DJANGO PROJESİ GELİŞTİRME KILAVUZU

*Başlangıçtan Canlı Yayına Kadar Adım Adım Rehber*

Oluşturulma Tarihi: 27.08.2025

# İÇİNDEKİLER

1. 1. GİRİŞ VE KAPSAM
2. 2. ÖN GEREKSİNİMLER
3. 3. PROJE AŞAMALARI DETAYLARİ
4. 4. YAYGIN HATALAR VE ÇÖZÜMLER
5. 5. İPUÇLARI VE BEST PRACTICES
6. 6. KAYNAKLAR VE REFERANSLAR

# 1. GİRİŞ VE KAPSAM

Bu kılavuz, Django web framework'ü kullanarak profesyonel seviyede web uygulamaları geliştirmek isteyen geliştiriciler için hazırlanmıştır.   
Başlangıç seviyesinden ileri seviyeye kadar tüm aşamaları kapsar ve real-world projelerde karşılaşabileceğiniz sorunlar için çözümler sunar.  
  
Bu kılavuzun özelliklerl:  
• 108 detaylı adım ile kapsamlı kapsam  
• Her adım için pratik örnekler  
• Yaygın hatalar ve çözüm yolları  
• Production deployment rehberi  
• Security best practices  
• Performance optimization teknikleri  
• Tek geliştirici odaklı yaklaşım

# 2. ÖN GEREKSİNİMLER

Bu kılavuzu takip etmek için aşağıdaki bilgi ve araçlara sahip olmanız gerekmektedir:  
  
• Temel Python bilgisi (değişkenler, fonksiyonlar, sınıflar)  
• HTML, CSS temel bilgisi  
• Command line/terminal kullanımı  
• Git version control temel bilgisi  
• PostgreSQL veritabanı temel kavramları  
  
Gerekli Araçlar:  
• Python 3.8 veya üzeri  
• Code editor (VS Code önerilir)  
• Git  
• PostgreSQL  
• Web browser  
• Terminal/Command Prompt

# 3. PROJE AŞAMALARI DETAYLARİ

## PROJE PLANLAMA VE ANALİZ AŞAMASI

Bu aşamada projenin temellerini atıyoruz. Doğru planlama yapılmayan projeler genellikle başarısız olur.

### ADIM 001: 📊 Proje Fikrini Netleştir

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1-2 gün |
| Bağımlılıklar | Yok |
| Kategori | 🎯 Planlama |
| İpuçları | Kağıt kalem ile brainstorming yap |

#### AÇIKLAMA:

Projenin tam olarak ne yapacağını, hangi problemi çözeceğini belirle. Hedef kitleyi tanımla.

#### NASIL YAPILIR:

1. Problemi Tanımlayın:  
 • Hangi sorunu çözüyorsunuz?  
 • Bu sorun gerçekten var mı?  
 • İnsanlar bunun için para ödeyebilir mi?  
  
2. Hedef Kitle Analizi:  
 • Uygulamanızı kimler kullanacak?  
 • Yaş, meslek, teknik seviye?  
 • Hangi platformları tercih ediyorlar?  
  
3. Rakip Analizi:  
 • Benzer uygulamalar var mı?  
 • Onlardan farkınız ne?  
 • Nasıl daha iyi olabilirsiniz?  
  
4. Özellik Listesi:  
 • Temel özellikler (MVP)  
 • Gelecekteki özellikler  
 • Nice-to-have özellikler  
  
5. Başarı Metrikleri:  
 • Hangi sayılar başarıyı gösterecek?  
 • Kullanıcı sayısı, işlem hacmi, gelir?

#### ÖRNEK PROJE FİKRİ:

Proje: "Öğrenci Yemekhane Sistemi"  
Problem: Öğrenciler yemekhane menüsünü göremiyorlar ve rezervasyon yapamiyorlar.  
Hedef Kitle: Üniversite öğrencileri (18-25 yaş)  
Temel Özellikler: Menü görüntüleme, rezervasyon, ödeme, admin paneli  
Başarı Metriği: Günlük 500+ aktif kullanıcı

────────────────────────────────────────────────────────────────────────────────

### ADIM 002: 🎯 Gereksinim Analizi Yap

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 001 |
| Kategori | 🎯 Planlama |
| İpuçları | Trello/Notion kullanarak organize ol |

#### AÇIKLAMA:

Projenin tüm fonksiyonel ve teknik gereksinimlerini listele. User story'leri yaz.

#### NASIL YAPILIR:

1. Fonksiyonel Gereksinimler:  
 • Kullanıcı neler yapabilecek?  
 • Sistemin hangi işlevleri olacak?  
 • Input ve output'lar neler?  
  
2. Teknik Gereksinimler:  
 • Hangi teknolojiler kullanılacak?  
 • Performans beklentileri?  
 • Güvenlik gereksinimleri?  
  
3. User Story Yazma:  
 Format: "Ben [kullanıcı tipi] olarak, [hedef] için [özellik] istiyorum."  
  
4. Acceptance Criteria:  
 • Her user story için kabul kriterleri  
 • Test edilebilir koşullar  
  
5. Non-Functional Requirements:  
 • Performance (yanıt süresi)  
 • Security (veri korunması)  
 • Usability (kullanım kolaylığı)  
 • Scalability (ölçeklenebilirlik)

#### ÖRNEK USER STORY:

"Ben öğrenci olarak, bugünkü yemek menüsünü görmek için uygulamaya giriş yapmak istiyorum."  
  
Acceptance Criteria:  
✓ Kullanıcı email/şifre ile giriş yapabilmeli  
✓ Giriş sonrası anasayfada günün menüsü görünmeli  
✓ Menüde yemek adı, fiyat ve kalori bilgisi olmalı  
✓ Yanlış şifre girişinde hata mesajı gösterilmeli

────────────────────────────────────────────────────────────────────────────────

### ADIM 003: 🗂️ Proje Klasör Yapısını Planla

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 002 |
| Kategori | 🎯 Planlama |
| İpuçları | Django best practices araştır |

#### AÇIKLAMA:

Django projesi için klasör hiyerarşisini kağıt üzerinde çiz

#### DİKKAT EDİLECEK NOKTALAR:

•   
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 004: 📊 Teknoloji Stack'i Belirle

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 002 |
| Kategori | 🎯 Planlama |
| İpuçları | LTS versiyonları tercih et |

#### AÇIKLAMA:

Django version, Python version, veritabanı, frontend teknolojilerini seç

#### DİKKAT EDİLECEK NOKTALAR:

• Güncel ve stabil sürümler seç  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 005: 🗄️ Veritabanı Şeması Tasarla

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 002,004 |
| Kategori | 🎯 Planlama |
| İpuçları | draw.io veya Lucidchart kullan |

#### AÇIKLAMA:

Tüm tabloları, ilişkileri, alanları kağıt/dijital olarak tasarla

#### DİKKAT EDİLECEK NOKTALAR:

• Normalizasyon kurallarına dikkat  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 006: 🎨 UI/UX Wireframe Hazırla

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2-4 gün |
| Bağımlılıklar | 002 |
| Kategori | 🎯 Planlama |
| İpuçları | Basit çizimlerle başla |

#### AÇIKLAMA:

Tüm sayfaların kabataslak çizimini yap (kağıt veya Figma)

#### DİKKAT EDİLECEK NOKTALAR:

• Kullanıcı deneyimini öncelikle düşün  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 007: 📋 Proje Zaman Çizelgesi Oluştur

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 001-006 |
| Kategori | 🎯 Planlama |
| İpuçları | Buffer süre ekle |

#### AÇIKLAMA:

Her aşama için gerçekçi süre planlaması yap

#### DİKKAT EDİLECEK NOKTALAR:

• Çok iyimser tahmin yapma  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## GELİŞTİRME ORTAMI KURULUMU

Development environment kurulumu ve temel araçların yapılandırılması.

### ADIM 008: 🐍 Python Kurulumu Kontrol Et

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | Yok |
| Kategori | ⚙️ Kurulum |
| İpuçları | python --version komutu çalıştır |

#### AÇIKLAMA:

Python 3.8+ kurulu olduğunu doğrula, gerekirse güncelle

#### NASIL YAPILIR:

1. Python Sürümü Kontrolü:  
 Terminalde: python --version veya python3 --version  
   
2. Windows için Kurulum:  
 • python.org'dan Python 3.8+ indirin  
 • "Add to PATH" seçeneğini işaretleyin  
 • pip otomatik olarak gelir  
  
3. macOS için Kurulum:  
 • Homebrew ile: brew install python3  
 • Veya python.org'dan installer  
  
4. Linux için Kurulum:  
 • Ubuntu/Debian: sudo apt-get install python3 python3-pip  
 • CentOS/RHEL: sudo yum install python3 python3-pip  
  
5. Kurulum Testi:  
 python --version  
 pip --version  
   
6. Pip Güncellemesi:  
 python -m pip install --upgrade pip

#### YAYGIN SORUNLAR:

❌ Problem: "python command not found"  
✅ Çözüm: PATH değişkenine Python ekleyin  
  
❌ Problem: Python 2.x kurulu ama 3.x gerekli   
✅ Çözüm: python3 komutunu kullanın veya alias oluşturun  
  
❌ Problem: pip çalışmıyor  
✅ Çözüm: python -m pip komutunu kullanın

────────────────────────────────────────────────────────────────────────────────

### ADIM 009: 📦 IDE/Editor Seç ve Konfigüre Et

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 008 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Linting, formatting araçlarını kur |

#### AÇIKLAMA:

VSCode, PyCharm vb. seç, Django extension'ları yükle

#### DİKKAT EDİLECEK NOKTALAR:

• Kod yazma hızını artırır  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 010: 📁 Ana Proje Klasörü Oluştur

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 0.1 gün |
| Bağımlılıklar | 003 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Türkçe karakter kullanma |

#### AÇIKLAMA:

Bilgisayarda projenin ana klasörünü oluştur

#### DİKKAT EDİLECEK NOKTALAR:

•   
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 011: 🔧 Virtual Environment Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 008,010 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Her projede ayrı venv kullan |

#### AÇIKLAMA:

python -m venv venv komutu ile sanal ortam oluştur

#### NASIL YAPILIR:

1. Virtual Environment Oluşturma:  
 Windows: python -m venv venv  
 macOS/Linux: python3 -m venv venv  
  
2. Aktivasyon:  
 Windows: venv\Scripts\activate  
 macOS/Linux: source venv/bin/activate  
  
3. Aktivasyon Kontrolü:  
 Terminal'de (venv) yazısını görmelisiniz  
 which python komutu venv içindeki python'u göstermeli  
  
4. Deaktivasyon:  
 deactivate komutu  
  
5. Paket Yükleme:  
 pip install django  
 pip install psycopg2-binary  
   
6. Requirements Dosyası:  
 pip freeze > requirements.txt  
 pip install -r requirements.txt

#### NEDEN GEREKLİ:

• Global Python'u kirletmez  
• Proje bağımlılıklarını izole eder  
• Farklı projeler farklı paket versiyonları kullanabilir  
• Production deployment kolaylaşır  
• Takım üyeleri aynı environment'ı kullanabilir

────────────────────────────────────────────────────────────────────────────────

### ADIM 012: 🔄 Virtual Environment Aktif Et

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.1 gün |
| Bağımlılıklar | 011 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Terminal'de (venv) görünmeli |

#### AÇIKLAMA:

venv\Scripts\activate (Windows) komutu çalıştır

#### NASIL YAPILIR:

1. Virtual Environment Oluşturma:  
 Windows: python -m venv venv  
 macOS/Linux: python3 -m venv venv  
  
2. Aktivasyon:  
 Windows: venv\Scripts\activate  
 macOS/Linux: source venv/bin/activate  
  
3. Aktivasyon Kontrolü:  
 Terminal'de (venv) yazısını görmelisiniz  
 which python komutu venv içindeki python'u göstermeli  
  
4. Deaktivasyon:  
 deactivate komutu  
  
5. Paket Yükleme:  
 pip install django  
 pip install psycopg2-binary  
   
6. Requirements Dosyası:  
 pip freeze > requirements.txt  
 pip install -r requirements.txt

#### NEDEN GEREKLİ:

• Global Python'u kirletmez  
• Proje bağımlılıklarını izole eder  
• Farklı projeler farklı paket versiyonları kullanabilir  
• Production deployment kolaylaşır  
• Takım üyeleri aynı environment'ı kullanabilir

────────────────────────────────────────────────────────────────────────────────

### ADIM 013: 📦 Django Kurulumu

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 012 |
| Kategori | ⚙️ Kurulum |
| İpuçları | pip install django==5.0 belirli versiyon |

#### AÇIKLAMA:

pip install django komutu ile Django'yu yükle

#### DİKKAT EDİLECEK NOKTALAR:

• LTS versiyonu tercih et  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 014: 🏗️ Django Projesi Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 013 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Proje adında Türkçe karakter yok |

#### AÇIKLAMA:

django-admin startproject myproject komutu çalıştır

#### NASIL YAPILIR:

1. Django Kurulumu:  
 pip install django  
  
2. Proje Oluşturma:  
 django-admin startproject myproject  
   
3. Klasör Yapısı İncelemesi:  
 myproject/  
 ├── manage.py (Django komutları için)  
 ├── myproject/  
 │ ├── \_\_init\_\_.py  
 │ ├── settings.py (ayarlar)  
 │ ├── urls.py (URL routing)  
 │ ├── wsgi.py (deployment için)  
 │ └── asgi.py (async için)  
  
4. İlk Çalıştırma:  
 cd myproject  
 python manage.py runserver  
   
5. Tarayıcıda Test:  
 http://127.0.0.1:8000/  
 Django welcome sayfasını görmelisiniz  
  
6. Admin Superuser:  
 python manage.py createsuperuser

#### PROJE ADI SEÇİMİ:

✅ İyi Örnekler: blog, ecommerce, taskmanager  
❌ Kötü Örnekler: mysite, project, django\_app  
  
Kurallar:  
• Küçük harflerle  
• Alt çizgi kullanabilirsiniz  
• Türkçe karakter yok  
• Python keyword'leri kullanmayın (if, for, class vb.)  
• Django'nun builtin modül adlarını kullanmayın

────────────────────────────────────────────────────────────────────────────────

### ADIM 015: 🎯 Django Apps Planla

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 005,014 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Her özellik için ayrı app |

#### AÇIKLAMA:

Projenin modülerlik için hangi app'lere bölüneceğini belirle

#### DİKKAT EDİLECEK NOKTALAR:

• Sonradan değiştirmek zor  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 016: 📱 Django Apps Oluştur

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 015 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Tekil isim kullan (user, not users) |

#### AÇIKLAMA:

python manage.py startapp appname komutları çalıştır

#### DİKKAT EDİLECEK NOKTALAR:

• INSTALLED\_APPS'e eklemeyi unutma  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 017: 🗄️ PostgreSQL Kurulumu

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 004 |
| Kategori | ⚙️ Kurulum |
| İpuçları | pgAdmin da kur |

#### AÇIKLAMA:

PostgreSQL'i indir, kur ve konfigüre et

#### DİKKAT EDİLECEK NOKTALAR:

• Varsayılan şifre not et  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 018: 🔗 Veritabanı Bağlantısı Kur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 017 |
| Kategori | ⚙️ Kurulum |
| İpuçları | psycopg2-binary paketi gerekli |

#### AÇIKLAMA:

Django settings.py'da PostgreSQL konfigürasyonu yap

#### DİKKAT EDİLECEK NOKTALAR:

• Bağlantı bilgilerini test et  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 019: 📋 Requirements.txt Oluştur

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 013,018 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Düzenli güncelle |

#### AÇIKLAMA:

pip freeze > requirements.txt komutu çalıştır

#### DİKKAT EDİLECEK NOKTALAR:

• Version control için önemli  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 020: 🌍 Git Repository Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 019 |
| Kategori | ⚙️ Kurulum |
| İpuçları | .gitignore dosyası ekle |

#### AÇIKLAMA:

git init, git add, git commit komutları çalıştır

#### DİKKAT EDİLECEK NOKTALAR:

• Her önemli değişiklikte commit  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 021: 🐙 GitHub'a Push Et

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 020 |
| Kategori | ⚙️ Kurulum |
| İpuçları | README.md dosyası ekle |

#### AÇIKLAMA:

GitHub'da repo oluştur ve push et

#### DİKKAT EDİLECEK NOKTALAR:

• Günlük push yapma alışkanlığı  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 022: 🔧 Django Settings Yapılandır

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 018 |
| Kategori | ⚙️ Kurulum |
| İpuçları | Çevresel değişkenler kullan |

#### AÇIKLAMA:

Development/Production ayarlarını düzenle

#### DİKKAT EDİLECEK NOKTALAR:

• SECRET\_KEY'i güvenli tut  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## VERİTABANI TASARIMI VE MODEL OLUŞTURMA

Django ORM kullanarak veritabanı modellerinin tasarlanması ve oluşturulması.

### ADIM 023: 📊 Model Sınıfları Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 3-5 gün |
| Bağımlılıklar | 005,016 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Field tiplerini doğru seç |

#### AÇIKLAMA:

Her app için models.py dosyalarında model sınıflarını yaz

#### NASIL YAPILIR:

1. Model Sınıfı Tanımlama:  
   
 class User(models.Model):  
 username = models.CharField(max\_length=150, unique=True)  
 email = models.EmailField()  
 first\_name = models.CharField(max\_length=30, blank=True)  
 is\_active = models.BooleanField(default=True)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
   
 class Meta:  
 db\_table = 'users'  
 ordering = ['-created\_at']  
   
 def \_\_str\_\_(self):  
 return self.username  
  
2. Field Tipleri:  
 • CharField: Kısa metin (max\_length zorunlu)  
 • TextField: Uzun metin  
 • IntegerField: Tam sayı  
 • BooleanField: True/False  
 • DateTimeField: Tarih ve saat  
 • EmailField: Email formatı  
 • ForeignKey: İlişki (başka model)  
  
3. Field Parametreleri:  
 • null=True: Database'de NULL olabilir  
 • blank=True: Form'da boş geçilebilir  
 • default: Varsayılan değer  
 • unique=True: Benzersiz olmalı  
 • db\_index=True: Index oluştur

────────────────────────────────────────────────────────────────────────────────

### ADIM 024: 🔗 Model İlişkileri Kur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 023 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | related\_name kullan |

#### AÇIKLAMA:

ForeignKey, ManyToMany, OneToOne ilişkilerini tanımla

#### NASIL YAPILIR:

1. Model Sınıfı Tanımlama:  
   
 class User(models.Model):  
 username = models.CharField(max\_length=150, unique=True)  
 email = models.EmailField()  
 first\_name = models.CharField(max\_length=30, blank=True)  
 is\_active = models.BooleanField(default=True)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
   
 class Meta:  
 db\_table = 'users'  
 ordering = ['-created\_at']  
   
 def \_\_str\_\_(self):  
 return self.username  
  
2. Field Tipleri:  
 • CharField: Kısa metin (max\_length zorunlu)  
 • TextField: Uzun metin  
 • IntegerField: Tam sayı  
 • BooleanField: True/False  
 • DateTimeField: Tarih ve saat  
 • EmailField: Email formatı  
 • ForeignKey: İlişki (başka model)  
  
3. Field Parametreleri:  
 • null=True: Database'de NULL olabilir  
 • blank=True: Form'da boş geçilebilir  
 • default: Varsayılan değer  
 • unique=True: Benzersiz olmalı  
 • db\_index=True: Index oluştur

────────────────────────────────────────────────────────────────────────────────

### ADIM 025: ✅ Model Validation Ekle

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 024 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Django validators kullan |

#### AÇIKLAMA:

clean() metodları ve field validationları ekle

#### NASIL YAPILIR:

1. Model Sınıfı Tanımlama:  
   
 class User(models.Model):  
 username = models.CharField(max\_length=150, unique=True)  
 email = models.EmailField()  
 first\_name = models.CharField(max\_length=30, blank=True)  
 is\_active = models.BooleanField(default=True)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
   
 class Meta:  
 db\_table = 'users'  
 ordering = ['-created\_at']  
   
 def \_\_str\_\_(self):  
 return self.username  
  
2. Field Tipleri:  
 • CharField: Kısa metin (max\_length zorunlu)  
 • TextField: Uzun metin  
 • IntegerField: Tam sayı  
 • BooleanField: True/False  
 • DateTimeField: Tarih ve saat  
 • EmailField: Email formatı  
 • ForeignKey: İlişki (başka model)  
  
3. Field Parametreleri:  
 • null=True: Database'de NULL olabilir  
 • blank=True: Form'da boş geçilebilir  
 • default: Varsayılan değer  
 • unique=True: Benzersiz olmalı  
 • db\_index=True: Index oluştur

────────────────────────────────────────────────────────────────────────────────

### ADIM 026: 📝 Model \_\_str\_\_ Metodları

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 023 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Admin panelde görünür |

#### AÇIKLAMA:

Her model için anlamlı \_\_str\_\_ metodu yaz

#### NASIL YAPILIR:

1. Model Sınıfı Tanımlama:  
   
 class User(models.Model):  
 username = models.CharField(max\_length=150, unique=True)  
 email = models.EmailField()  
 first\_name = models.CharField(max\_length=30, blank=True)  
 is\_active = models.BooleanField(default=True)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
   
 class Meta:  
 db\_table = 'users'  
 ordering = ['-created\_at']  
   
 def \_\_str\_\_(self):  
 return self.username  
  
2. Field Tipleri:  
 • CharField: Kısa metin (max\_length zorunlu)  
 • TextField: Uzun metin  
 • IntegerField: Tam sayı  
 • BooleanField: True/False  
 • DateTimeField: Tarih ve saat  
 • EmailField: Email formatı  
 • ForeignKey: İlişki (başka model)  
  
3. Field Parametreleri:  
 • null=True: Database'de NULL olabilir  
 • blank=True: Form'da boş geçilebilir  
 • default: Varsayılan değer  
 • unique=True: Benzersiz olmalı  
 • db\_index=True: Index oluştur

────────────────────────────────────────────────────────────────────────────────

### ADIM 027: 🗂️ İlk Migration Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 025 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Migration dosyalarını kontrol et |

#### AÇIKLAMA:

python manage.py makemigrations komutu çalıştır

#### NASIL YAPILIR:

1. Migration Dosyası Oluşturma:  
 python manage.py makemigrations  
   
2. Migration Dosyasını İnceleme:  
 migrations/0001\_initial.py dosyasını açın  
 Ne değiştiğini kontrol edin  
  
3. Migration Uygulama:  
 python manage.py migrate  
  
4. Migration Durumu:  
 python manage.py showmigrations  
  
5. Belirli App için Migration:  
 python manage.py makemigrations myapp  
 python manage.py migrate myapp  
  
6. Migration Geri Alma:  
 python manage.py migrate myapp 0001  
  
7. Fake Migration:  
 python manage.py migrate --fake

#### ÖNEMLİ UYARILAR:

⚠️ Migration dosyalarını silmeyin!  
⚠️ Migration dosyalarını manuel düzenlemeyin!  
⚠️ Production'da migration öncesi backup alın!  
⚠️ Migration conflict'ları dikkatli çözün!  
⚠️ Büyük tablolarda downtime olabileceğini düşünün!

────────────────────────────────────────────────────────────────────────────────

### ADIM 028: ⚡ Migration Uygula

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 027 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Hata mesajlarını dikkatle oku |

#### AÇIKLAMA:

python manage.py migrate komutu çalıştır

#### NASIL YAPILIR:

1. Migration Dosyası Oluşturma:  
 python manage.py makemigrations  
   
2. Migration Dosyasını İnceleme:  
 migrations/0001\_initial.py dosyasını açın  
 Ne değiştiğini kontrol edin  
  
3. Migration Uygulama:  
 python manage.py migrate  
  
4. Migration Durumu:  
 python manage.py showmigrations  
  
5. Belirli App için Migration:  
 python manage.py makemigrations myapp  
 python manage.py migrate myapp  
  
6. Migration Geri Alma:  
 python manage.py migrate myapp 0001  
  
7. Fake Migration:  
 python manage.py migrate --fake

#### ÖNEMLİ UYARILAR:

⚠️ Migration dosyalarını silmeyin!  
⚠️ Migration dosyalarını manuel düzenlemeyin!  
⚠️ Production'da migration öncesi backup alın!  
⚠️ Migration conflict'ları dikkatli çözün!  
⚠️ Büyük tablolarda downtime olabileceğini düşünün!

────────────────────────────────────────────────────────────────────────────────

### ADIM 029: 👑 Superuser Oluştur

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 028 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Güçlü şifre kullan |

#### AÇIKLAMA:

python manage.py createsuperuser komutu çalıştır

#### DİKKAT EDİLECEK NOKTALAR:

• Admin paneline giriş için  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 030: 🔧 Admin Panel Konfigürasyonu

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 029 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | list\_display, search\_fields ekle |

#### AÇIKLAMA:

admin.py dosyalarında model kayıtlarını yap

#### DİKKAT EDİLECEK NOKTALAR:

• Veri girişi kolaylaşır  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 031: 📊 Sample Data Ekleme

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1-2 gün |
| Bağımlılıklar | 030 |
| Kategori | 🗄️ Veritabanı |
| İpuçları | Gerçekçi veriler kullan |

#### AÇIKLAMA:

Fixture'lar veya management command ile test verisi ekle

#### DİKKAT EDİLECEK NOKTALAR:

• Development'ta çok yardımcı  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## BACKEND GELİŞTİRME

Django views, URLs ve API endpoint'lerinin geliştirilmesi.

### ADIM 032: 🗺️ URL Patterns Planla

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 006 |
| Kategori | 🌐 Backend |
| İpuçları | RESTful naming convention |

#### AÇIKLAMA:

Tüm sayfa URL'lerini ve API endpoint'lerini planla

#### DİKKAT EDİLECEK NOKTALAR:

• SEO dostu URL'ler  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 033: 🏠 Ana Sayfa View Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 032 |
| Kategori | 🌐 Backend |
| İpuçları | TemplateView veya function view |

#### AÇIKLAMA:

Home page için view fonksiyonu/sınıfı yaz

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

### ADIM 034: 🔗 URL Configuration

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 033 |
| Kategori | 🌐 Backend |
| İpuçları | App seviyesinde URL'ler |

#### AÇIKLAMA:

URLconf'ları (urls.py) yapılandır

#### DİKKAT EDİLECEK NOKTALAR:

• name parametreleri ekle  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 035: 📋 CRUD Views Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 5-7 gün |
| Bağımlılıklar | 031,034 |
| Kategori | 🌐 Backend |
| İpuçları | Class-based views kullan |

#### AÇIKLAMA:

Her model için Create, Read, Update, Delete view'ları

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

### ADIM 036: 🔐 Authentication Views

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 035 |
| Kategori | 🌐 Backend |
| İpuçları | Django'nun hazır view'larını kullan |

#### AÇIKLAMA:

Login, logout, register view'larını oluştur

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

### ADIM 037: 🛡️ Permission Kontrolü

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 036 |
| Kategori | 🌐 Backend |
| İpuçları | login\_required decorator |

#### AÇIKLAMA:

View'larda izin kontrollerini ekle

#### DİKKAT EDİLECEK NOKTALAR:

• Security için kritik  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 038: 📊 Pagination Ekle

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 035 |
| Kategori | 🌐 Backend |
| İpuçları | Django Paginator kullan |

#### AÇIKLAMA:

Uzun listelerde sayfa numaralandırma

#### DİKKAT EDİLECEK NOKTALAR:

• Performance için önemli  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 039: 🔍 Search Functionality

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 035 |
| Kategori | 🌐 Backend |
| İpuçları | Q objects kullan |

#### AÇIKLAMA:

Arama özelliği ekle

#### DİKKAT EDİLECEK NOKTALAR:

• User experience  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 040: 📱 API Views (Opsiyonel)

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 3-5 gün |
| Bağımlılıklar | 035 |
| Kategori | 🌐 Backend |
| İpuçları | Django REST Framework |

#### AÇIKLAMA:

REST API için DRF view'ları

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

## FRONTEND GELİŞTİRME

Template'ler, static files ve kullanıcı arayüzünün geliştirilmesi.

### ADIM 041: 📁 Template Klasör Yapısı

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 006 |
| Kategori | 🎨 Frontend |
| İpuçları | App seviyesinde organize et |

#### AÇIKLAMA:

templates klasörünü ve alt klasörlerini oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 042: 🏗️ Base Template Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 041 |
| Kategori | 🎨 Frontend |
| İpuçları | Block'ları doğru tanımla |

#### AÇIKLAMA:

base.html ana şablon dosyasını oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 043: 🎨 CSS Framework Entegrasyonu

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 042 |
| Kategori | 🎨 Frontend |
| İpuçları | CDN veya lokal kurulum |

#### AÇIKLAMA:

Bootstrap, Tailwind vb. framework ekle

#### DİKKAT EDİLECEK NOKTALAR:

• Responsive tasarım için  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 044: 🏠 Ana Sayfa Template

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 033,043 |
| Kategori | 🎨 Frontend |
| İpuçları | SEO meta tag'ları ekle |

#### AÇIKLAMA:

index.html ana sayfa şablonunu oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 045: 📝 Form Templates

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 3-4 gün |
| Bağımlılıklar | 035,044 |
| Kategori | 🎨 Frontend |
| İpuçları | CSRF token'ı unutma |

#### AÇIKLAMA:

Tüm form sayfalarının template'lerini oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 046: 📋 List Templates

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 035,038 |
| Kategori | 🎨 Frontend |
| İpuçları | Pagination template ekle |

#### AÇIKLAMA:

Listeleme sayfalarının template'lerini oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 047: 📄 Detail Templates

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 035,046 |
| Kategori | 🎨 Frontend |
| İpuçları | Breadcrumb ekle |

#### AÇIKLAMA:

Detay sayfalarının template'lerini oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 048: 🔐 Auth Templates

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 036,045 |
| Kategori | 🎨 Frontend |
| İpuçları | Error message'ları ekle |

#### AÇIKLAMA:

Login, register template'lerini oluştur

#### NASIL YAPILIR:

1. Template Klasör Yapısı:  
 templates/  
 ├── base.html  
 ├── home.html  
 └── users/  
 ├── list.html  
 └── detail.html  
  
2. Base Template (base.html):  
   
 <!DOCTYPE html>  
 <html>  
 <head>  
 <title>{% block title %}My Site{% endblock %}</title>  
 </head>  
 <body>  
 <nav>  
 <!-- Navigation -->  
 </nav>  
   
 <main>  
 {% block content %}  
 {% endblock %}  
 </main>  
 </body>  
 </html>  
  
3. Child Template:  
   
 {% extends 'base.html' %}  
   
 {% block title %}Home Page{% endblock %}  
   
 {% block content %}  
 <h1>Welcome!</h1>  
 {% for user in users %}  
 <p>{{ user.username }}</p>  
 {% endfor %}  
 {% endblock %}  
  
4. Template Tags:  
 • {{ variable }}: Değişken yazdırma  
 • {% tag %}: Mantık bloğu  
 • {% comment %}: Yorum  
 • {% load static %}: Static dosyalar

────────────────────────────────────────────────────────────────────────────────

### ADIM 049: 📱 Responsive Design

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 3-4 gün |
| Bağımlılıklar | 043-048 |
| Kategori | 🎨 Frontend |
| İpuçları | Media queries kullan |

#### AÇIKLAMA:

Mobil uyumlu tasarım yap

#### DİKKAT EDİLECEK NOKTALAR:

• Mobil kullanım artıyor  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 050: ⚡ JavaScript Functionality

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 049 |
| Kategori | 🎨 Frontend |
| İpuçları | jQuery veya Vanilla JS |

#### AÇIKLAMA:

İnteraktif özellikler ekle

#### DİKKAT EDİLECEK NOKTALAR:

• User experience artırır  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## FORM YÖNETİMİ VE VALİDASYON

Django forms kullanarak veri girişi ve validation işlemlerinin yapılandırılması.

### ADIM 051: 📝 Django Forms Oluştur

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 025,045 |
| Kategori | 📝 Forms |
| İpuçları | ModelForm kullan |

#### AÇIKLAMA:

Her model için form sınıfları oluştur

#### NASIL YAPILIR:

1. Model Form:  
   
 class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ['username', 'email', 'first\_name']  
 widgets = {  
 'username': forms.TextInput(attrs={'class': 'form-control'}),  
 'email': forms.EmailInput(attrs={'class': 'form-control'}),  
 }  
  
2. View'da Form Kullanımı:  
   
 def create\_user(request):  
 if request.method == 'POST':  
 form = UserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('user\_list')  
 else:  
 form = UserForm()  
   
 return render(request, 'create\_user.html', {'form': form})  
  
3. Template'te Form:  
   
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kaydet</button>  
 </form>  
  
4. Custom Validation:  
   
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError("Bu kullanıcı adı zaten alınmış!")  
 return username

────────────────────────────────────────────────────────────────────────────────

### ADIM 052: ✅ Form Validation

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 051 |
| Kategori | 📝 Forms |
| İpuçları | clean() metodları yaz |

#### AÇIKLAMA:

Form seviyesinde validation kuralları ekle

#### NASIL YAPILIR:

1. Model Form:  
   
 class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ['username', 'email', 'first\_name']  
 widgets = {  
 'username': forms.TextInput(attrs={'class': 'form-control'}),  
 'email': forms.EmailInput(attrs={'class': 'form-control'}),  
 }  
  
2. View'da Form Kullanımı:  
   
 def create\_user(request):  
 if request.method == 'POST':  
 form = UserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('user\_list')  
 else:  
 form = UserForm()  
   
 return render(request, 'create\_user.html', {'form': form})  
  
3. Template'te Form:  
   
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kaydet</button>  
 </form>  
  
4. Custom Validation:  
   
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError("Bu kullanıcı adı zaten alınmış!")  
 return username

────────────────────────────────────────────────────────────────────────────────

### ADIM 053: 🎨 Form Styling

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1-2 gün |
| Bağımlılıklar | 052 |
| Kategori | 📝 Forms |
| İpuçları | Widget'ları özelleştir |

#### AÇIKLAMA:

Form'ları güzel görünüm için stil ver

#### NASIL YAPILIR:

1. Model Form:  
   
 class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ['username', 'email', 'first\_name']  
 widgets = {  
 'username': forms.TextInput(attrs={'class': 'form-control'}),  
 'email': forms.EmailInput(attrs={'class': 'form-control'}),  
 }  
  
2. View'da Form Kullanımı:  
   
 def create\_user(request):  
 if request.method == 'POST':  
 form = UserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('user\_list')  
 else:  
 form = UserForm()  
   
 return render(request, 'create\_user.html', {'form': form})  
  
3. Template'te Form:  
   
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kaydet</button>  
 </form>  
  
4. Custom Validation:  
   
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError("Bu kullanıcı adı zaten alınmış!")  
 return username

────────────────────────────────────────────────────────────────────────────────

### ADIM 054: 💾 File Upload Forms

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 051 |
| Kategori | 📝 Forms |
| İpuçları | FileField/ImageField |

#### AÇIKLAMA:

Dosya yükleme form'larını ekle

#### NASIL YAPILIR:

1. Model Form:  
   
 class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ['username', 'email', 'first\_name']  
 widgets = {  
 'username': forms.TextInput(attrs={'class': 'form-control'}),  
 'email': forms.EmailInput(attrs={'class': 'form-control'}),  
 }  
  
2. View'da Form Kullanımı:  
   
 def create\_user(request):  
 if request.method == 'POST':  
 form = UserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('user\_list')  
 else:  
 form = UserForm()  
   
 return render(request, 'create\_user.html', {'form': form})  
  
3. Template'te Form:  
   
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kaydet</button>  
 </form>  
  
4. Custom Validation:  
   
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError("Bu kullanıcı adı zaten alınmış!")  
 return username

────────────────────────────────────────────────────────────────────────────────

### ADIM 055: 🔒 CSRF Protection

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 045 |
| Kategori | 📝 Forms |
| İpuçları | {% csrf\_token %} template'lerde |

#### AÇIKLAMA:

Tüm form'larda CSRF koruması kontrol et

#### DİKKAT EDİLECEK NOKTALAR:

• Security açığı önler  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## STATIC FILES VE MEDIA YÖNETİMİ

CSS, JavaScript ve media dosyalarının yönetimi.

### ADIM 056: 📁 Static Files Yapılandırma

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 022 |
| Kategori | 📁 Static |
| İpuçları | App seviyesinde static klasör |

#### AÇIKLAMA:

CSS, JS, images için static klasör yapısı

#### DİKKAT EDİLECEK NOKTALAR:

• Development/Production farkı  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 057: 🎨 CSS Dosyaları Oluştur

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 056 |
| Kategori | 📁 Static |
| İpuçları | Modüler CSS yaz |

#### AÇIKLAMA:

Custom CSS stillerini yaz

#### DİKKAT EDİLECEK NOKTALAR:

• Brand identity için  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 058: ⚡ JavaScript Dosyaları

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 057 |
| Kategori | 📁 Static |
| İpuçları | ES6 syntax kullan |

#### AÇIKLAMA:

Custom JavaScript kodlarını yaz

#### DİKKAT EDİLECEK NOKTALAR:

• İnteraktivite için  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 059: 🖼️ Media Files Konfigürasyonu

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 054 |
| Kategori | 📁 Static |
| İpuçları | MEDIA\_URL ve MEDIA\_ROOT |

#### AÇIKLAMA:

Kullanıcı yükleme dosyaları için media ayarları

#### DİKKAT EDİLECEK NOKTALAR:

• File upload için gerekli  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 060: 🗜️ Static Files Optimize

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 058 |
| Kategori | 📁 Static |
| İpuçları | Django Compressor |

#### AÇIKLAMA:

CSS/JS dosyalarını minimize et

#### DİKKAT EDİLECEK NOKTALAR:

• Performance için  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## GÜVENLİK VE İZİN YÖNETİMİ

Uygulama güvenliği ve kullanıcı yetki sistemlerinin kurulması.

### ADIM 061: 🔐 User Permission System

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 3 gün |
| Bağımlılıklar | 037 |
| Kategori | 🔒 Security |
| İpuçları | Django Groups kullan |

#### AÇIKLAMA:

Kullanıcı yetki sistemi kur

#### DİKKAT EDİLECEK NOKTALAR:

• Role-based access  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 062: 🛡️ Security Settings

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 022 |
| Kategori | 🔒 Security |
| İpuçları | HTTPS, SECURE\_\* settings |

#### AÇIKLAMA:

Production güvenlik ayarlarını yap

#### DİKKAT EDİLECEK NOKTALAR:

• Canlıda mutlaka gerekli  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 063: ⚡ Rate Limiting

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 036 |
| Kategori | 🔒 Security |
| İpuçları | django-ratelimit |

#### AÇIKLAMA:

Brute force saldırılarına karşı koruma

#### DİKKAT EDİLECEK NOKTALAR:

• API koruma  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 064: 🔍 SQL Injection Koruması

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 035 |
| Kategori | 🔒 Security |
| İpuçları | Raw SQL kullanma |

#### AÇIKLAMA:

ORM kullanımını kontrol et

#### DİKKAT EDİLECEK NOKTALAR:

• Güvenlik açığı  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 065: 🍪 Session Security

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 062 |
| Kategori | 🔒 Security |
| İpuçları | SESSION\_COOKIE\_\* settings |

#### AÇIKLAMA:

Oturum güvenliği ayarları

#### DİKKAT EDİLECEK NOKTALAR:

• Oturum hijacking önleme  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## TEST YAZMA VE KALİTE KONTROL

Unit test, integration test ve kalite kontrol süreçlerinin uygulanması.

### ADIM 066: 🧪 Unit Test Kurulumu

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 031 |
| Kategori | 🧪 Testing |
| İpuçları | Django TestCase kullan |

#### AÇIKLAMA:

Test ortamını ve framework'ünü hazırla

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 067: 📊 Model Tests

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 066 |
| Kategori | 🧪 Testing |
| İpuçları | Edge case'leri test et |

#### AÇIKLAMA:

Model sınıfları için test yaz

#### NASIL YAPILIR:

1. Model Sınıfı Tanımlama:  
   
 class User(models.Model):  
 username = models.CharField(max\_length=150, unique=True)  
 email = models.EmailField()  
 first\_name = models.CharField(max\_length=30, blank=True)  
 is\_active = models.BooleanField(default=True)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
   
 class Meta:  
 db\_table = 'users'  
 ordering = ['-created\_at']  
   
 def \_\_str\_\_(self):  
 return self.username  
  
2. Field Tipleri:  
 • CharField: Kısa metin (max\_length zorunlu)  
 • TextField: Uzun metin  
 • IntegerField: Tam sayı  
 • BooleanField: True/False  
 • DateTimeField: Tarih ve saat  
 • EmailField: Email formatı  
 • ForeignKey: İlişki (başka model)  
  
3. Field Parametreleri:  
 • null=True: Database'de NULL olabilir  
 • blank=True: Form'da boş geçilebilir  
 • default: Varsayılan değer  
 • unique=True: Benzersiz olmalı  
 • db\_index=True: Index oluştur

────────────────────────────────────────────────────────────────────────────────

### ADIM 068: 🌐 View Tests

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 3-4 gün |
| Bağımlılıklar | 067 |
| Kategori | 🧪 Testing |
| İpuçları | Response status code'ları |

#### AÇIKLAMA:

View fonksiyonları için test yaz

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

### ADIM 069: 📝 Form Tests

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 068 |
| Kategori | 🧪 Testing |
| İpuçları | Valid/invalid input test |

#### AÇIKLAMA:

Form validation'ları test et

#### NASIL YAPILIR:

1. Model Form:  
   
 class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ['username', 'email', 'first\_name']  
 widgets = {  
 'username': forms.TextInput(attrs={'class': 'form-control'}),  
 'email': forms.EmailInput(attrs={'class': 'form-control'}),  
 }  
  
2. View'da Form Kullanımı:  
   
 def create\_user(request):  
 if request.method == 'POST':  
 form = UserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('user\_list')  
 else:  
 form = UserForm()  
   
 return render(request, 'create\_user.html', {'form': form})  
  
3. Template'te Form:  
   
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kaydet</button>  
 </form>  
  
4. Custom Validation:  
   
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError("Bu kullanıcı adı zaten alınmış!")  
 return username

────────────────────────────────────────────────────────────────────────────────

### ADIM 070: 🔐 Authentication Tests

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 069 |
| Kategori | 🧪 Testing |
| İpuçları | Permission test'leri |

#### AÇIKLAMA:

Login/logout süreçlerini test et

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 071: 📊 Test Coverage Analizi

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 0.5 gün |
| Bağımlılıklar | 070 |
| Kategori | 🧪 Testing |
| İpuçları | coverage.py kullan |

#### AÇIKLAMA:

Test coverage raporunu çıkar

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

## PERFORMANS OPTİMİZASYONU

Uygulama performansının artırılması ve optimizasyon teknikleri.

### ADIM 072: 🗄️ Database Optimization

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 071 |
| Kategori | ⚡ Performance |
| İpuçları | Django Debug Toolbar |

#### AÇIKLAMA:

Veritabanı sorgularını optimize et

#### DİKKAT EDİLECEK NOKTALAR:

• N+1 problem çöz  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 073: 📊 Index Ekleme

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 072 |
| Kategori | ⚡ Performance |
| İpuçları | db\_index=True |

#### AÇIKLAMA:

Veritabanı indexlerini ekle

#### DİKKAT EDİLECEK NOKTALAR:

• Query performance  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 074: 💾 Caching Sistemi

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 073 |
| Kategori | ⚡ Performance |
| İpuçları | Redis veya Memcached |

#### AÇIKLAMA:

Cache framework'ünü entegre et

#### DİKKAT EDİLECEK NOKTALAR:

• Response time iyileştir  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 075: 🗜️ File Compression

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 060 |
| Kategori | ⚡ Performance |
| İpuçları | Whitenoise kullan |

#### AÇIKLAMA:

CSS/JS dosyalarını sıkıştır

#### DİKKAT EDİLECEK NOKTALAR:

• Load time azalt  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 076: 🖼️ Image Optimization

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 075 |
| Kategori | ⚡ Performance |
| İpuçları | Pillow ile resize |

#### AÇIKLAMA:

Resim dosyalarını optimize et

#### DİKKAT EDİLECEK NOKTALAR:

• Bandwidth tasarrufu  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## DEPLOYMENT VE CANLI YAYIN

Uygulamanın production ortamına deploy edilmesi.

### ADIM 077: 🐳 Docker Configuration

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 076 |
| Kategori | 🚀 Deployment |
| İpuçları | Multi-stage build |

#### AÇIKLAMA:

Docker container'ı hazırla

#### DİKKAT EDİLECEK NOKTALAR:

• Consistent deployment  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 078: ☁️ Production Settings

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 062 |
| Kategori | 🚀 Deployment |
| İpuçları | Environment variables |

#### AÇIKLAMA:

Canlı ortam ayarlarını hazırla

#### DİKKAT EDİLECEK NOKTALAR:

• Security için ayrı ayar  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 079: 📊 Database Migration

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 078 |
| Kategori | 🚀 Deployment |
| İpuçları | Data migration da düşün |

#### AÇIKLAMA:

Production migration planını yap

#### NASIL YAPILIR:

1. Migration Dosyası Oluşturma:  
 python manage.py makemigrations  
   
2. Migration Dosyasını İnceleme:  
 migrations/0001\_initial.py dosyasını açın  
 Ne değiştiğini kontrol edin  
  
3. Migration Uygulama:  
 python manage.py migrate  
  
4. Migration Durumu:  
 python manage.py showmigrations  
  
5. Belirli App için Migration:  
 python manage.py makemigrations myapp  
 python manage.py migrate myapp  
  
6. Migration Geri Alma:  
 python manage.py migrate myapp 0001  
  
7. Fake Migration:  
 python manage.py migrate --fake

#### ÖNEMLİ UYARILAR:

⚠️ Migration dosyalarını silmeyin!  
⚠️ Migration dosyalarını manuel düzenlemeyin!  
⚠️ Production'da migration öncesi backup alın!  
⚠️ Migration conflict'ları dikkatli çözün!  
⚠️ Büyük tablolarda downtime olabileceğini düşünün!

────────────────────────────────────────────────────────────────────────────────

### ADIM 080: 🔧 CI/CD Pipeline

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 3 gün |
| Bağımlılıklar | 021,071 |
| Kategori | 🚀 Deployment |
| İpuçları | Test → Build → Deploy |

#### AÇIKLAMA:

GitHub Actions workflow'unu yaz

#### DİKKAT EDİLECEK NOKTALAR:

• Otomatik deployment  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 081: 🌐 Domain & SSL

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 080 |
| Kategori | 🚀 Deployment |
| İpuçları | Let's Encrypt ücretsiz |

#### AÇIKLAMA:

Domain satın al ve SSL sertifikası kur

#### DİKKAT EDİLECEK NOKTALAR:

• HTTPS zorunlu  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 082: ☁️ Server Setup

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 081 |
| Kategori | 🚀 Deployment |
| İpuçları | DigitalOcean, AWS vb. |

#### AÇIKLAMA:

Production sunucusunu hazırla

#### DİKKAT EDİLECEK NOKTALAR:

• Resource planning yap  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 083: 🚀 First Deployment

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 082 |
| Kategori | 🚀 Deployment |
| İpuçları | Staging ortamında test et |

#### AÇIKLAMA:

İlk canlı deployment'ı yap

#### NASIL YAPILIR:

1. Production Settings:  
   
 # settings/production.py  
 DEBUG = False  
 ALLOWED\_HOSTS = ['yourdomain.com', 'www.yourdomain.com']  
   
 DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql',  
 'NAME': os.environ.get('DB\_NAME'),  
 'USER': os.environ.get('DB\_USER'),  
 'PASSWORD': os.environ.get('DB\_PASSWORD'),  
 'HOST': os.environ.get('DB\_HOST'),  
 'PORT': os.environ.get('DB\_PORT'),  
 }  
 }  
  
2. Static Files:  
 python manage.py collectstatic  
  
3. Gunicorn (WSGI Server):  
 pip install gunicorn  
 gunicorn myproject.wsgi:application  
  
4. Nginx Configuration:  
   
 server {  
 listen 80;  
 server\_name yourdomain.com;  
   
 location /static/ {  
 alias /path/to/static/files/;  
 }  
   
 location / {  
 proxy\_pass http://127.0.0.1:8000;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 }  
 }  
  
5. Environment Variables:  
 # .env file  
 SECRET\_KEY=your-secret-key  
 DEBUG=False  
 DB\_NAME=myproject  
 DB\_USER=postgres  
 DB\_PASSWORD=password

────────────────────────────────────────────────────────────────────────────────

## MONİTORING VE LOGGİNG

Uygulama izleme, log yönetimi ve error tracking.

### ADIM 084: 📊 Logging Setup

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 083 |
| Kategori | 📊 Monitoring |
| İpuçları | Python logging module |

#### AÇIKLAMA:

Uygulama log'larını yapılandır

#### DİKKAT EDİLECEK NOKTALAR:

• Debug için gerekli  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 085: ⚡ Performance Monitoring

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 084 |
| Kategori | 📊 Monitoring |
| İpuçları | Sentry, New Relic |

#### AÇIKLAMA:

Uygulama performansını izle

#### NASIL YAPILIR:

1. Model Form:  
   
 class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ['username', 'email', 'first\_name']  
 widgets = {  
 'username': forms.TextInput(attrs={'class': 'form-control'}),  
 'email': forms.EmailInput(attrs={'class': 'form-control'}),  
 }  
  
2. View'da Form Kullanımı:  
   
 def create\_user(request):  
 if request.method == 'POST':  
 form = UserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('user\_list')  
 else:  
 form = UserForm()  
   
 return render(request, 'create\_user.html', {'form': form})  
  
3. Template'te Form:  
   
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kaydet</button>  
 </form>  
  
4. Custom Validation:  
   
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError("Bu kullanıcı adı zaten alınmış!")  
 return username

────────────────────────────────────────────────────────────────────────────────

### ADIM 086: 🔔 Error Tracking

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 085 |
| Kategori | 📊 Monitoring |
| İpuçları | Sentry entegrasyonu |

#### AÇIKLAMA:

Hata takip sistemi kur

#### DİKKAT EDİLECEK NOKTALAR:

• Proaktif hata çözme  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 087: 💾 Backup System

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 086 |
| Kategori | 📊 Monitoring |
| İpuçları | Database + media files |

#### AÇIKLAMA:

Otomatik backup sistemi kur

#### DİKKAT EDİLECEK NOKTALAR:

• Data loss önleme  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 088: 📈 Analytics

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 087 |
| Kategori | 📊 Monitoring |
| İpuçları | Google Analytics |

#### AÇIKLAMA:

Kullanıcı davranışları analizi

#### DİKKAT EDİLECEK NOKTALAR:

• User insights  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## DOKÜMANTASYON

Teknik ve kullanıcı dokümantasyonunun hazırlanması.

### ADIM 089: 📚 API Documentation

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 040 |
| Kategori | 📚 Documentation |
| İpuçları | Swagger/OpenAPI |

#### AÇIKLAMA:

API endpoint'lerini dokümante et

#### DİKKAT EDİLECEK NOKTALAR:

• Developer experience  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 090: 📖 User Manual

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2-3 gün |
| Bağımlılıklar | 089 |
| Kategori | 📚 Documentation |
| İpuçları | Screenshot'lar ekle |

#### AÇIKLAMA:

Son kullanıcı kılavuzu yaz

#### DİKKAT EDİLECEK NOKTALAR:

• User adoption  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 091: 🔧 Technical Documentation

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 090 |
| Kategori | 📚 Documentation |
| İpuçları | Code comment'ları |

#### AÇIKLAMA:

Teknik dokümantasyon yaz

#### DİKKAT EDİLECEK NOKTALAR:

• Maintenance için  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 092: 🚀 Deployment Guide

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 091 |
| Kategori | 📚 Documentation |
| İpuçları | Step-by-step guide |

#### AÇIKLAMA:

Deployment talimatları yaz

#### NASIL YAPILIR:

1. Production Settings:  
   
 # settings/production.py  
 DEBUG = False  
 ALLOWED\_HOSTS = ['yourdomain.com', 'www.yourdomain.com']  
   
 DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql',  
 'NAME': os.environ.get('DB\_NAME'),  
 'USER': os.environ.get('DB\_USER'),  
 'PASSWORD': os.environ.get('DB\_PASSWORD'),  
 'HOST': os.environ.get('DB\_HOST'),  
 'PORT': os.environ.get('DB\_PORT'),  
 }  
 }  
  
2. Static Files:  
 python manage.py collectstatic  
  
3. Gunicorn (WSGI Server):  
 pip install gunicorn  
 gunicorn myproject.wsgi:application  
  
4. Nginx Configuration:  
   
 server {  
 listen 80;  
 server\_name yourdomain.com;  
   
 location /static/ {  
 alias /path/to/static/files/;  
 }  
   
 location / {  
 proxy\_pass http://127.0.0.1:8000;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 }  
 }  
  
5. Environment Variables:  
 # .env file  
 SECRET\_KEY=your-secret-key  
 DEBUG=False  
 DB\_NAME=myproject  
 DB\_USER=postgres  
 DB\_PASSWORD=password

────────────────────────────────────────────────────────────────────────────────

### ADIM 093: 📋 README.md Güncelle

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 092 |
| Kategori | 📚 Documentation |
| İpuçları | Installation guide |

#### AÇIKLAMA:

GitHub README'yi detaylandır

#### DİKKAT EDİLECEK NOKTALAR:

• Open source için önemli  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## KALİTE GÜVENCESİ VE TEST

Kapsamlı test süreçleri ve kalite kontrol işlemleri.

### ADIM 094: 🧪 Manual Testing

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 3-4 gün |
| Bağımlılıklar | 071 |
| Kategori | ✅ QA |
| İpuçları | Test case'leri hazırla |

#### AÇIKLAMA:

Tüm özellikleri manuel olarak test et

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 095: 📱 Cross-browser Testing

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 094 |
| Kategori | ✅ QA |
| İpuçları | Chrome, Firefox, Safari |

#### AÇIKLAMA:

Farklı browser'larda test et

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 096: 📱 Mobile Testing

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 095 |
| Kategori | ✅ QA |
| İpuçları | Responsive tasarım kontrol |

#### AÇIKLAMA:

Mobil cihazlarda test et

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 097: ⚡ Load Testing

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 096 |
| Kategori | ✅ QA |
| İpuçları | Apache Bench kullan |

#### AÇIKLAMA:

Yüksek trafik altında test et

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 098: 🔒 Security Testing

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 097 |
| Kategori | ✅ QA |
| İpuçları | OWASP Top 10 |

#### AÇIKLAMA:

Güvenlik açıklarını test et

#### NASIL YAPILIR:

1. Test Dosyası (tests.py):  
   
 from django.test import TestCase  
 from django.urls import reverse  
 from .models import User  
  
 class UserModelTest(TestCase):  
 def setUp(self):  
 self.user = User.objects.create(  
 username='testuser',  
 email='test@example.com'  
 )  
  
 def test\_user\_creation(self):  
 self.assertEqual(self.user.username, 'testuser')  
 self.assertTrue(self.user.is\_active)  
  
 def test\_str\_method(self):  
 self.assertEqual(str(self.user), 'testuser')  
  
2. View Test:  
   
 class UserViewTest(TestCase):  
 def test\_home\_page(self):  
 response = self.client.get(reverse('home'))  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, 'Welcome')  
  
3. Test Çalıştırma:  
 python manage.py test  
 python manage.py test myapp  
 python manage.py test myapp.tests.UserModelTest  
  
4. Coverage (Test Kapsamı):  
 pip install coverage  
 coverage run --source='.' manage.py test  
 coverage report  
 coverage html

────────────────────────────────────────────────────────────────────────────────

### ADIM 099: 🐛 Bug Fixes

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 3-5 gün |
| Bağımlılıklar | 098 |
| Kategori | ✅ QA |
| İpuçları | Priority sıralaması yap |

#### AÇIKLAMA:

Testte bulunan hataları düzelt

#### DİKKAT EDİLECEK NOKTALAR:

• Quality assurance  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## PROJE TAMAMLAMA VE LAUNCH

Projenin tamamlanması ve canlı yayına çıkış süreçleri.

### ADIM 100: 🚀 Final Deployment

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 099 |
| Kategori | 🎯 Launch |
| İpuçları | Backup aldıktan sonra |

#### AÇIKLAMA:

Son hali ile canlı yayın

#### NASIL YAPILIR:

1. Production Settings:  
   
 # settings/production.py  
 DEBUG = False  
 ALLOWED\_HOSTS = ['yourdomain.com', 'www.yourdomain.com']  
   
 DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql',  
 'NAME': os.environ.get('DB\_NAME'),  
 'USER': os.environ.get('DB\_USER'),  
 'PASSWORD': os.environ.get('DB\_PASSWORD'),  
 'HOST': os.environ.get('DB\_HOST'),  
 'PORT': os.environ.get('DB\_PORT'),  
 }  
 }  
  
2. Static Files:  
 python manage.py collectstatic  
  
3. Gunicorn (WSGI Server):  
 pip install gunicorn  
 gunicorn myproject.wsgi:application  
  
4. Nginx Configuration:  
   
 server {  
 listen 80;  
 server\_name yourdomain.com;  
   
 location /static/ {  
 alias /path/to/static/files/;  
 }  
   
 location / {  
 proxy\_pass http://127.0.0.1:8000;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 }  
 }  
  
5. Environment Variables:  
 # .env file  
 SECRET\_KEY=your-secret-key  
 DEBUG=False  
 DB\_NAME=myproject  
 DB\_USER=postgres  
 DB\_PASSWORD=password

────────────────────────────────────────────────────────────────────────────────

### ADIM 101: 📊 Post-launch Monitoring

|  |  |
| --- | --- |
| Öncelik | 🔴 Kritik |
| Tahmini Süre | 3 gün |
| Bağımlılıklar | 100 |
| Kategori | 🎯 Launch |
| İpuçları | Error rate'leri izle |

#### AÇIKLAMA:

Canlı yayın sonrası takip

#### DİKKAT EDİLECEK NOKTALAR:

• Immediate issue response  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 102: 📈 Performance Review

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 2 gün |
| Bağımlılıklar | 101 |
| Kategori | 🎯 Launch |
| İpuçları | Response time, load |

#### AÇIKLAMA:

Performance metrikleri analiz et

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

### ADIM 103: 📝 Project Retrospective

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 102 |
| Kategori | 🎯 Launch |
| İpuçları | Lessons learned |

#### AÇIKLAMA:

Proje retrospektifi yap

#### DİKKAT EDİLECEK NOKTALAR:

• Future project improvement  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 104: 🎉 Project Completion

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 1 gün |
| Bağımlılıklar | 103 |
| Kategori | 🎯 Launch |
| İpuçları | Final report hazırla |

#### AÇIKLAMA:

Proje tamamlama dokümantasyonu

#### DİKKAT EDİLECEK NOKTALAR:

• Official closure  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

## BAKIM VE DESTEK

Proje sonrası bakım, güncelleme ve destek süreçleri.

### ADIM 105: 🔄 Regular Updates

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | Sürekli |
| Bağımlılıklar | 104 |
| Kategori | 🔧 Maintenance |
| İpuçları | Security patch'leri |

#### AÇIKLAMA:

Düzenli güncelleme planı

#### DİKKAT EDİLECEK NOKTALAR:

• System maintenance  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 106: 📊 Analytics Review

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | Aylık |
| Bağımlılıklar | 105 |
| Kategori | 🔧 Maintenance |
| İpuçları | User behavior patterns |

#### AÇIKLAMA:

Kullanım analitikleri inceleme

#### NASIL YAPILIR:

1. Function-Based View:  
   
 def home(request):  
 if request.method == 'POST':  
 # POST işlemleri  
 pass  
   
 context = {  
 'title': 'Ana Sayfa',  
 'users': User.objects.all()  
 }  
 return render(request, 'home.html', context)  
  
2. Class-Based View:  
   
 class HomeView(TemplateView):  
 template\_name = 'home.html'  
   
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['users'] = User.objects.all()  
 return context  
  
3. URL Bağlama (urls.py):  
   
 urlpatterns = [  
 path('', views.home, name='home'),  
 path('home/', HomeView.as\_view(), name='home\_class'),  
 ]  
  
4. Generic Views:  
 • ListView: Liste görünümü  
 • DetailView: Detay görünümü   
 • CreateView: Oluşturma formu  
 • UpdateView: Güncelleme formu  
 • DeleteView: Silme onayı

────────────────────────────────────────────────────────────────────────────────

### ADIM 107: 🆕 Feature Requests

|  |  |
| --- | --- |
| Öncelik | 🟢 İyi Olur |
| Tahmini Süre | Sürekli |
| Bağımlılıklar | 106 |
| Kategori | 🔧 Maintenance |
| İpuçları | User feedback priority |

#### AÇIKLAMA:

Yeni özellik talepleri değerlendirme

#### DİKKAT EDİLECEK NOKTALAR:

• Product evolution  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

### ADIM 108: 🔒 Security Audits

|  |  |
| --- | --- |
| Öncelik | 🟡 Önemli |
| Tahmini Süre | 3 Aylık |
| Bağımlılıklar | 107 |
| Kategori | 🔧 Maintenance |
| İpuçları | Vulnerability scanning |

#### AÇIKLAMA:

Düzenli güvenlik denetimleri

#### DİKKAT EDİLECEK NOKTALAR:

• Security maintenance  
• Adımı tamamlamadan sonrakine geçmeyin  
• Hata alırsanız önce dokümantasyonu kontrol edin  
• Stack Overflow ve Django documentation'ı kullanın  
• Backup almayı unutmayın  
• Git commit'lerini düzenli yapın

────────────────────────────────────────────────────────────────────────────────

# 4. YAYGYN HATALAR VE ÇÖZÜMLER

Bu bölümde Django geliştirme sürecinde sık karşılaşılan hatalar ve çözüm yolları yer almaktadır.  
  
IMPORT HATALARI:  
❌ ModuleNotFoundError: No module named 'django'  
✅ Virtual environment aktif olup olmadığını kontrol edin  
  
❌ ImportError: cannot import name 'User' from 'django.contrib.auth.models'  
✅ Import statement'ı kontrol edin: from django.contrib.auth.models import User  
  
VERİTABANI HATALARI:  
❌ django.db.utils.OperationalError: no such table  
✅ Migration'ları çalıştırın: python manage.py migrate  
  
❌ UNIQUE constraint failed  
✅ Duplicate veri var, önceki veriyi silin veya unique=False yapın  
  
URL HATALARI:  
❌ NoReverseMatch at / Reverse for 'home' not found  
✅ urls.py'da 'home' adında bir URL pattern olup olmadığını kontrol edin  
  
❌ Page not found (404)  
✅ URL pattern'i doğru yazılmış mı kontrol edin  
  
TEMPLATE HATALARI:  
❌ TemplateDoesNotExist at /  
✅ Template dosyasının doğru yerde olup olmadığını kontrol edin  
  
❌ Invalid block tag: 'endfor'  
✅ {% for %} ve {% endfor %} eşleşmesini kontrol edin  
  
STATIC FILES HATALARI:  
❌ Static files yüklenmiyor  
✅ python manage.py collectstatic çalıştırın  
✅ STATIC\_URL ve STATIC\_ROOT ayarlarını kontrol edin  
  
GÜVENLİK HATALARI:  
❌ Forbidden (CSRF token missing)  
✅ Form'larda {% csrf\_token %} eklemeyi unutmayın  
  
❌ DisallowedHost at /  
✅ ALLOWED\_HOSTS ayarını kontrol edin

# 5. İPUÇLARI VE BEST PRACTICES

Bu bölümde Django development sürecini hızlandıracak ve kaliteyi artıracak ipuçları bulacaksınız.  
  
GELİŞTİRME İPUÇLARI:  
• Django Debug Toolbar kullanın - SQL query'leri görebilirsiniz  
• Shell kullanmayı öğrenin: python manage.py shell  
• Django admin'i customize edin - development sürecini hızlandırır  
• Fixture'lar ile test data oluşturun  
• Management command'ları yazın tekrarlayan işler için  
• django-extensions paketini kurun (shell\_plus, reset\_db vs.)  
  
KOD KALİTESİ:  
• PEP 8 standartlarına uyun  
• Black code formatter kullanın  
• flake8 ile linting yapın  
• Type hints kullanın (Python 3.5+)  
• Docstring yazın fonksiyonlar için  
• Magic number kullanmayın, constant tanımlayın  
  
PERFORMANs:  
• select\_related() ve prefetch\_related() kullanın N+1 problem için  
• Database index'leri ekleyin sık sorgulanan alanlara  
• Caching kullanın (Redis önerilir)  
• pagination kullanın uzun listeler için  
• Static files için CDN kullanın  
• Database query'leri optimize edin  
  
GÜVENLİK:  
• Hiçbir zaman DEBUG=True ile production'a çıkmayın  
• SECRET\_KEY'i güvenli tutun ve rotate edin  
• HTTPS kullanın production'da  
• User input'larını validate edin  
• SQL injection'a karşı ORM kullanın  
• XSS'e karşı template escaping aktif olsun  
  
DEPLOYMENT:  
• Environment variables kullanın hassas bilgiler için  
• Separate settings dosyaları kullanın (dev/staging/prod)  
• Automated deployment setup edin (CI/CD)  
• Database backup stratejinizi belirleyin  
• Monitoring ve logging sistemini kurun  
• Health check endpoint'i ekleyin  
  
PROJE YÖNETİMİ:  
• Git flow kullanın  
• Meaningful commit message'ları yazın  
• Branch protection rules ekleyin  
• Code review sürecini uygulayın  
• Documentation güncel tutun  
• Issue tracking kullanın (GitHub Issues, Jira)  
  
TEST STRATEGY:  
• Test-driven development (TDD) approach'u deneyin  
• Unit test coverage'ı %80+ tutun  
• Integration test yazın kritik flow'lar için  
• Factory boy kullanın test data için  
• Continuous integration setup edin  
• Load testing yapın production öncesi  
  
ÖĞRENİM KAYNAKLARI:  
• Django documentation (docs.djangoproject.com)  
• Django Girls Tutorial  
• Two Scoops of Django (kitap)  
• Classy Class-Based Views (ccbv.co.uk)  
• Django packages (djangopackages.org)  
• Real Python Django tutorials  
  
VERİMLİLİK:  
• IDE shortcuts öğrenin (VS Code/PyCharm)  
• Terminal shortcuts kullanın  
• Snippet'lar oluşturun sık kullanılan kod parçaları için  
• Multiple monitor setup yapın mümkünse  
• Pomodoro tekniği kullanın focus için  
• Regular break'ler alın burnout'u önlemek için

# 6. KAYNAKLAR VE REFERANSLAR

Django geliştirme sürecinde ihtiyaç duyabileceğiniz kaynaklar:  
  
RESMİ DOKÜMANTASYON:  
• Django Documentation: https://docs.djangoproject.com/  
• Django REST Framework: https://www.django-rest-framework.org/  
• Django Tutorial: https://docs.djangoproject.com/en/stable/intro/tutorial01/  
  
ÖĞRETİM KAYNAKLARI:  
• Django Girls Tutorial: https://tutorial.djangogirls.org/  
• Mozilla Django Tutorial: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django  
• Real Python Django: https://realpython.com/tutorials/django/  
• Django for Beginners (kitap): William S. Vincent  
• Two Scoops of Django (kitap): Daniel ve Audrey Feldroy  
  
ARAÇLAR VE PAKETLER:  
• Django Debug Toolbar: Debug ve profiling  
• Django Extensions: Faydalı management command'ları  
• Pillow: Image processing  
• Celery: Background tasks  
• Redis: Caching ve message broker  
• Gunicorn: WSGI server  
• Nginx: Web server ve reverse proxy  
  
TEST ARAÇLARI:  
• Factory Boy: Test data generation  
• Pytest-django: Gelişmiş test framework  
• Coverage.py: Test coverage ölçümü  
• Selenium: Browser automation  
  
DEPLOYMENT PLATFORMLARI:  
• Heroku: Kolay deployment  
• DigitalOcean: VPS hosting  
• AWS: Cloud services  
• PythonAnywhere: Python hosting  
• Railway: Modern deployment platform  
  
CODE QUALITY:  
• Black: Code formatter  
• flake8: Linting  
• isort: Import sorting  
• pre-commit: Git hooks  
• mypy: Type checking  
  
VERİTABANI ARAÇLARI:  
• PostgreSQL: Production database  
• pgAdmin: PostgreSQL GUI  
• DBeaver: Universal database tool  
• Redis CLI: Redis command line  
  
MONİTORING:  
• Sentry: Error tracking  
• New Relic: Performance monitoring  
• DataDog: Infrastructure monitoring  
• Uptime Robot: Uptime monitoring  
  
TOPLULUK KAYNAKLARI:  
• Django Forum: https://forum.djangoproject.com/  
• r/django (Reddit): https://reddit.com/r/django  
• Django Discord: Django community chat  
• Stack Overflow: Programming Q&A  
• GitHub: Open source Django projects  
  
BLOGLAR VE NEWSLETTER:  
• Django News: Weekly newsletter  
• Real Python: Python tutorials  
• Planet Django: Django blog aggregator  
• Django Tricks: Tips and tricks  
• Simple is Better Than Complex: Django tutorials  
  
GÜVENLİK KAYNAKLARI:  
• OWASP: Web application security  
• Django Security Documentation  
• Python Security: Security best practices  
• Security Headers: HTTP security headers  
  
Bu kaynakları bookmark'layın ve ihtiyaç duyduğunuzda başvurun. Django topluluğu çok yardımseverdir,   
sorularınızı forum ve Discord kanallarında sorabilirsiniz.