

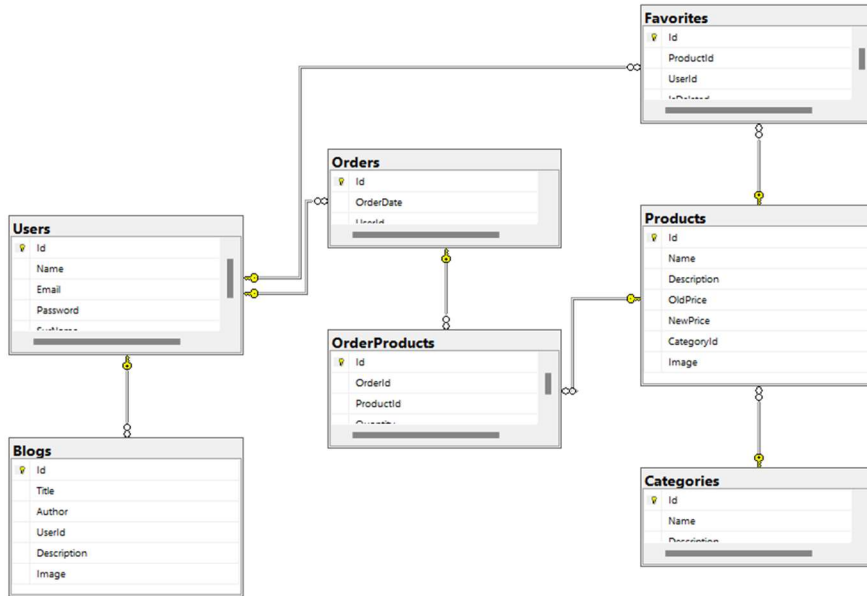
Özet

Çalışma ASP.NET Core API kullanarak MVC mimarisi ile servisler aracılığı ile API üzerinden ve aynı zamanda entityframework kullanılarak MSSQL DB 'den gelen verilerin gerekli kontrolleri yapılarak ön yüze ulaştırma amacıyla hazırlanmıştır.

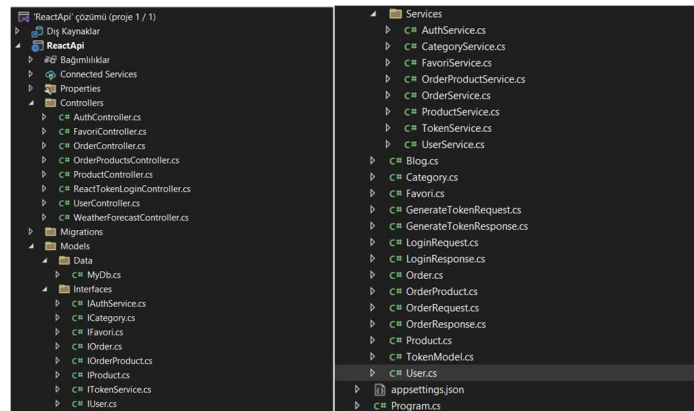
Database Modeli

MyDb adlı bir DbContext sınıfı kullanılarak çeşitli modeller arasında ilişkiler kurulmuş ve veritabanındaki tabloların temsili yapılmıştır. Bu sınıf, uygulamanın veritabanına erişim sağlar.

Database Diyagramı



Çözüm Gezgini



Proje MVC modelinde oluşturulmuştur. Db güvenliğini sağlamak için Interface'ler oluşturularak servis yapıları kurulmuştur.

Proje API isteklerine cevap vermek üzere geliştirilmiştir. Bu noktada swagger UI aracı veya postman aracı ile API isteklerinin düzgün çalışması kontrol edilmiştir.

Örnek vermek gerekirse ürünlerin ön yüze gitmesi için hazırlanan product servisinin yapısı şu şekildedir:

Interface

```
6 başvuru
public interface IProduct
{
    2 başvuru
    Task<List<Product>> GetAllProductsAsync();

    4 başvuru
    Task<Product> GetProductByIdAsync(int id);

    1 başvuru
    Task<bool> AddProductAsync(Product product);
    1 başvuru
    Task<bool> UpdateProductAsync(Product product);
    1 başvuru
    Task<bool> DeleteProductAsync(int id);
}
```

Service

```
2 başvuru
public class ProductService : IProduct
{
    MyDb db;

    0 başvuru
    public ProductService(MyDb db)
    {
        this.db = db;
    }

    1 başvuru
    public Task<bool> AddProductAsync(Product product) {...}

    1 başvuru
    public async Task<bool> DeleteProductAsync(int id) {...}

    2 başvuru
    public Task<List<Product>> GetAllProductsAsync() {...}

    4 başvuru
    public Task<Product> GetProductByIdAsync(int id) {...}

    1 başvuru
    public async Task<bool> UpdateProductAsync(Product product) {...}
}
```

Burada interface servise implamente edilerek her bir metotun içeriği doldurulmuştur.

GetAllProduct() controller sayfası ise aşağıda verilmiştir:

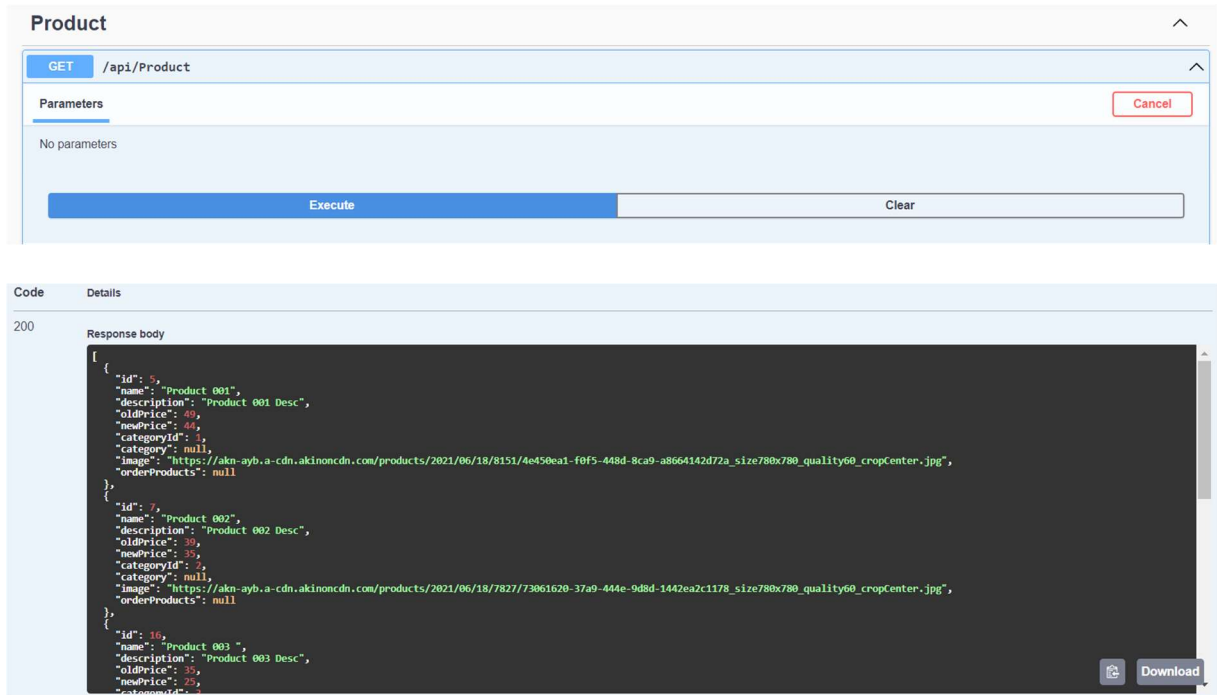
Controller

```
[HttpGet]

0 başvuru
public Task<List<Product>> Get()
{
    return _product.GetAllProductsAsync();
}
```

[HttpGet] sayesinde url üzerinden tüm ürünler çekilebilmektedir.


Tüm API metotları bu şekilde oluşturulduktan sonra swagger UI ve postman araçlarında test gerçekleştirilmiştir.



Figürde görüldüğü üzere Code:200 ile response dönmüştür. API'den başarılı bir şekilde ürünler çekilmiştir.

React UI

Login Karşılama Ekranı

[Login](#) [SignUp](#)


Login

Welcome back! Please enter your username and password to login.

☐ I have read and accept the user agreement

LOGIN

- Login Yapıldıktan Sonra Giriş Yapan Kullanıcının Sayfa Ekranı

[Shop](#) [MyPage](#) [Logout](#)

Hoşgeldin senes

Dashboard

Orders

Logout

My account

Dashboard

Hello, [senes](#) (If Not [senes!](#) [Logout](#))

From your account dashboard, you can easily check & view your recent orders, manage your shipping and billing addresses and edit your password and account details.

- Orders Butonuna Basılınca kullanıcının geçmiş siparişlerini takip edebiliyoruz.

Dashboard

Orders

Logout

My account

Orders

OrderId	Date	Detail
1	2024-01-10T16:05:48.81	<button>Go</button>
2	2024-01-10T23:37:55.164014	<button>Go</button>
3	2024-01-10T23:40:50.8236183	<button>Go</button>

- Ardından her bir siparişin detayını inceleyebiliyoruz.

Dashboard

Orders

Logout

My account

Orders

Product Id	Quantity	Go Back
7	2	<button>Back</button>

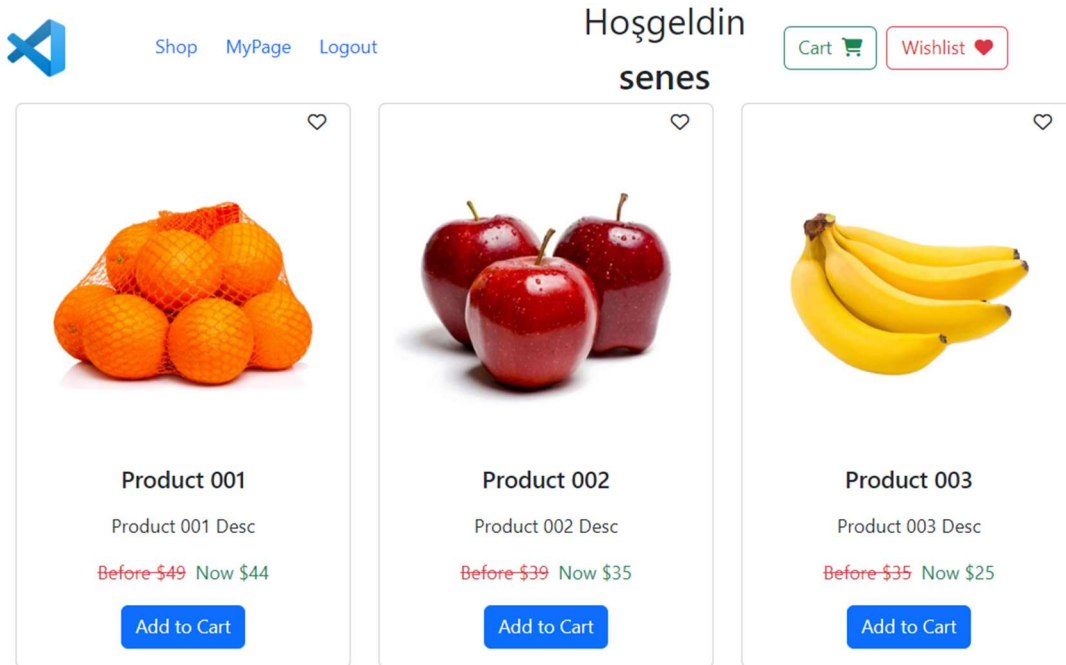
Burada giriş yapan kullanıcı kontrolü backend tarafından gelen token ile kontrol edilmektedir. Aşağıdaki figürde alınan token konsolda gösterilmektedir.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWw Login.js:27
zb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWItcy9uYW11Ijoic2VuZXMlLCJlbWVpYyI6
InNlbnVzIiwibmJmIjoxNzA2MTgxMDQ5LCJleHAiOjE3MDYxODI4NDksImZcyI6ImJ1YmVuaW1pc
3N1ZXIiLCJhdWQiOiJidWJlbnltYXVkaWVuY2UiQ.ktFGQc-6KxSsI-
t1ONEi_ypMVuxkTPUQdv6la5fUrHo
```

Bu konsol çıktısı Login sayfasının 27. Satırındaki fonksiyonun çıktısıdır.

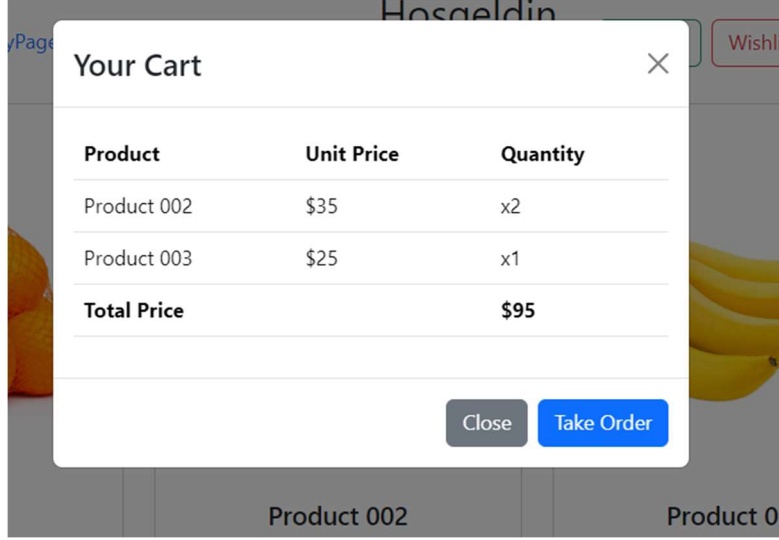
Alınan token ve user bilgisi sessionda tutulmaktadır. Böylece her bir çalışma zamanında user bilgisi ve token güncellenecektir.

Shop Page

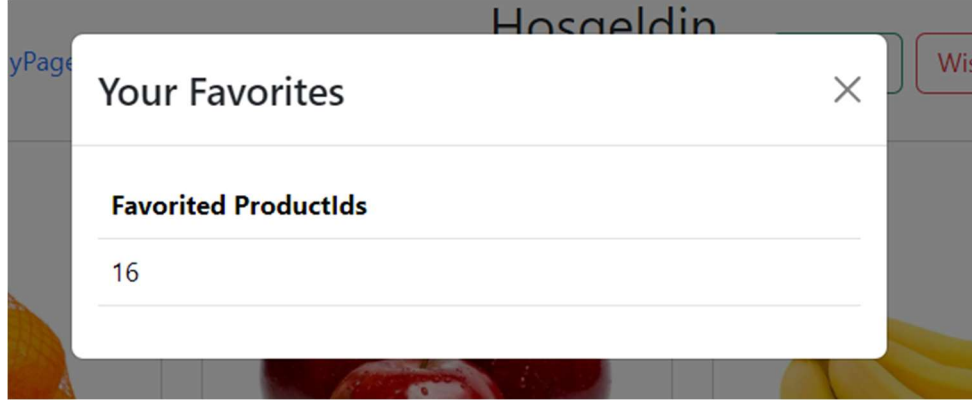


Figürde görüldüğü üzere giriş yapan kullanıcı olduğunda bu sayfaya gidilebilmektedir. Add to Cart fonksiyonu kullanıldığında navbardaki cart güncellenmektedir. Aynı zamanda her bir ürün cartının içerisindeki kalp butonu sayesinde ürünün favori durumu güncellenebilmektedir.

Navbardaki cart butonuna tıklandığında ortaya çıkan modalda gerekli bilgiler bulunmaktadır.



- Burada take order butonuna tıklandığında giriş yapan kullanıcı sipariş oluşturmaktadır.



- Burada ise navbarda bulunan wishlist butonuna basılınca görünen modala yer verilmiştir. Kullanıcının favori ürünlerinin ID'si listelenmektedir.

Karşılaşılan Zorluklar

Backend'den ön yüze token gönderimi yapılırken JWT kullanılmıştır. Gerekli konfigürasyonların yapılması ve fetch ile API'den dönen token verisinin session'da tutulması çalışma yapılırken en zorlanılan alanlardır.

Tartışma

Bu proje sonucunda API servisleri oluşturma, MVC modelinin kullanma ve asenkron fonksiyonların geliştirilmesi alanlarında bireysel gelişim sağlanmıştır. Aynı zamanda React

tarafında fetch fonksiyonunun farklı metotlarda(post,get,put) kullanılması sağlanmıştır. Bu noktada ortaya fullstack bir proje çıkartılmıştır.