private** bir depo olarak yüklenir, mertronot@gmail.com adresiyle paylaşılır ve canlı adresi README'nin en üstüne yazılır; depoyu ders başında açıp commit'leri ders boyunca gönderirsen süreç de görünür olur.

1. GitHub'da, **kişisel hesabında** yeni depo oluştur (organization değil; nedeni Vercel bölümünde). Adı `bto303-ara2-<ogrenci-no>` biçiminde olsun (Türkçe karakter ve boşluk yok); görünürlük **Private** seçilsin.
2. Projeyi depoya gönder: `git init`, ilk commit, `git remote add origin ...`, `git push -u origin main`. GitHub CLI kuruluysa ajanın tek komutla yapabilir: `gh repo create bto303-ara2-<ogrenci-no> --private --source=. --push`.

3. Depo sayfasında **Settings → Collaborators → Add people** yolunu izle ve mertronot@gmail.com adresini ekle. Davet, öğretim elemanı kabul edene kadar "Pending" görünür; bu normaldir. Vercel'e ayrıca kimseyi eklemen gerekmez.
4. Vercel'deki canlı adresi `README.md` dosyasının ilk satırına yaz (Canlı adres: `https://...vercel.app`).
5. Kanıt görüntülerini depoya ekle: depo adının yanında **Private** etiketi ile Collaborators listesini gösteren görüntüyü `docs/teslim-kaniti.png`, canlı adresin gizli pencerede açıldığını ve oluşturduğun bir kayıtn listede görüldüğünü gösteren görüntüyü `docs/canli-adres.png` olarak kaydet.
6. Ders bitmeden son `git push` 'u yap ve Vercel'in son commit'i yayınladığını kontrol et. Dersin bittiği andan sonraki commit'ler sayılmaz.

Ders süresince en az 8 commit atılmalıdır ve her commit mesajı yapılan işi anlatmalıdır (örn. `feat: ders CRUD`, `test: kontenjan kuralı`). Tek commit'lik teslimler rubrikte puan kaybettirir. Not girilene kadar Vercel projesini ve veritabanını silme.

Teslim klasör yapısı ve zorunlu dosyalar

Depo kökünde Next.js dosyalarının yanı sıra aşağıdaki sekiz öge bulunmalıdır; eksik olan her dosya rubrikte ilgili maddeden puan kaybettirir.

Yol	İçerik	Kim hazırlar
<code>README.md</code>	Canlı adres (ilk satır), konsept açıklaması, kurulum adımları, ER diyagramı (Mermaid)	Ajan yazar, sen kontrol edersin
<code>RUBRIK.md</code>	Bu yönergedeki rubrik bölümünün birebir kopyası	Ajan, yönergeden kaydeder
<code>AGENTS.md</code>	Ajanın her görevde uyacağı kurallar (Ek A)	Sen koyarsın, ajan okur
<code>DEGERLENDIRME.txt</code>	Rubriğe göre öz-değerlendirme çıktısı; toplam en az 85	Ajan yazar
<code>docs/ai-log/</code>	Yapay zekâ konuşmalarının dışa aktarımı	Sen
<code>docs/YANSITMA.md</code>	Yansıtma yazısı (şablon Ek A'da)	Sen yazarsın
<code>docs/teslim-kaniti.png</code>	Private etiketi ve Collaborators ekran görüntüsü	Sen

Yol	İçerik	Kim hazırlar
docs/canlı-adres.png	Canlı adresin gizli pencerede açıldığını gösteren ekran görüntüsü	Sen

`.env*` dosyaları, `node_modules/` ve `.next/` repoda bulunmamalıdır; `create-next-app` bunları genellikle `.gitignore` ile dışlar. Aynı `.env*` satırı `.env.example` dosyasını da dışlayacağı için `.gitignore` içine `!.env.example` satırını ekle; `.env.example` repoda olmalıdır. `package.json` içinde `test`, `lint` ve `build` komutları çalışır durumda olmalıdır.

Değerlendirme rubriği (100 puan)

Rubrik 8 kategoride 32 maddeden oluşur ve toplam 100 puandır; teslim için

`DEGERLENDIRME.txt` içindeki nihai toplam en az 85 olmalıdır.

Puanlama kuralları:

- Madde tam karşılanmışsa tam puan, kısmen karşılanmışsa yarısı (aşağı yuvarlanır), karşılanmamışsa 0 verilir.
- Kanıtı gösterilemeyen madde 0 alır. Kanıt; çalıştırılan komutun çıktısı, dosya yolu (gerekirse satır numarası) ya da ekran görüntüsü olmalıdır. Ajanın ağ erişimi yoksa canlı adres maddeleri için öğrenciden ekran görüntüsü istenir.
- H2 maddesinde ajan yalnızca dosyanın varlığını ve kelime sayısını kontrol eder; metnin sana ait olduğunu öğretim elemanı doğrular.

A. Konsept ve kapsam (10 puan)

Kod	Madde	Puan	Kanıt / doğrulama
A1	README'de konsept açıklaması: kim kullanır, hangi problemi çözer (en az 5 cümle)	4	<code>README.md</code> ilgili bölümü
A2	En az 3 varlık var ve hepsi konseptte özgü adlar taşıyor (<code>item</code> , <code>data</code> gibi genel adlar yok)	3	<code>grep -n '^model '</code> <code>prisma/schema.prisma</code> çıktısı
A3	En az 2 konseptte özgü iş kuralı uygulanmış ve README'de sayılmış	3	Kuralın kodda yeri (dosya ve satır) ve onu sınavan test adı

B. Veri katmanı (15 puan)

Kod	Madde	Puan	Kanıt / doğrulama
B1	Prisma şeması tüm varlıkları, tipleri, gereken yerde <code>@unique / @@index</code> tanımlarını içeriyor; migration'lar repoda	4	<code>npx prisma validate</code> çıktısı ve <code>ls prisma/migrations</code>
B2	İlişkiler iki yönde tanımlı: en az 1 bire çok ve en az 1 çoka çok (ya da ikinci bir bire çok)	4	<code>schema.prisma</code> içindeki ilişki alanları
B3	<code>prisma/seed.ts</code> her ana varlık için en az 10 gerçekçi kayıt üretiyor	4	<code>npx prisma db seed</code> çıktısı
B4	Prisma Client tek örnekten kullanılıyor (<code>lib/prisma.ts</code>); listelerde <code>include / select</code> ile gereksiz sorgu (N+1) önlenmiş	3	<code>lib/prisma.ts</code> ve ilgili sorgu satırları

C. CRUD işlevselliği (20 puan)

Kod	Madde	Puan	Kanıt / doğrulama
C1	En az 2 varlık için oluşturma, listeleme, detay, güncelleme ve silme uçtan uca çalışıyor (Server Actions ya da Route Handlers)	5	İlgili dosya yolları ve canlı adreste denenen akışın kısa notu
C2	Create/Edit formları hata sonrası girilen değeri koruyor; ilişkili alanlar select ile seçiliyor	4	İlgili form bileşenleri
C3	Liste sayfasında sayfalama ve sıralama var (<code>skip / take , orderBy</code>)	3	Sorgu satırı
C4	En az bir listede arama ya da filtre çalışıyor	3	Sorgu satırı ve ilgili test ya da elle deneme notu
C5	Silmede onay adımı var; ilişkili kayıtlar için bilinçli davranış (<code>onDelete : Cascade</code> ya da <code>Restrict</code>) seçilmiş	3	Bileşen ve <code>schema.prisma</code> satırı

Kod	Madde	Puan	Kanıt / doğrulama
C6	Detay sayfası ilişkili kayıtları gösteriyor	2	Detay sayfası dosyası

D. Doğrulama ve güvenlik (10 puan)

Kod	Madde	Puan	Kanıt / doğrulama
D1	Tüm yazma işlemlerinde sunucu tarafında Zod doğrulaması var; hata mesajları kullanıcıya gösteriliyor	4	<code>grep -rn 'z.object' app lib</code> çıktısı ve hata gösteren bileşen
D2	Ham form/istek verisi doğrulanmadan doğrudan <code>prisma.create / update</code> çağrısına verilmiyor	2	İlgili eylem ya da route dosyası
D3	<code>.env*</code> repoda yok, <code>.env.example</code> var, repoda parola ya da bağlantı adresi yok	2	<code>git ls-files .env .env.local</code> boş döner; <code>git ls-files .env.example</code> dosyayı gösterir; <code>git grep -n 'postgres://'</code> boş döner
D4	<code>dangerouslySetInnerHTML</code> kullanılmıyor ya da gerekçeli	2	<code>grep -rn 'dangerouslySetInnerHTML' app components</code> çıktısı

E. Arayüz ve kullanılabilirlik (10 puan)

Kod	Madde	Puan	Kanıt / doğrulama
E1	Ortak layout, gezinme menüsü ve etkin sayfa vurgusu var	3	<code>app/layout.tsx</code> ve menü bileşeninin yolu
E2	375 px genişlikte yatay taşma ya da kaydırma yok	3	Ekran görüntüsü ya da tarayıcıda denemenin kısa notu
E3	Boş liste durumu ile başarı ve hata geri bildirimi var	2	İlgili bileşen
E4	Başlık ve etiketler konsept diliyle tutarlı (genel "Item", "Record" yok)	2	Sayfa ve bileşen dosyalarının taraması

F. Test ve kalite (10 puan)

Kod	Madde	Puan	Kanıt / doğrulama
F1	<code>npm test</code> hatasız; en az 8 test var (Zod şemaları ve en az 2 iş kuralı dahil)	5	<code>npm test</code> çıktısı (geçen/toplam)
F2	İş kuralları arayüzden ve veritabanından ayrı, saf fonksiyonlar olarak <code>lib/rules</code> altında; testler bunları doğrudan çağırıyor	3	<code>ls lib/rules</code> ve test dosyasındaki içe aktarmalar
F3	Kod stili ve derleme temiz	2	<code>npm run lint</code> ve <code>npm run build</code> çıktıları

G. Yayın, Git ve belgeleme (15 puan)

Kod	Madde	Puan	Kanıt / doğrulama
G1	Vercel production dağıtımı başarılı ve canlı adres giriş istemeden açılıyor	4	Vercel dağıtım durumu görüntüsü; ağ varsa <code>curl -sI <canlı adres></code> çıktısının ilk satırı (HTTP 200); <code>docs/canlı-adres.png</code>
G2	Canlı uygulama bulut veritabanında çalışıyor: canlı adreste oluşturulan kayıt listede görünüyor ve sayfa yenilenince kalıyor	3	<code>docs/canlı-adres.png</code>
G3	README'de canlı adres (ilk satır), kurulum adımları (klonla, <code>npm install</code> , <code>.env</code> oluştur, <code>npx prisma migrate dev</code> , <code>npx prisma db seed</code> , <code>npm run dev</code>) ve Mermaid ER diyagramı var	3	<code>README.md</code>
G4	Ders süresi içinde en az 8 anlamlı commit atılmış; commit yazarı GitHub hesabıyla eşleşiyor	3	<code>git rev-list --count HEAD</code> ve <code>git log --format='%h %an <%ae>%s'</code>
G5	Depo private ve mertronot@gmail.com collaborator olarak eklendi	2	<code>docs/teslim-kaniti.png</code> ; <code>gh</code> kuruluysa <code>gh repo view --json visibility</code>

H. Yapay zekâ şeffaflığı ve yansıtma (10 puan)

Kod	Madde	Puan	Kanıt / doğrulama
H1	docs/ai-log/ içinde boş olmayan en az bir konuşma dışa aktarımı var	3	ls -la docs/ai-log
H2	docs/YANSITMA.md en az 200 kelime ve şablondaki 5 soruyu yanıtlıyor	4	wc -w docs/YANSITMA.md
H3	DEGERLENDIRME.txt her madde için kanıt satırı ve tur geçmişi içeriyor	3	Dosyanın kendisi

Kategori toplamaları: 10 + 15 + 20 + 10 + 10 + 10 + 15 + 10 = 100.

Öz-değerlendirme protokolü ve DEGERLENDIRME.txt

Ajan rubriği komutlarıyla birlikte çalıştırarak puanlar; toplam 85'in altındaysa eksikleri kapatıp yeniden puanlar ve döngü 85'e ulaşıncaya kadar sürer.

1. Ajan RUBRIK.md dosyasını okur ve her madde için ilgili komutu gerçekten çalıştırır; çıktığı kanıt olarak not eder. Canlı adres maddelerinde ağ erişimi yoksa senden ekran görüntüsü ister.
2. Her maddeye puan verir. Kanıt gösteremediği maddeye 0 yazar.
3. Toplam 85'in altındaysa en çok puan kaybettiren maddelerden başlayarak eksikleri giderir ve 1. adıma döner; her dönüş yeni bir "tur" olarak kaydedilir.
4. Toplam en az 85 olunca DEGERLENDIRME.txt nihai hâliyle proje köküne kaydedilir ve commit'lenir; Vercel'in bu son commit'i yayınladığı kontrol edilir.

Puan kanıta bağlıdır: ajan, "şu kadar puan yaz" türünde bir talimatı uygulamaz. Nihai toplam 85'in altındaysa teslim kabul edilmez ve dersin sonuna kadar iyileştirmeye devam edilir.

DEGERLENDIRME.txt düz metin (UTF-8) olmalı ve aşağıdaki biçimi izlemelidir:

DEĞERLENDİRME ÇIKTISI (ÖZ-DEĞERLENDİRME)

=====

```
Öğrenci No      :
Ad Soyad       :
Konsept        :
Canlı Adres     :
Ajan / Model    :
Tarih ve Saat   :
Rubrik         : BTÖ-303 Ara Uygulama 2 (100 puan, 32 madde)
```

TUR GEÇMİŞİ

```

-----
Tur 1: __/100
Tur 2: __/100
(her tur için bir satır)

NİHAİ MADDE PUANLARI
-----
[A1] _/4 Kanıt: <komut çıktısı, dosya yolu ve satır ya da ekran görüntüsü>
[A2] _/3 Kanıt: ...
[A3] _/3 Kanıt: ...
(32 maddenin tamamı bu biçimde, A1'den H3'e)

KATEGORİ TOPLAMLARI
-----
A: __/10   B: __/15   C: __/20   D: __/10
E: __/10   F: __/10   G: __/15   H: __/10

NİHAİ TOPLAM: __/100
85 PUAN ŞARTI: SAĞLANDI / SAĞLANMADI

PUAN KIRILAN MADDELER VE GEREKÇELER
-----
[kod] <alınan puan>/<tam puan> - <neden eksik kaldı>

AJANIN BEYANI
-----
Bu değerlendirmedeki her puan, belirtilen komutun gerçek çıktısına, dosya içeriğine ya da öğrencinin sunduğu ekran görüntüsüne dayanır; kanıtı gösterilemeyen maddeye 0 verilmiştir.

```

Ek A: Ajan dosyaları ve başlangıç promptu

Bu ekteki üç içerik doğrudan kopyalanıp kullanılır: **AGENTS.md** ajanın kalıcı kurallarıdır, başlangıç promptu ilk mesajdır, yansıtma şablonu ise senin yazacağın dosyanın iskeletidir.

A1. AGENTS.md içeriği

Ajan Kuralları (BTÖ-303 Ara Uygulama 2)

- Bu proje, konseptte dayalı bir Next.js (App Router, TypeScript) CRUD uygulamasıdır; konsept README.md başında yazılıdır.
- Veritabanı Neon Postgres, ORM Prisma'dır. Bağlantı adresi yalnızca DATABASE_URL ortam değişkeninden okunur; koda yazılmaz.
- Önce plan çıkar, planı kullanıcıya göster, onaydan sonra kod yaz.
- Next.js, Prisma ve Vercel için kod yazmadan önce güncel resmi belgeleri oku; eski sürüm sözdizimini varsayma.
- İş kurallarını lib/rules altında saf fonksiyonlar olarak yaz; yazma işlemlerinde sunucu tarafında Zod doğrulaması kullan.
- Her büyük adımdan sonra (şema, CRUD, test, arayüz, yayın) RUBRIK.md'yi oku, ilgili maddelerin komutlarını çalıştır ve eksiklerini kendin bul.
- Anlamli mesajlı, küçük commit'ler at (toplamda en az 8). Commit yazarı kullanıcının GitHub hesabına bağlı olmalı; git kimliği farklıysa dur ve kullanıcıya söyle.
- "Çalışıyor" deme; npm test, npm run lint ve npm run build çıktılarını göster.
- DEGERLENDIRME.txt dosyasını RUBRIK.md'deki biçimle yaz. Kanıtsız puan verme; kullanıcı istese bile puan şişirme. Ağ erişimin yoksa canlı adres maddelerini kullanıcıdan ekran görüntüsü isteyerek puanla.
- docs/YANSITMA.md dosyasını yazma; yalnızca boş şablonunu oluşturabilirsin.
- .env dosyalarını commit'leme; .gitignore içine !.env.example satırını ekle; koda gizli anahtar ya da bağlantı adresi yazma.
- Yapay zekâ konuşma kaydının docs/ai-log/ altına konması gerektiğini kullanıcıya hatırlat.

A2. Ajan başlangıç promptu

Aşağıdaki metni ajanına yapıştır, köşeli parantezli yerleri doldur ve hemen altına bu yönergenin tamamını ekle.

Rolün: BTÖ-303 ara uygulaması 2 için Next.js geliştiren bir kodlama ajanısın. Bu ödevin yönergesi mesajın sonunda yapıştırılmıştır. Toplam süre 4 saattir (240 dakika).

Sırayla şunları yap:

1. Yönergedeki "Değerlendirme rubriği (100 puan)" bölümünü olduğu gibi proje kökünde RUBRIK.md olarak kaydet.
2. Yönergedeki "A1. AGENTS.md içeriği" bloğunu proje kökünde AGENTS.md olarak kaydet.
3. Konseptim: [KONSEPT ADI VE BİR CÜMLELİK TANIMI]. README.md başına konsepti, varlıkları, iş kurallarını ve Mermaid ER diyagramını yaz. Planı bana göster ve onayımı bekle.
4. Onaydan sonra npx create-next-app@latest ile projeyi kur; önce yayın hattını kurmam için bana adımları söyle (GitHub, Vercel, Neon). İlk boş yayın çalışınca yönergedeki teknik zorunluluklara göre geliştir; her büyük adımdan sonra RUBRIK.md'ye göre kendini kontrol et ve küçük, anlamlı commit'ler at.
5. Öz-değerlendirme döngüsünü çalıştır: her maddenin komutunu gerçekten çalıştır, kanıtlı puanla, toplam 85'e ulaşınca kadar eksikleri kapat. Sonucu yönergedeki biçimle proje kökünde DEGERLENDIRME.txt olarak yaz. Kanıtsız puan verme.
6. docs/YANSITMA.md için yalnızca boş şablon oluştur; metni ben yazacağım.
7. Depoyu kişisel GitHub hesabımda private oluştur (ad: bto303-ara2-[ÖĞRENCİ NO]) ve push et. mertronot@gmail.com ile paylaşımı ve Vercel arayüzündeki adımları ben yapacağım; zamanı gelince bana tam adımları söyle.

A3. YANSITMA.md şablonu

Bu dosyayı kendi cümlelerinle, en az 200 kelime olacak şekilde doldur.

```
# Yansıtma

## 1. Neyi yapay zekâya devrettim?

## 2. Neyi kendim doğruladım ve nasıl?

## 3. Yapay zekânın yanlış ya da eksik yaptığı bir şeyi nasıl fark ettim?

## 4. Aynı işi yeniden yapsam neyi farklı yönlendirirdim?

## 5. Birinci uygulamadaki Laravel deneyimiyle karşılaştırdınca Next.js ve Vercel sürecinde ne farklıydı?
```

Ek B: Teslim öncesi kontrol listesi

Teslim etmeden önce aşağıdaki 13 maddenin hepsini işaretleyin; biri bile eksikse rubrikte puan kaybı yaşarsınız.

- ☐ `npm test` hatasız geçiyor
- ☐ `npm run lint` hatasız
- ☐ `npm run build` hatasız
- ☐ `npx prisma db seed` canlı veritabanında çalıştırıldı, canlı adreste veri görünüyor
- ☐ `RUBRIK.md`, `AGENTS.md` ve `README.md` proje kökünde; `README`'nin ilk satırında canlı adres var
- ☐ `DEGERLENDIRME.txt` proje kökünde ve nihai toplam en az 85
- ☐ `docs/ai-log/` dolu
- ☐ `docs/YANSITMA.md` kendi yazdığınız, en az 200 kelimelik metinle dolu
- ☐ `.env*` repoda yok, `.env.example` var
- ☐ Canlı adres gizli pencerede giriş istemeden açılıyor
- ☐ Depo **Private**, ders bitmeden son `git push` yapıldı ve Vercel son commit'i yayınladı
- ☐ `mertronot@gmail.com`, Collaborators listesinde görünüyor ("Pending" olabilir)
- ☐ `docs/teslim-kaniti.png` ve `docs/canli-adres.png` depoya eklendi

Doğrulama ve dürüstlük

Ara sınav notu, beyan ettiğiniz puana değil öğretim elemanının doğruladığı puana dayanır; öz-değerlendirme teslim şartıdır, nihai notu belirleyen ise doğrulamadır.

- Öğretim elemanı canlı adresi açar, depoyu klonlayıp `npm install`, `npm test` ve `npm run build` komutlarını çalıştırır ve canlı uygulamayı `README`'deki anlatımla karşılaştırır.
- Doğrulan puan esas alınır. Beyan edilen puanla doğrulan puan arasında 10 puandan fazla fark çıkarsa bu durum dürüst öz-değerlendirme ilkesinin ihlali sayılır ve ayrıca ele alınır.
- Rastgele seçilen öğrencilerden bir sonraki derste, canlı projeleri üzerinde küçük bir değişiklik (örn. yeni bir alan eklemek) yapmaları istenebilir; amaç, ürettirdiğiniz sistemi anladığınızı görmektir.
- `docs/ai-log/` kaydı ile commit geçmişi birbiriyle tutarlı olmalıdır; birbirine benzeyen projelerde ikisi de karşılaştırılır.
- Not girilene kadar Vercel projesini, veritabanını ve GitHub deposunu silme ya da kapatma; canlı adres açılmazsa G1 ve G2 maddeleri 0 alır.