

**Akdeniz University Computer Science**

**Advance Application Development Full Stack Project**

**Zayimhan Korkmaz - 20220808055**

<https://github.com/zayimhan/Full-Stack-Ecommerce-Project>

## **1. Introduction**

This project is a full-stack web application developed using **Angular** for the frontend and **Spring Boot** for the backend. The architecture follows a clear separation of concerns, where the frontend is responsible for user interface and interaction logic, while the backend handles business logic, data persistence, and security.

The Angular frontend leverages modular components, services, validators, and interceptors to provide a responsive and maintainable user experience. Each component is designed with a specific function, ensuring scalability and ease of maintenance.

On the backend, Spring Boot is used to build a robust and secure RESTful API. The backend structure includes clearly defined layers such as Controllers, Services, Repositories, and Entities. It incorporates best practices for business logic separation, data access with JPA, and JWT-based security configuration.

This combination of Angular and Spring Boot provides a powerful and scalable foundation suitable for modern web applications, ensuring a maintainable and testable codebase on both the client and server sides.

## 2. Frontend Architecture (Angular)

The frontend of the application is built using **Angular**, a robust and modular web framework that facilitates the development of scalable and maintainable user interfaces. The architecture emphasizes component-based development, service abstraction, route management, and form validation.

### Component Structure

The user interface is entirely composed of Angular components. Each component is a self-contained unit with its own HTML, CSS, and TypeScript files. The application's UI is organized into thematic domains such as **Cart**, **Auth**, **User**, **Product**, and **Seller**. These domain-based component groups improve code clarity, reusability, and ease of maintenance.

Angular's **routing system** enables seamless navigation between components, offering a fluid and user-friendly experience across different screens—such as product listings, login pages, cart views, and seller dashboards.

### Services

Angular services are the core units responsible for business logic and data management. They act as intermediaries between components and external APIs, ensuring components remain lightweight and focused solely on the presentation logic. For instance:

- `cart.service.ts` handles cart operations,
- `auth.service.ts` manages authentication workflows,
- `checkout.service.ts` oversees the order placement process.

This separation improves testability and code reusability.

### Interceptors and Validators

Cross-cutting concerns are handled using interceptors. For example, `auth-interceptor.service.ts` automatically appends JWT tokens to all HTTP requests, ensuring secure communication with the backend.

Form validation is centralized through custom validator classes like `shopValidators.ts`, which enforce frontend-level data integrity. These validators reduce

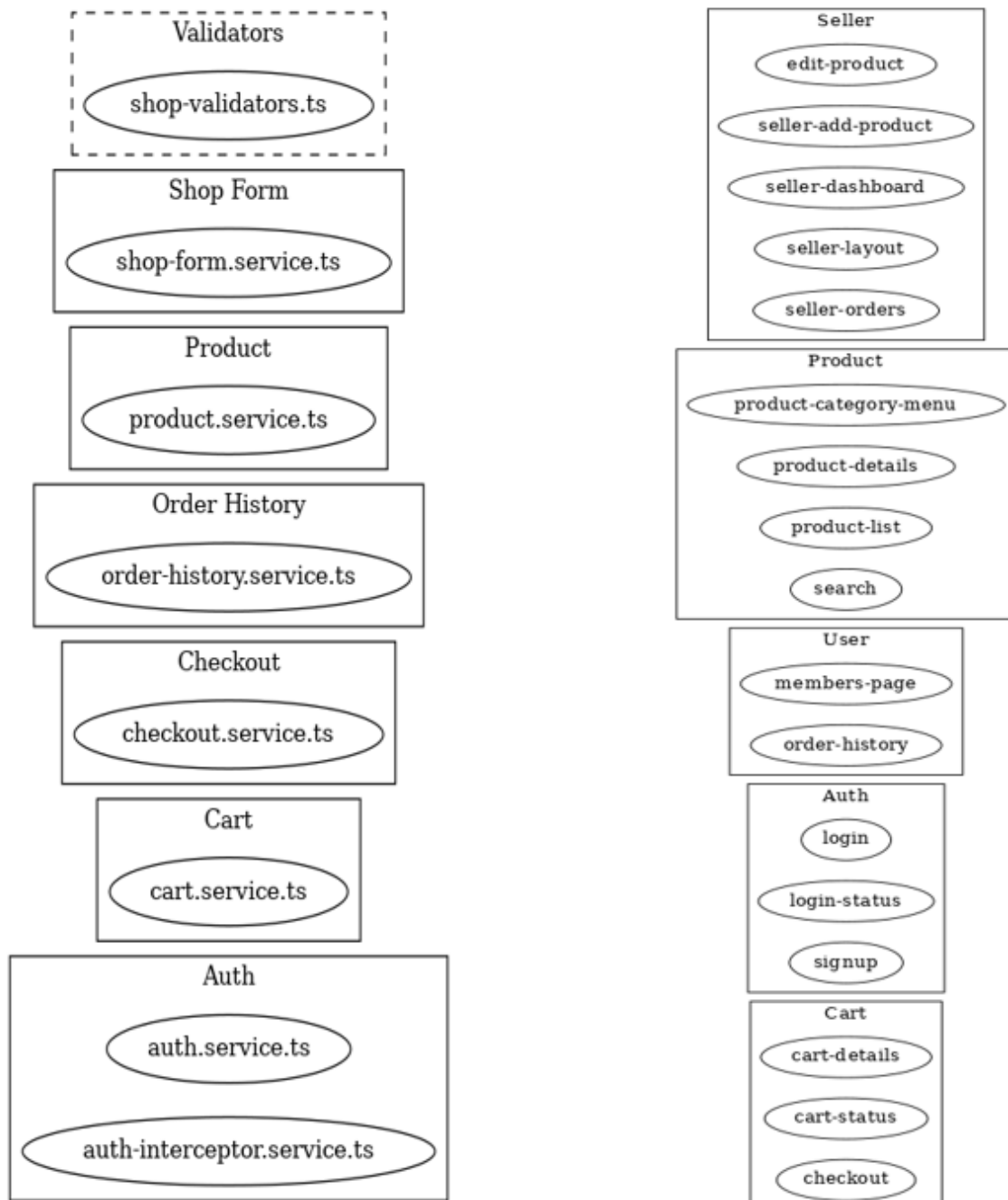
code duplication and strengthen input control mechanisms, improving the overall user experience.

## **Admin Module**

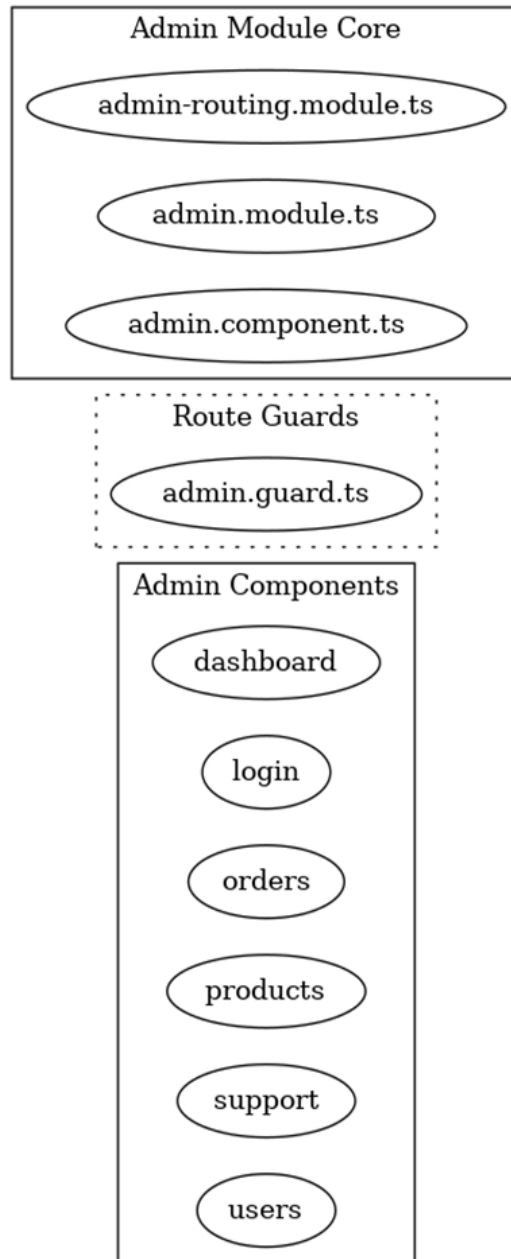
An administrative interface is developed as a dedicated module for system administrators. It is organized into three parts:

- **Admin Core:** Includes `admin.module.ts`, `admin-routing.module.ts`, and `admin.component.ts` to configure routing and layout.
- **Admin Components:** Functional components for dashboard, orders, products, support, and users.
- **Route Guards:** Such as `admin.guard.ts`, which restricts access to authorized users by verifying roles.

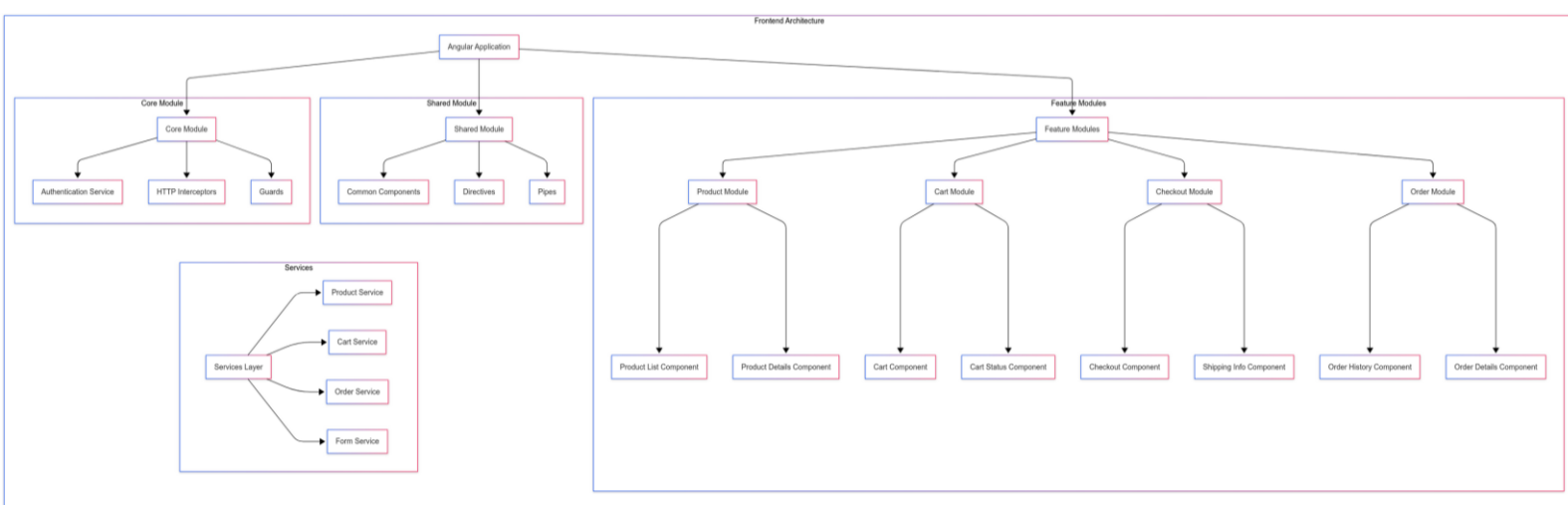
This modular approach ensures that the admin panel is secure, scalable, and easily maintainable.



## Listing All Frontend Files



## Listing All Frontend Files



## Frontend Architecture – Explanations

- **Angular Application:** The root of the Angular project, hosting all modules and components.

### Core Module

- **Core Module:** Contains essential services and infrastructure used across the entire application.
- **Authentication Service:** Handles user login, registration, and session management.
- **HTTP Interceptors:** Automatically attaches headers (e.g., JWT) to outgoing HTTP requests.
- **Guards:** Restricts access to routes based on user roles or authentication status.

### Shared Module

- **Shared Module:** Provides reusable components, directives, and pipes for the whole application.
- **Common Components:** UI components that are used across multiple feature modules.
- **Directives:** Adds custom behavior to HTML elements.
- **Pipes:** Transforms and formats data in templates.

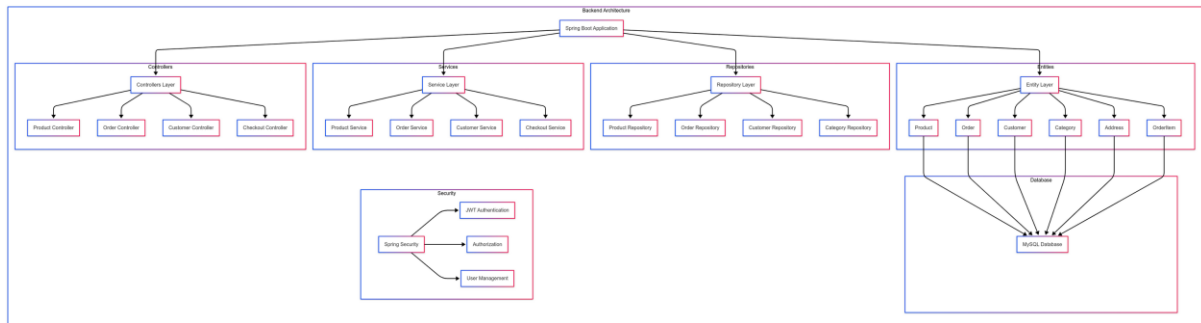
### Feature Modules

- **Feature Modules:** Encapsulates domain-specific functionality into modular units.
- **Product Module:** Manages product-related views and logic.
  - **Product List Component:** Displays a list of available products.

- **Product Details Component:** Shows detailed information about a selected product.
- **Cart Module:** Manages the shopping cart experience.
  - **Cart Component:** Lists products currently in the user's cart.
  - **Cart Status Component:** Shows a cart icon with the item count and total.
- **Checkout Module:** Handles the order completion process.
  - **Checkout Component:** Allows the user to finalize the purchase.
  - **Shipping Info Component:** Collects delivery address and shipping options.
- **Order Module:** Displays past orders and order details.
  - **Order History Component:** Lists all previous purchases made by the user.
  - **Order Details Component:** Provides itemized information about a specific order.

#### Services Layer

- **Services Layer:** Contains business logic and API interaction logic separate from components.
  - **Product Service:** Fetches and manages product data via APIs.
  - **Cart Service:** Handles operations like adding/removing items from the cart.
  - **Order Service:** Processes order creation and retrieval.
  - **Form Service:** Manages form input, validation, and submission logic.



## Backend Architecture – Explanations

- **Spring Boot Application:** The root backend application managing API endpoints, business logic, and data access.

### Controllers Layer

- **Controllers Layer:** Handles HTTP requests and routes them to appropriate services.
  - **Product Controller:** Manages endpoints related to product listing, creation, and updates.
  - **Order Controller:** Processes order-related requests such as placing and viewing orders.
  - **Customer Controller:** Handles user-related actions like profile management.
  - **Checkout Controller:** Manages the order confirmation and payment processes.

### Service Layer

- **Service Layer:** Implements core business logic and coordinates between controllers and repositories.
  - **Product Service:** Contains business rules for managing product operations.
  - **Order Service:** Processes order workflows, including status updates and totals.
  - **Customer Service:** Manages customer account information and authentication.
  - **Checkout Service:** Oversees the logic for finalizing and validating purchases.

### Repository Layer

- **Repository Layer:** Interfaces with the database using JPA for CRUD operations.
  - **Product Repository:** Performs database operations related to product entities.
  - **Order Repository:** Queries and saves order data to the database.



- **Customer Repository:** Interacts with tables containing customer records.
- **Category Repository:** Manages product categories in the data layer.

#### Entity Layer

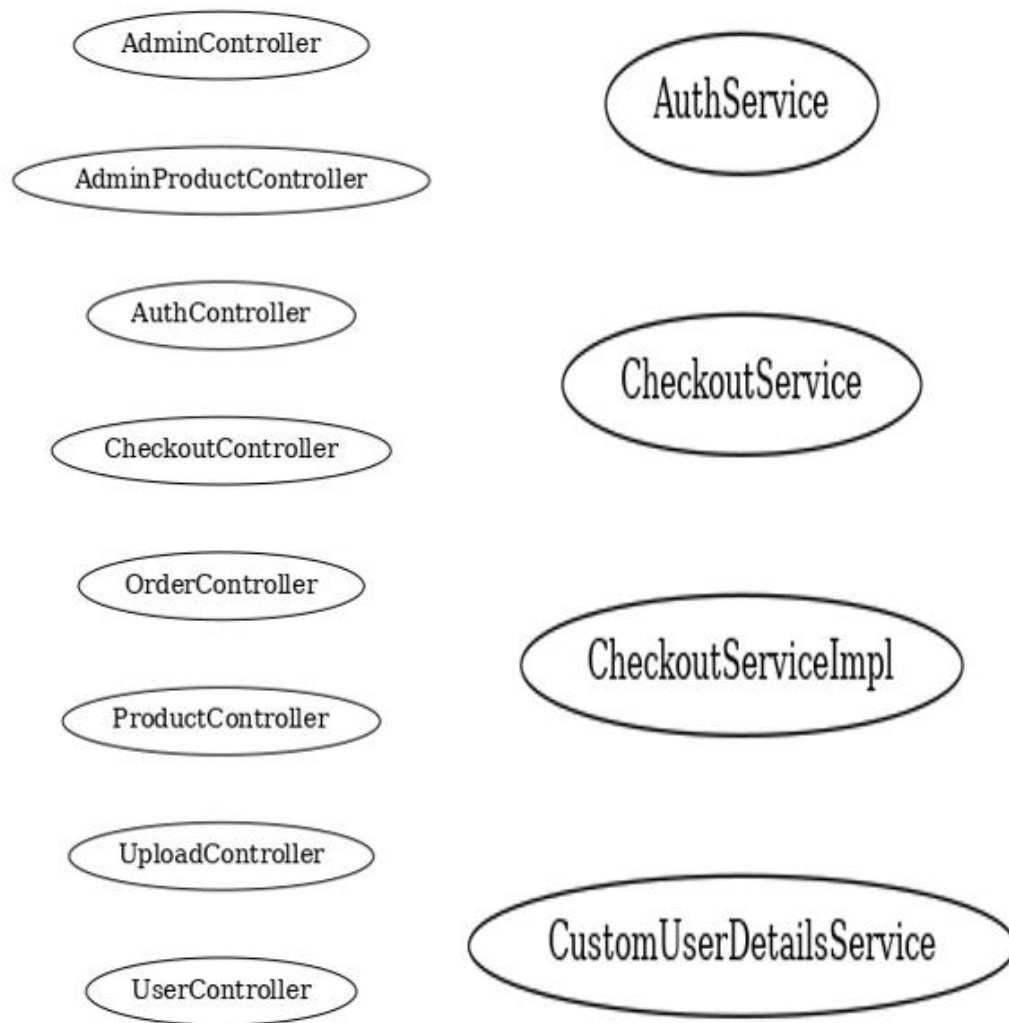
- **Entity Layer:** Represents database tables through Java classes using ORM.
  - **Product:** Entity representing a product listing.
  - **Order:** Represents a complete user order.
  - **Customer:** Entity for storing user account information.
  - **Category:** Defines a grouping or classification for products.
  - **Address:** Stores address-related information for orders and users.
  - **OrderItem:** Connects specific products to a particular order.

#### Database

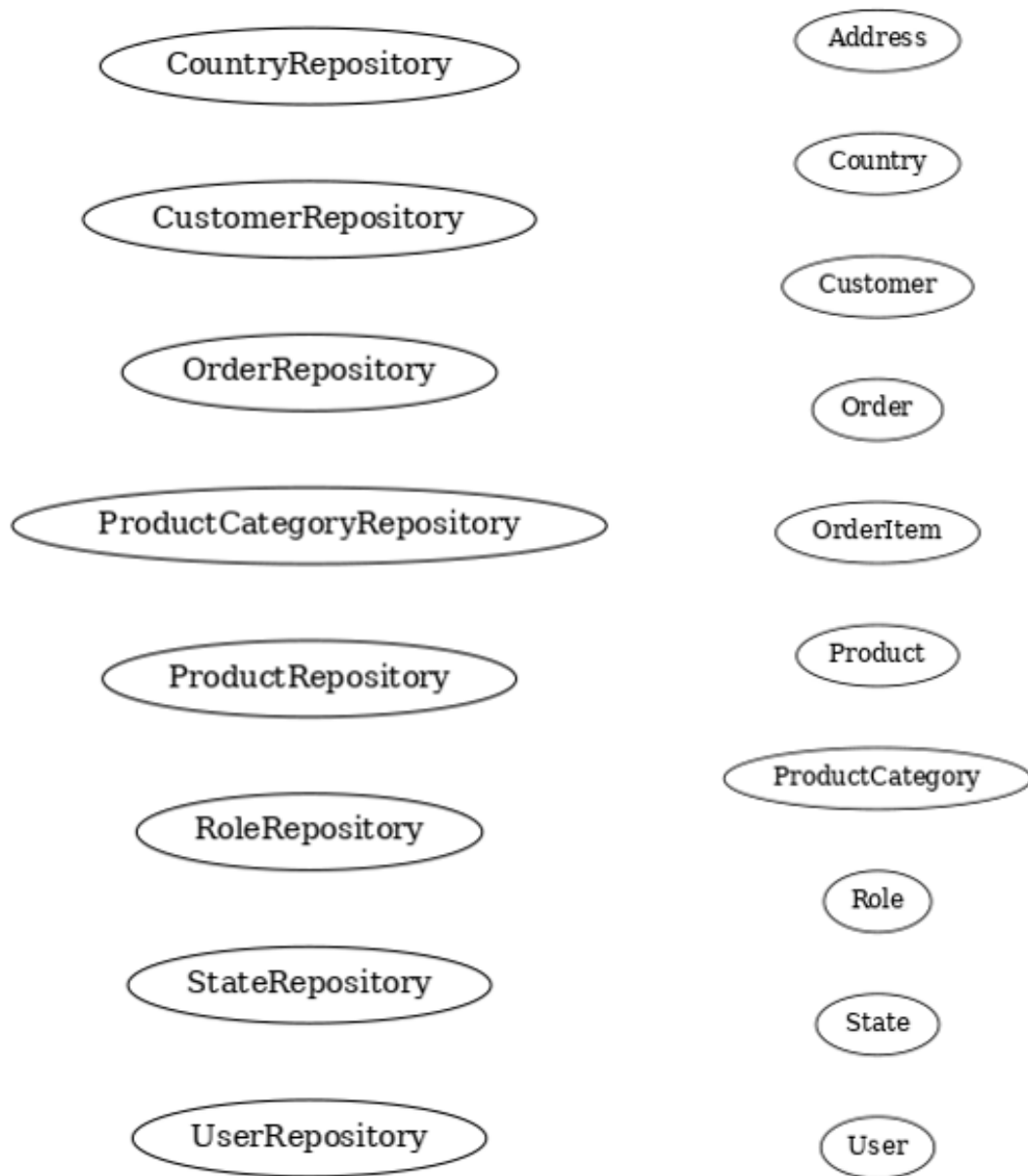
- **MySQL Database:** Stores all persistent data from the entities; used by repositories for querying.

#### Security Layer

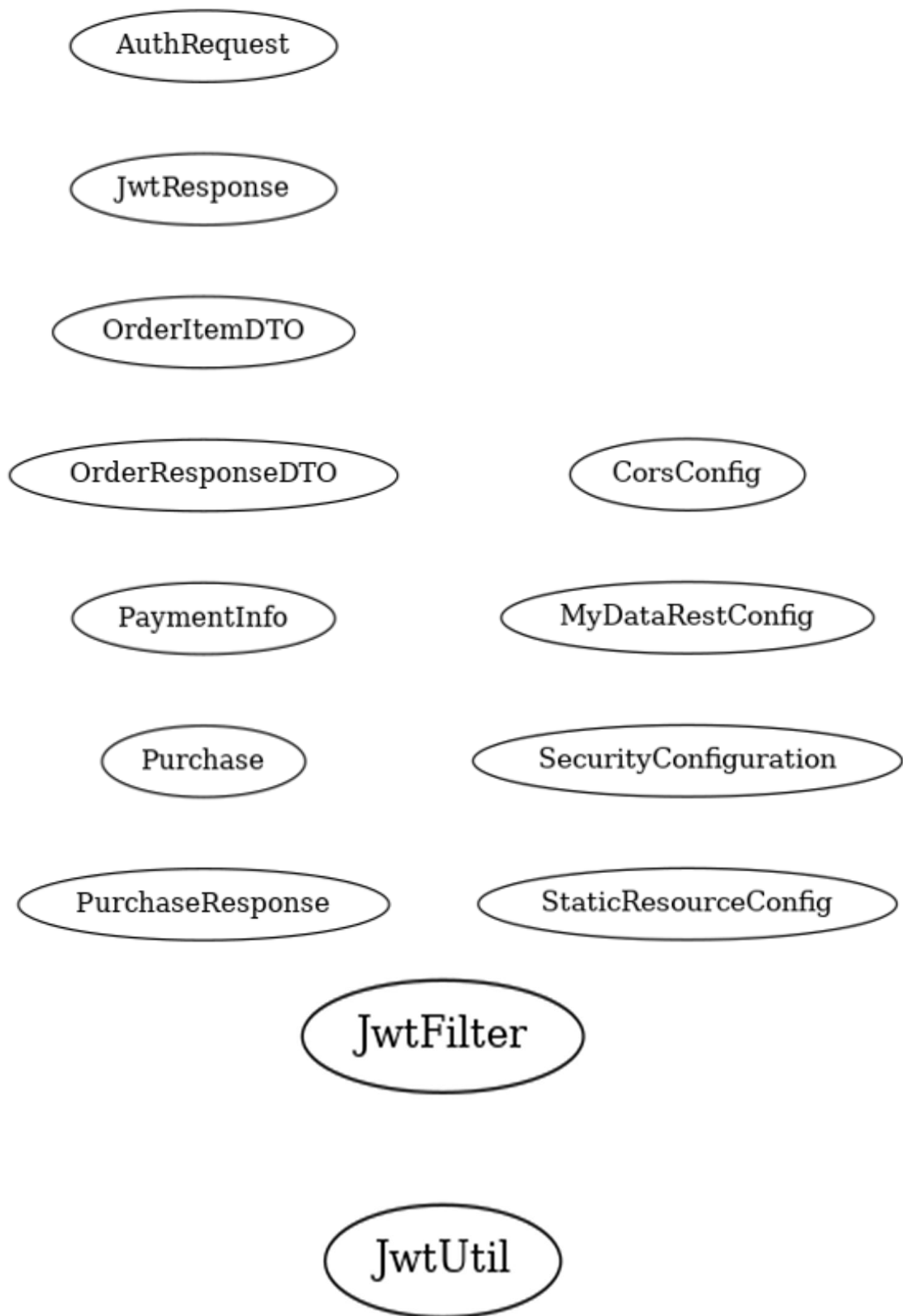
- **Spring Security:** Provides authentication and access control across the backend.
  - **JWT Authentication:** Uses JSON Web Tokens to authenticate users.
  - **Authorization:** Enforces access rules based on user roles or scopes.
  - **User Management:** Handles user login credentials and access rights.



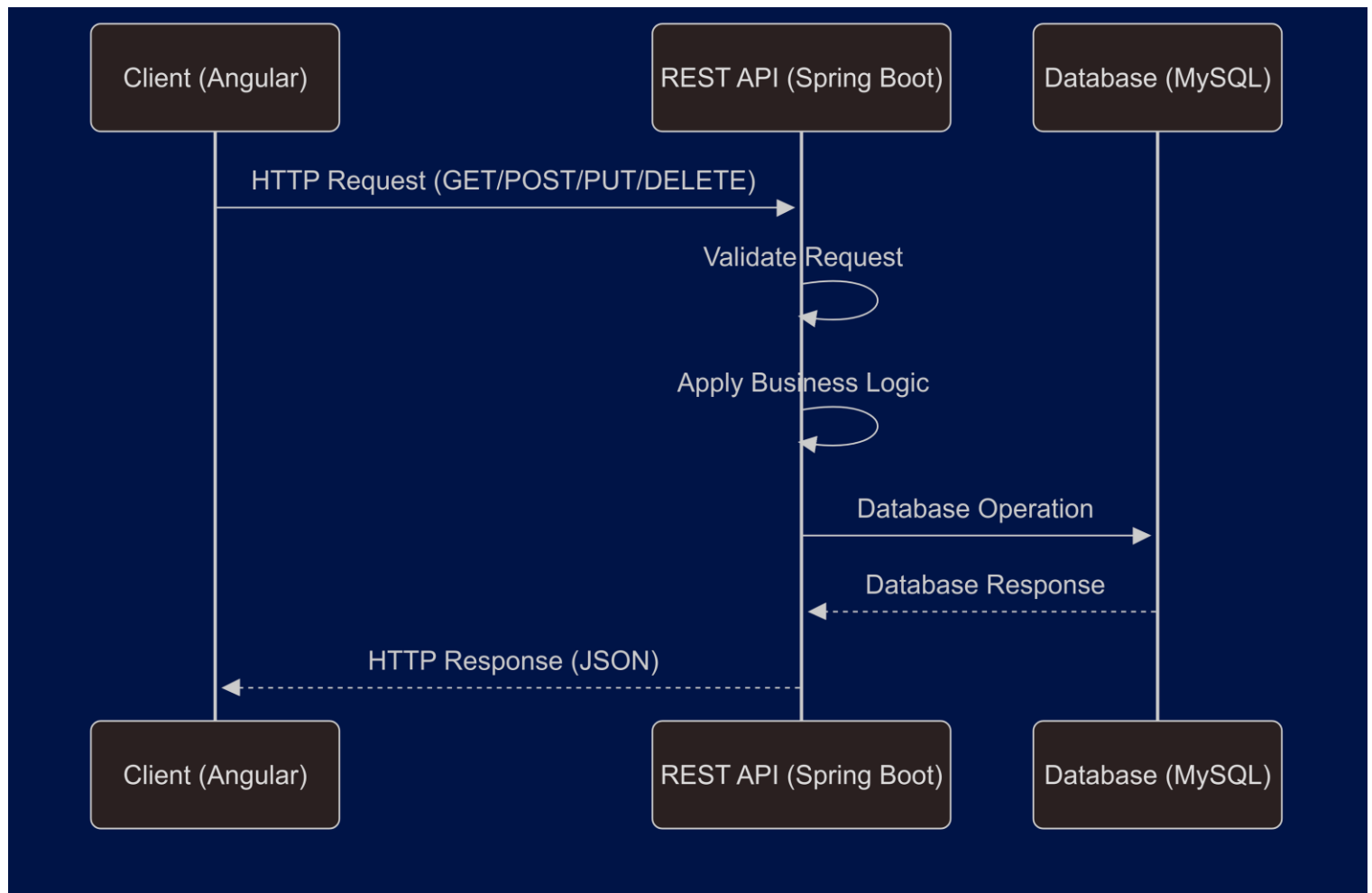
## Listing All Backend Files



## Listing All Backend Files



## Listing All Backend Files



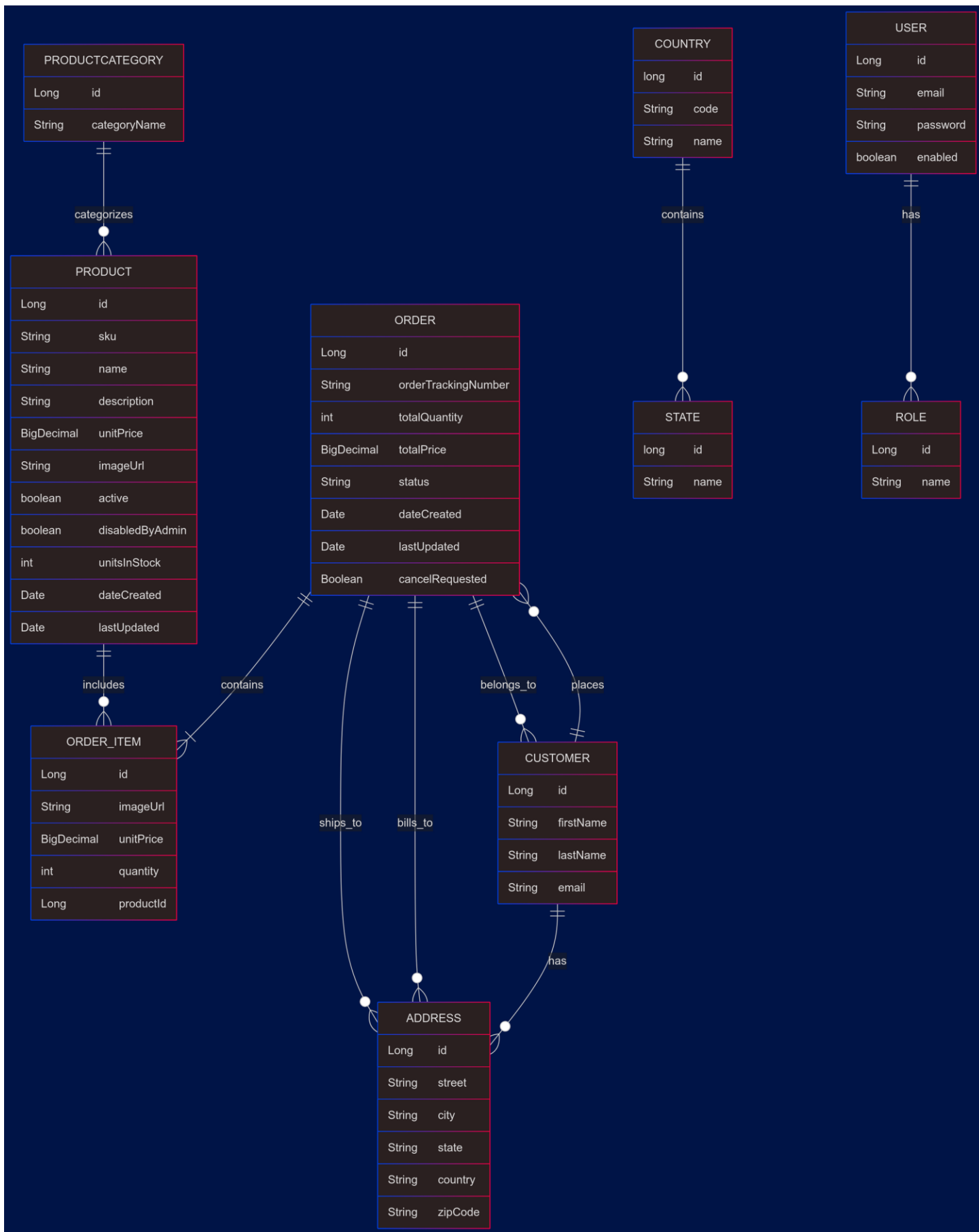
## Client-Server Communication – Explanations

- **Client (Angular):** The frontend application that sends HTTP requests to interact with backend services.
- **REST API (Spring Boot):** The backend server that handles requests from the client and performs validation, business logic, and database operations.
- **Database (MySQL):** The relational database that stores and returns persistent application data.

Request Flow:

1. **HTTP Request (GET/POST/PUT/DELETE):** The Angular client sends a RESTful request to the backend to fetch or modify data.
2. **Validate Request:** The Spring Boot backend checks if the request is authenticated and properly structured.

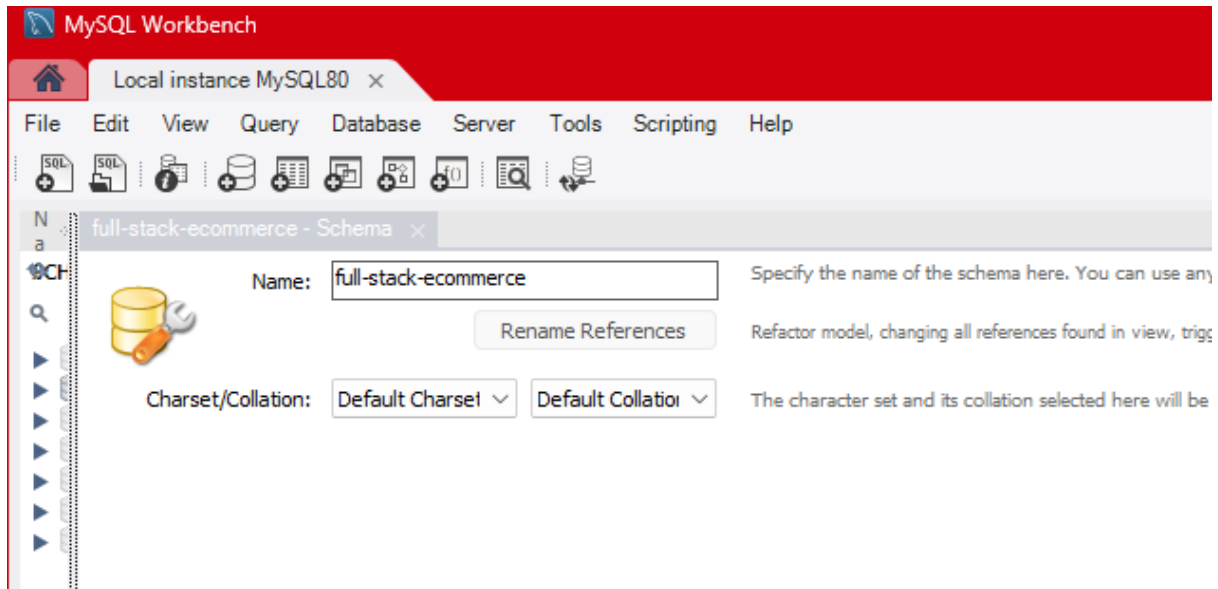
3. **Apply Business Logic:** Based on the request, appropriate service-layer logic is executed (e.g., order validation, total calculation).
4. **Database Operation:** The backend queries or updates the database as needed.
5. **Database Response:** The MySQL database returns the result of the operation (e.g., retrieved data, confirmation of update).
6. **HTTP Response (JSON):** The backend sends a structured JSON response back to the Angular client.



# ER Diagram for entities

# Steps to start project

## Create new schema in mySql



Go to MySQL Workbench and create new shema 'full-stack-ecommerce'



And run the **public class** SpringBootEcommerceApplication, database will automatically create.

Run the sql script countries-and-states.sql which provided github Project page

Then go to Frontend part and start fronted by using command **ng serve**




# Present For Application Part

### Kayıt Ol

Customer

▼

Sign Up



Search for products ...

Search

Welcome back  
report@sunum.com

Logout

Members

Order History


\$78.96 6

Books

Coffee Mugs

Mouse Pads


Luggage Tags



Exploring Vue.js

\$14.99


Add to cart



Advanced Techniques in Big Data

\$13.99


Add to cart



Crash Course in Big Data

\$18.99

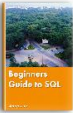
Add to cart



JavaScript Cookbook

\$23.99


Add to cart



Beginners Guide to SQL

\$14.99


Add to cart



Advanced Techniques in JavaScript

\$16.99


Add to cart



Introduction to Spring Boot

\$25.99

Add to cart








Become a Guru in React.js

\$23.99

Add to cart

Add some product and go to cart

	Advanced Techniques in JavaScript \$13.99	<a href="#">Remove</a> Subtotal: \$13.99
	Crash Course in Big Data \$18.99	Quantity: <a href="#">+</a> 2 <a href="#">-</a> <a href="#">Remove</a> Subtotal: \$37.98
	sol ayağım \$5.00	Quantity: <a href="#">+</a> 2 <a href="#">-</a> <a href="#">Remove</a> Subtotal: \$10.00
	Advanced Techniques in JavaScript \$16.99	Quantity: <a href="#">+</a> 1 <a href="#">-</a> <a href="#">Remove</a> Subtotal: \$16.99
		<b>Total Quantity: 6</b> <b>Shipping: FREE</b> <b>Total Price: \$78.96</b> <a href="#">Checkout</a>



Welcome back [report@sumum.com](#)

[Books](#)  
[Coffee Mugs](#)  
[Mouse Pads](#)  
[Luggage Tags](#)

First Name  
Last Name  
Email

Shipping Address



Country  
Street  
City  
State  
Zip Code  
☒ Billing Address same as Shipping Address

Billing Address

Country  
Street  
City  
State  
Zip Code

Credit or Debit Card

04 / 28 111

Review Your Order

Total Quantity: 6  
Shipping: FREE  
Total Price: \$78.96

localhost:4200/order-history

ExampleShop

Search for products ...

Search

Welcome back  
report@sunum.com

Logout

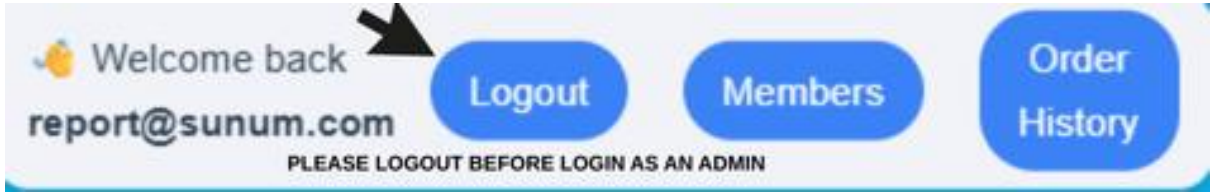
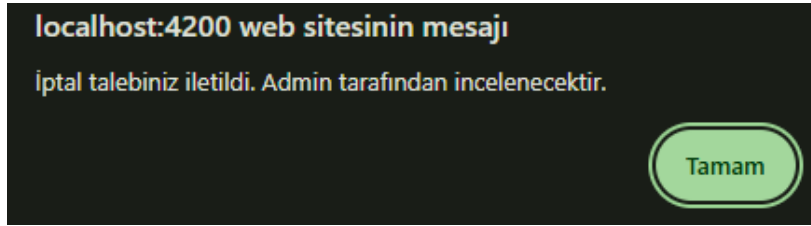
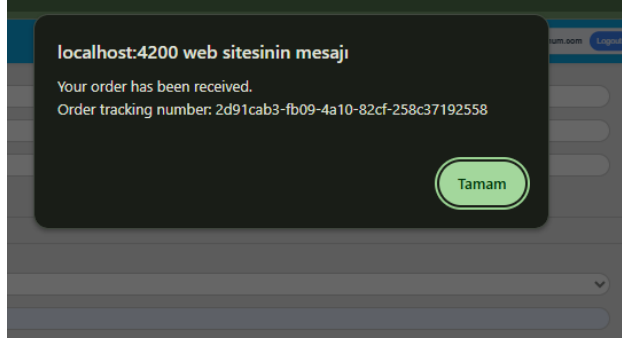
Members

Order History

### Your Orders

Tracking Number	Total Price	Total Quantity	Date	Status	Actions
2d91cab3-fb09-4a10-82cf-258c37192558	\$78.96	6	May 12, 2025, 8:09:05 PM	PENDING	<a href="#">İptal Talebi Gönder</a> <a href="#">Details</a>

**ORDER HISTORY PAGE**



http://localhost:4200/admin/login

## Admin GiriŖi

E-posta

Ŗifre

```
SELECT * FROM users WHERE email = 'admin@admin.com';
SELECT * FROM users_roles WHERE user_id = <admin_user_id>;
SELECT * FROM roles WHERE id = <role_id>;
```

GiriŖ Yap

HoŖ Geldiniz, Admin!

Bu panelden kullanıcıları, siparişleri, ürünleri ve destek taleplerini yönetebilirsiniz.

Kullanıcılar Siparişler Ürünler Destek

#	Müşteri	Tarih	Tutar	Durum	İŖlem
29	report@sunum.com	5/12/25, 8:09 PM	TRY78.96	PENDING	<a href="#">İptal Onayla</a>

HoŖ Geldiniz, Admin!

Bu panelden kullanıcıları, siparişleri, ürünleri ve destek taleplerini yönetebilirsiniz.

Kullanıcılar Siparişler Ürünler Destek




#	Müşteri	Tarih	Toplam Tutar	Durum	İŖlemler
29	report@sunum.com	5/12/25, 8:09 PM	TRY78.96	<a href="#">İPTAL EDİLDİ</a>	<a href="#">Gözetle</a> <a href="#">İptal</a>

HoŖ Geldiniz, Admin!

Bu panelden kullanıcıları, siparişleri, ürünleri ve destek taleplerini yönetebilirsiniz.

Kullanıcılar Siparişler Ürünler Destek

## 🛒 Ürün Listesi


#	Görsel	İsim	Fiyat	Stok	İşlem
1		Crash Course in Python	\$14.99	100	<button>Deactivate</button> <button>Aktif Et</button>
2		Become a Guru in JavaScript	\$20.99	49	<button>Deactivate</button> <button>Aktif Et</button>
3		Exploring Vue.js	\$14.99	100	<button>Deactivate</button>

Satıcı Paneli


Homepage

View Orders

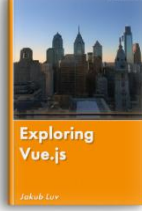
Yeni Ürün Ekle





**Crash Course in Python**  
Learn Python at your own pace. The author explains how the technology works in easy-to-understand language. This book includes working examples that you can apply to your own projects. Purchase the book and get started today!  
**Price:** \$14.99  
**Stock:** 100  
[Edit Product](#) [Deactive Product](#)



**Become a Guru in JavaScript**  
Learn JavaScript at your own pace. The author explains how the technology works in easy-to-understand language. This book includes working examples that you can apply to your own projects. Purchase the book and get started today!  
**Price:** \$20.99  
**Stock:** 49  
[Edit Product](#) [Deactive Product](#)



**Exploring Vue.js**  
Learn Vue.js at your own pace. The author explains how the technology works in easy-to-understand language. This book includes working examples that you can apply to your own projects. Purchase the book and get started today!  
**Price:** \$14.99  
**Stock:** 100  
[Edit Product](#) [Deactive Product](#) [✓ Activate Product](#)



### + Yeni Ürün Ekle

Ürün Adı

Kategori

Açıklama

Fiyat

Stok Adedi

Görsel Yükle

Dosya Seç

Dosya seçilmedi

Ekle

İptal

# Some seller activities

## Conclusion

Thank you for taking the time to read through my project report. I appreciate your interest and hope this document provided a clear overview of the application's architecture and functionality.

If you are interested in exploring more features or reviewing the full source code, I kindly invite you to visit the GitHub repository linked at the beginning of the report.

Your feedback and suggestions are always welcome.

Thank you once again!