FoodOrderingProject

Bülay Ecem Çakallı

Bu proje Full Stack Development kullanılarak geliştirilmiştir. Full Stack Development;

1. Frontend
2. Database
3. Backend

olmak üzere 3 ana bileşenden oluşur, bu üç katmanı etkili bir şekilde oluşturma yeteneğine de zaten Full Stack Development denir.
Bu projemin amacı da bu üç ana bileşeni tam anlamıyla anlayıp düzgün anlaşılır bir proje oluşturmaktır.
Projenin gidişatı şu şekildedir:

Frontend Başlangıç Süreci

İlk olarak frontend kısmının presentation tarafını MVVM (Model-View-ViewModel) yapısı kullanarak geliştirdim. View katmanıyla başlayarak ekranları ve temel dialogları ekledim. Mock Data kullanarak arkada bu dataları yönlendirdiğim geçici bir ViewModel oluşturdum. Böylece projenin genel görsel yapısını oluşturmuş oldum ve projenin bütünsel tasarımı zihnimde netleşti.

Veritabanı Tasarımı

Kullanacağım temel entity'ler belirginleştikten sonra veritabanı oluşturmaya karar verdim. FoodOrdering adlı basit bir veritabanı kurdum. Veritabanını kurduktan sonra fark ettim ki ANSI olacak şekilde kurmamışım, bundan ötürü bir daha kurmak zorunda kaldım. ANSI kullanmak istememin nedeni, diğer SQL platformlarıyla uyumundan ötürü.

Backend Geliştirme Süreci

Veritabanı kurulumundan sonra Web Service katmanına geçiş yaptım. Projenin üç ana bileşeni arasında tam anlamıyla bitirdiğim ilk kısım Web Service oldu. Bu katmanı oluştururken Clean Architecture ve Vertical Slice Architecture yaklaşımlarına benzer yaklaşımlar kullandım.

Architecture and Approaches - Backend

Tam anlamıyla saf bir Clean Architecture (Presentation, Application, Domain ve Infrastructure katmanları) kullanmadım. Ancak Core, Feature ve Infrastructure olmak üzere 3 ana katmandan oluşan, Clean Architecture'ı andıran bir yapı oluşturdum. Ek olarak Program.cs dosyası, uygulamanın başlangıç noktası olarak görev yapmaktadır. Database Controller ise veritabanı bağlantısını sağladığı için diğer katmanlardan ayrı bir konumdadır. Feature klasöründe her bir entity için yeni klasörler oluşturdum ve sonrasında onların feature'larını da bu klasör içinde tekrardan ayırdım. Postman'da endpoint'lerimi test ettim ve test başarı ile sonuçlandığında Frontend tarafına tekrardan geçiş yaptım.

Frontend Geliştirme Süreci

Web servisinin hazır olmasıyla birlikte, frontend tarafında Mock Data kullanımından gerçek veritabanı bağlantısına geçiş sürecini tamamladım. Bu değişiklik için öncelikle API bağlantılarını yönetecek ApiClient ve ApiService adlı iki dosyayı data/remote kısmına ekledim. Bu sayede Retrofit kullanarak bağlantıyı kurdum. Bundan sonra Repository pattern'ini ve bu repository'lerin implementation'larını oluşturarak data'yı projeme çekebildim. Ayrıca app/di (dependency injection) kısmında use case'leri kullanarak metodları kullanılabilir hale getirdim ancak tam anlamıyla istenilen bir DI modeline ulaşamadım. Constructor düzeyinde kaldı, genel olarak use case'leri kullandığım kısımlarda, data tekrardan oluşmakta. Bu da istenmeyen gereksiz bir şey aslında ancak proje devam edilirse ilerleyen süreçte DI rahat bir şekilde kullanımının kodda biraz değişmesiyle düzeltilebilir. Ayrıca, DTO (Data Transfer Object) çevirme işlemi de mapper sınıflarında yapılmakta. Sonrasında encryption eklemek adına ConfigHelper adlı bir sayfa daha oluşturdum ve bunun sonucunda AES256 kullanarak sensitive bilgileri güven altına almış bulundum.

Workflow

Projeyi açtığınızda karşınıza startup page gelmekte. Bu sayfada database ile bir bağlantı sağlamadan mainscrollable page'e, yani asıl sayfaya girememektesiniz. Database configuration işlemi öncesi default bir şifre sorulmakta. Bu şifre doğru girildiğinde configscreen adlı bir sayfaya yönlendiriliyorsunuz. Bundan sonra doğru girildiği takdirde otomatik olarak mainscrollable page açılmakta. Ayrıca ilk girişte back button bulunmakta (startup page'e), ama

sonrasında bulunmamakta. Mainscrollable page'de girdiğinizde confighelper'da zaten datalar loadalldata ile birlikte yüklenmişti. Şimdi burada birkaç filtreleme yapılıyor. Search bar'ın altındaki butonlarla search bar'da adı üstünde spesifik yiyecek ya da içecek aramak için var. Herhangi bir yiyecek ya da içecek seçilme durumunda sağ altta bir market arabası butonu çıkmakta. Üstünde ne kadar yiyecek, içecek vs. aldığınızın sayısı görülmekte. Bu market arabası butonuna basıldığında CartPage'e yönlendiriliyorsunuz. Bundan sonra ilerlerseniz PaymentMethodDialog. Creditmedcenter ve multinet'te payment başarılı olmakta, kalanlarda olmamakta. Bu yapılan 'order'lar da DB'ye gönderilmekte. Hali hazırda çalışan bir workflow'um olduktan sonra biraz daha detaya girdim. Encryption eklemeye karar verdim. Bunun sonucunda ConfigHelper adlı bir file daha açtım.

Son Eklemeler

Fark ettim ki benden istenilen, başta startup page'de database bağlantısının olmadığını göstermek sonucu config'e gitmek değil de, başta config açılıp orada şifre belirlenmip sonra o config'i tekrar açmak için sorulan dialog'a o şifreyi girmekmiş. Kodumu bu şekilde düzelttim ve baştaki start up page yapısını değiştirmiş oldum MainActivity'de bu da yaptığım son eklenti oldu. Sonucunda projemin bu üç katmanını bitirmiş bulundum.

İlerleyen Süreç

İlerleyen süreçte bahsettiğim DI implement edilebilir. Bunun yanında DevOps katmanı üstünde çalışılabilir. Ayrıca Unit Test yapılabilir.

Hangi Cihazlarda Kullanılabilir - SDK Level Uyumluluk

Proje geliştirilirken farklı Android SDK seviyelerine uyumluluk göz önünde bulundurulmuştur. Minimum SDK seviyesi API Level 21 (Android 5.0 Lollipop) olarak belirlenerek geniş bir kullanıcı kitlesine ulaşılması hedeflenmiştir. Target SDK seviyesi API Level 34 (Android 14) olarak güncel tutularak Google Play Store gereksinimlerine uygun olması sağlanmıştır. Bu sayede Android 5.0 ve üzeri tüm Android cihazlarda sorunsuz çalışabilmektedir. Ayrıca deprecated API'lerin kullanımından kaçınılarak gelecekteki Android sürümlerine uyumluluk sürdürülmesi planlanmıştır.