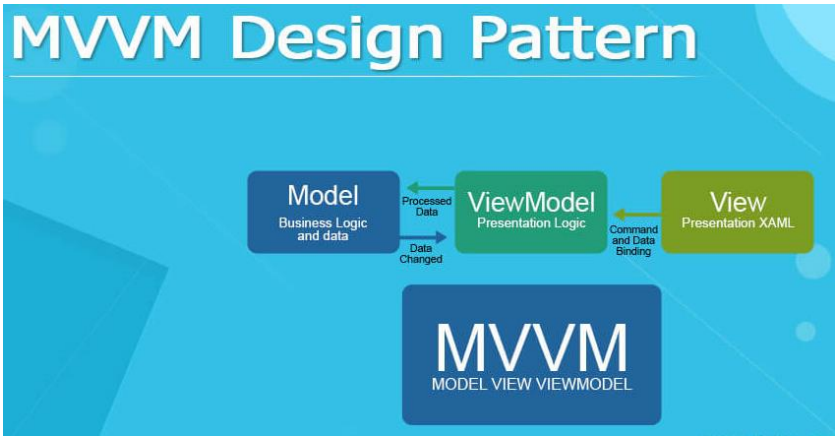
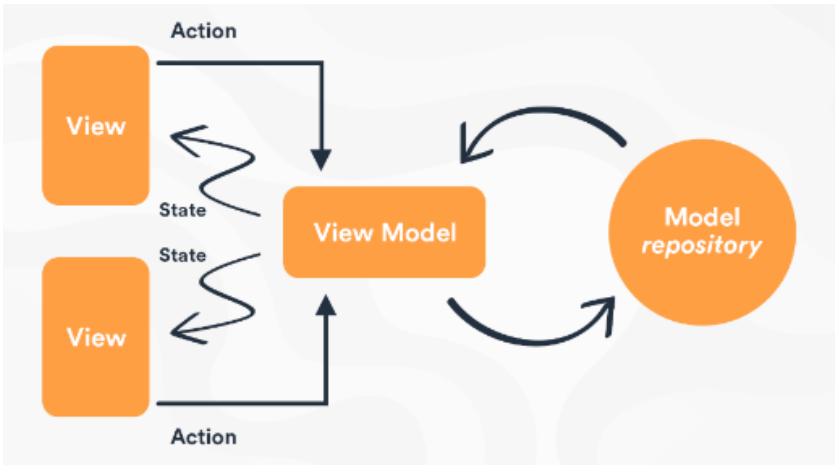
Mimari nedir ve neden ihtiyaç duyulur?

Mimarilerin amacı uygulamayı hızlı geliştirebilmek, gelecekte rahat güncelleyebilmek ve kodun rahat okunabilmesi ve temiz olmasıdır. Yazılım mimarisine ihtiyaç duymamızın en önemli sebeplerinden biri de sistemin karmaşıklığını yönetmek ve bütünlüğünü korumak için pratik bir yapı sunmasıdır.

Mimariler, üst seviye soyutlama yaparak geliştirilecek sistemi üst seviyeden daha iyi anlamamıza olanak sağlar. Detaylardan soyutlanarak karmaşık sistemleri daha iyi analiz etmemizi sağlar.

1) Uygulama Mimarisi: MVVM (Model–View–Viewmodel)





MVVM üç temel bileşenden oluşur. Bunlar; model, view ve viewmodel'dir.

Model: Veri tabanına erişim, veri tabanı ilişkileri gibi data ile ilgili işlemleri bulunduran, data(veri) işlemlerinin gerçekleştiği, veritabanı veya API ile haberleşen servislerimizin bulunduğu kısımdır. Uygulamamızın tüm hizmetlerini ve modellenen verileri çağırma süreçlerini içerir.

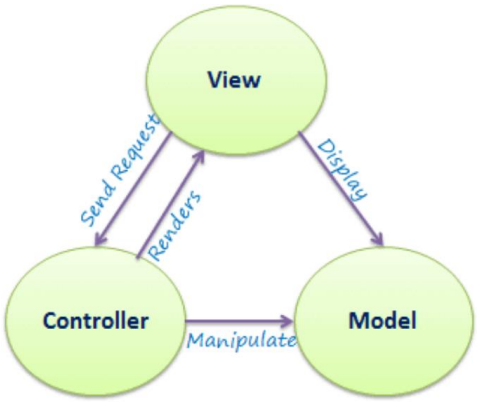
View: Uygulamamızdaki tüm tasarım yapılarının bulunduğu bölümdür. Kullanıcılar tarafından görülen son arayüzdür. Yazdığımız uygulama ile kullanıcı arasındaki bağlantıyı sağlar.

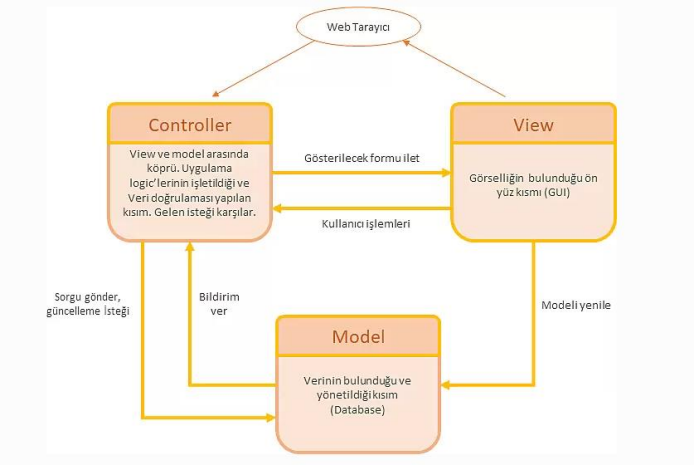
ViewModel: ViewModel, model yapısını view yapısına bağlayan ve uygulamanın mantıksal akışını ve state bilgisi tutan kısımdır. ViewModel yapısı doğrudan view'e erişemez ancak ViewModel, view'e erişebilir. ViewModel yapısı, model yapısına erişebilir ancak model yapısı, ViewModel yapısına erişemez.

MVMM'nin bir "Separation of Concerns" yapısı vardır. Bu sayede uzun ve karmaşık kodlarla uğraşmak zorunda kalmazsınız. Birçok yapıyı birbirinden ayırarak kullanabilirsiniz.

MVVM, iş mantığını ve arayüzü ayırır. Böylelikle tasarımcı, tasarımını yaparken geliştirici de arka işlemleri yapabilir. Bu, uygulamanın geliştirilmesini ve test edilmesini kolaylaştırır. Model tarafında değişiklik yapmak istediğinizde UI tarafında herhangi bir değişiklik yapmadan iyileştirmeler yapabilirsiniz.

2) Uygulama Mimarisi: MVC (Model–View–Controller)





MVC Mimarisi, kodumuzun farklı amaçlara hizmet eden yapılarını birbirinden ayırarak, kodu daha rahat geliştirilebilir ve test hale getirmemize yardımcı olur. Yani kod yapımız daha az hata çıkartma potansiyeline ulaşır. Kullanıcı arayüzlerini birbirinden ayrıştıran ve de uygulamanın çok farklı amaçlara hizmeter veren bölümlerinin birbirine girmesini önleyen bir yazılım mimarisidir.

MVC üç temel bileşenden oluşur. Bunlar; model, view ve controller'dır.

Model: Veri tabanına erişim, veri tabanı ilişkileri gibi data ile ilgili işlemleri bulunduran, data(veri) işlemlerinin gerçekleştiği, veritabanı veya API ile haberleşen servislerimizin bulunduğu kısımdır. Uygulamamızın tüm hizmetlerini ve modellenen verileri çağırma süreçlerini içerir.

View: Bu katman kullanıcının ekranda gördüğü katman olarak adlandırılır. Bu kısımda Html, Css, Javascript ara yüz teknolojileri kullanılır ve işin en keyif veren müdahalelerin yapıldığı kısımdır.

Controller: Kullanıcıların view üzerinden verdiği komutların Controller aracılığı ile model işlenmesini sağladığı katmandır, yani view ve model arasında kalan katmandır. Metotlar ve fonksiyonlar bu katmanda çağırılarak kullanılır.

MVC'nin birbirinden bağımsız oluşu en kullanışlı yönüdür. Şöyle açıklayabiliriz ki, ileride projemizin kullanıcı katmanında görsel bir değişiklik yapmak istersek sadece görünümü değiştirecek kısımla yani View ile ilgileniriz. Spagetti kodla uğraşmak zorunda kalmayız.

MVC, genel olarak web uygulaması geliştirmek amacı ile kullanılmakta olan bir sistemdir.

Büyük ve kompleks projelerde Front End, Back End ve Database gibi projelerin farklı kısımlarında görev yapacak geliştiricilerin yaptığı işlerin biribirine karışmaması gerekir. Bu durumda mvc mimarisini kullanarak rahatlıkla takım çalışmaları yapabiliriz.