🚀 DJANGO PROJESİ GELİŞTİRME KILAVUZU - BÖLÜM 3

# 🗄️ Veritabanı Tasarımı ve Model Oluşturma

# 🗄️ 4. VERİTABANI ŞEMASI TASARLA

## 🎯 Ne Yapacaksın?

Projenin tüm veri ihtiyaçlarını analiz et, tabloları ve ilişkileri tasarla.

### 📝 Yapılacaklar:

* • Entity-Relationship Diagram (ERD) çiz
* • Ana varlıkları (entities) belirle
* • Tablolar arası ilişkileri tanımla (1:1, 1:N, N:N)
* • Her tablonun alanlarını listele
* • Primary key ve foreign key'leri belirle
* • Normalizasyon kurallarını uygula

### 💡 Örnek Proje: E-Ticaret Veritabanı

#### 🔹 Ana Tablolar:

|  |  |  |
| --- | --- | --- |
| Tablo Adı | Açıklama | Ana Alanlar |
| User | Kullanıcı bilgileri | id, username, email, password, first\_name, last\_name, is\_active |
| Category | Ürün kategorileri | id, name, slug, description, parent\_category |
| Product | Ürün bilgileri | id, name, slug, description, price, stock, category, image |
| Cart | Alışveriş sepeti | id, user, created\_at, updated\_at |
| CartItem | Sepet ürünleri | id, cart, product, quantity, price |
| Order | Siparişler | id, user, total\_amount, status, created\_at, shipping\_address |
| OrderItem | Sipariş kalemleri | id, order, product, quantity, price |
| Address | Adres bilgileri | id, user, title, address\_line, city, country, postal\_code |

#### 🔹 İlişki Türleri:

* • User → Address (1:N) - Bir kullanıcının birden fazla adresi olabilir
* • Category → Product (1:N) - Bir kategoride birden fazla ürün olabilir
* • User → Cart (1:1) - Her kullanıcının bir sepeti var
* • Cart → CartItem (1:N) - Sepette birden fazla ürün olabilir
* • User → Order (1:N) - Kullanıcının birden fazla siparişi olabilir
* • Order → OrderItem (1:N) - Siparişte birden fazla ürün olabilir

# 🏗️ 5. DJANGO MODELLERİNİ OLUŞTUR

## 🎯 Ne Yapacaksın?

Tasarladığın veritabanı şemasını Django modellerine dönüştür.

### 📝 Yapılacaklar:

* • Her app için models.py dosyasında model sınıflarını yaz
* • Doğru field tiplerini seç
* • İlişkileri doğru şekilde tanımla (ForeignKey, ManyToMany, OneToOne)
* • Meta sınıflarını ekle (verbose\_name, ordering, db\_table)
* • \_\_str\_\_ metodlarını yaz
* • Validation metodları ekle

### 💡 Örnek Model: Product

from django.db import models  
from django.urls import reverse  
from django.contrib.auth.models import User  
  
class Category(models.Model):  
 name = models.CharField(max\_length=100, verbose\_name="Kategori Adı")  
 slug = models.SlugField(max\_length=100, unique=True)  
 description = models.TextField(blank=True, verbose\_name="Açıklama")  
 parent = models.ForeignKey('self', on\_delete=models.CASCADE,   
 null=True, blank=True,   
 related\_name='children')  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
   
 class Meta:  
 verbose\_name = "Kategori"  
 verbose\_name\_plural = "Kategoriler"  
 ordering = ['name']  
   
 def \_\_str\_\_(self):  
 return self.name  
  
class Product(models.Model):  
 STOCK\_STATUS = [  
 ('in\_stock', 'Stokta'),  
 ('out\_of\_stock', 'Stok Yok'),  
 ('discontinued', 'Üretim Durduruldu'),  
 ]  
   
 name = models.CharField(max\_length=200, verbose\_name="Ürün Adı")  
 slug = models.SlugField(max\_length=200, unique=True)  
 description = models.TextField(verbose\_name="Ürün Açıklaması")  
 price = models.DecimalField(max\_digits=10, decimal\_places=2,   
 verbose\_name="Fiyat")  
 stock\_quantity = models.PositiveIntegerField(default=0,   
 verbose\_name="Stok Miktarı")  
 stock\_status = models.CharField(max\_length=20,   
 choices=STOCK\_STATUS,  
 default='in\_stock')  
 category = models.ForeignKey(Category, on\_delete=models.CASCADE,  
 related\_name='products')  
 image = models.ImageField(upload\_to='products/',   
 blank=True, null=True)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
 updated\_at = models.DateTimeField(auto\_now=True)  
 is\_active = models.BooleanField(default=True)  
   
 class Meta:  
 verbose\_name = "Ürün"  
 verbose\_name\_plural = "Ürünler"  
 ordering = ['-created\_at']  
 indexes = [  
 models.Index(fields=['slug']),  
 models.Index(fields=['category']),  
 models.Index(fields=['price']),  
 ]  
   
 def \_\_str\_\_(self):  
 return self.name  
   
 def get\_absolute\_url(self):  
 return reverse('product\_detail', args=[self.slug])  
   
 def is\_in\_stock(self):  
 return self.stock\_quantity > 0 and self.stock\_status == 'in\_stock'