Genyazılım Stajyer Grubu Toplantı 3

MVC, Backend/API/Frontend ve Django

Kuzey Koç

TL;DR

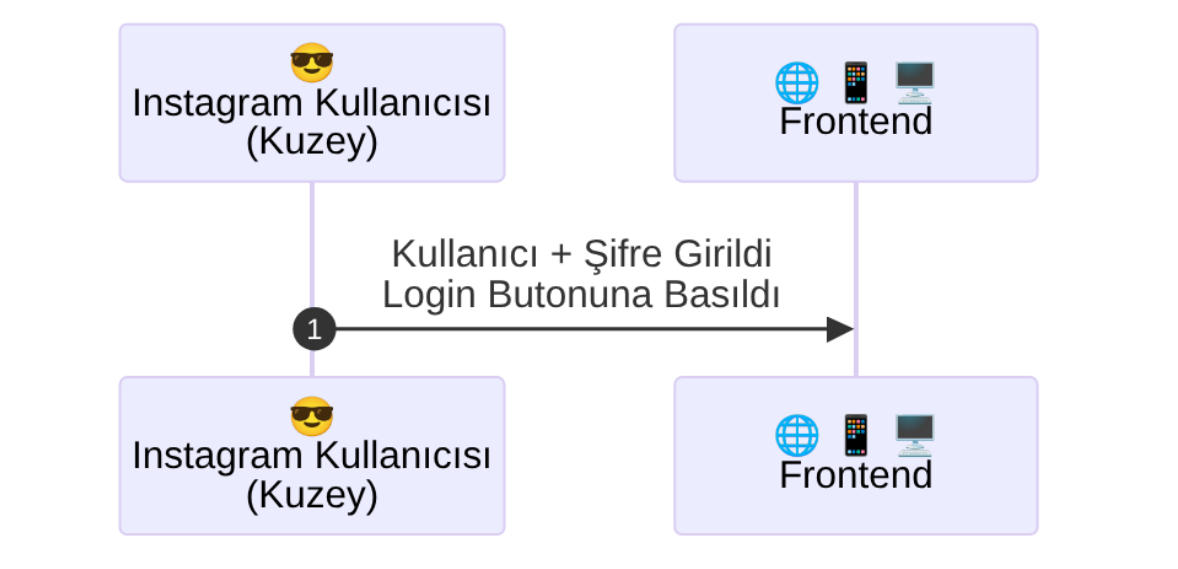
Toplantının özeti şu şekildedir;

- Öğrencilere **Scrimba: Python Tutorial** kursunda anlaşılmayan yerlerin olup olmadığı soruldu ve bu konu üstüne konuşuldu.
- **MVC Mimarisi** anlatıldı
- **Frontend** ve **Backend** mantığı anlatıldı
- **API** anlatıldı
- Django'dan bahsedildi ve bir **demo** yapıldı
- Django ödevi verildi

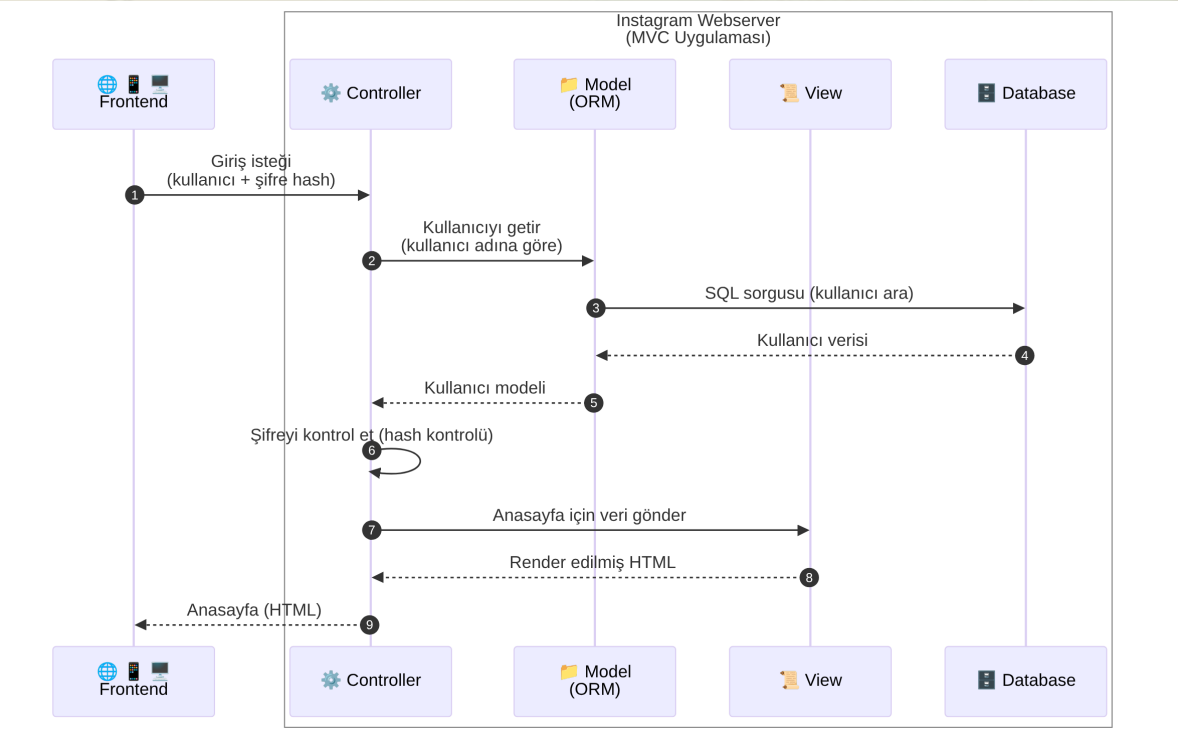
MVC Mimarisi

MVC, Açılımı *Model View Controller* olan bir **Yazılım Mimarisidir**. Birçok yazılım mimarisinin temelini oluşturur. MVC mimarisini örnek bir senaryo ile açıklamaya çalışalım.

Kuzey isminde bir instagram kullanıcımız olsun ve bu uygulamaya giriş yaptığını var sayalım. Kullanıcı adını ve şifresini giriyor ve "giriş yap" butonuna tıklıyor. Burası tamamen Kuzey ile kullanıcı arayüzü arasında olan bir interaksiyon ve henüz MVC ile ilgili birşey yok. Aşağıda bir sequence diyagram ile olan biten gösteriliyor



Kuzey ”Login” butonuna tıkladıktan sonra bu bilgi, instagram arayüzünden Amerika’da yer alan Instagram web serverına gönderilir. Burada tabiki konuyu en basit düzeyde tartışıyoruz. Çünkü işler o kadar da basit değil ama şimdilik bu şekilde kabul edelim.



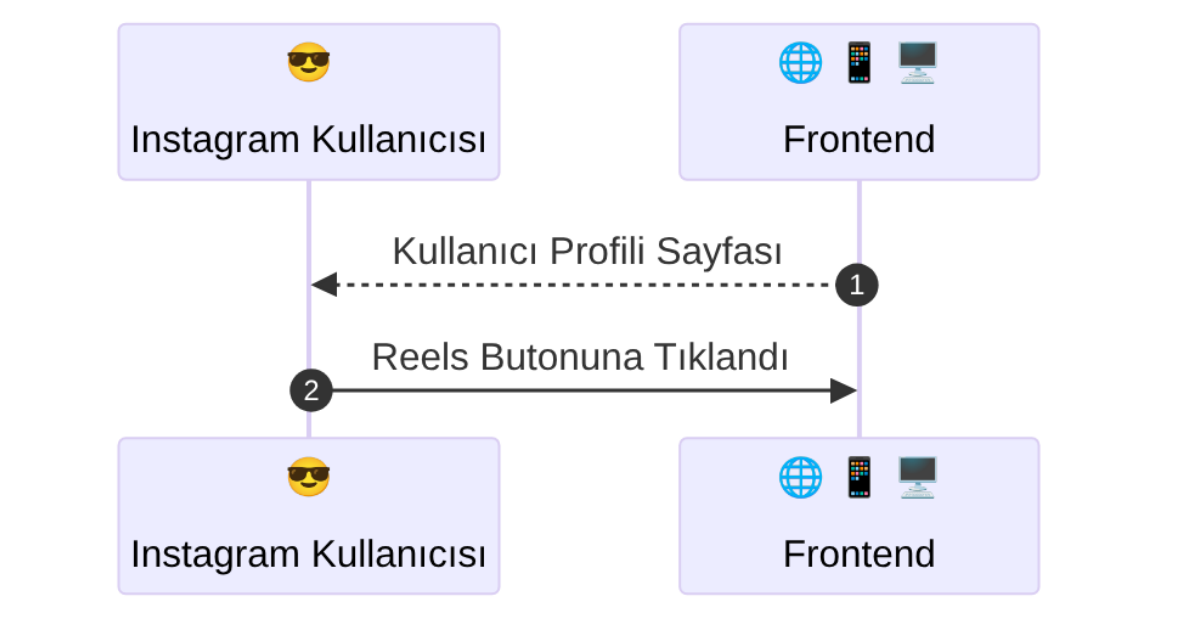
Fronten’imiz, bilgilerimizi Instagram sunucusuna gönderdikten sonra bu bilgiler bu sunucu içinde çalışan Instagram uygulamasına gider ve ilk olarak **Controller** modülü karşılar. (Aslında Router karşılar ancak şimdilik bu ayrıntıyı es geçebiliriz).

Controller, gelen kullanıcı, şifre ve ”butona tıklandı” bilgilerini alır ve gerekli işlemleri yapmaya başlar ancak Instagramdaki bütün kullanıcılar arasından bizi tanıması için öncelikle veritabanından bizi bulup şifremizin doğru olup olmadığını kontrol etmesi gerekir. Bunun için öncelikle ”Veriden Sorumlu” olan **Model** modülüne bu bilgileri gönderir. Model ise veritabanından kullanıcı ismimizi ve şifremizin hashini çeker ve Controllera geri verir.

Contorller bu noktada Model’den gelen bilgilerle işlem yapar. Bu işlemlere örnek olaraka;

- Böyle bir instagram kullanıcısı gerçekten var mı? (veritabanında var mı?)
- Frontend’den gelen şifre hashi, veritabasnından gelen hash ile uyuşuyor mu?

Eğer herşey doğru ise ve Kuzey gerçekten şifresini doğru girdiyse, Controller, sayfaları barındıran **View** modülünden, Kuzey'in profil sayfasını oluşturmasını ister. View ise Kuzeyin sayfasını Controller'a HTML formatında iletir. Contorller da bunu Fronten'de gönderir.



Bu aşamada artık Frontendin elinde Bakcend'den gelen bir HTML verisi bulunur. Tek yapması gereken ise Kuzey'e bu sayfayı render etmek ve göstermek olur.

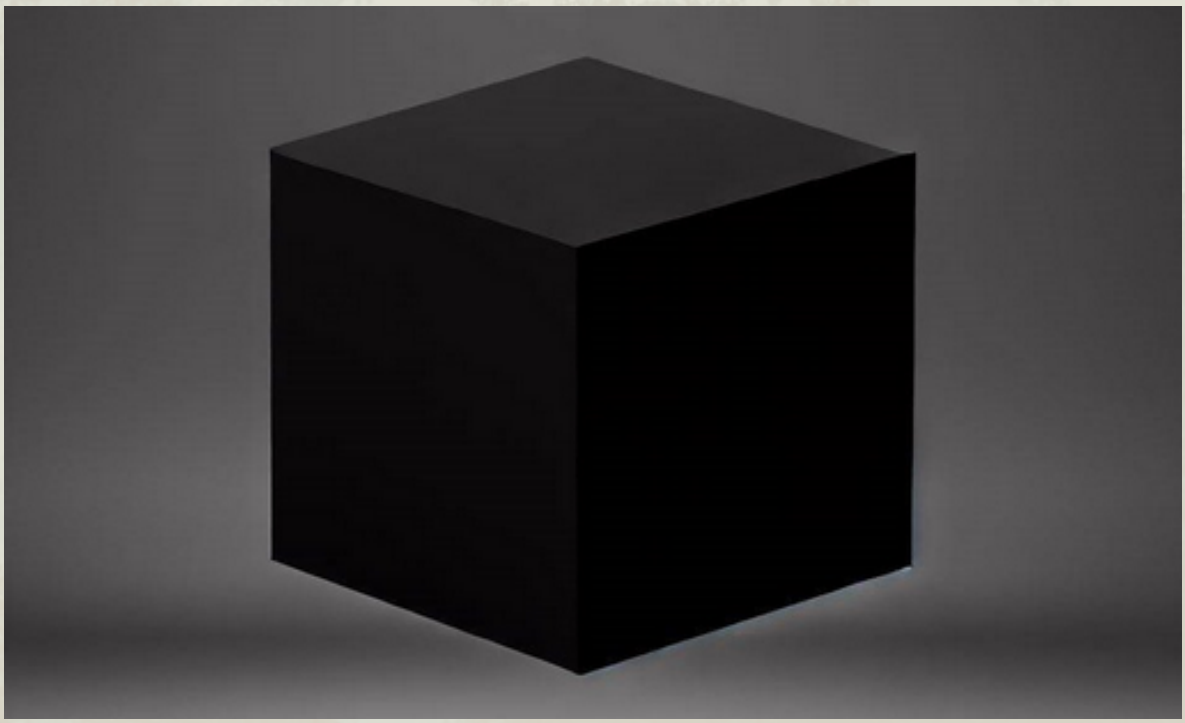
Kuzey artık profiline giriş yaptığını görür ve **doom scrolling**'ine başlayabilir hale gelir. Yine Frontend'den Reels butonuna tıkladıktan sonra MVC tarafında benzer işler gerçekleşir. Bunu artık MVC mantığını kavradıktan sonra aklımızda canlandırmak daha kolay olacaktır.

- Reels butonuna tıklandı bilgisi Controller'a ulaşır
- Controller, Model'i kullanarak veritabanından ya da storage'dan videoları çeker
- Controller, View'a aldığı videoyu gönderir ve sayfa oluşturmasını ister
- View reels barındıran bir sayfa oluşturup HTML formatında Controllera geri verir
- Controller Frontend'e sayfayı gönderir
- Frontend, Controllerdan aldığı HTML verisini sayfada görüntüler
- Kuzey reelsı sayfasında görür

Bir sonraki adım ise Kuzey'den Frontend'e giden "yukarı kaydırma" eylemi. MVC tarafında bir sonraki videonun frontende getirilmesi nasıl olur kendiniz yorumlayın.

Backend, Frontend ve API

Demin görmüş olduğumuz MVC mimarisi bir Backend mimarisidir. Backend dediğimizde aklımıza kapalı bir kutu ve sadece gelen isteklere cevap verebilen ve veri alış verişi yapabilen bir yapı gelmelidir.



Bu kapalı kutu içinde çalışan yazılımın günün sonunda dışarıdan veri alması gerekiyor. Deminki örnekteki gibi kullanıcının giriş bilgileri, bastığı buttonlar ve yüklediği fotoğrafların bir şekilde bu kapalı kutunun içine girmesi gerekiyor. Bunun için ise bu kapalı kutunun dış dünya ile bağlantısını sağlayacak küçük deliklere ihtiyacı vardır. Biz bu dış dünya ile haberleşmeyi sağlayan yapıya API diyoruz.

API, bu kapalı kutunun bir parçasıdır ve sadece API içinde tanımlanan şeylerin yapılmasına izin verir. Şimdi kapalı kutu örneğimizi alıp bir üst düzeye taşıyalım.

API ve Backend'i bir otomata benzetebiliriz! Otomatlar da aslında back-end gibi "kapalı kutular"dır. ancak dış dünya ile de bağlantı kurup, işlem yaparlar. Otomatın dış dünyayla haberleşmek için kullandığı iki şey vardır;

- ürün seçme fonksiyonu
- para giriş fonksiyonu

Bu iki fonksiyonu, bu kapalı kutunun API'ı olarak kabul edelim. Kullanıcı gelir, ürününü seçer ve parasını verir. Bu dakikadan sonra "kapalı kutu" arka planda sihirli birşey yapar ve ürünümüzü bize verir.

Otomata dikkatli bakarsak aslında ön kısmının ise tamamen bir "arayüz" olduğunu da görürüz. Yani bütün bir "Frontend" de karşımızda duruyordur. Ürünleri kullanıcıya gösteren bir yapısı da vardır.

Özet olarak bir web uygulamasını en basit şekilde örneklendirecek olursak bir Otomata benzetebiliriz;

Otomat	Web Uygulaması
Ürünleri Gösteren Cam	Frontend
Para giriş fonksiyonu	API fonksiyonu
Ürün seçme fonksiyonu	API fonksiyonu
Otomatın İç Mekanizmaları	Backend

Django

Eğer biz de Instagram gibi bir web uygulaması yapmak istersek, gün sonunda bir backend yazmamız gerekecektir. Dökümanda Instagram örneğini vermemin sebebi, Instagramın backendinin gerçekten de Django ile yazılmış olmasıdır: **Instagram Architecture**

Django, Python ile çalışan bir backend frameworküdür ve daha da süperi, MVC mimarisiyle beraber geliyor ve mimariyi kendimiz oluşturmamız gerekmiyor. Biz sadece gerekli **Model**, **View** ve **Controller** modüllerini yazdığımızda, Django gerisini bizim için halletmiş oluyor.

Ben bu toplantıdan önce Django ile Todo uygulamamızın backendini yazdım ve API'ın nasıl çalıştığını göstermek için bir demo yapacağım.

Demo

Ödev

- **Django Tutorial for Beginners – Build Powerful Backends - YouTube** İzlenecek
- Bugün sunumu yapılan Django uygulaması gibi bir backend'in yapılması *(youtube ve chatgpt'den yardım alın)*
- **Insomnia** ile API testi yapılacak ve aşağıdaki **CRUD** requestleri çalışır hale getirilecek;

`GET /api/todos/`	Tüm Todo'ları listele
`GET /api/todos/{id}/`	Tek Todo listele
`POST /api/todos/`	Yeni Todo oluştur
`PUT /api/todos/{id}/`	Bir Todo güncelle
`PATCH /api/todos/{id}/`	Bir Todo'nun bir kısmını güncelle
`DELETE /api/todos/{id}/`	Bir Todo sil