🚀 DJANGO PROJESİ GELİŞTİRME KILAVUZU - BÖLÜM 4

# 🎯 Uygulama Geliştirme ve Deployment

# 🔗 6. VIEWS VE URL YAPISINI KURGULA

## 🎯 Ne Yapacaksın?

Kullanıcı isteklerini karşılayacak view fonksiyonları ve URL yapısını oluştur.

### 📝 Yapılacaklar:

* • URL patterns tasarla (ana site, admin, api)
* • Function-based veya Class-based views seç
* • View fonksiyonlarını yaz
* • URL dispatcher oluştur
* • Template yapısını planla
* • Forms oluştur

### 💡 Örnek URL Yapısı:

|  |  |  |
| --- | --- | --- |
| URL Pattern | View | Açıklama |
| / | HomeView | Ana sayfa |
| /products/ | ProductListView | Ürün listesi |
| /products/<slug>/ | ProductDetailView | Ürün detay |
| /cart/ | CartView | Sepet görüntüle |
| /cart/add/ | AddToCartView | Sepete ekle |
| /checkout/ | CheckoutView | Ödeme sayfası |
| /orders/ | OrderListView | Sipariş geçmişi |
| /auth/login/ | LoginView | Giriş yap |
| /auth/register/ | RegisterView | Kayıt ol |
| /admin/ | admin.site.urls | Yönetim paneli |

# 🧪 7. TEST YAZIMI VE KALİTE KONTROL

## 🎯 Ne Yapacaksın?

Kodunun doğru çalıştığından emin olmak için kapsamlı testler yaz.

### 📝 Test Türleri:

* • Unit Tests (Model testleri)
* • Integration Tests (View testleri)
* • Functional Tests (Selenium ile)
* • API Tests (DRF için)
* • Performance Tests

### 💡 Örnek Test Kodu:

from django.test import TestCase, Client  
from django.contrib.auth.models import User  
from django.urls import reverse  
from myapp.models import Product, Category  
  
class ProductModelTest(TestCase):  
 def setUp(self):  
 self.category = Category.objects.create(  
 name="Test Kategori",  
 slug="test-kategori"  
 )  
 self.product = Product.objects.create(  
 name="Test Ürün",  
 slug="test-urun",  
 description="Test açıklaması",  
 price=99.99,  
 stock\_quantity=10,  
 category=self.category  
 )  
   
 def test\_product\_creation(self):  
 self.assertEqual(self.product.name, "Test Ürün")  
 self.assertTrue(self.product.is\_in\_stock())  
   
 def test\_product\_str\_representation(self):  
 self.assertEqual(str(self.product), "Test Ürün")  
  
class ProductViewTest(TestCase):  
 def setUp(self):  
 self.client = Client()  
 self.category = Category.objects.create(  
 name="Test Kategori",  
 slug="test-kategori"  
 )  
 self.product = Product.objects.create(  
 name="Test Ürün",  
 slug="test-urun",  
 description="Test açıklaması",  
 price=99.99,  
 category=self.category  
 )  
   
 def test\_product\_list\_view(self):  
 response = self.client.get('/products/')  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, "Test Ürün")  
   
 def test\_product\_detail\_view(self):  
 url = reverse('product\_detail', args=[self.product.slug])  
 response = self.client.get(url)  
 self.assertEqual(response.status\_code, 200)  
 self.assertContains(response, self.product.name)  
  
# Test çalıştırma komutları:  
# python manage.py test # Tüm testleri çalıştır  
# python manage.py test myapp # Belirli app testleri  
# python manage.py test myapp.tests.ProductModelTest # Belirli test sınıfı  
# coverage run --source='.' manage.py test # Coverage ile test

# 🚀 8. DEPLOYMENT VE YAYIN

## 🎯 Ne Yapacaksın?

Projeyi canlı sunucuya yayınla ve kullanıcıların erişimine aç.

### 📝 Deployment Adımları:

#### 🔹 1. Production Ayarları:

* • DEBUG = False yap
* • SECRET\_KEY'i çevre değişkeninde tut
* • ALLOWED\_HOSTS ayarla
* • Static files yapılandırması
* • Database ayarları (PostgreSQL)
* • Email backend ayarları

#### 🔹 2. Server Seçenekleri:

* • Heroku (Kolay başlangıç)
* • DigitalOcean (Güçlü ve ekonomik)
* • AWS (Kurumsal çözüm)
* • VPS (Kendi sunucun)

#### 🔹 3. Heroku Deployment:

# 1. Gerekli dosyaları oluştur:  
# requirements.txt  
pip freeze > requirements.txt  
  
# Procfile  
echo "web: gunicorn myproject.wsgi" > Procfile  
  
# runtime.txt  
echo "python-3.11.0" > runtime.txt  
  
# 2. Git repository hazırla:  
git init  
git add .  
git commit -m "Initial commit"  
  
# 3. Heroku CLI ile deploy:  
heroku create myproject-name  
heroku addons:create heroku-postgresql:mini  
heroku config:set SECRET\_KEY="your-secret-key"  
heroku config:set DEBUG=False  
git push heroku main  
heroku run python manage.py migrate  
heroku run python manage.py collectstatic --noinput  
heroku run python manage.py createsuperuser

# 📊 9. İZLEME VE BAKIM

## 🎯 Ne Yapacaksın?

Projeyi sürekli izle, güncellemeler yap ve sorunları çöz.

### 📝 İzleme Araçları:

* • Error tracking (Sentry)
* • Performance monitoring (New Relic)
* • Uptime monitoring (UptimeRobot)
* • Log analizi (Papertrail)
* • User analytics (Google Analytics)

### 📝 Bakım Görevleri:

* • Güvenlik güncellemeleri
* • Database optimizasyonu
* • Backup stratejisi
* • Cache stratejisi
* • CDN kullanımı

# ✅ 10. PROJE TAMAMLAMA CHECKLİSTİ

### 🔍 Kalite Kontrol:

* ☐ Tüm testler geçiyor
* ☐ Code coverage %80 üzeri
* ☐ Security scan temiz
* ☐ Performance testleri yapıldı
* ☐ Cross-browser uyumlu
* ☐ Mobile responsive
* ☐ SEO optimizasyonu
* ☐ Accessibility (WCAG) uyumu
* ☐ Error handling tamamlandı
* ☐ Logging yapılandırıldı

### 📚 Dokümantasyon:

* ☐ README.md dosyası
* ☐ API dokümantasyonu
* ☐ Kurulum kılavuzu
* ☐ Kullanıcı kılavuzu
* ☐ Developer dokümantasyonu
* ☐ Deployment kılavuzu
* ☐ Troubleshooting rehberi
* ☐ Changelog dosyası
* ☐ License bilgisi
* ☐ Contributing guidelines

# 🎉 TEBRİKLER!

Django projen başarıyla tamamlandı! Artık profesyonel bir web uygulamasına sahipsin. Unutma: Geliştirme süreci hiç bitmez. Kullanıcı geri bildirimlerini dinle, sürekli iyileştirmeler yap ve teknolojik gelişmeleri takip et.

### 🔄 Sonraki Adımlar:

* • Kullanıcı geri bildirimlerini topla
* • Analytics verilerini analiz et
* • Yeni özellikler planla
* • Teknik borçları temizle
* • Kod refactoring yap
* • Yeni teknolojileri araştır