

**BEGINNER  
FULL-STACK  
BOOTCAMP SUNUMU**

**ASP .NET CORE  
E-TİCARET  
BİTİRME PROJESİ**



**Serhat KARADAĞ**

# PROJENİN AMACI

Bu e-ticaret projesi, modern web teknolojileri ve endüstri standartlarına uygun yazılım pratiklerini kullanarak güvenli, ölçeklenebilir ve kullanıcı dostu bir alışveriş deneyimi sunmayı amaçlamaktadır. Projenin temel hedefleri:

1

## Kullanıcılar için sorunsuz bir alışveriş deneyimi sağlamak

- Ürün arama, filtreleme ve sayfalama
- Güvenli ödeme işlemleri (Stripe entegrasyonu)
- Şifre resetleme (Mail entegrasyonu)

2

## Adminler için etkili bir yönetim platformu oluşturmak

- Ürünler için CRUD işlemleri ve stok kontrolü
- Ürünler için fotoğraf yükleme (Cloudinary ile)
- Gerçek zamanlı bildirimler (SignalR ile)

3

## Yüksek performans ve güvenlik standartlarını karşılamak

- Onion Architecture ile sürdürülebilir ve test edilebilir kod
- Redis önbelleğe alma ile hızlı yanıt süreleri
- ASP.NET Core Identity ile güvenli kimlik doğrulama











4

## Ölçeklenebilir ve bakımı kolay bir sistem oluşturmak





- Docker konteynerizasyonu ile kolay dağıtım
- Microservice mimarisine geçiş için uygun altyapı
- Kapsamlı loglama (Serilog ile)

# KULLANILAN TEKNOLOJİLER

## Backend

- Framework :  ASP .NET Core
- Database :  Entity Framework Core,  Microsoft SQL Server
- Authentication / Authorization : ASP .NET Core Identity (Cookie-Based Auth, Role Based Auth)
- Caching :  Redis
- Logging :  Serilog
- Real-Time Communication :  SignalR
- Containerization :  Docker
- APIs & Services :  Stripe,  Cloudinary,  Mail Service

## Frontend

- Framework :  Angular
- Styling :  Tailwind CSS,  Angular Material
- State Management :  Angular Signals

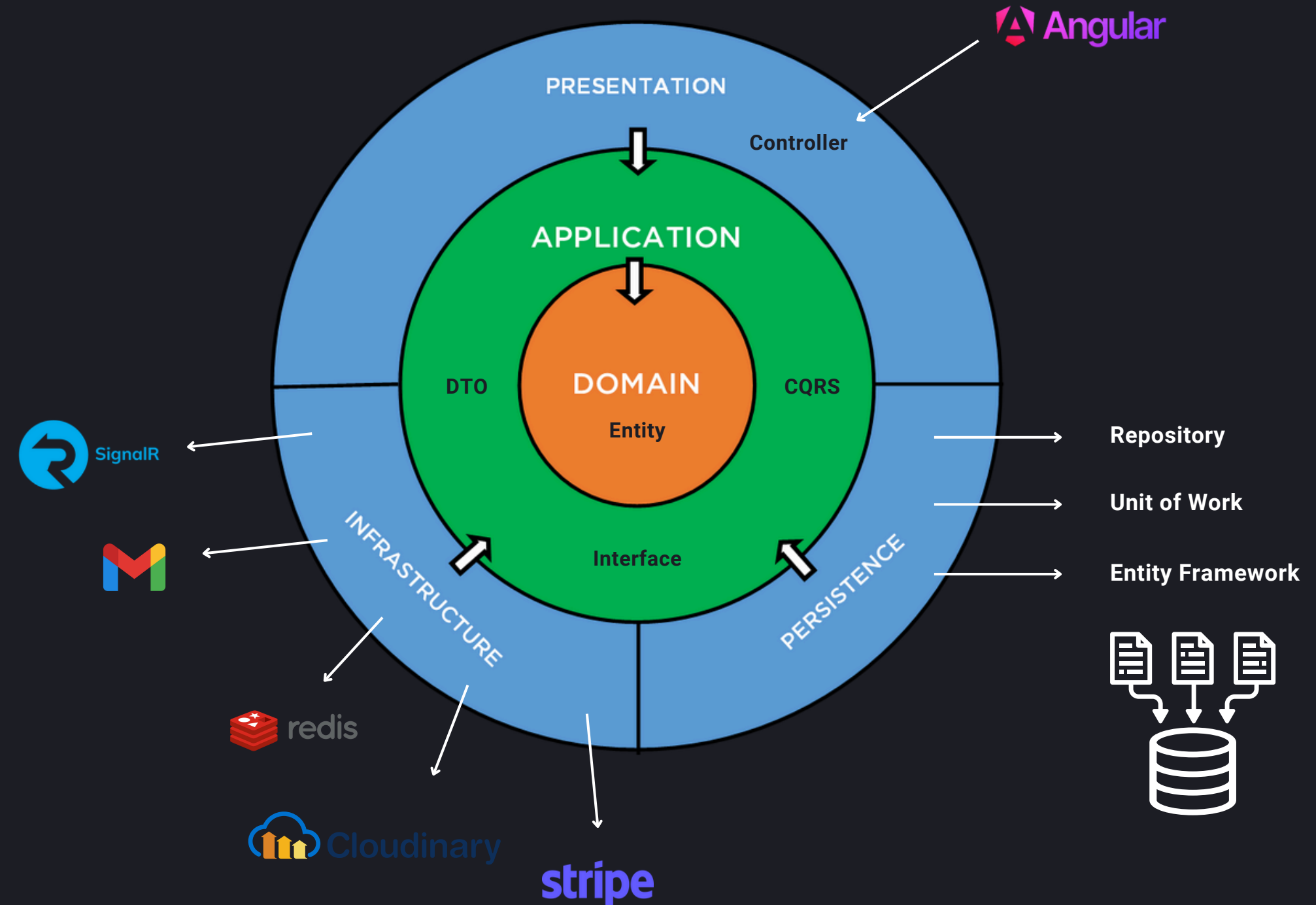
# SİSTEM MİMARİSİ VE TASARIM DESENLERİ

## Sistem Mimarisi

- Onion Architecture

## Tasarım Desenleri

- Generic Repository Pattern
- Unit Of Work Pattern
- CQRS / Mediator Pattern



# FRONTEND'DE NELER VAR?



Light - Dark Mode



Reusable Components



Angular Reactive Forms



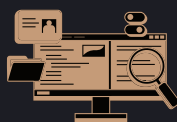
Auth - Admin Guards



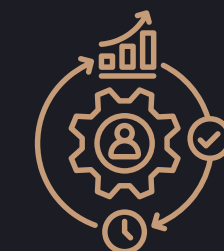
Dialog, Loading, Notification Service



Lazy Loading



Modern UI Styling

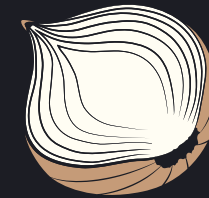


State Management (Angular Signals)

# BACKEND'DE NELER VAR?



CRUD Operations



Onion Architecture



Sorting, Filtering, Searching, Pagination



Global Error Handling



Authentication / Authorization



Real Time Communication (SignalR)



File Upload (Cloundinary)



Caching, Logging, Payment, Mail Service

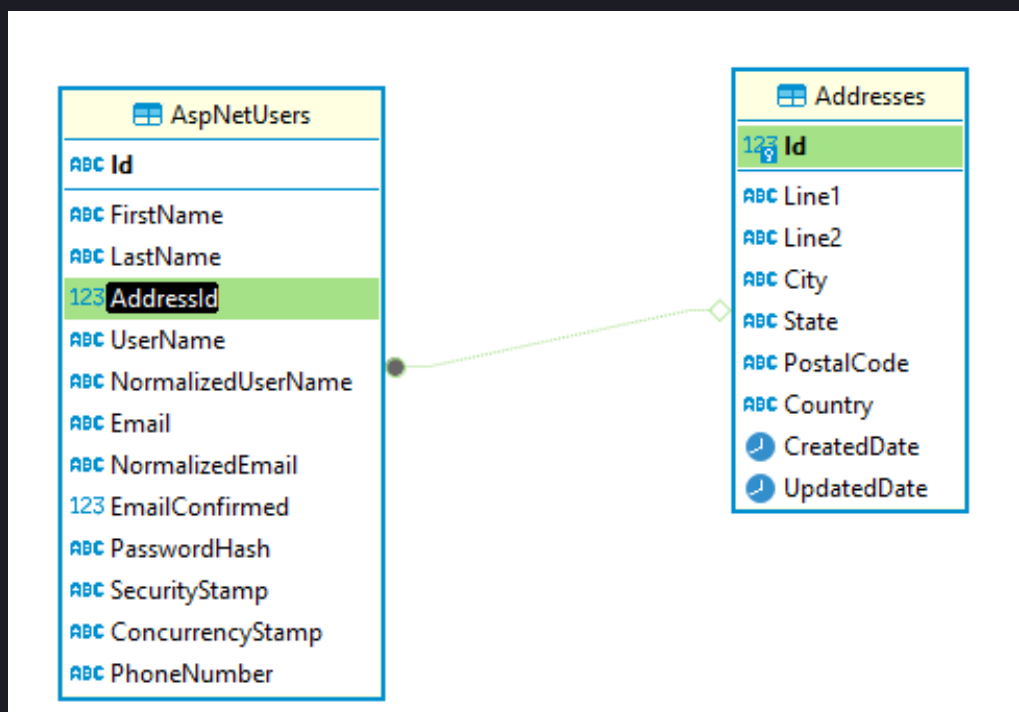
# .NET CORE IDENTITY (COOKIE-BASED AUTHENTICATION)

## Register

Register

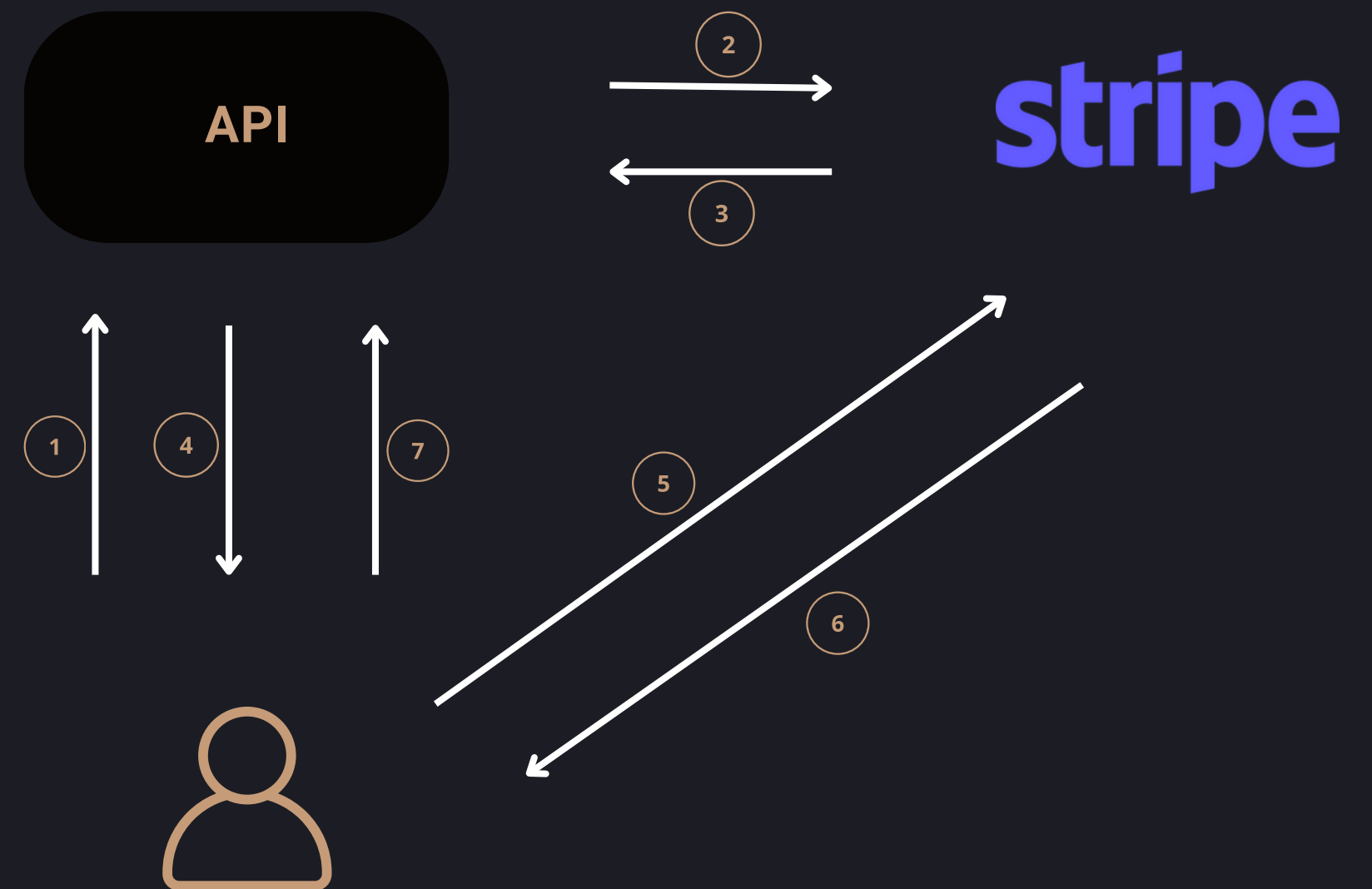
- .NET Core Identity ile cookie-based authentication kullanılarak kullanıcı kimlik doğrulama işlemleri güvenli bir şekilde yönetildi.
- User ve Address arasında 1-1 ilişki kuruldu. User ödeme yaptığında adres bilgilerini kaydedebilir.

```
public class AppUser : IdentityUser
{
    2 references
    public string? FirstName { get; set; }
    2 references
    public string? LastName { get; set; }
    6 references
    public Address? Address { get; set; }
}
```



# STRIPE (PAYMENT SERVICE)

- 1 Create 'payment intent' with API
- 2 API sends 'payment intent' to Stripe
- 3 Stripe creates 'payment intent' returns 'client secret'
- 4 API returns 'client secret' to client
- 5 Client sends payment to Stripe using the 'client secret'
- 6 Stripe sends confirmation to client, payment was successful
- 7 Client creates order with API (not implemented yet)





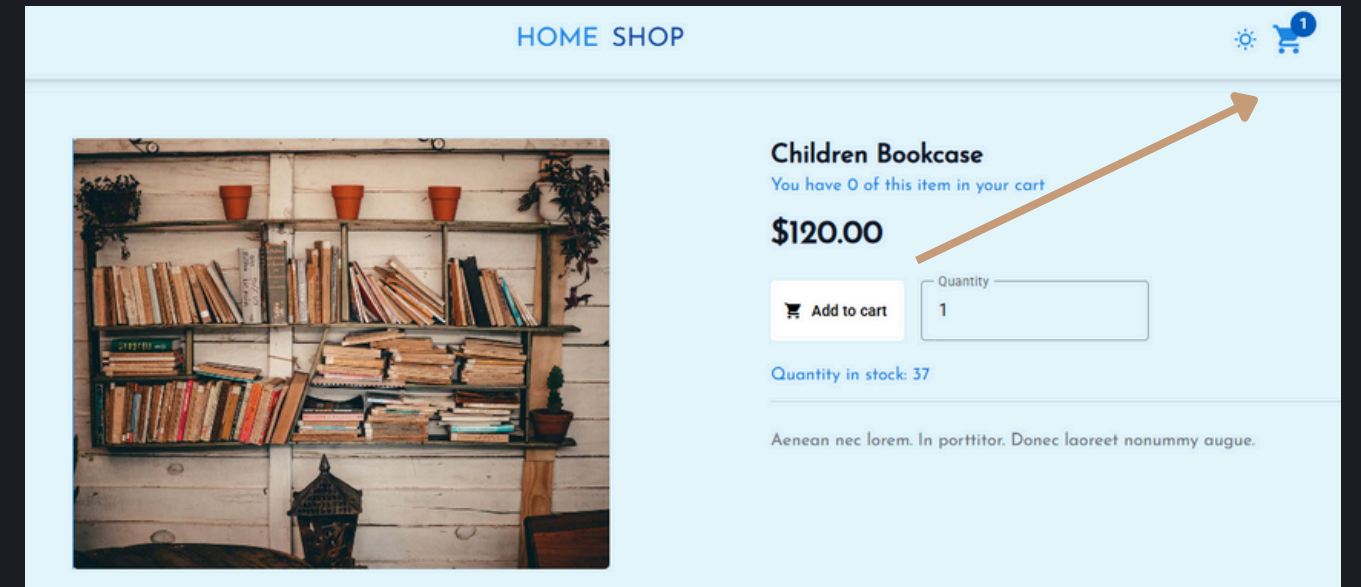
# REDIS (IN-MEMORY DATABASE / CACHING)

Projede Redis 2 farklı amaç için kullanıldı.

1

## In-Memory Database

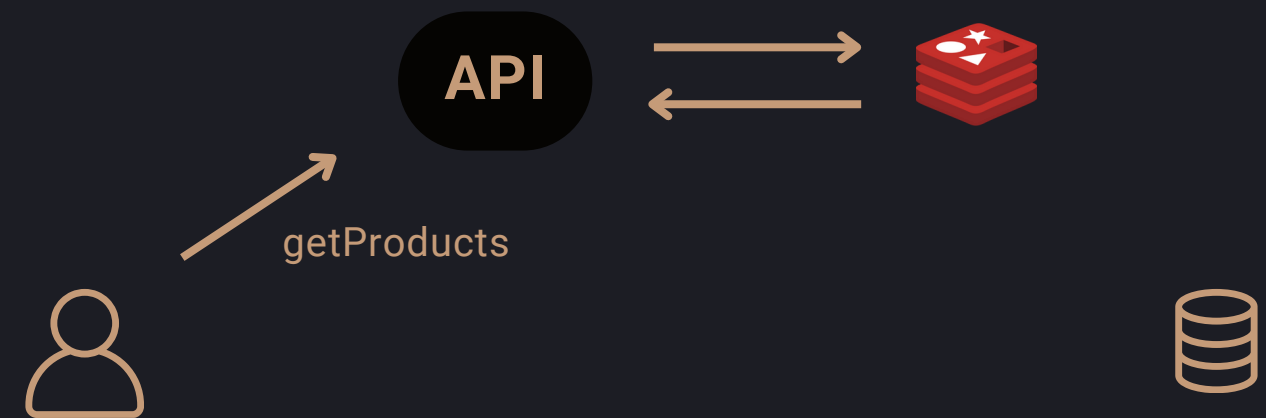
Kullanıcı 'Add to Cart' butonuna bastığında sepet bilgileri Redis'e kaydedilir.



2

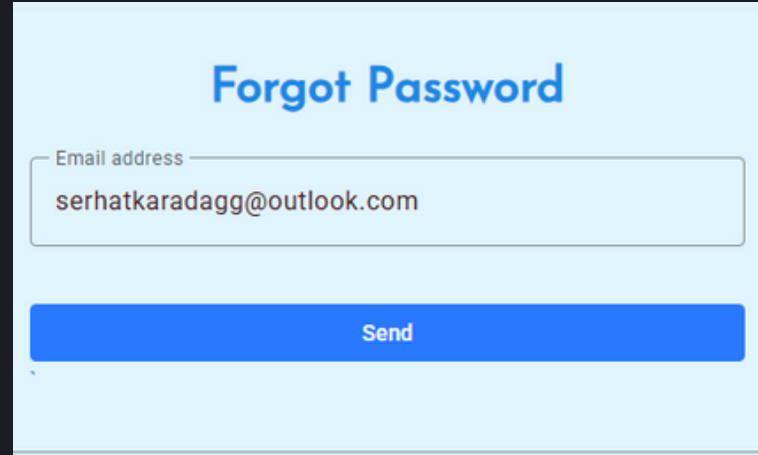
## Caching

Sık kullanılan veriler Redis'e cache'lenir, veritabanına yapılan okuma istekleri azalır.



# MAIL SERVICE

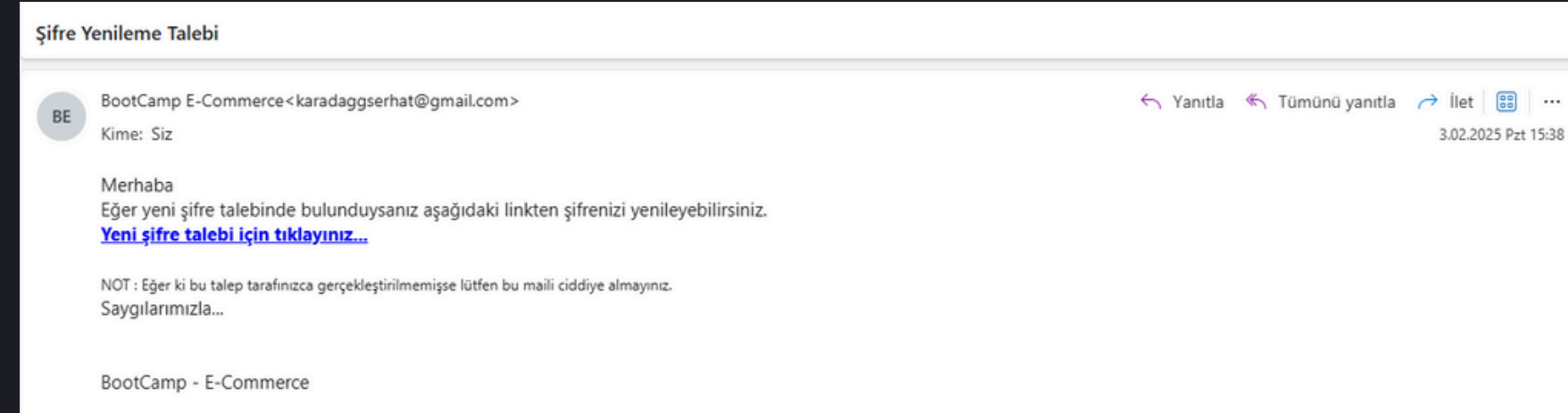
1



A light blue form titled "Forgot Password". It contains a text input field labeled "Email address" with the value "serhatkaradagg@outlook.com". Below the input field is a blue button labeled "Send".

Kullanıcı şifresini unuttuğunda mail adresiyle istek atar.

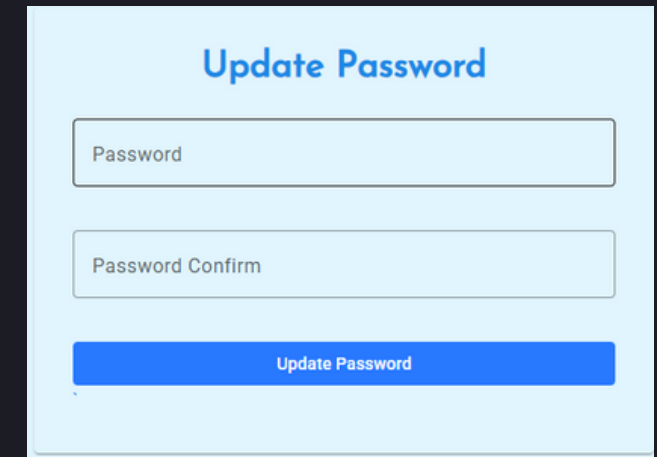
2



An email interface showing a message from "BootCamp E-Commerce <karadaggserhat@gmail.com>". The subject is "Şifre Yenileme Talebi". The email body says: "Merhaba, Eğer yeni şifre talebinde bulunduysanız aşağıdaki linkten şifrenizi yenileyebilirsiniz. [Yeni şifre talebi için tıklayınız...](#)". Below this is a disclaimer: "NOT : Eğer ki bu talep tarafınızca gerçekleştirilmemişse lütfen bu maili ciddiye almayınız. Saygılarımızla...". The footer says "BootCamp - E-Commerce".

Kullacının mail hesabına şifresini yenilemek için mail gönderilir. Tek kullanımlık reset token oluşturulur. Linke tıkladığında 'update-password' sayfasına yönlendirilir.

3

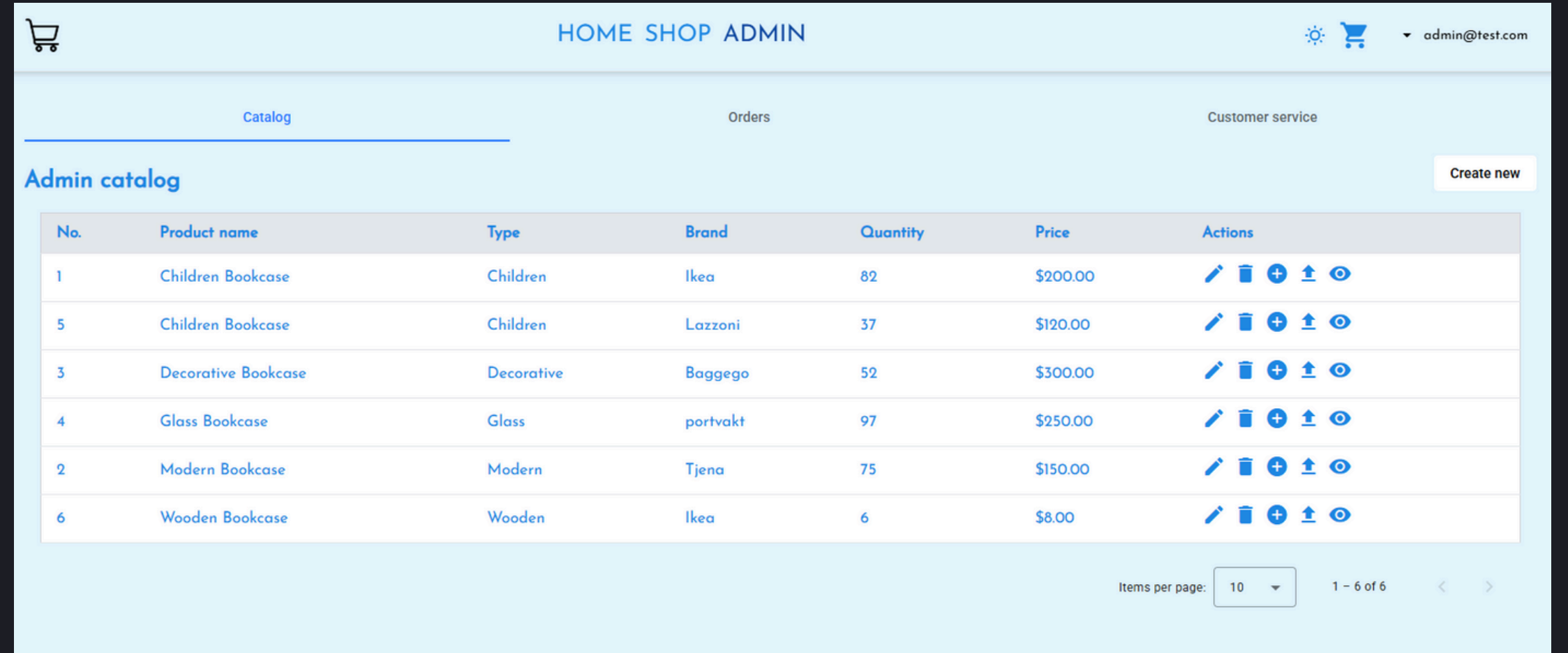


A light blue form titled "Update Password". It contains two text input fields: "Password" and "Password Confirm". Below these fields is a blue button labeled "Update Password".































Kullanıcı şifresini güncelleyebilir. Tek kullanımlık reset token sayesinde şifre güncellendikten sonra bu sayfada tekrar işlem yapılamaz.

# .NET CORE IDENTITY (ROLE-BASED AUTHORIZATION)

- .Net Core Identity kullanılarak Role-Based Authorization uygulandı. Projeye 'Admin' ve 'Customer' rolleri eklendi.
- Admin'ler ürünler için CRUD işlemlerini yapabilir. Ürünlere resim ekleyebilir ve ürün stoğunu güncelleyebilir.



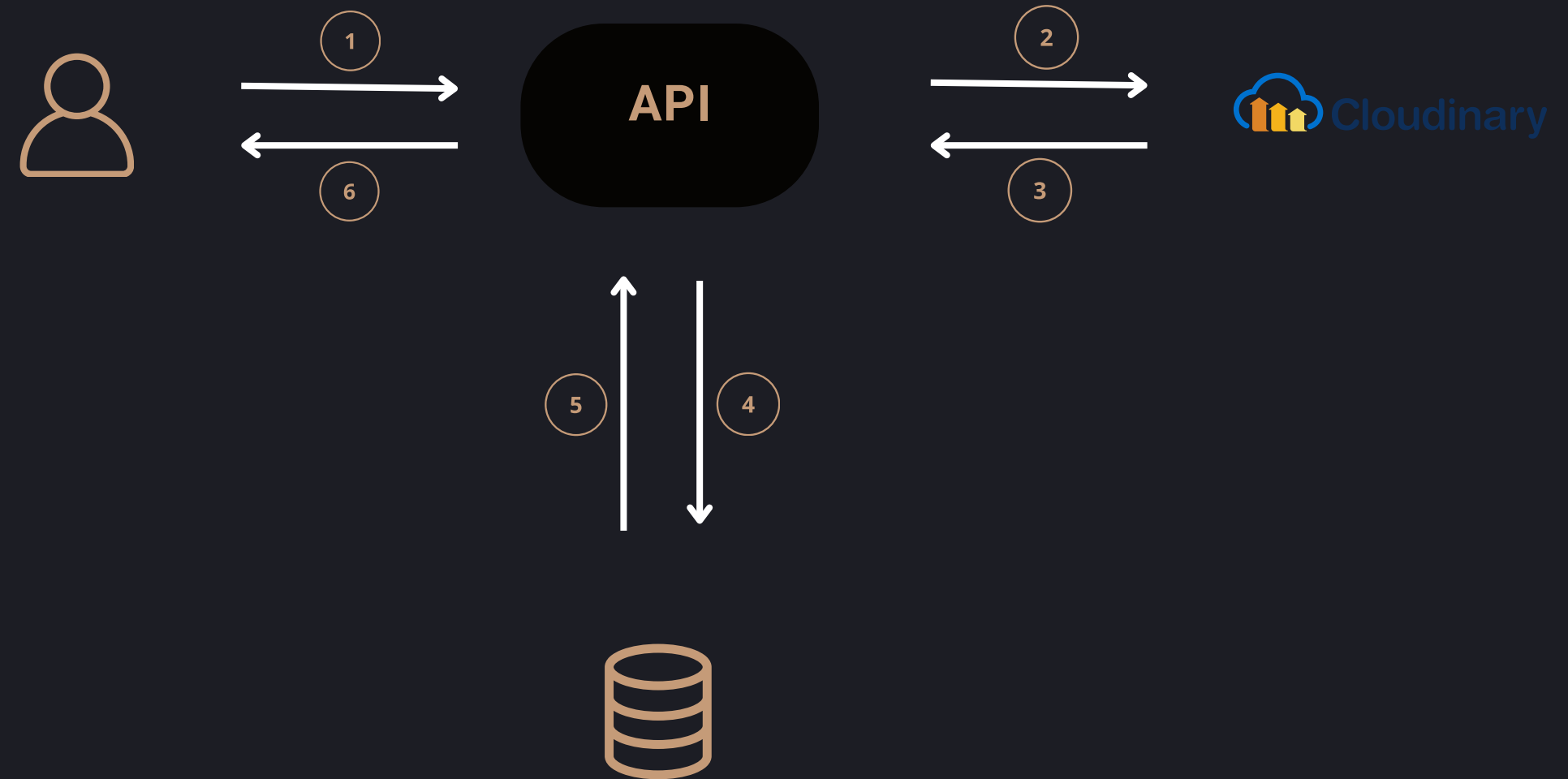
The screenshot displays the 'HOME SHOP ADMIN' dashboard. At the top, there's a navigation bar with a shopping cart icon, the title 'HOME SHOP ADMIN', a settings gear icon, a shopping cart icon, and a user profile dropdown for 'admin@test.com'. Below the navigation bar, there are three tabs: 'Catalog' (selected), 'Orders', and 'Customer service'. The 'Admin catalog' section features a 'Create new' button and a table with the following data:

No.	Product name	Type	Brand	Quantity	Price	Actions
1	Children Bookcase	Children	Ikea	82	\$200.00	    
5	Children Bookcase	Children	Lazzoni	37	\$120.00	    
3	Decorative Bookcase	Decorative	Baggego	52	\$300.00	    
4	Glass Bookcase	Glass	portvakt	97	\$250.00	    
2	Modern Bookcase	Modern	Tjena	75	\$150.00	    
6	Wooden Bookcase	Wooden	Ikea	6	\$8.00	    

At the bottom right, there's a pagination control showing 'Items per page: 10' and '1 - 6 of 6'.

# CLOUDINARY (PHOTO UPLOAD SERVICE)

- 1 Client uploads photo to API
- 2 Server uploads the photo to Cloudinary
- 3 Cloudinary stores photo, sends response
- 4 API saves photo URL and Public ID to Database
- 5 Saved in Database and given auto generated ID
- 6 API response sent to client



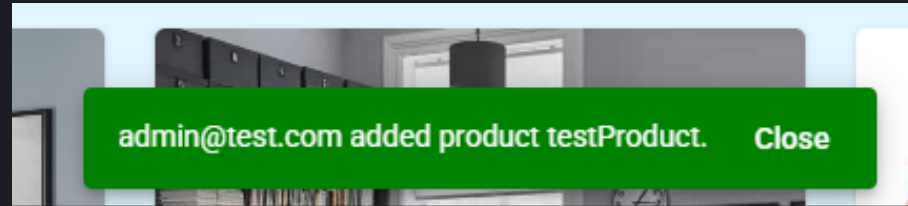
# SignalR (REAL TIME COMMUNICATION)

SignalR kütüphanesi sayesinde kullanıcılar arasında canlı bildirimler gönderilir.

Örneğin; admin@test.com isimli kullanıcı createProduct işlemini yaptıktan sonra diğer adminlere canlı bildirim gönderilir.

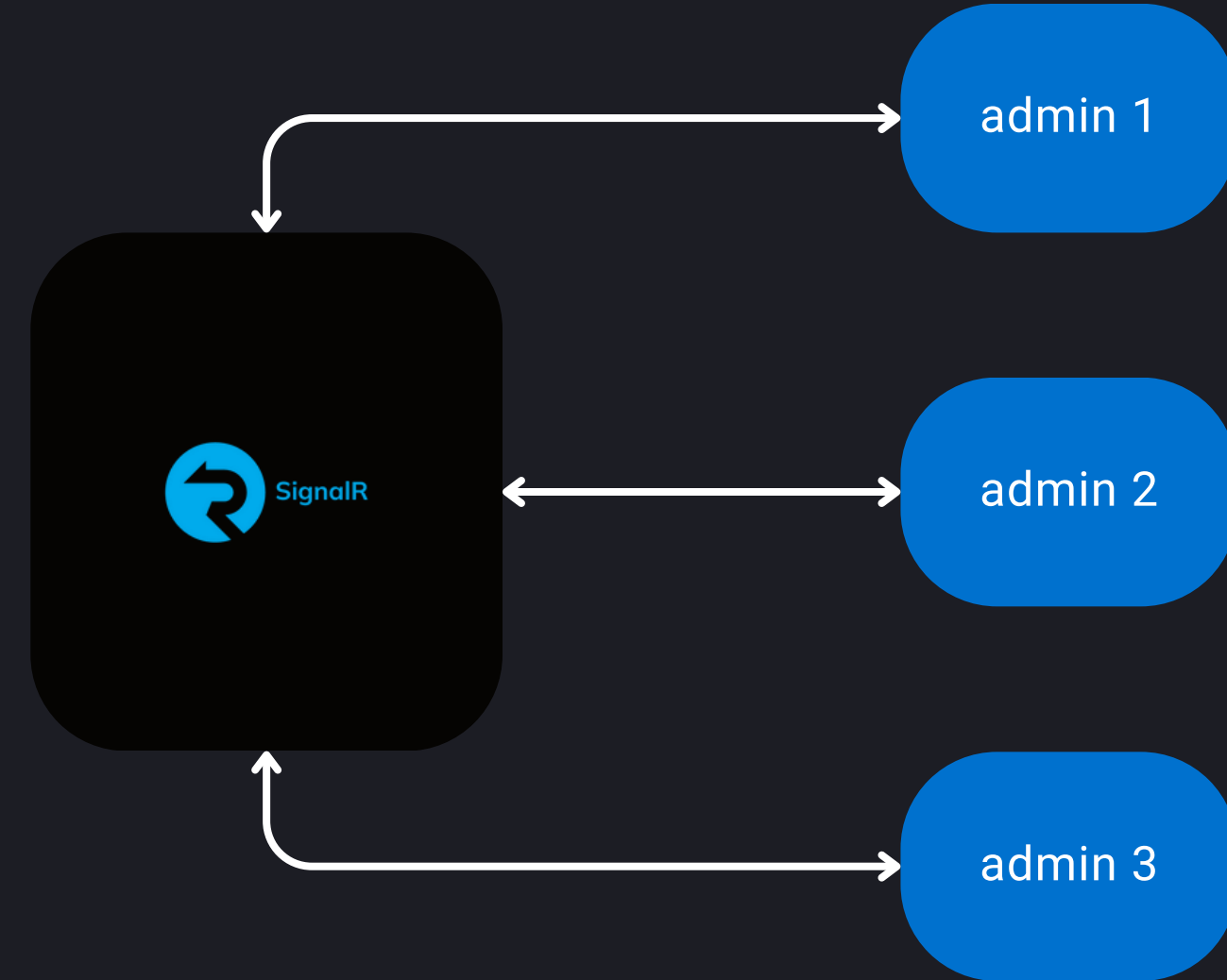


admin 1



admin 2

POST / CreateProduct



# LOGGING (SERILOG)

Serilog kütüphanesi sayesinde uygulamanın çalışma sırasında meydana gelen olayları ve hataları kaydederiz.

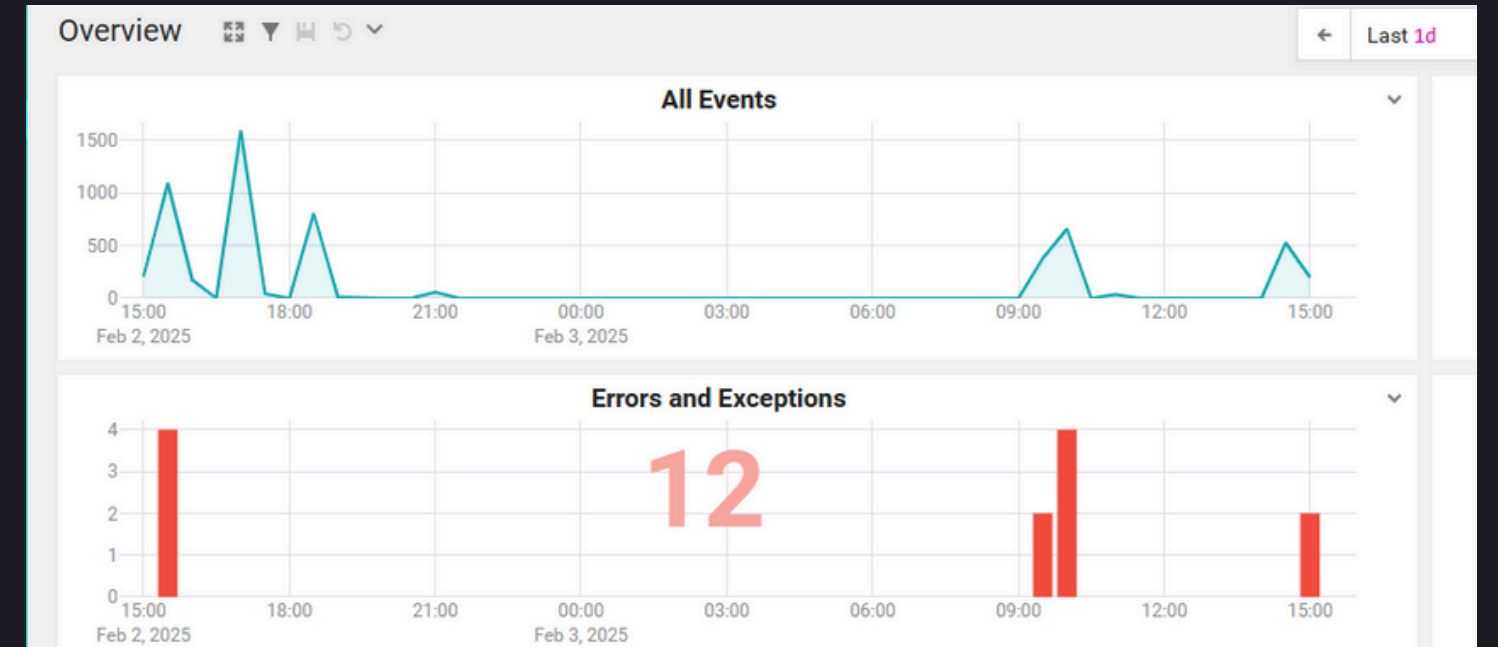
Global error handling ile uygulamadaki hataların merkezi bir yerde ele alınması sağlandı. Loglama sayesinde bu hataların detaylarını kaydederiz.

Yandaki görsellerde uygulamanın çalışma sırasında meydana gelen olayları ve hataları 3 farklı yolla kaydederiz.

- 1- Text dosyası
- 2- Database
- 3- Seq kütüphanesiyle görselleştirme

```
log.txt
Presentation > ECommerceBackend.API > logs > log.txt
26756
26757 2025-02-03 15:11:30.060 +03:00 [ERR] DuplicateUserName - Username 'serhatkaradagg@outlook.com' is already taken.
26758
26759 2025-02-03 15:11:30.062 +03:00 [INF] Request finished HTTP/2 POST https://localhost:5001/api/account/register - 400 nu
26760
```

Id	Message	MessageTemplate	Level	TimeSpan
6,585	DuplicateUserName - Username 'serhatkaradagg@outlook.com' is already taken.	DuplicateUserName - Username 'serhatkaradagg@outlook.com' is already taken.	Error	2025-02-03 15:11:30.060 +03:00
6,584	An unhandled exception has occurred while executing the request.	An unhandled exception has occurred while executing the request.	Error	2025-02-03 15:11:30.060 +03:00
5,318	DuplicateUserName - Username 'serhatkaradagg@outlook.com' is already taken.	DuplicateUserName - Username 'serhatkaradagg@outlook.com' is already taken.	Error	2025-02-03 15:11:30.060 +03:00
5,317	An unhandled exception has occurred while executing the request.	An unhandled exception has occurred while executing the request.	Error	2025-02-03 15:11:30.060 +03:00



# Projeye Neler Eklenebilir?

- **Responsive Design** (Mobil telefonlara uyumlu olacak bir tasarım eklenebilir.)
- **NgRx** (Signals ile state yönetimi sağlanıyordu, ancak proje büyüdükçe global state yönetimine geçerek merkezi bir kontrol sağlamak isteyebiliriz.)
- **Order Service** (Payment işlemlerinin sorunsuz çalışabilmesi için projeye Order ile ilgili yeni eklemeler yapılabilir.)
- **Testing** (Unit, Integration gibi testler kullanılarak projenin güvenilirliği, kalitesi ve bakımı sağlanabilir.)
- **Microservice geçiş** (Projede zaten CQRS pattern uygulanmıştı. Bunun üstüne, Command ve Query işlemleri için farklı veritabanları ve servisler kullanılabilir. Veri tutarlılığı için Eventual Consistency modeli kullanılabilir.)

**BEGINNER  
FULL-STACK  
BOOTCAMP SUNUMU**



**Serhat KARADAĞ**

**BENİ  
DİNLEDİĞİNİZ İÇİN  
TEŞEKKÜRLER**