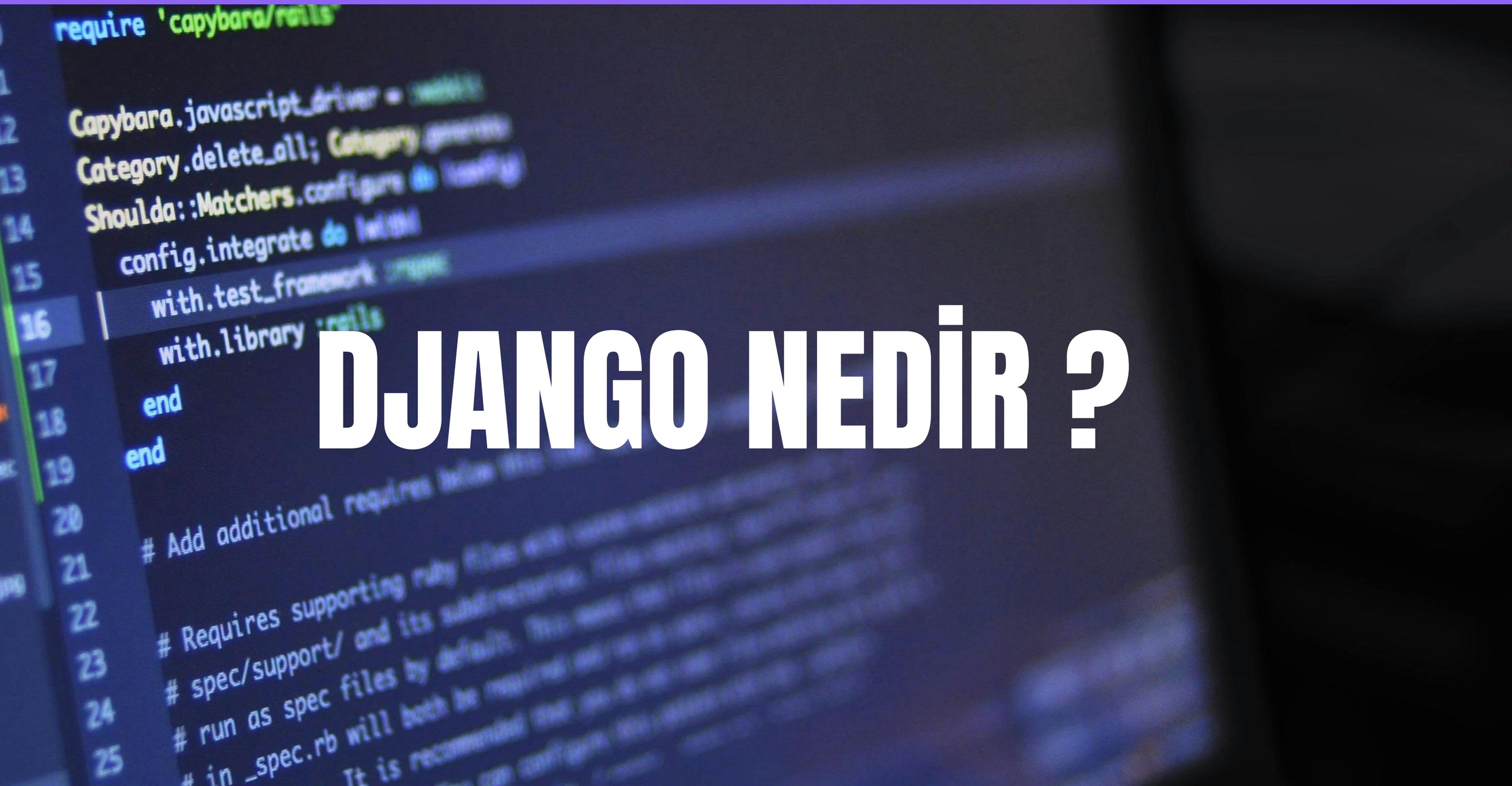
Django İle Python Arka Yüz Yazılım Geliştirme

Teknik Bilimler
Meslek Yüksekokulu

Öğr. Gör. Furkan DURMUŞ

Bu Haftanın Ders Kazanımları

- Django nedir?
- MVT mimarisi
- Django'nun kullanım alanları
- Django'nun genel yapısı
- Web nedir? HTTP temelleri

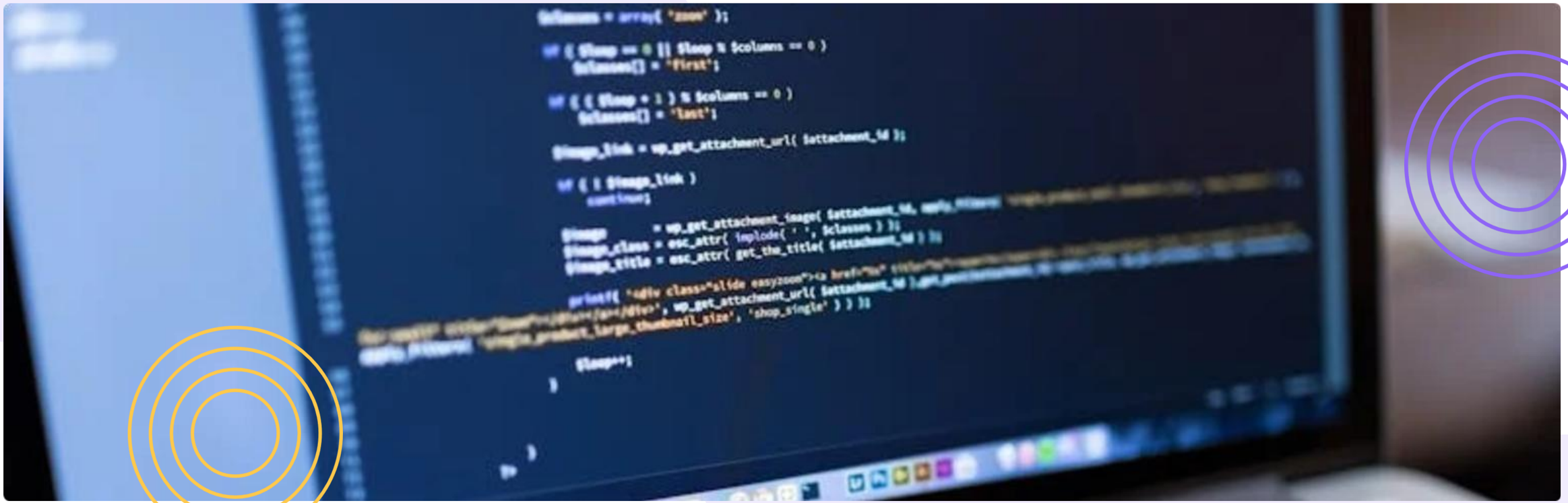


DJANGO NEDİR ?

Django nedir?

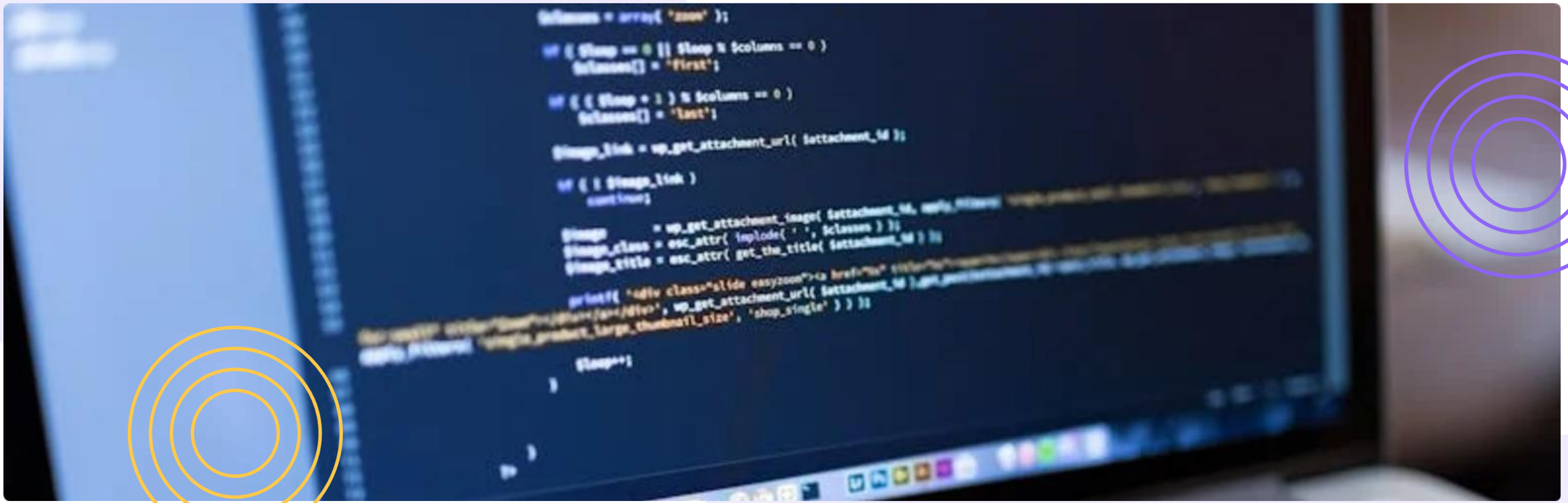
Bu derste sıfırdan bir web uygulaması geliştirmeyi öğreneceğiz. Ama önce şunu anlamamız gerekiyor: Web uygulaması nedir ve Django bunun neresinde?

Web sitesi nedir?



Web sitesi nedir?

Web sitesi, internet üzerinden erişilen ve bir sunucu üzerinde çalışan yazılım sistemidir.



Web sitesi nedir?

Web sitesi, internet üzerinden erişilen ve bir sunucu üzerinde çalışan yazılım sistemidir.

Bilgi sunmaya odaklıdır.

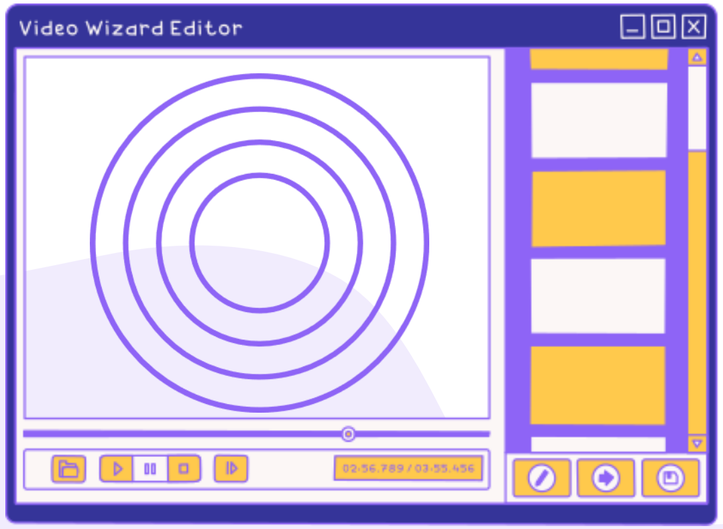
Kullanıcı genellikle:

- İçerik okur
- Sayfalar arasında gezinir
- Form doldurabilir (iletişim formu gibi)

Web sitesi nedir?

Web sitesi, internet üzerinden erişilen ve bir sunucu üzerinde çalışan yazılım sistemidir.

Bilgi sunmaya odaklıdır.

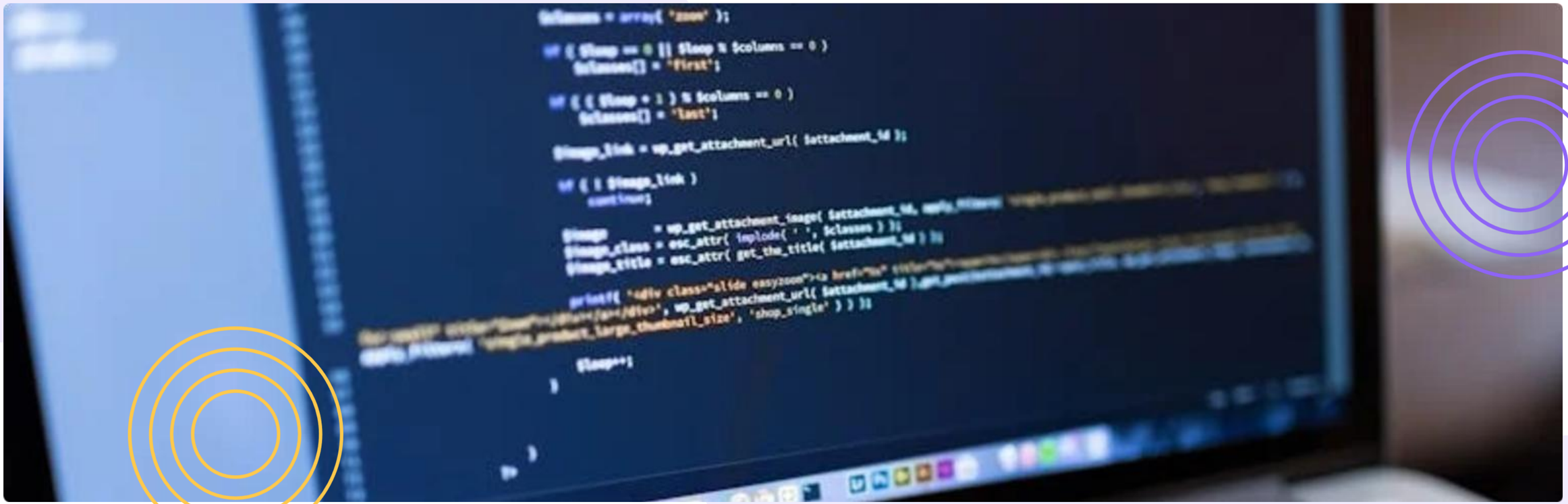


Özellikleri:

- Daha statiktir.
- İçerik ağırlıklıdır.
- Etkileşim sınırlıdır.
- Genelde tanıtım, kurumsal kimlik veya blog amaçlıdır.

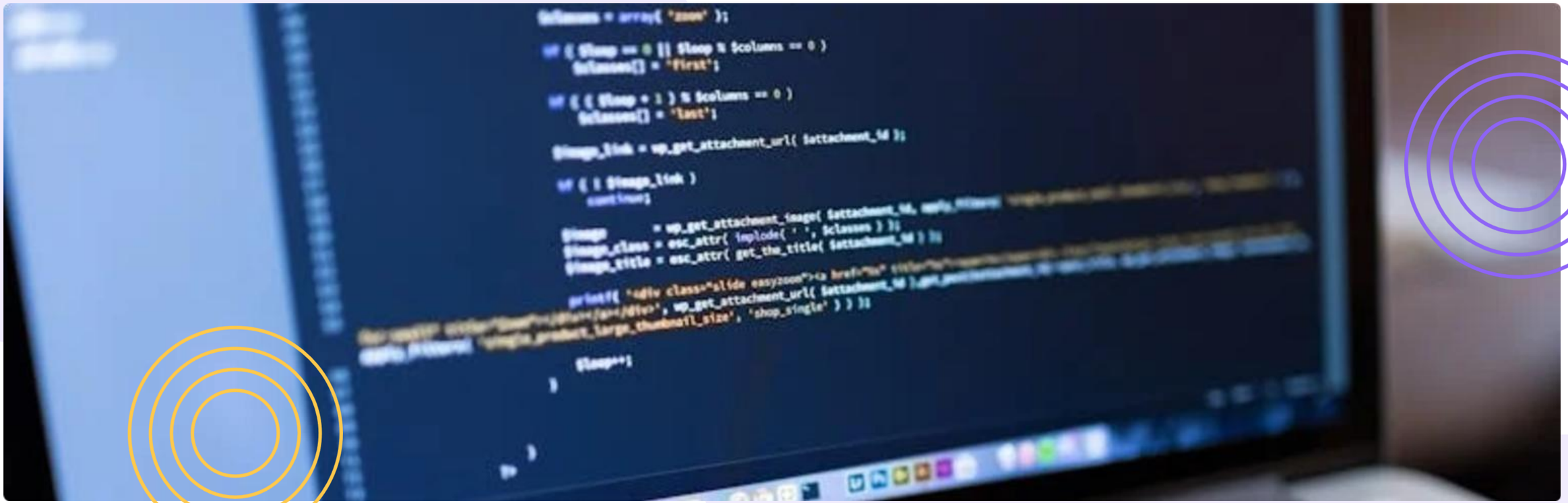
Örnekler: Kurumsal firma sitesi - Blog sitesi - Haber sitesi - Portföy sitesi

Web uygulaması nedir?



Web uygulaması nedir?

Kullanıcıyla aktif etkileşim kurar ve işlem yaptırır.



Web uygulaması nedir?

Kullanıcıyla aktif etkileşim kurar ve işlem yaptırır.

Kullanıcı genellikle:

- Giriş yapar
- Veri girer
- Veri kaydeder
- İşlem yapar
- Kendi hesabını yönetir

Web uygulaması nedir?

Kullanıcıyla aktif etkileşim kurar ve işlem yaptırır.

Özellikleri:

- Dinamiktir.
- Backend + veritabanı içerir.
- Kullanıcı hesapları vardır.
- İş mantığı (business logic) barındırır.

Örnekler ?

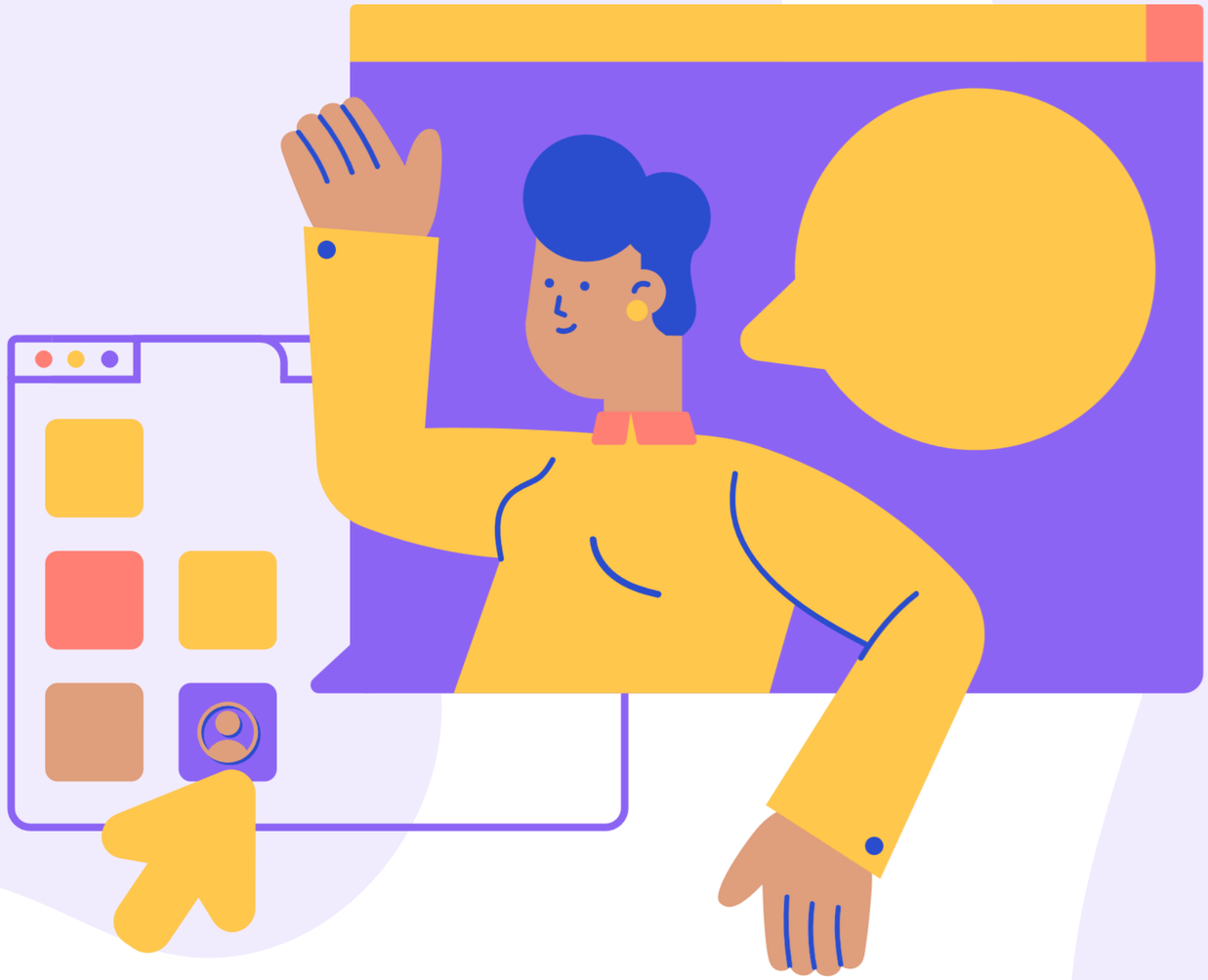
Web uygulaması nedir?

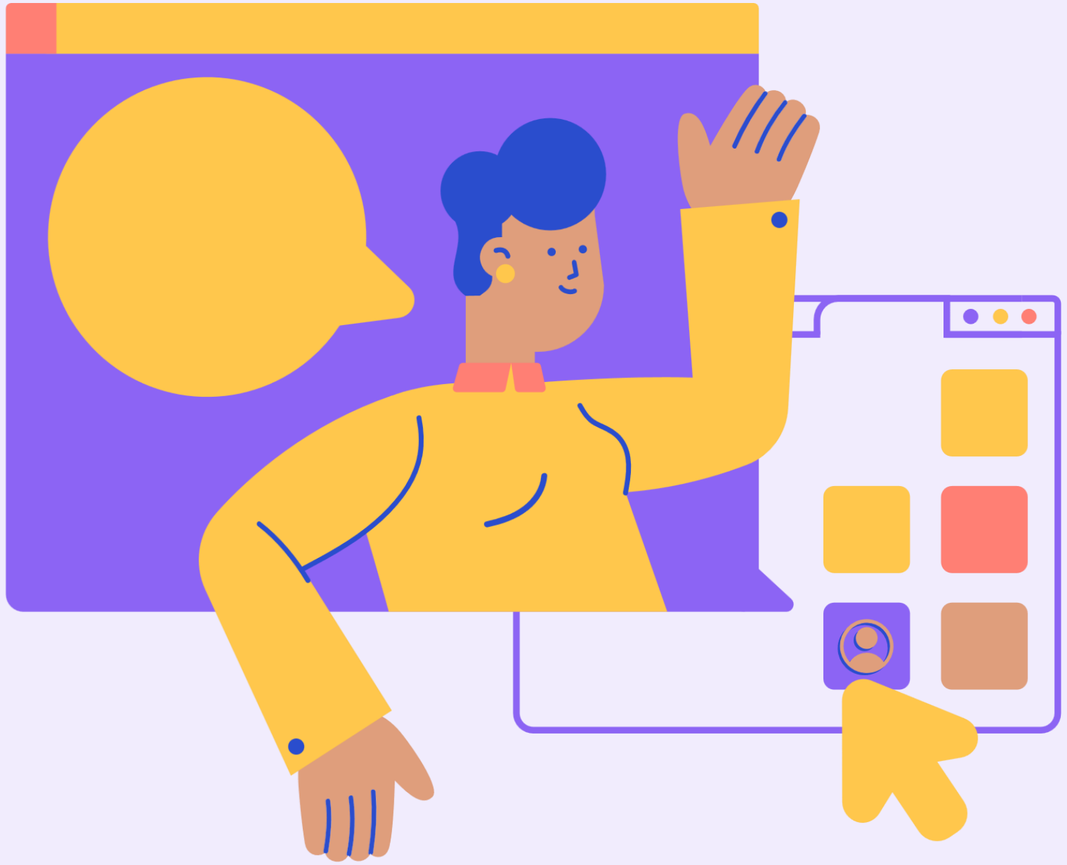
Kullanıcıyla aktif etkileşim kurar ve işlem yaptırır.

Örnekler :

- Gmail
- E-devlet
- Online bankacılık

Instagram web sitesi mi?





Bir web sitesinde veri nerede tutulur?

İstemci-Sunucu Mimarisi

Kullanıcı

Kullanıcı istek gönderir.

Tarayıcı

Sunucu

Sunucu işler.

Veri Tabanı

Gerekirse veritabanına gider.

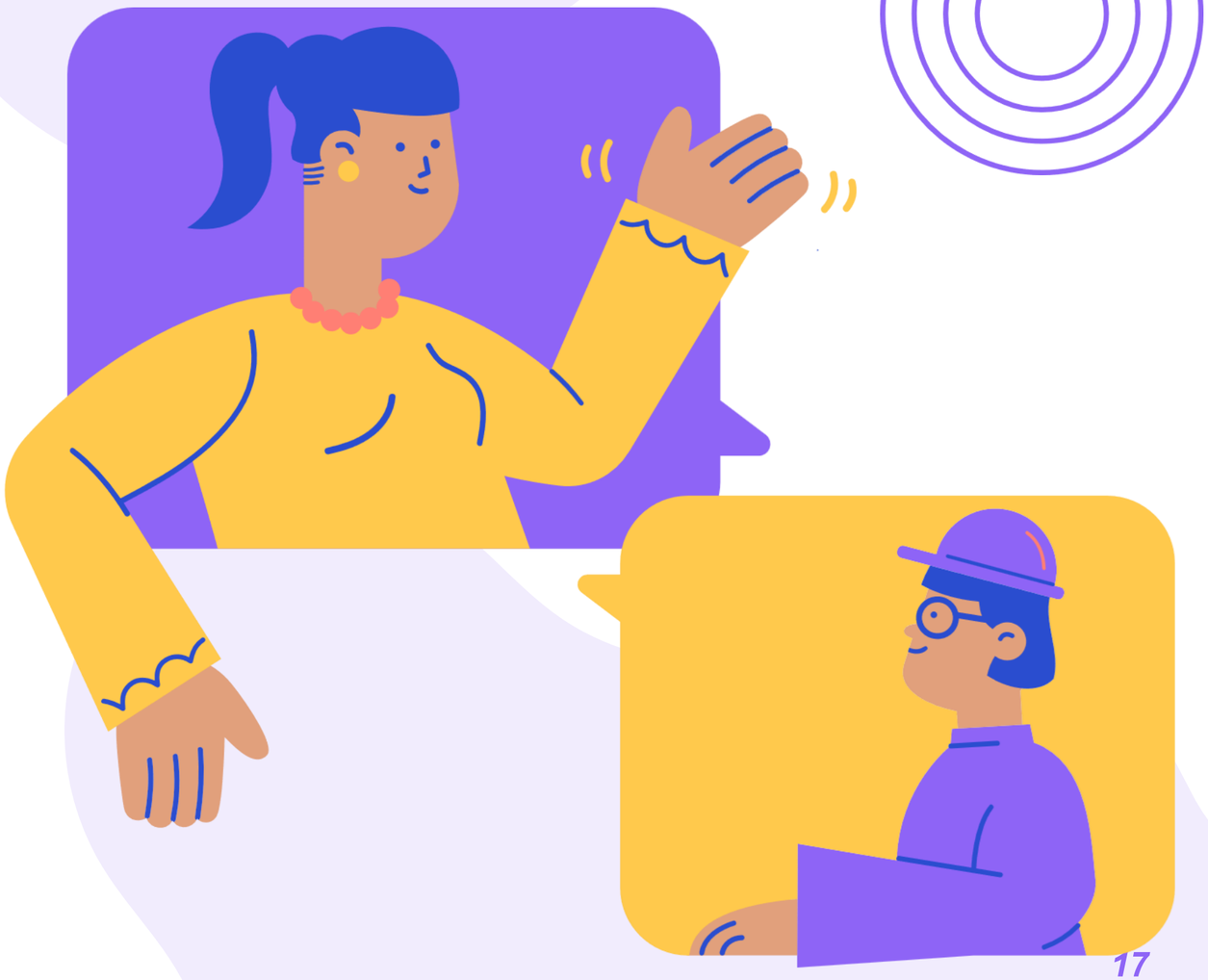
Sonuç HTML olarak döner.

ÖN YÜZ

Nedir ?

ARKA YÜZ

Nedir ?



ÖN YÜZ

Kullanıcının gördüğü arayüzdür. (HTML, CSS, JS)

ARKA YÜZ

Nedir ?

ÖN YÜZ

Kullanıcının gördüğü arayüzdür. (HTML, CSS, JS)

ARKA YÜZ

Kullanıcının görmediği, sistemin çalışmasını sağlayan arka plan yazılımıdır.

Django nedir?

Django, Python ile yazılmış açık kaynaklı bir ***web backend framework***'üdür.

Web uygulamalarını hızlı, güvenli ve düzenli bir şekilde geliştirmek için kullanılır.

Framework nedir?

Framework, yazılım geliştirirken işleri kolaylaştıran, hazır yapı ve kurallar sunan bir geliştirme iskeletidir.

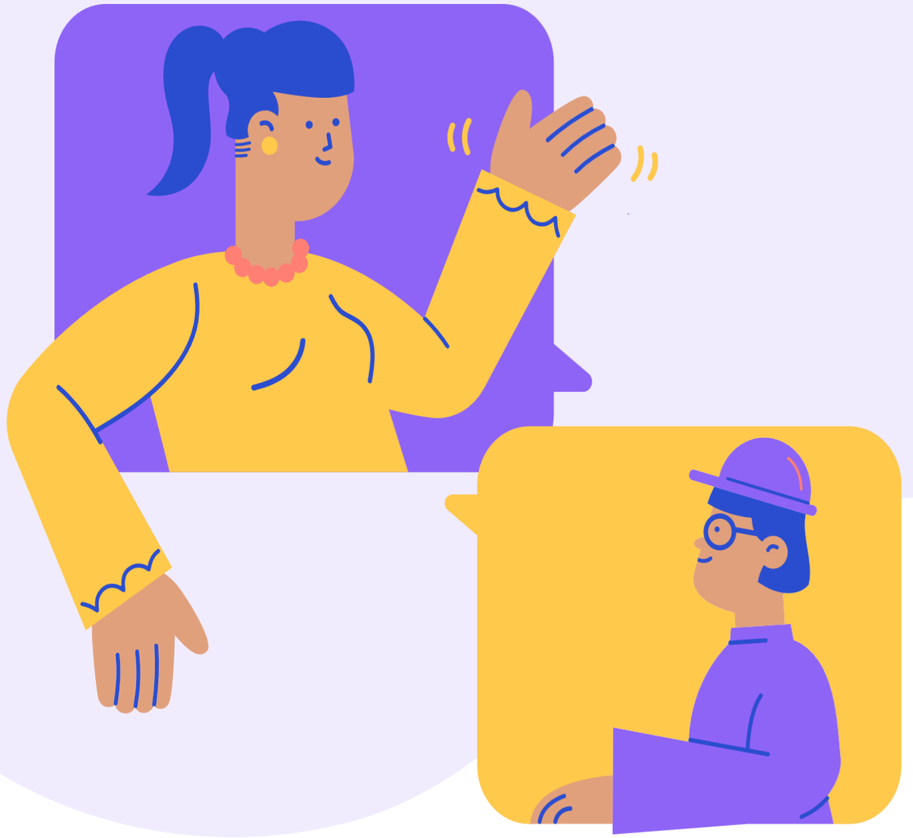
Framework =

Hazır sistem + kurallar + araçlar

Framework nedir?

Framework, yazılım geliştirirken işleri kolaylaştıran, hazır yapı ve kurallar sunan bir geliştirme iskeletidir.

Neden Kullanılır?



Framework nedir?

Neden Kullanılır?

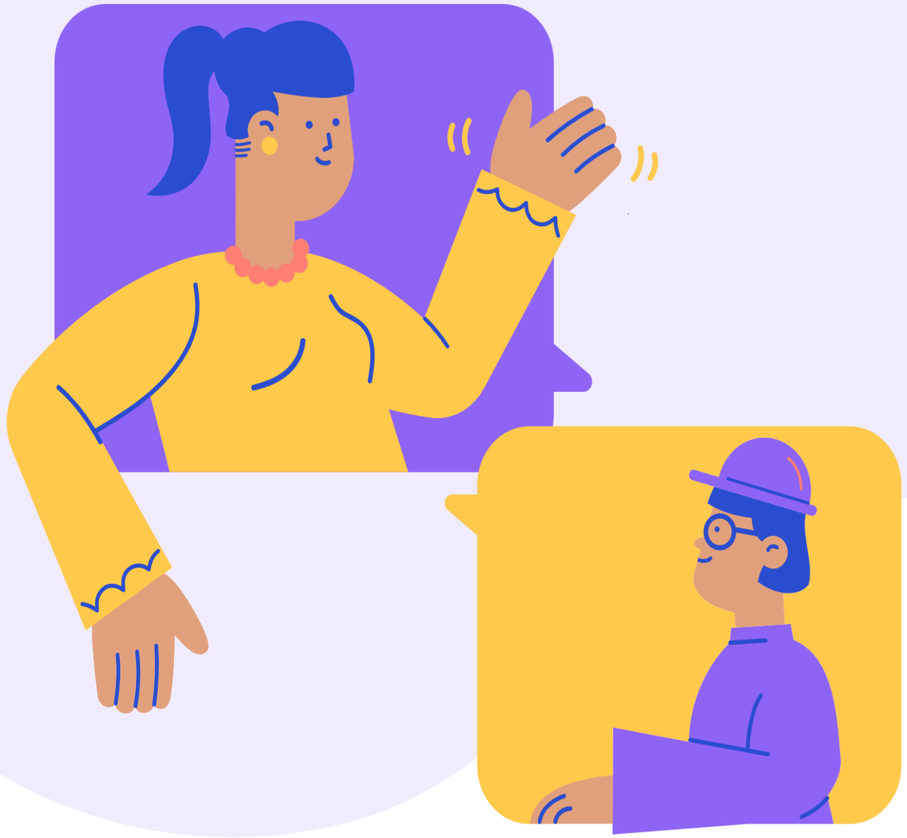
Çünkü sıfırdan her şeyi yazmak yerine:

Hazır güvenlik yapıları gelir.

Veritabanı bağlantısı hazırdır.

Dosya yapısı bellidir.

Tekrar eden işler otomatikleşir.



Bu da; Daha hızlı geliştirme - Daha düzenli kod - Daha az hata sağlar.

Örnek Frameworkler

Backend:

Django (Python)

Laravel (PHP)

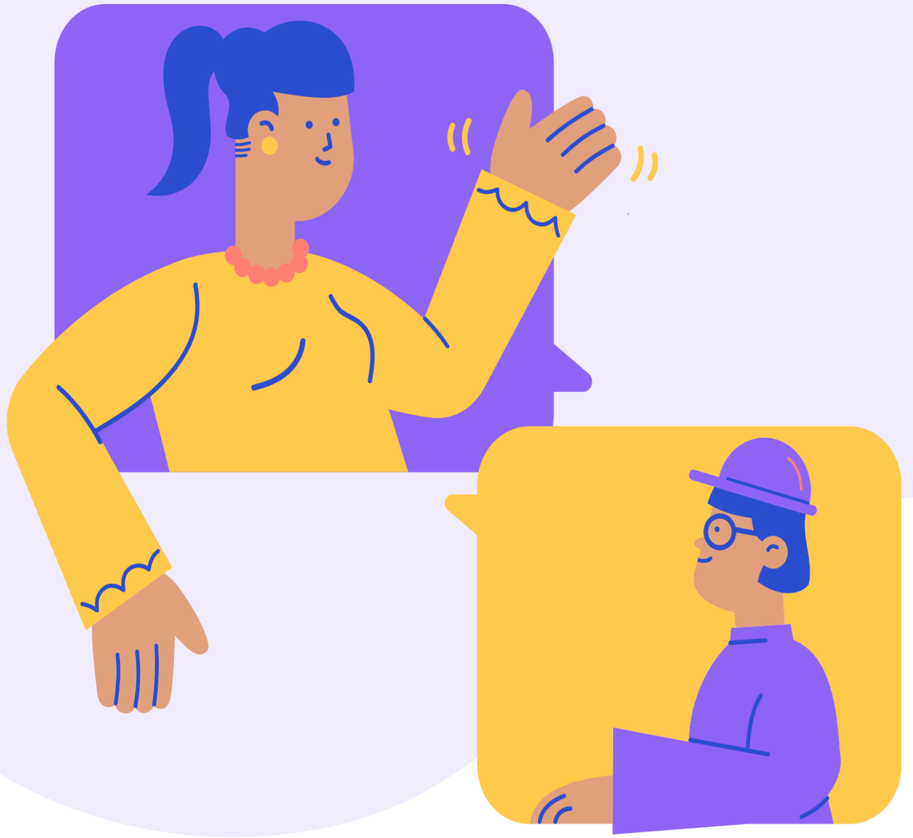
Spring (Java)

Frontend:

React

Angular

Vue



Django ile

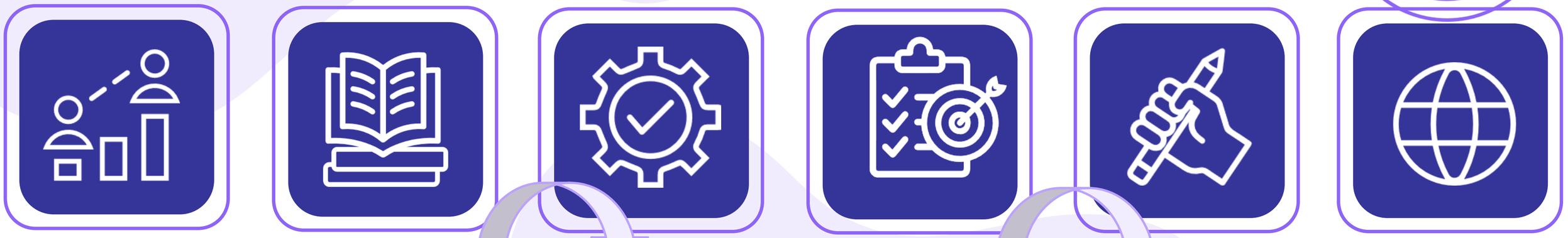
- Web sitesi geliştirme
- Web uygulaması geliştirme
- API yazma
- Kullanıcı giriş / kayıt sistemi kurma
- Veritabanı işlemleri yapma
- Admin paneli oluşturma

Django Özellikleri

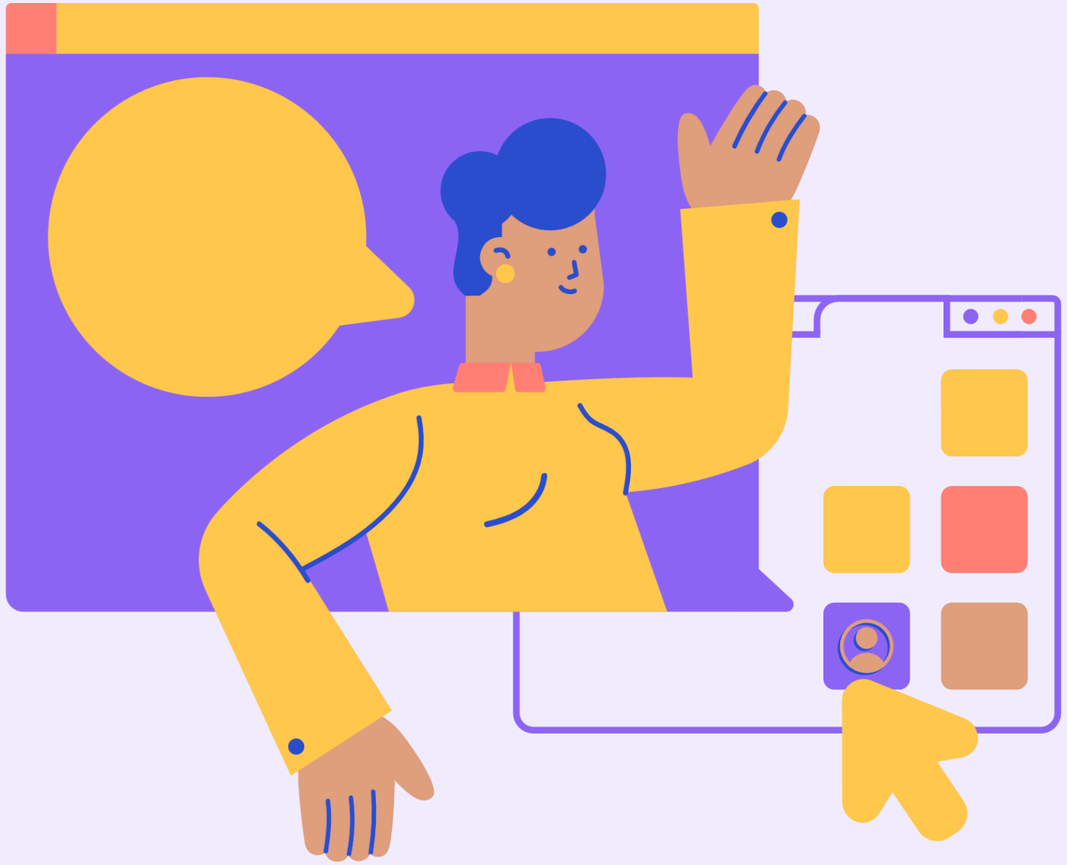
- Hazır admin paneli
- Güvenlik önlemleri (SQL injection, XSS koruması vb.)
- ORM (Veritabanını Python kodu ile yönetme -Object Relational Mapping)
- Hızlı geliştirme
- MVC benzeri yapı (Django'da MVT)

Django = Hız + Güvenlik + Yapı

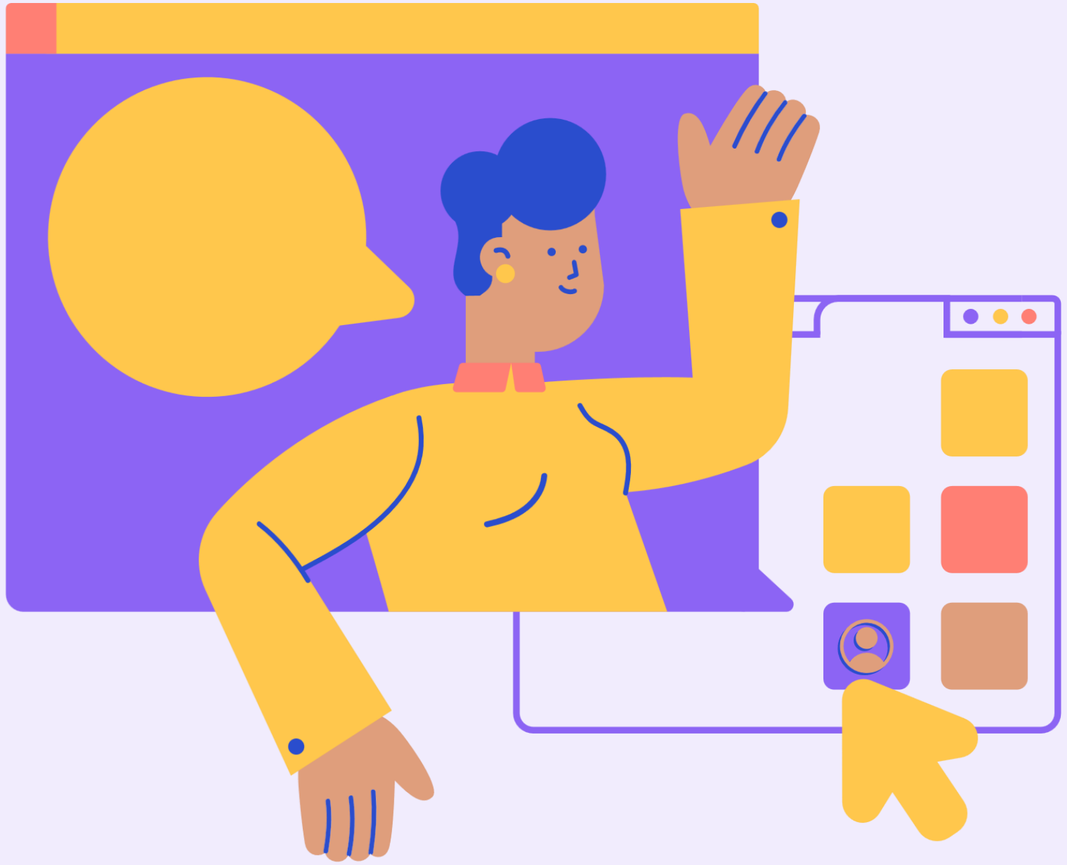
Django Nasıl Çalışır?



Django'nun kullanım alanları

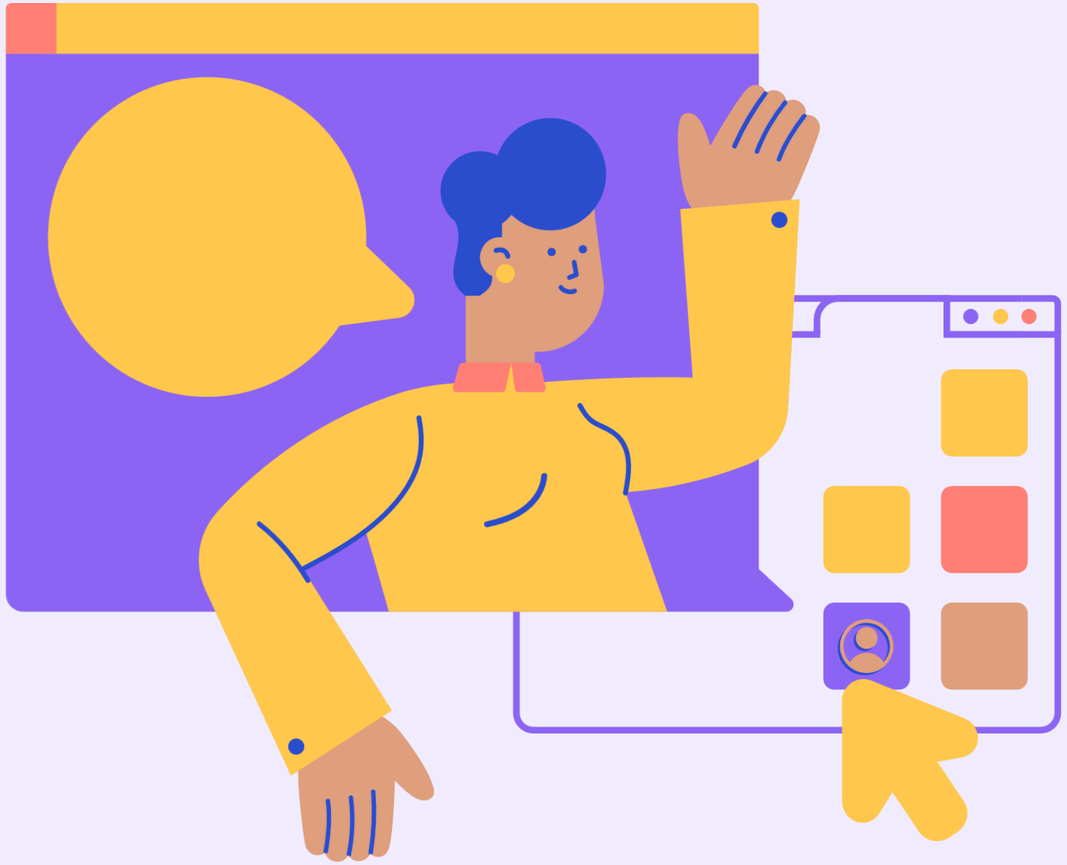


Django her projeye uygun mu?



Django her projeye uygun mu?

Basit bir 1 sayfalık tanıtım sitesi için Django gerekir mi?



Django her projeye uygun mu?

Basit bir 1 sayfalık tanıtım sitesi için Django gerekir mi?

Django genellikle veri tabanlı ve iş mantığı olan projelerde kullanılır.

Nerelerde Kullanılır ?

E-ticaret siteleri

Ürün listeleme

Sepet

Sipariş

Kullanıcı hesabı

Stok kontrolü

Nerelerde Kullanılır ?

İçerik Yönetim Sistemleri

Haber siteleri

Blog siteleri

Kurumsal içerik panelleri

Nerelerde Kullanılır ?

Kurumsal Yönetim Panelleri

Personel yönetimi

Stok takip sistemi

CRM sistemleri

Öğrenci bilgi sistemi

Nerelerde Kullanılır ?

API Servisleri (Backend API)

Mobil uygulamalar

React / Vue frontendler

Mikroservis mimarileri

Django ile...

Instagram

Firefox

NATIONAL GEOGRAPHIC

Pinterest

Spotify®

YouTube

Dropbox

Django Güçlü Yanları

Hızlı geliştirme

Hazır admin

Hazır auth

ORM (Object Relational Mapping)

Django Güçlü Yanları

Güvenlik

CSRF koruması (Cross-Site Request Forgery)

SQL injection koruması

XSS koruma yardımcıları

Şifre hashleme

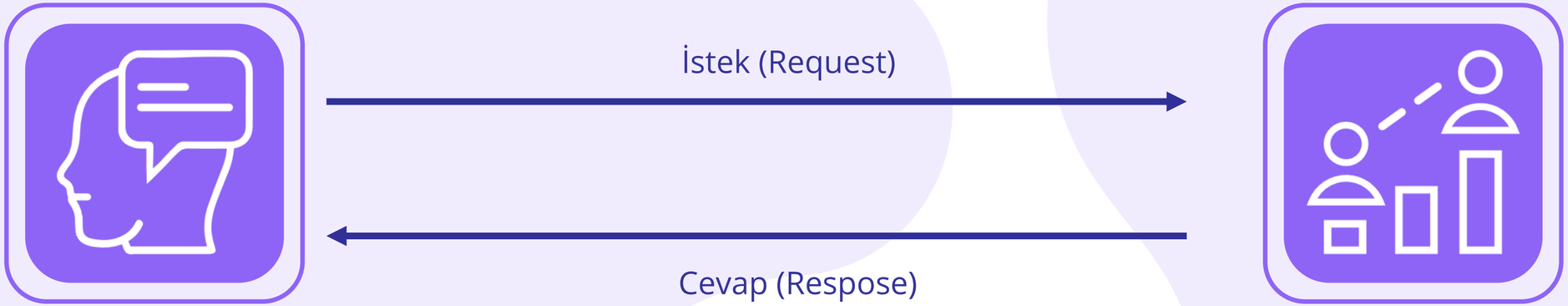
Web nedir? HTTP temelleri

Web

Web dediğimiz şey aslında bilgisayarların birbiriyle konuşma sistemidir.

Web = Client (İstemci) + Server (Sunucu) + Protokol

Web'in Temel Mantığı



HTTP NEDİR?

HyperText

- HTML gibi bağlantılı metin

Transfer

- Veri aktarımı

Protocol

- Kurallar bütünü

- HTTP, web üzerinde veri alışverişini sağlayan iletişim protokolüdür.

HTTP Metotları

GET

Veri almak için

POST

Veri göndermek için

PUT

Var olan veriyi güncelleme

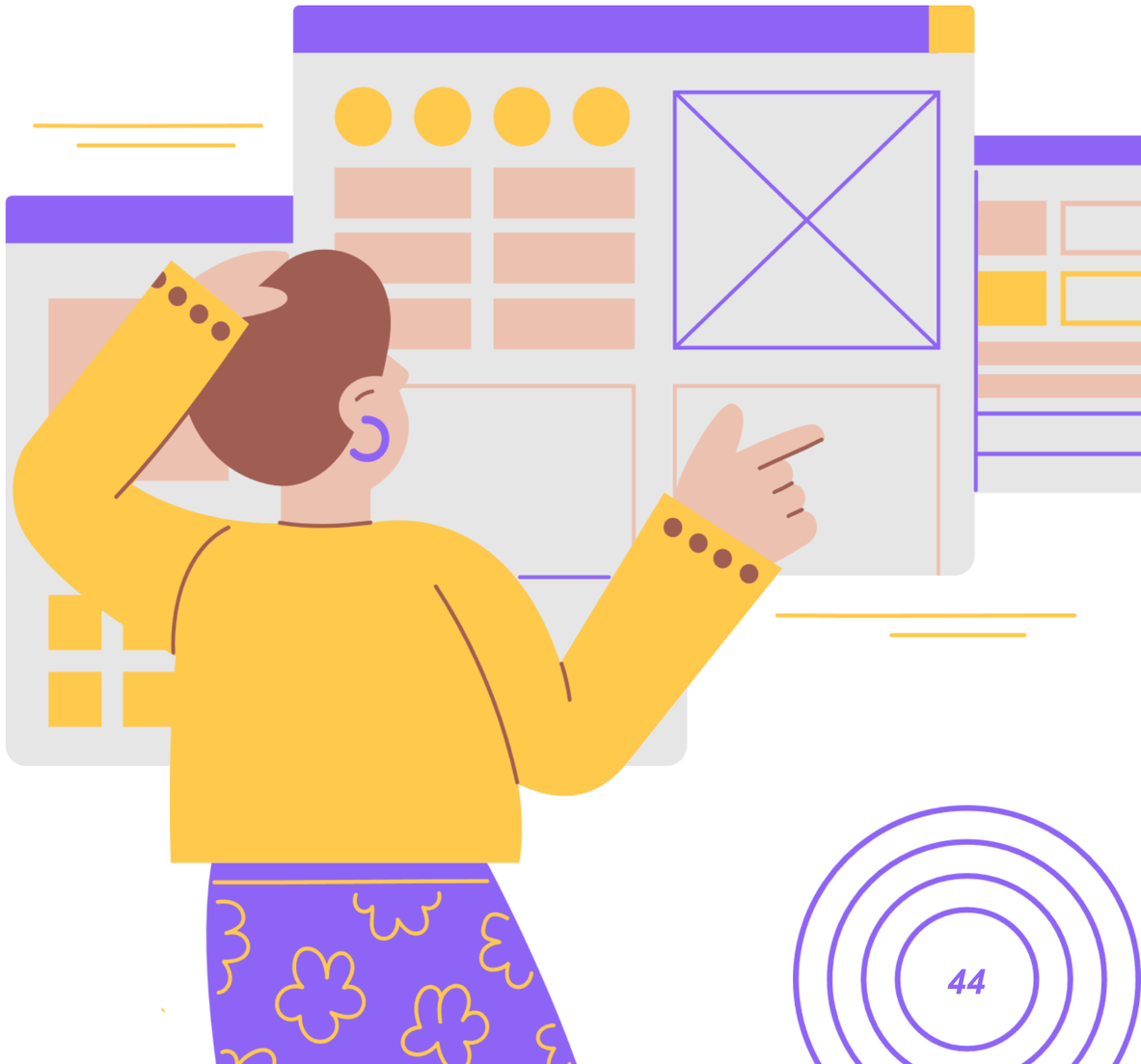
DELETE

Var olan veriyi güncelleme

Login ve Ürün listeleme hangi http metotları ile yapılır yorumlayınız.

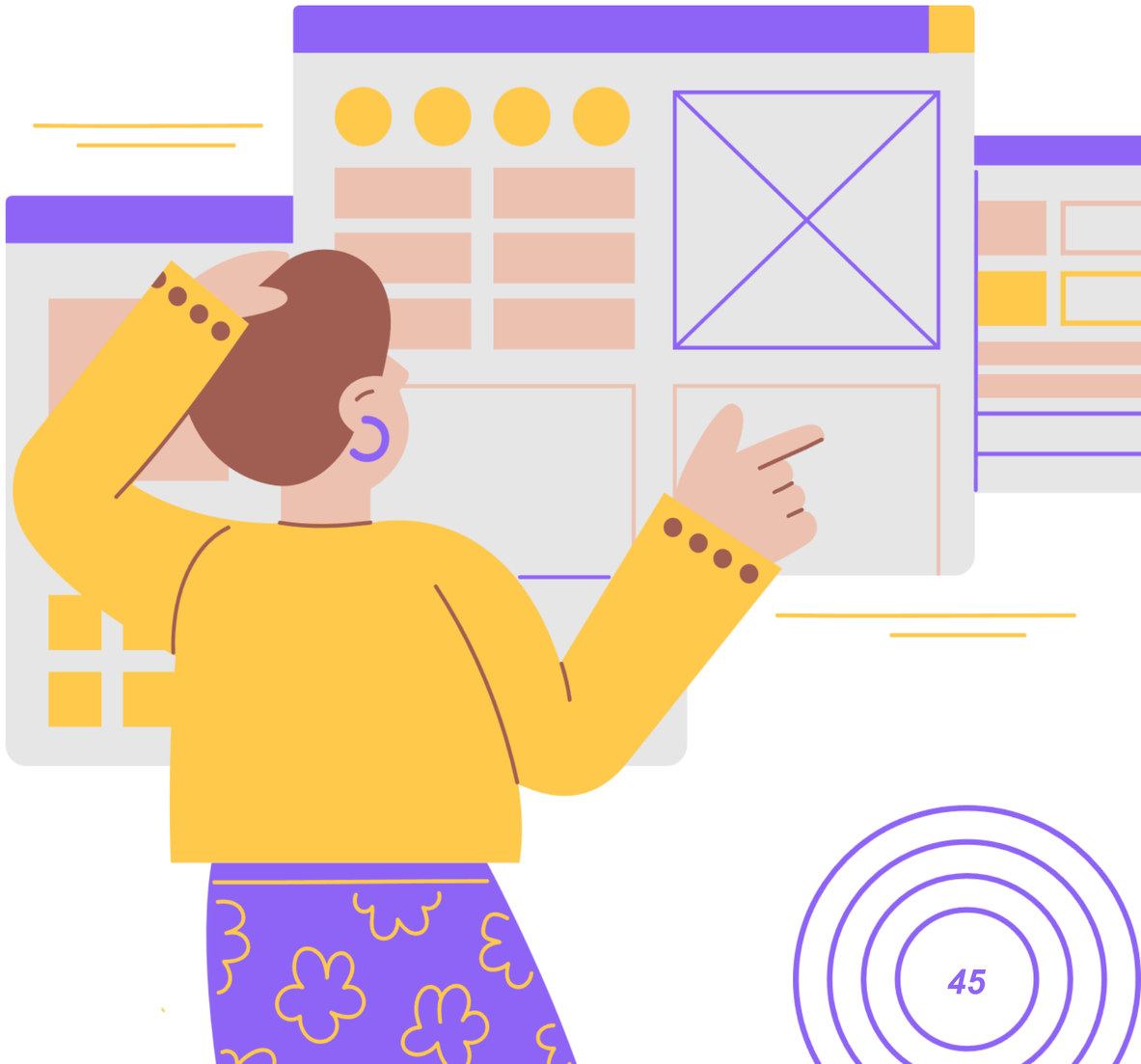
HTTP Durum Kodları

- 200
- 201
- 400
- 401
- 403
- 404
- 500



HTTP Durum Kodları

200 → OK	(İstek başarılı)
201 → Created	(Yeni veri oluşturuldu)
400 → Bad Request	(Hatalı istek)
401 → Unauthorized	(Yetkisiz)
403 → Forbidden	(Erişim yasak)
404 → Not Found	(Sayfa bulunamadı)
500 → Internal Server Error	(Sunucu hatası)



MVT Mimarisi

MVT Nedir? Neyi Çözer?

MVT, Django’nun uygulamayı parçalara ayırma biçimidir. Web uygulamasını 3 parçaya bölerek düzenli çalışmasını sağlar.

Amaç:

- Kodun düzenli olması (okunabilirlik)
- Sorumlulukların ayrılması (bakım kolaylığı)
- Tekrarın azaltılması
- Test edilebilirlik

MVT Nedir? Neyi Çözer?

Model

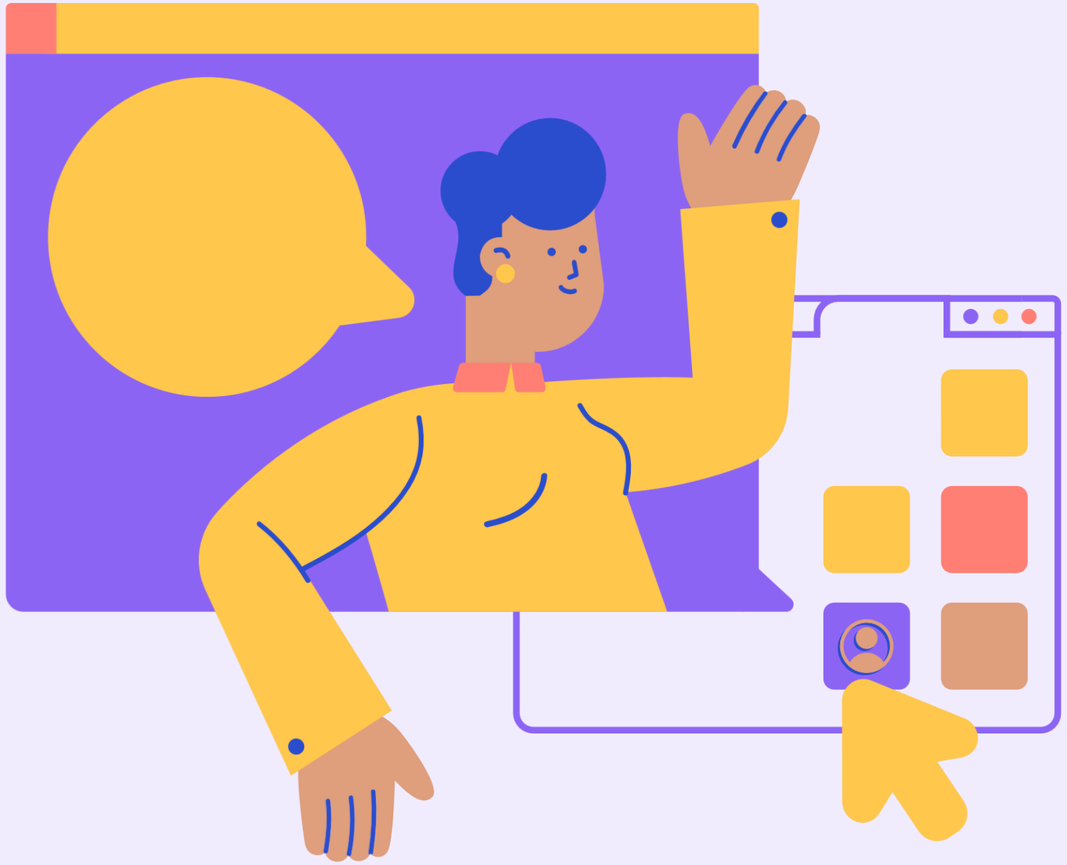
- Verinin kendisi
- Veriyle ilgili kurallar

View

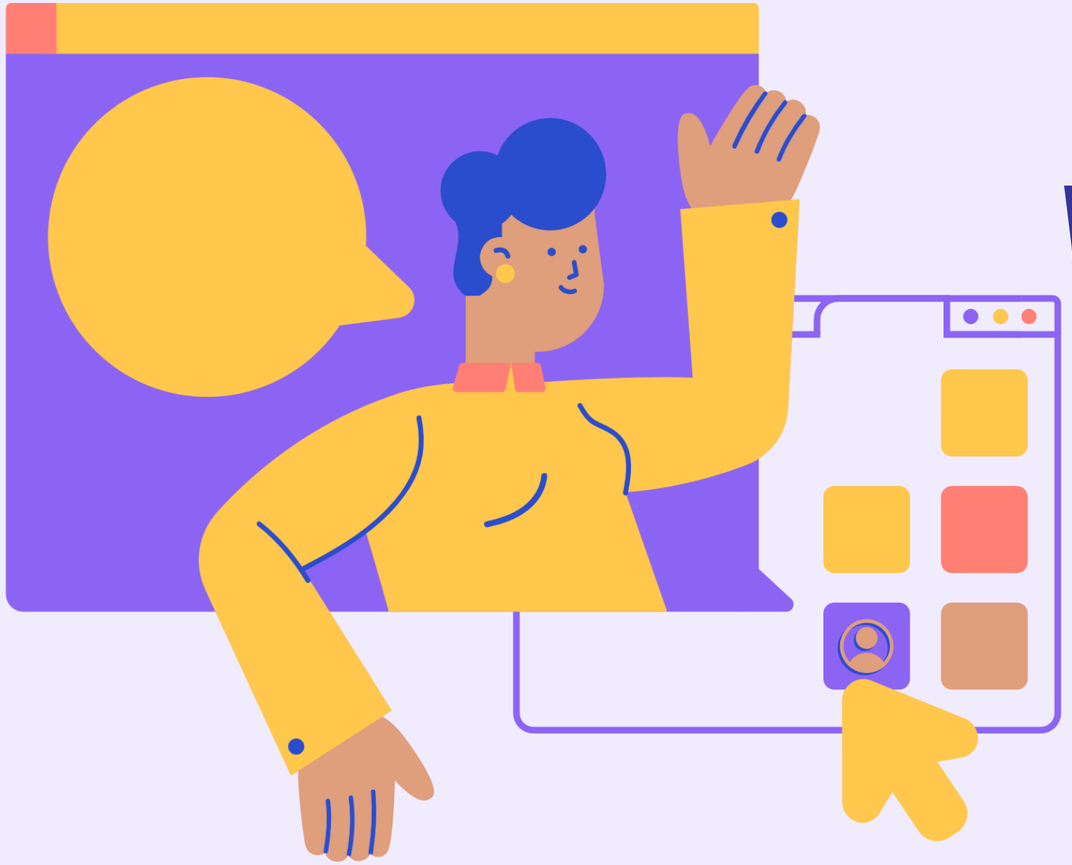
- İş mantığı
- hangi verinin hangi şablona gideceğine karar

Template

- Kullanıcıya gösterilecek arayüz (HTML)



Veri?
İşlem?
Görünüm?



Veri → MODEL

İşlem → VIEW

Görünüm → TEMP.

Model: “Veri + Kurallar”

Veri Yapısı (Database Şeması)

Bir tablonun alanlarını tanımlanır.

Örneğin Öğrenci tablosu : ad, soyad, sinif

Model: “Veri + Kurallar”

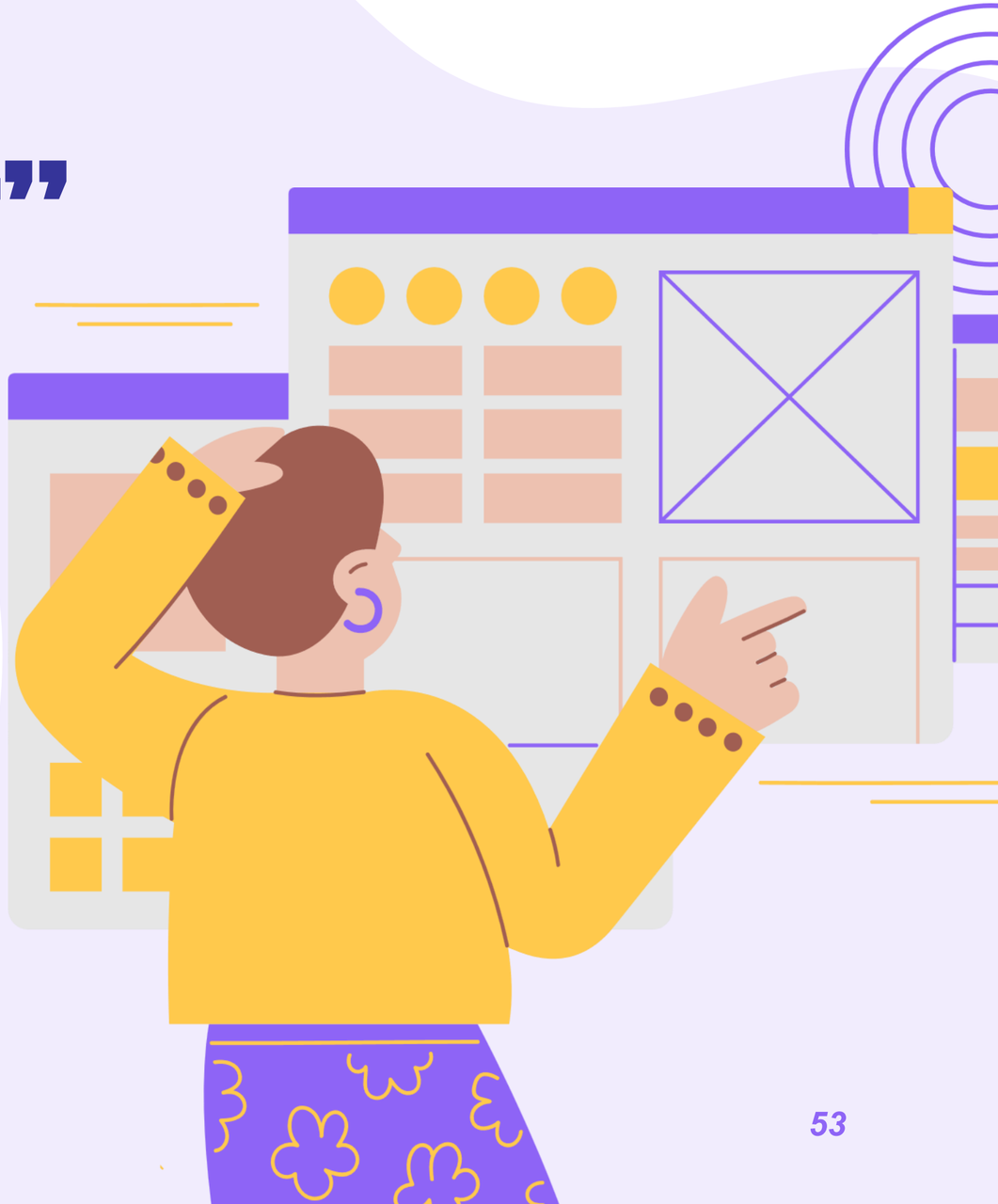
İş Kuralları (Domain Logic)

- Veri doğrulama kuralları
- Hesaplanan özellikler (ör. full_name)
- İlişkiler (ForeignKey/ManyToMany vs.)

Model: “Veri + Kurallar”

ORM Mantığı (“Veriye Erişim”)

- “Model sınıfı” üzerinden kayıt ekleme, okuma, güncelleme, silme



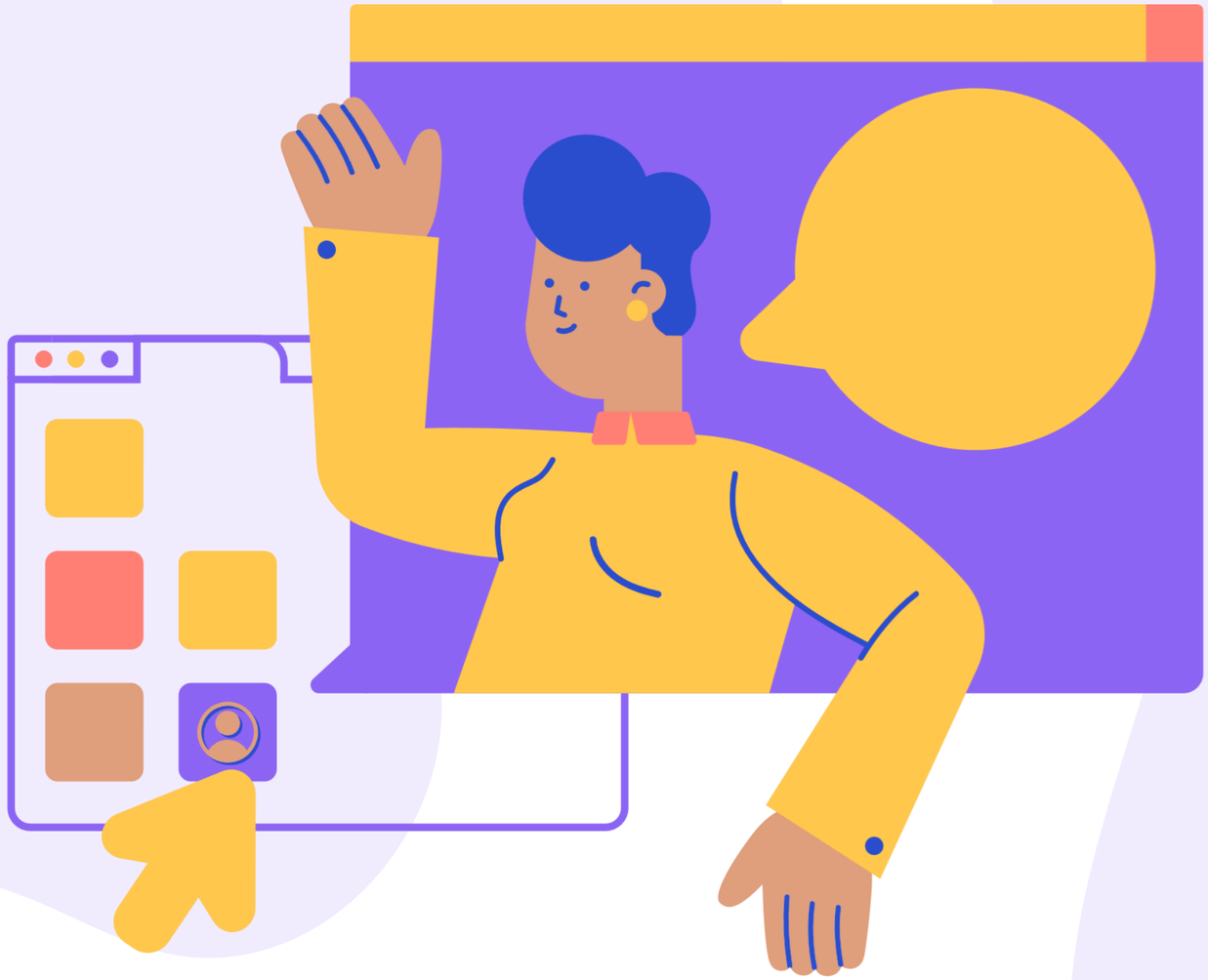
Model

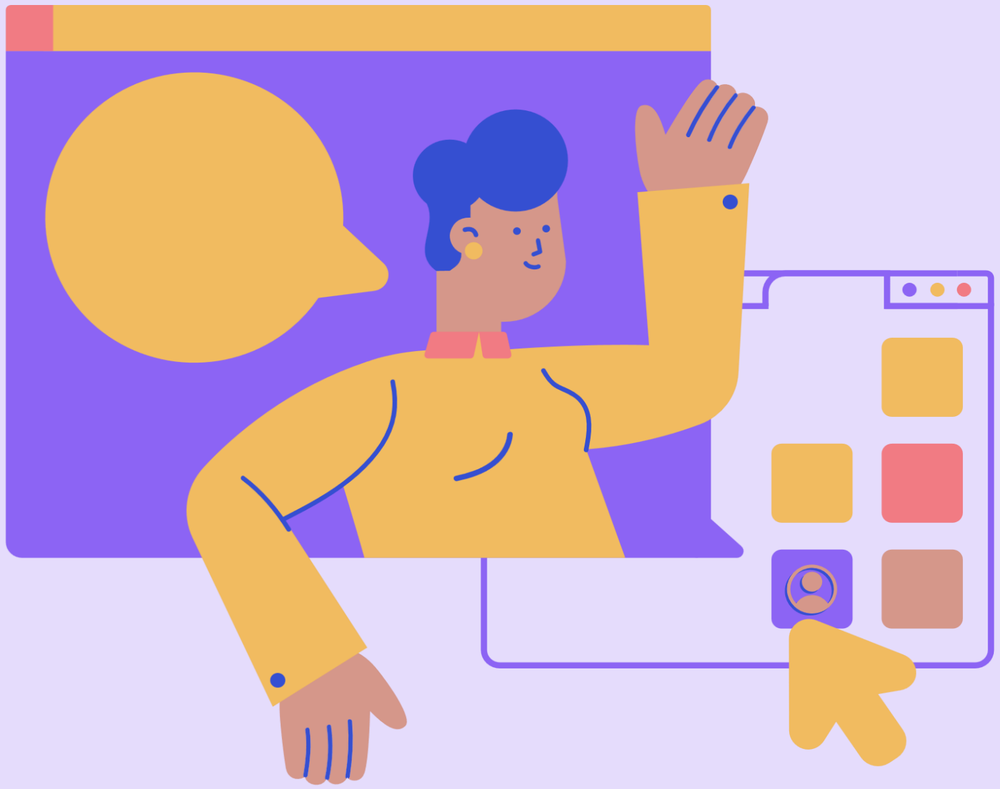
Özetle veritabanı ile ilgilenir.

Tabloları tanımlar

Veri ekler, siler, günceller

ORM burada çalışır





MODEL → Veri Katmanı

Template: “Sunum Katmanı”

Template, Django’da genellikle HTML dosyalarıdır.

- Kullanıcıya gösterilecek sayfanın iskeletini kurar.
- View’den gelen veriyi yerleştirir.
- Basit sunum işlemleri yapar (listeleme, koşul, formatlama gibi).

Template: “Sunum Katmanı”

Template katmanında

- Döngü (for)
- Koşul (if)
- Basit filtreleme (tarih formatlama vs.)

Gibi işlemler yapılabilir.

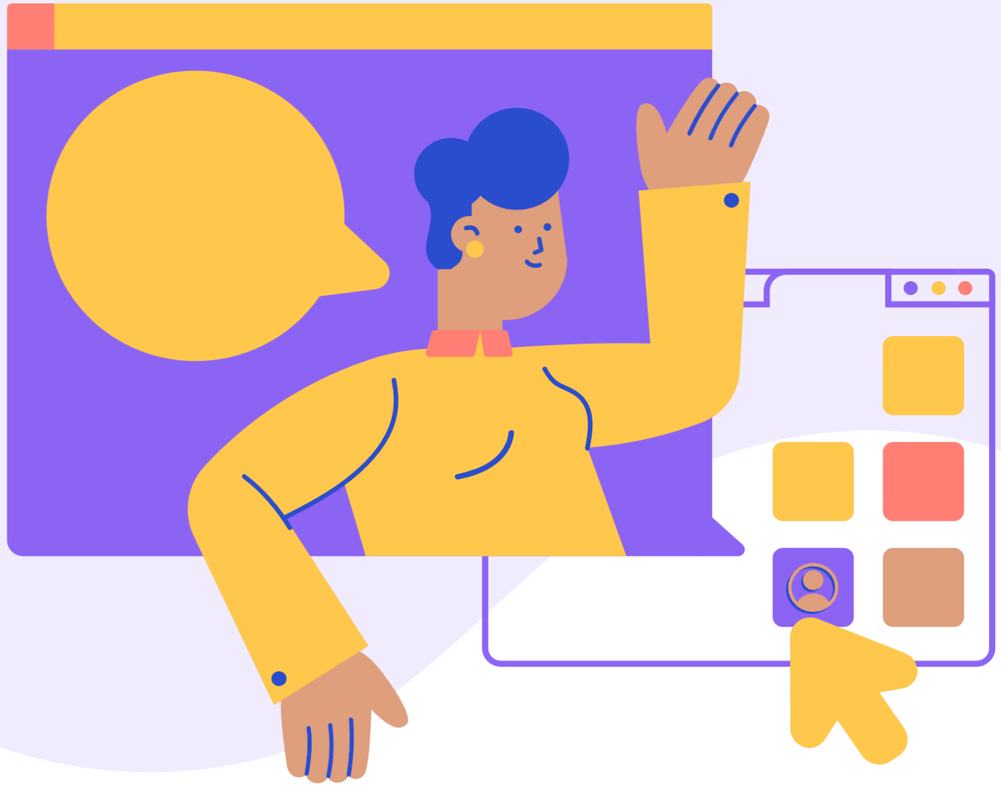
```
<ul>
  {% for product in products %}
      <li>{{ product.name }} - {{ product.price }}</li>
  {% endfor %}
</ul>
```

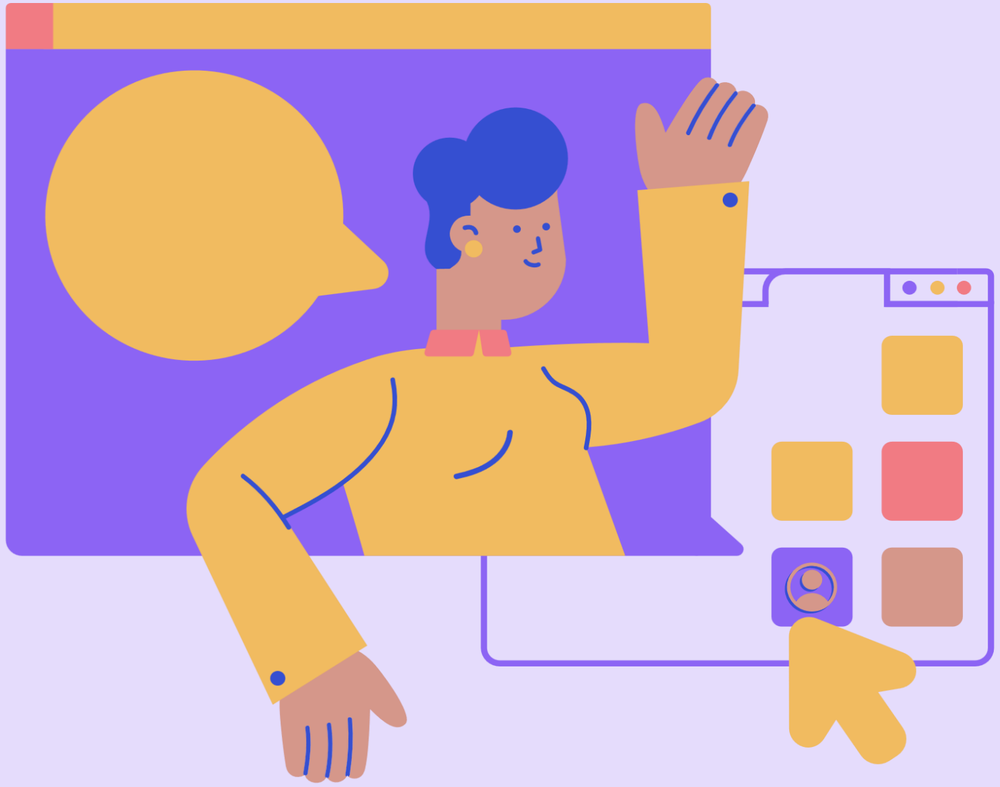
Template: “Sunum Katmanı”

Template katmanında

- Ağır iş mantığı
- DB sorgusu
- Karmaşık hesap

Gibi işlemler **yapılamaz** !





TEMPLATE→ Görüntü Katmanı

Template'in "temiz" kalması çok kritik

Template

Template katmanı kullanıcıya görünen arayüzdür.

Özetle:

- HTML içerir
- Dinamik veri gösterir
- Tasarım burada olur

View: “Karar Verici Katman”

Template, Django’da genellikle HTML dosyalarıdır.

- Hangi verinin kullanılacağını belirler. (modelden veriyi alır)
- Hangi şablonun render edileceğini seçer.
- Şablona hangi verilerin gönderileceğini belirler.
- Kullanıcıdan gelen form verisini değerlendirir.

View

Özetle:

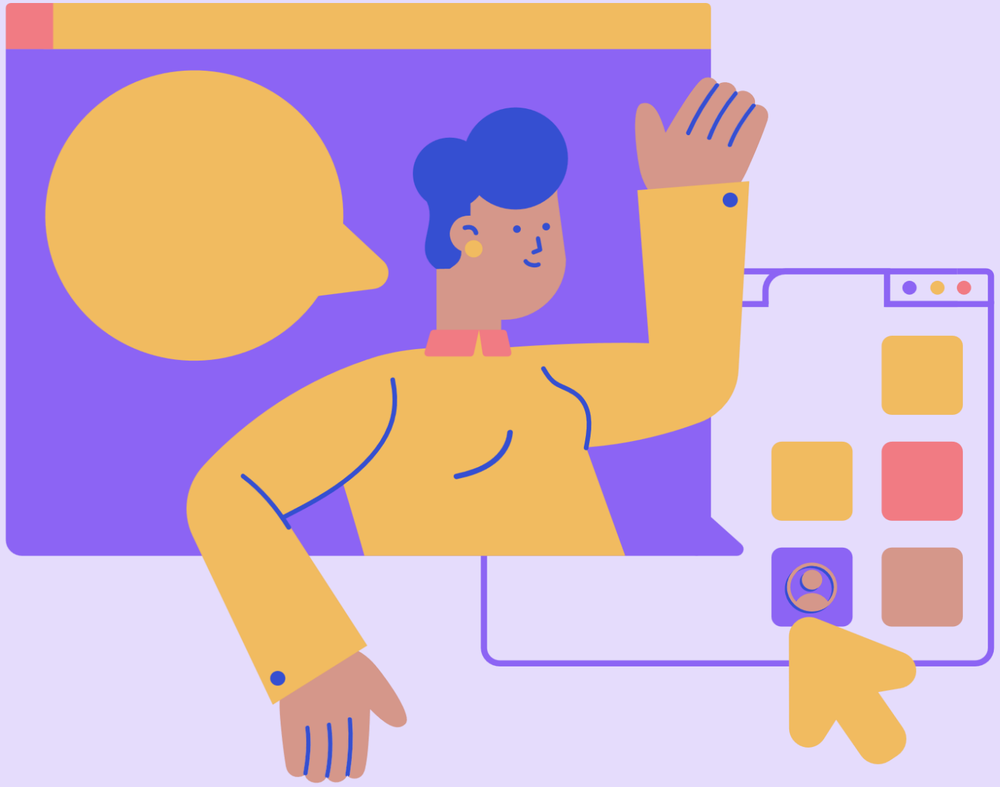
- Kullanıcıdan gelen isteği alır
- Gerekirse modelden veri çeker
- Sonucu template’e gönderir

View

View katmanı :

- “Bu sayfada hangi veriler listelenecek?”
- “Detay sayfasında hangi kayıt gösterilecek?”
- “Yetki var mı?” (auth haftasında ayrıntı)
- “Hata varsa ne göstereceğim?”

Sorularının cevap bulduğu yerdir.



VIEW→ İşlem Katmanı

View'ı orkestra şefi olarak düşünebiliriz.

MVT

“Liste sayfasında hangi kayıtlar gösterilecek?”

“Post’un başlığı ve tarihi hangi alanlar?”

“Post listesi HTML’de nasıl görünecek?”

MVT

“Liste sayfasında hangi kayıtlar gösterilecek?” → View

“Post’un başlığı ve tarihi hangi alanlar?” → Model

“Post listesi HTML’de nasıl görünecek?” → Template

Django'nun genel yapısı

Djangonun Genel Yapısı

```
├── core/                          ← Ana sayfa & genel sayfalar
│   ├── __init__.py
│   ├── views.py
│   ├── urls.py
│   ├── models.py
│   └── templates/
│       └── core/
│           ├── home.html
│           └── about.html
```

```
myproject/
│
├── manage.py
│
├── myproject/                     ← Proje ayarları
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
│
├── core/                          ← Ana sayfa & genel sayfalar
│   ├── __init__.py
│   ├── views.py
│   ├── urls.py
│   ├── models.py
│   └── templates/
│       └── core/
│           ├── home.html
│           └── about.html
│
├── blog/                          ← Blog sistemi
│   ├── __init__.py
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   └── templates/
│       └── blog/
│           ├── blog_list.html
│           └── blog_detail.html
│
└── templates/                     ← Ortak template klasörü (opsiyonel)
    └── base.html
```

Son