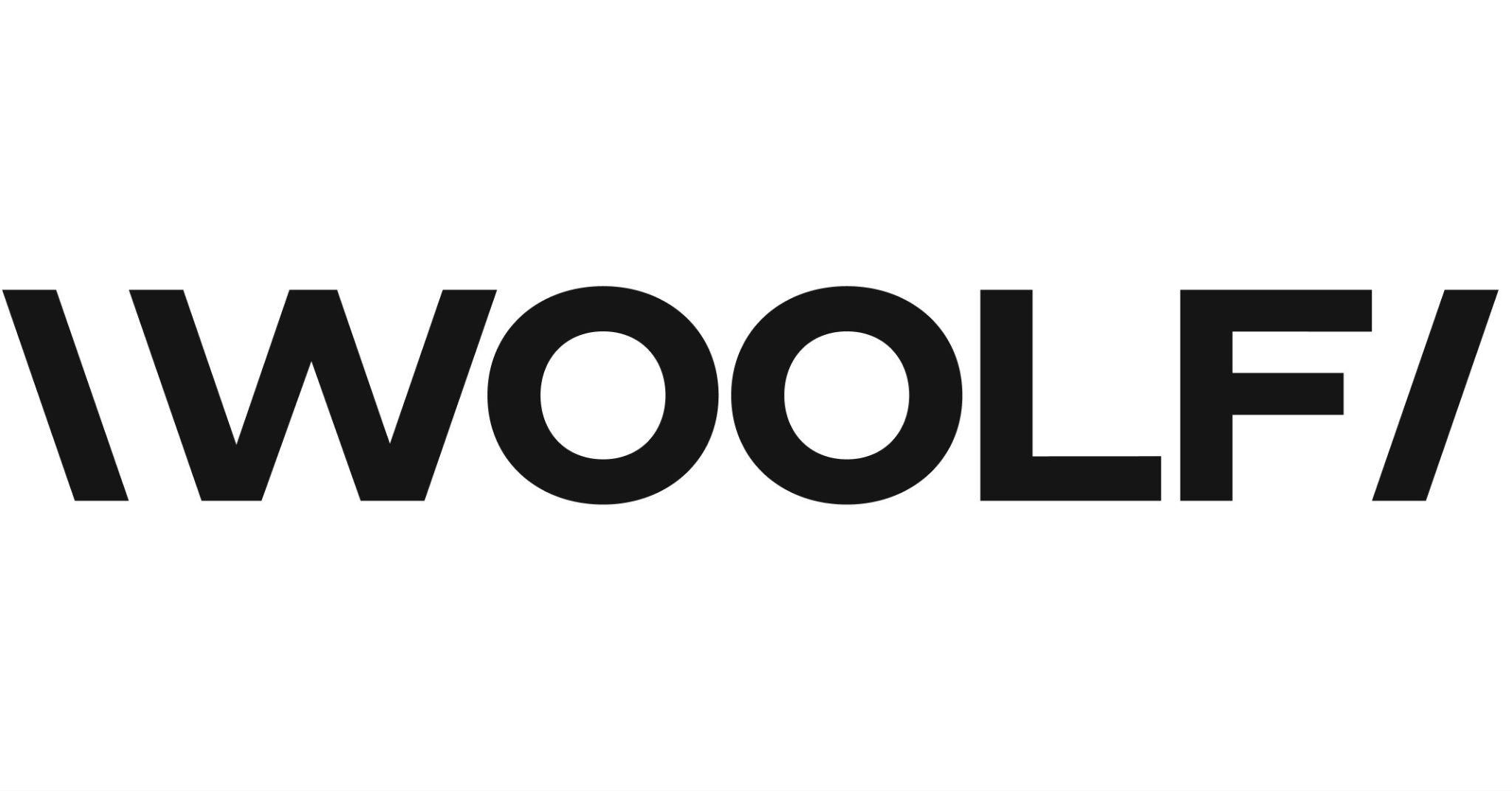
**Applied Software Project Report**

By

Abhijeet Moreshwar Kale

**A Master’s Project Report submitted to Scaler Neovarsity - Woolf in partial fulfillment of the requirements for the degree of Master of Science in Computer Science**

June 2025



**Scaler Mentee Email ID :** mr.abhijeetkale@gmail.com

**Thesis Supervisor :** Naman Bhalla

**Date of Submission :** 08/06/2025

**Certification**

I confirm that I have overseen / reviewed this applied project and, in my judgment, it adheres to the appropriate standards of academic presentation. I believe it satisfactorily meets the criteria, in terms of both quality and breadth, to serve as an applied project report for the attainment of Master of Science in Computer Science degree. This applied project report has been submitted to Woolf and is deemed sufficient to fulfill the prerequisites for the Master of Science in Computer Science degree.

Naman Bhalla

…………………

Project Guide / Supervisor

**DECLARATION**

I confirm that this project report, submitted to fulfill the requirements for the Master of Science in Computer Science degree, completed by me from 26/07/2024 to 30/09/2024, is the result of my own individual endeavor. The Project has been made on my own under the guidance of my supervisor with proper acknowledgement and without plagiarism. Any contributions from external sources or individuals, including the use of AI tools, are appropriately acknowledged through citation. By making this declaration, I acknowledge that any violation of this statement constitutes academic misconduct. I understand that such misconduct may lead to expulsion from the program and/or disqualification from receiving the degree.

**Abhijeet Moreshwar Kale**

**<Signature of the Candidate> Date: 08 June 2025**

**ACKNOWLEDGMENT**

I am profoundly grateful to everyone who contributed to the successful completion of the backend module for the *E-commerce Website* project.

First and foremost, I extend my deepest thanks to Anurag Khanna, module instructor for their invaluable guidance and expertise throughout the development of this project. Their constructive feedback and technical insights were instrumental in designing and implementing the backend architecture using Java, Spring Boot, MySQL, Elasticsearch, Apache Kafka, Redis, MongoDB and API Gateway.

I also wish to express my gratitude to Scaler for providing the resources, tools, and collaborative environment necessary to bring this project to life. Special thanks to module colleagues for their support in debugging, optimizing queries, and ensuring seamless integration of core functionalities such as user authentication, product management, and payment processing.

I would like to acknowledge my peers and colleagues for their encouragement and assistance in tackling challenges during various stages of development, from creating RESTful APIs to optimizing database performance. Their input played a vital role in improving the robustness and scalability of the backend system.

Finally, I am immensely thankful to my family for their unwavering support, patience, and encouragement, which gave me the strength to complete this project successfully.

This achievement would not have been possible without the collective effort and contributions of everyone involved. Thank you for being a part of this journey.

**Table of Contents**

[**List of Tables 6**](#_9nnr2lniv90f)

[**List of Figures 7**](#_ju1gc9w3iuai)

[**Applied Software Project 8**](#_b4cf8683b1wd)

[Abstract 8](#_sj7c7bghlznr)

[Project Description 8](#_1z5fx61h0cc)

[Requirement Gathering 9](#_joagy45av5k0)

[Class Diagrams 9](#_nvf4h831fm8o)

[Database Schema Design 9](#_ydqs8nkbe6m9)

[Feature Development Process 11](#_p6mfl8dwb9sy)

[Deployment Flow 12](#_2mk44ad33gi)

[Technologies Used 12](#_wn68bn10ag78)

[Conclusion 13](#_4yf46wt6rx84)

[**References 14**](#_z0iyzog9l959)

## List of Tables

**(To be written sequentially as they appear in the text)**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| **1** |  |  |
| **2** |  |  |

## List of Figures

**(List of Images, Graphs, Charts sequentially as they appear in the text)**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| **1** |  |  |
| **2** |  |  |

## Applied Software Project

### Abstract

### This project presents the design and implementation of a scalable, microservices-based **e-commerce application** that provides a seamless and efficient backend infrastructure for managing an online shopping platform. In response to the growing need for modular, high-performance systems, this application adopts a microservices architecture that allows for independent development, deployment, and scaling of core services.

### The platform is composed of multiple dedicated microservices:

### **User Management Service** handles user registration, authentication, and session management.

### **Product Catalog Service** maintains product listings, categories, and inventory data.

### **Cart Service** enables users to manage their shopping cart contents.

### **Order Management Service** processes order placement and tracks their status.

### **Payment Service** simulates secure transaction handling.

### **Notification Service** dispatches real-time updates such as order confirmations and payment status.

### These services communicate via **REST APIs** and utilize **Apache Kafka** for asynchronous, event-driven interactions. An **API Gateway** is implemented to act as a unified entry point, managing routing, authentication, and rate limiting.

### The backend system is built using **Spring Boot**, with **MySQL** as the primary data store for persistence. Using **MongoDB** as a primary data store for Cart Service to handle dynamic nature of the cart. **Redis caching** is integrated to improve performance by reducing latency and database load, particularly for frequently accessed data such as product listings and user sessions.

### This project demonstrates the practical application of modern backend architectural principles in building a robust e-commerce system. It lays the groundwork for further enhancements such as containerized deployment, horizontal scalability, and CI/CD automation.

### Project Description

### The goal of this project is to design and implement a cloud-ready, microservices-based **e-commerce backend system** that facilitates end-to-end management of online shopping operations. The project aims to address the core requirements of a modern e-commerce platform—scalability, modularity, maintainability, and performance—by decomposing the application into independently deployable services.

**Key Features**

The system is divided into multiple microservices, each handling a specific domain:

1. **User Management Service**
   * Handles user registration, authentication (JWT-based), and profile management.
   * Implements secure session handling with MySQL for performance and scalability.
2. **Product Catalog Service**
   * Manages product data, categories, and inventory.
   * Exposes APIs to add, update, retrieve, and delete products.
   * Uses Elasticsearch for faster product lookups with full text search.
3. **Cart Service**
   * Allows users to add, update, and remove items from their shopping cart.
   * Stores cart state independently and ensures consistency across user sessions.
   * Uses Redis for faster retrieval of Cart
4. **Order Management Service**
   * Handles order creation, order status tracking, and historical order retrieval.
   * Integrates with the Payment Service and sends events to Kafka.
5. **Payment Service**
   * Simulates payment processing using test flows.
   * Generates transaction IDs and validates payment status.
6. **Notification Service**
   * Sends order confirmation and payment status notifications via email or sms.
   * Listens to Kafka topics for event-driven messaging.
7. **API Gateway**
   * Acts as a single-entry point to the system.
   * Handles request routing, load balancing.

**Technology Stack**

* **Backend Framework**: Spring Boot (Java)
* **Database:** MySQL, MongoDB
* **Cache:** Redis
* **Messaging:** Apache Kafka
* **Security:** JWT-based authentication
* **API Gateway/Load Balancer:** HAProxy

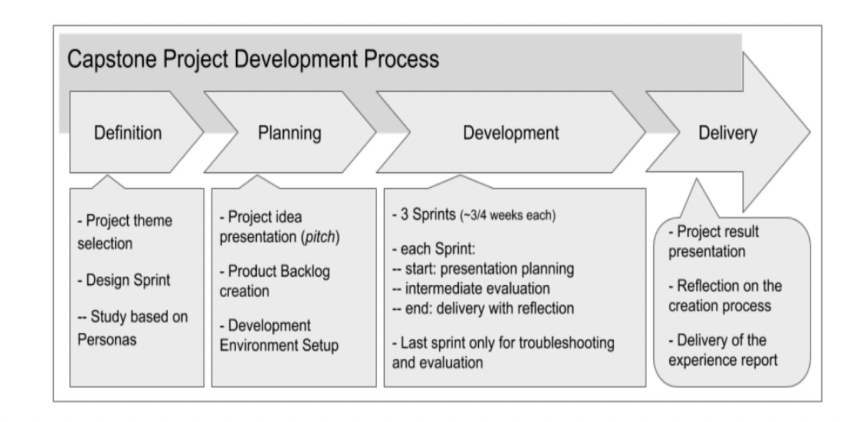
**Inter-Service Communication**

Services communicate using **REST APIs** for synchronous operations and **Kafka** for asynchronous event-driven processing. This design ensures high availability and responsiveness even under heavy load.

**Objectives Achieved**

* Clear separation of concerns with dedicated services
* Scalable and loosely coupled architecture
* High-performance backend with Redis caching
* Resilient inter-service communication via Kafka
* Centralized access and routing through an API Gateway

<<Diagram>>



**Figure 1.1**: Project Development Process (Figure captions go below figures.)

### 

### Requirement Gathering

* + 1. **User Management**
       - Registration: Allow new users to create an account using their email or social media profiles.
       - Login: Users should be able to securely log in using their credentials.
       - Profile Management: Users should have the ability to view and modify their profile details.
       - Password Reset: Users must have the option to reset their password through a secure link.
    2. **Product Catalog**
       - Browsing: Users should be able to browse products by different categories.
       - Product Details: Detailed product pages with product images, descriptions, specifications, and other relevant information.
       - Search: Users must be able to search for products using keywords.
    3. **Cart & Checkout**
       - Add to Cart: Users should be able to add products to their cart.
       - Cart Review: View selected items in the cart with price, quantity, and total details.
       - Checkout: Seamless process to finalize the purchase, including specifying delivery address and payment method.
    4. **Order Management**
       - Order Confirmation: After making a purchase, users should receive a confirmation with order details.
       - Order History: Users should be able to view their past orders.
       - Order Tracking: Provide users with a way to track their order's delivery status.
    5. **Payment**
       - Multiple Payment Options: Support for credit/debit cards, online banking, and other popular payment methods.
       - Secure Transactions: Ensure user trust by facilitating secure payment transactions.
       - Payment Receipt: Provide users with a receipt after a successful payment.
    6. **Authentication**
       - Secure Authentication: Ensure that user data remains private and secure during login and throughout their session.
       - Session Management: Users should remain logged in for a specified duration or until they decide to log out.

Use Case: Typical Flow with Kafka & Elasticsearch Integration

**Part 1**

* User logs in and searches for a product.
* Request reaches LB, then passed to API Gateway.
* API Gateway routes the search request to Product Catalog Service.
* Product Catalog Service queries Elasticsearch for a fast product search.

**Part 2**

* User adds a product to the cart.
* Cart Service produces a message to Kafka about this action.

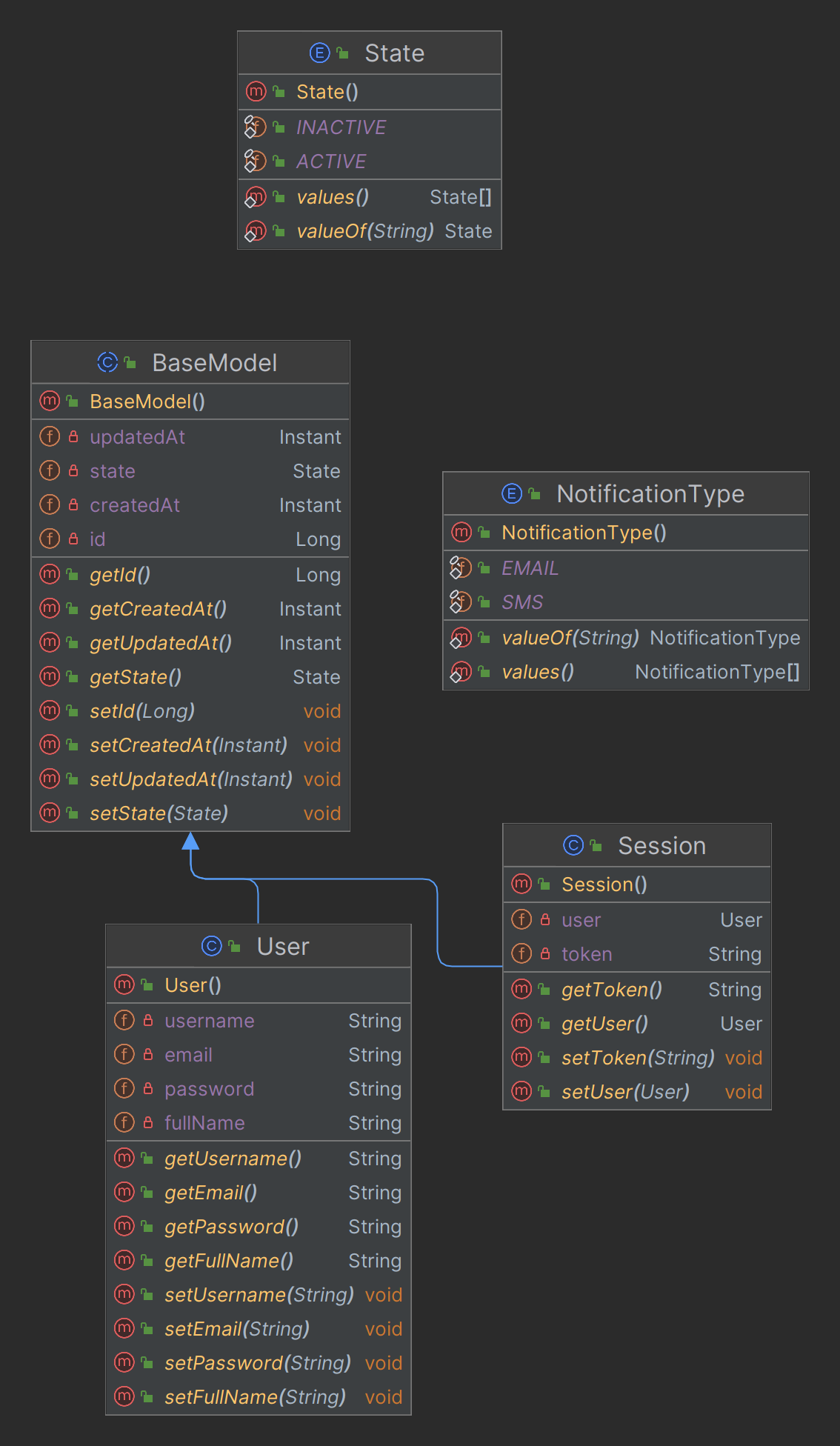
**Part 3**

* User checks out, triggering the Order Management Service.
* After placing the order, a message is sent to Kafka.
* Payment Service consumes the Kafka message to process payment.

<<Diagram>>

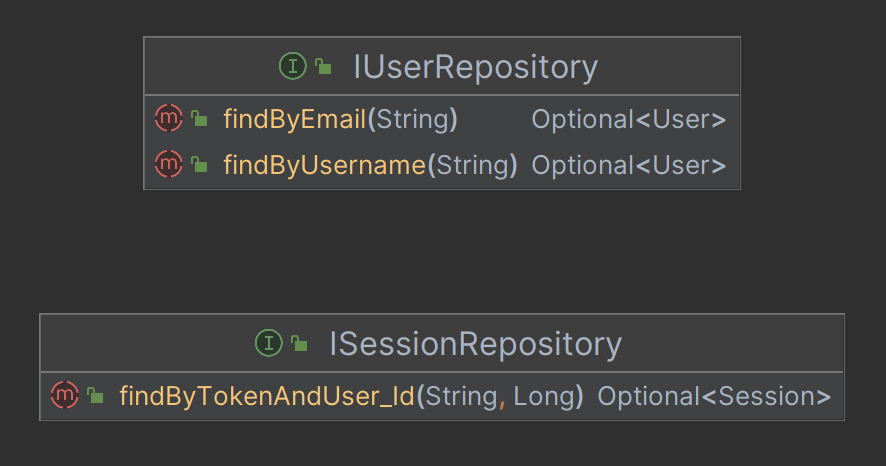
### Class Diagrams

1. **User Management Service**
   * **Models**



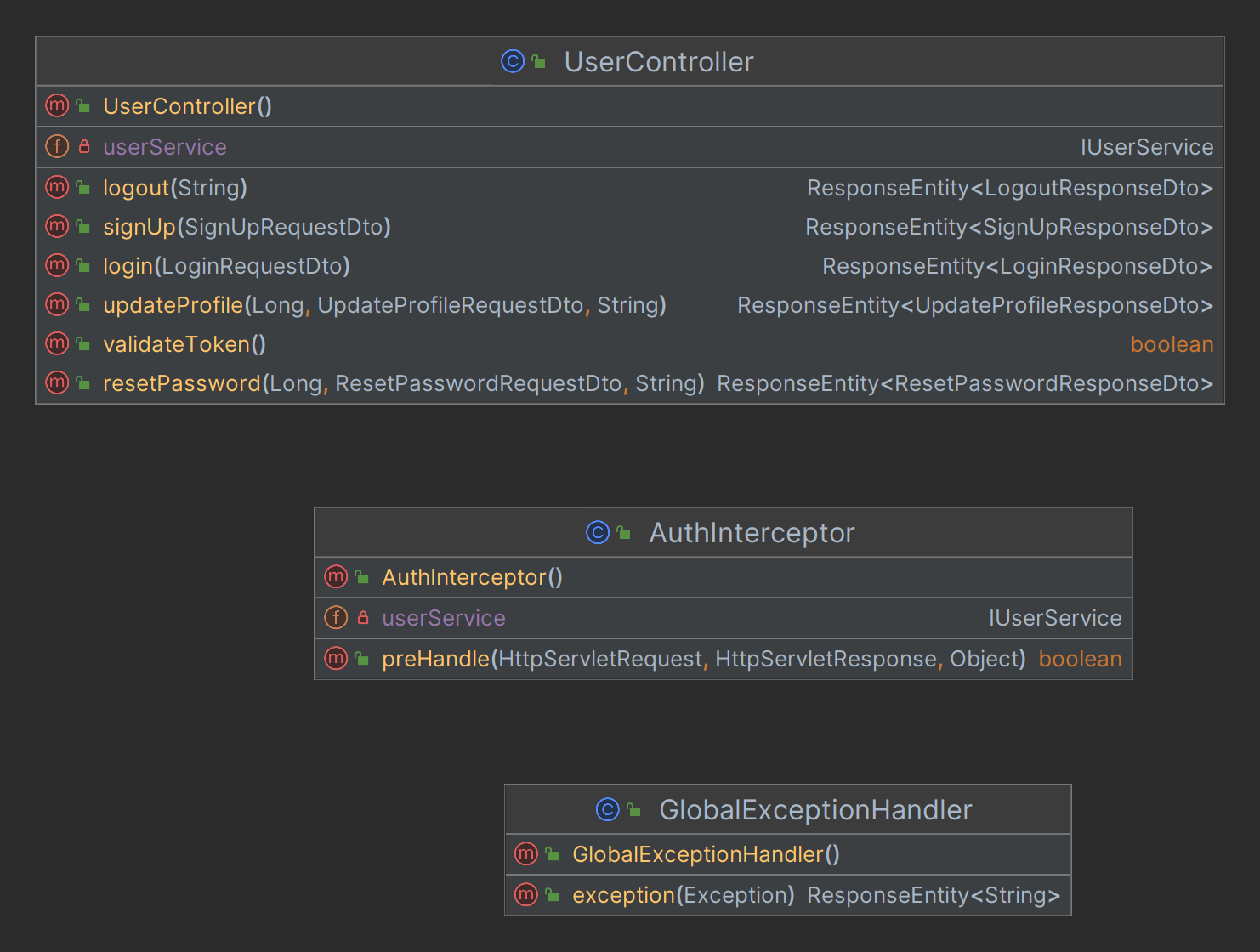
**Figure 3.1.1**: User Management Models

* + **Repositories**



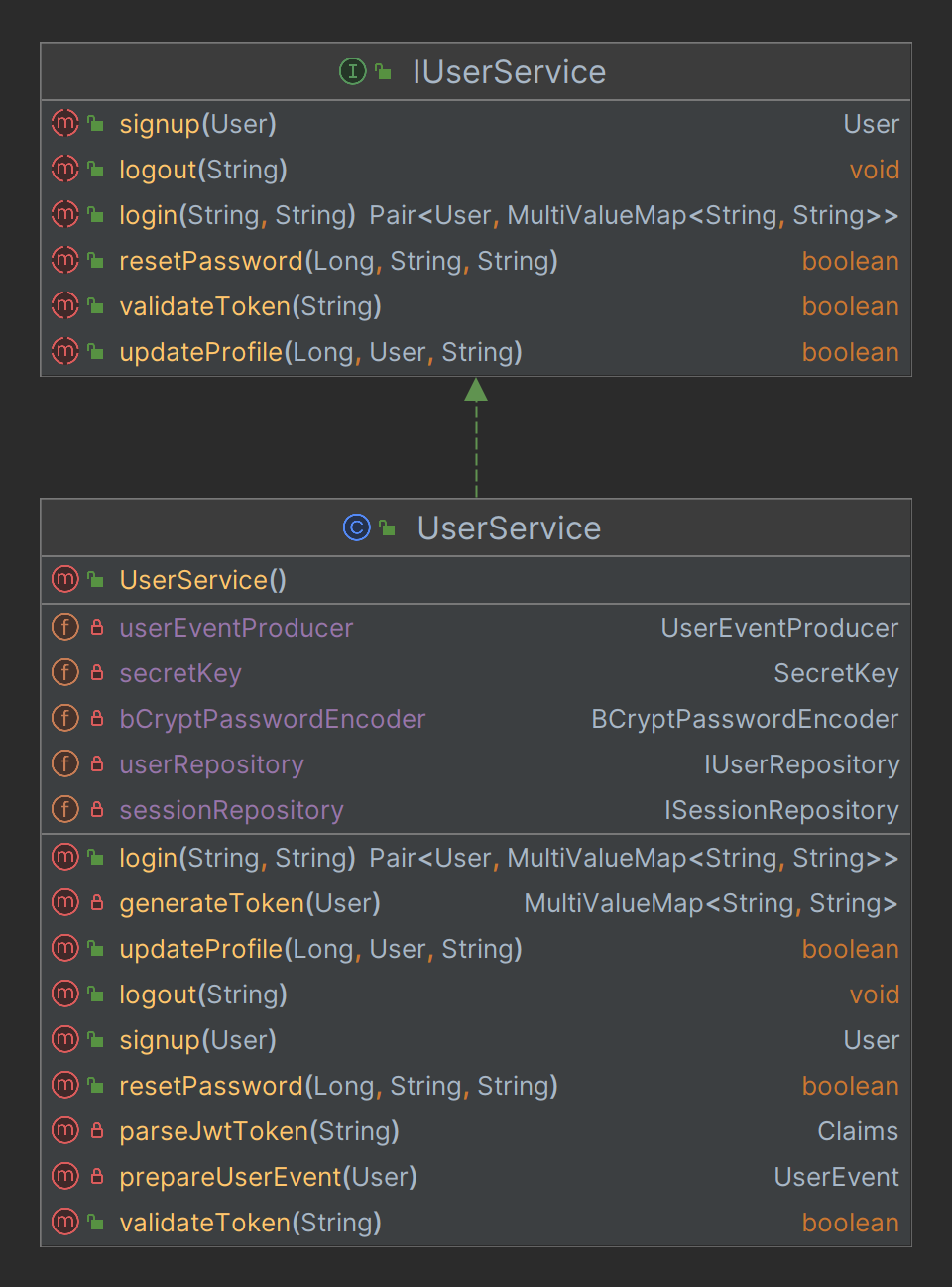
**Figure 3.1.2**: User Management Repository

* + **Controllers**



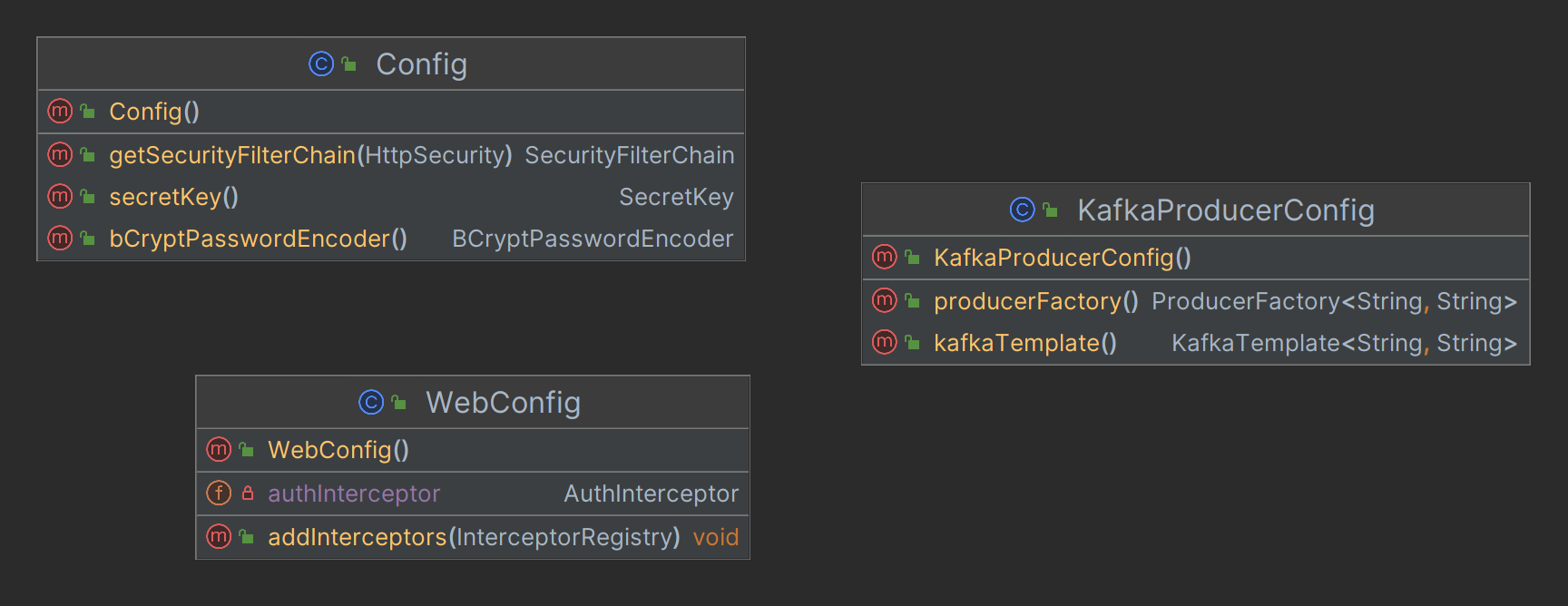
**Figure 3.1.3**: User Management Controller

* + **Services**



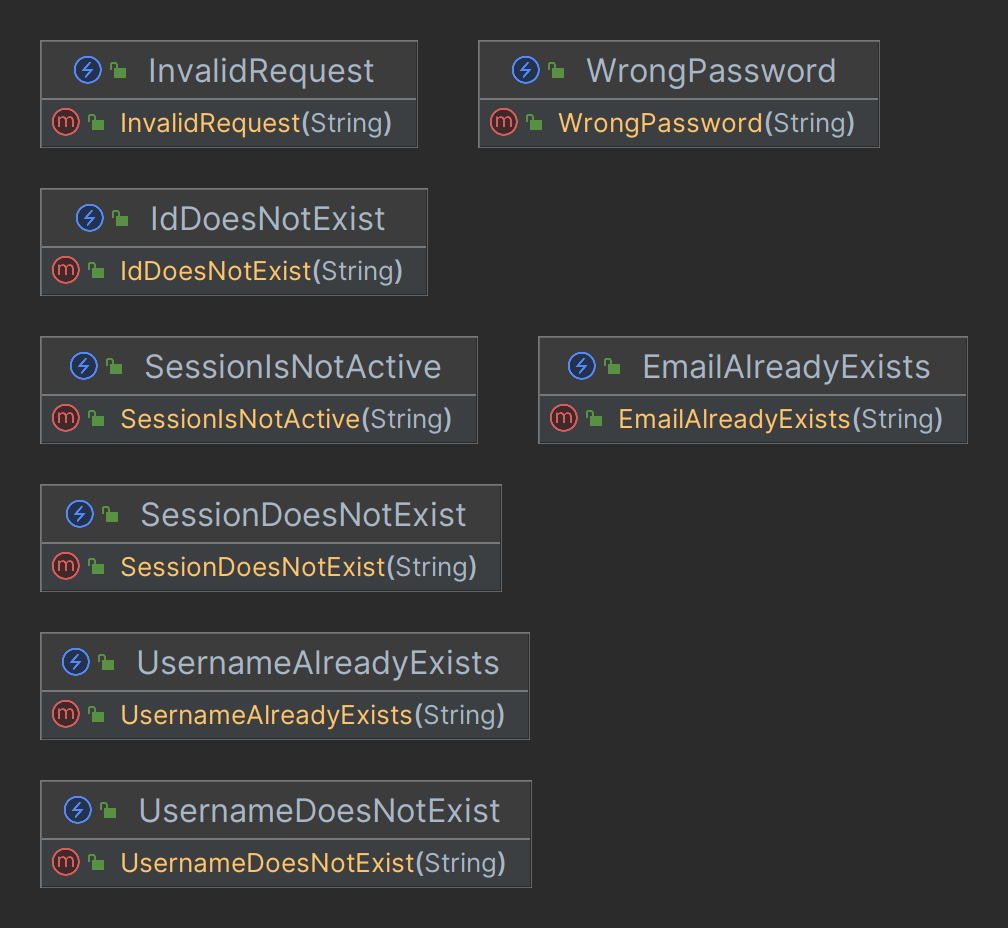
**Figure 3.1.4**: User Management Service

* + **Configs**



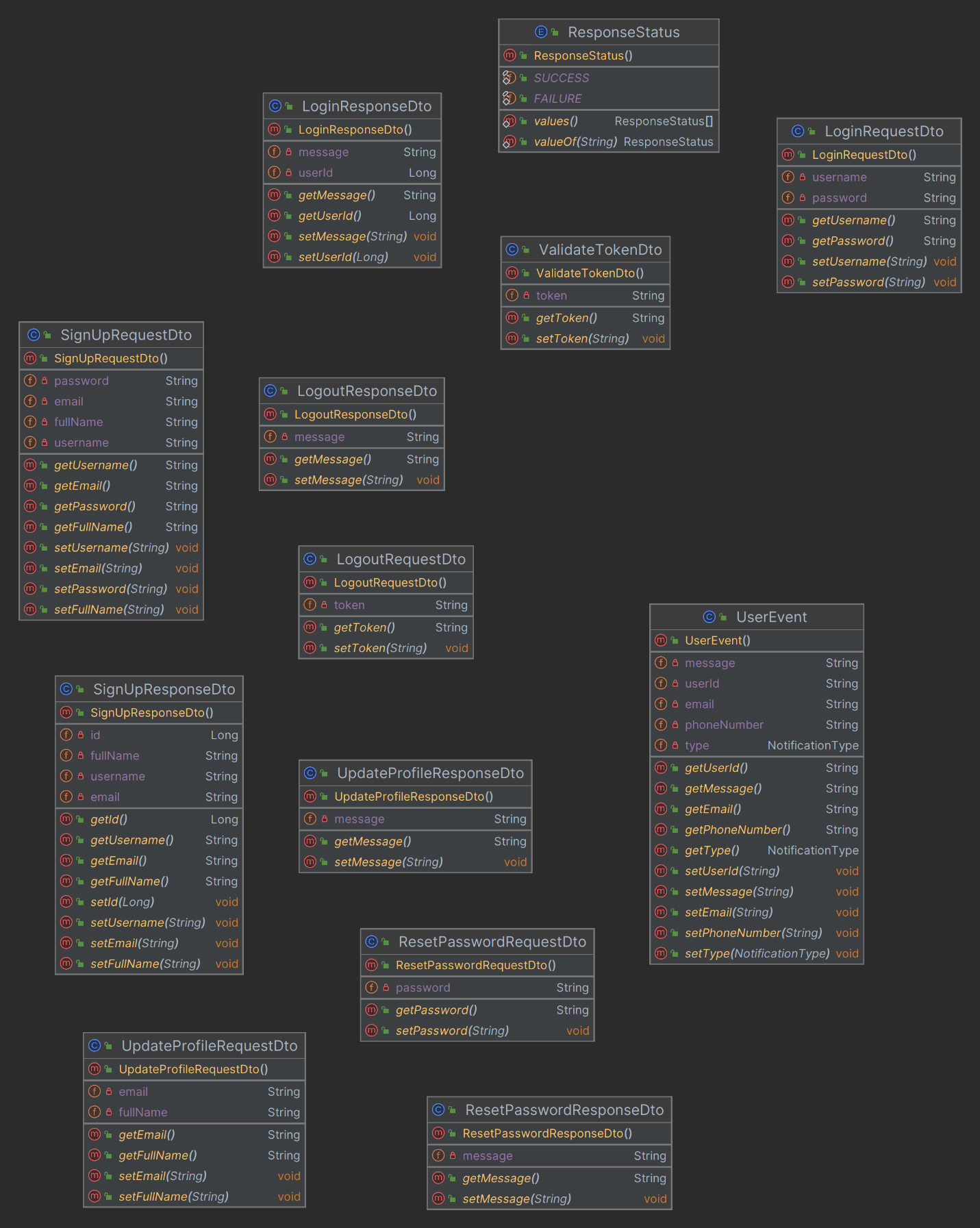
**Figure 3.1.5**: User Management Configurations

* + **Exceptions**



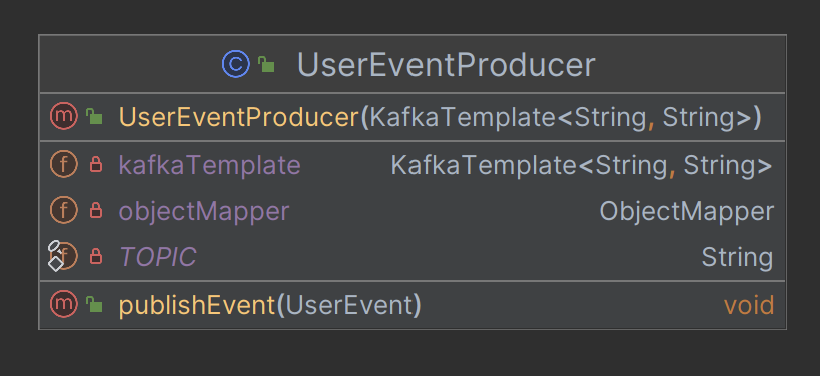
**Figure 3.1.6**: User Management Exceptions

* + **Dtos**



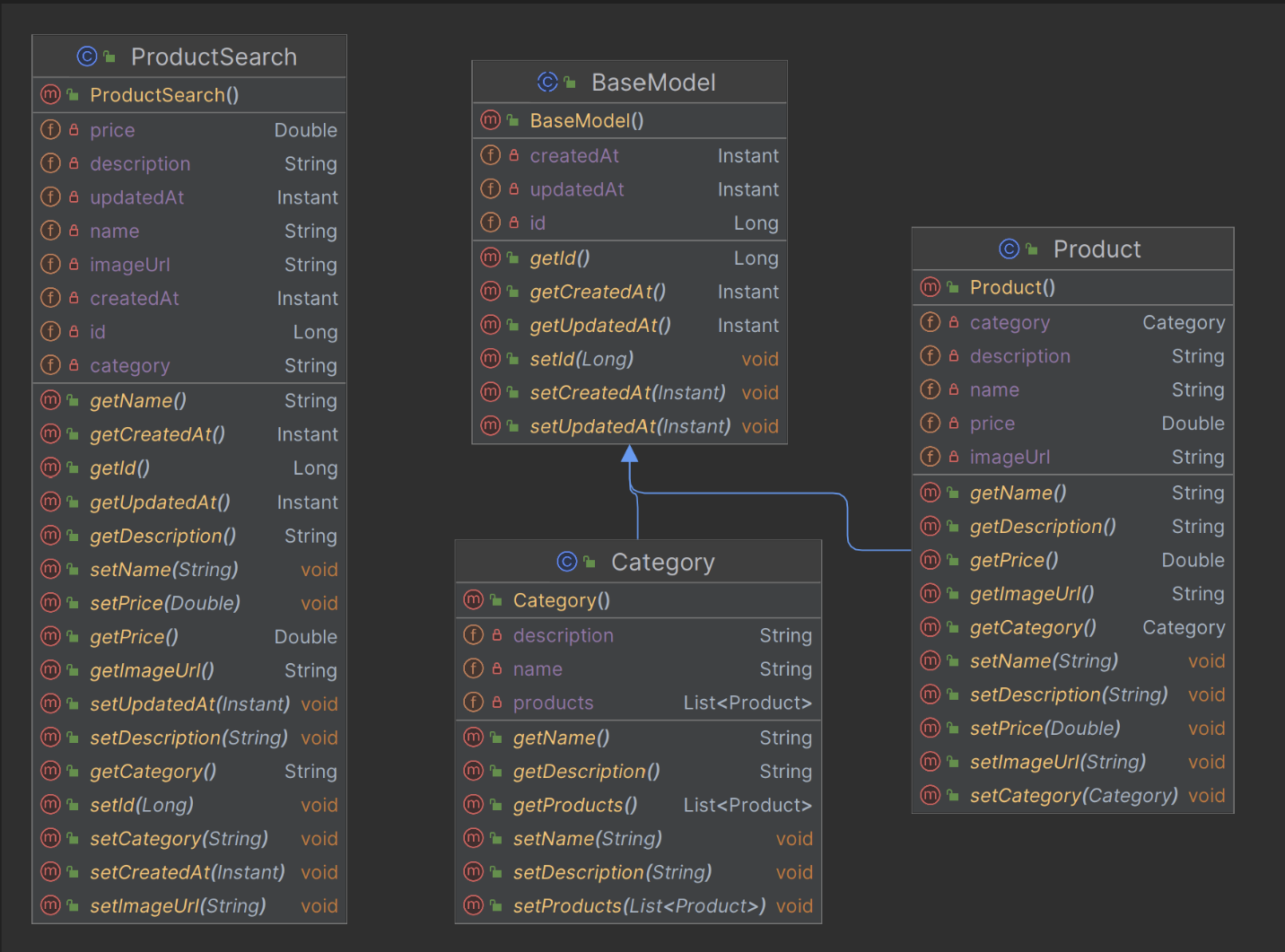
**Figure 3.1.7**: User Management Dtos

* + **Producers**



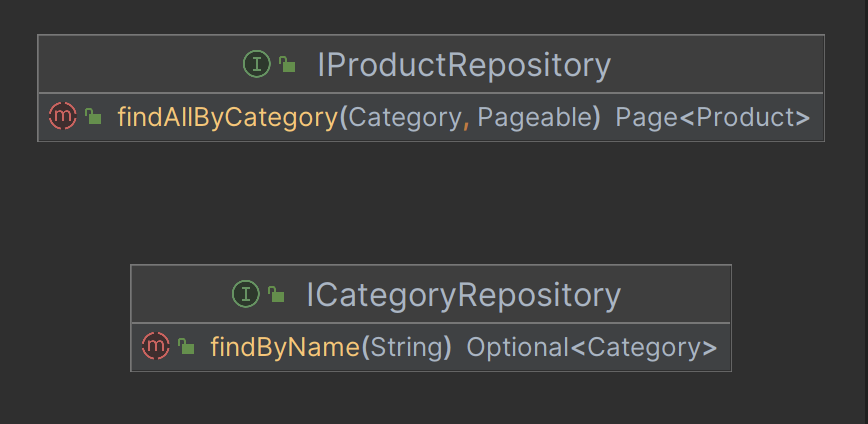
**Figure 3.1.8**: User Management Kafka Event Producer

1. **Product Catalog Service**
   * **Models**

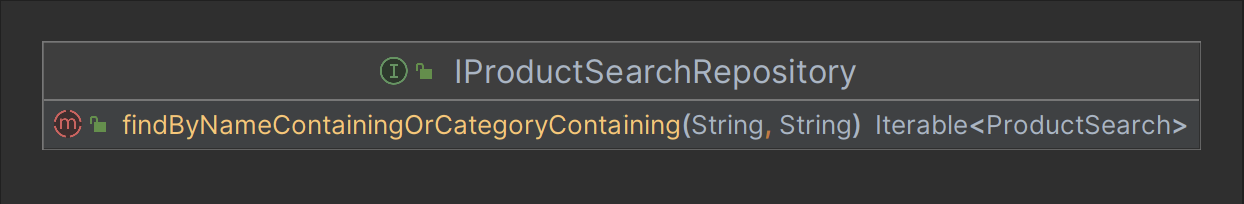


**Figure 3.2.1**: Product Catalog Models

* + **Repositories**

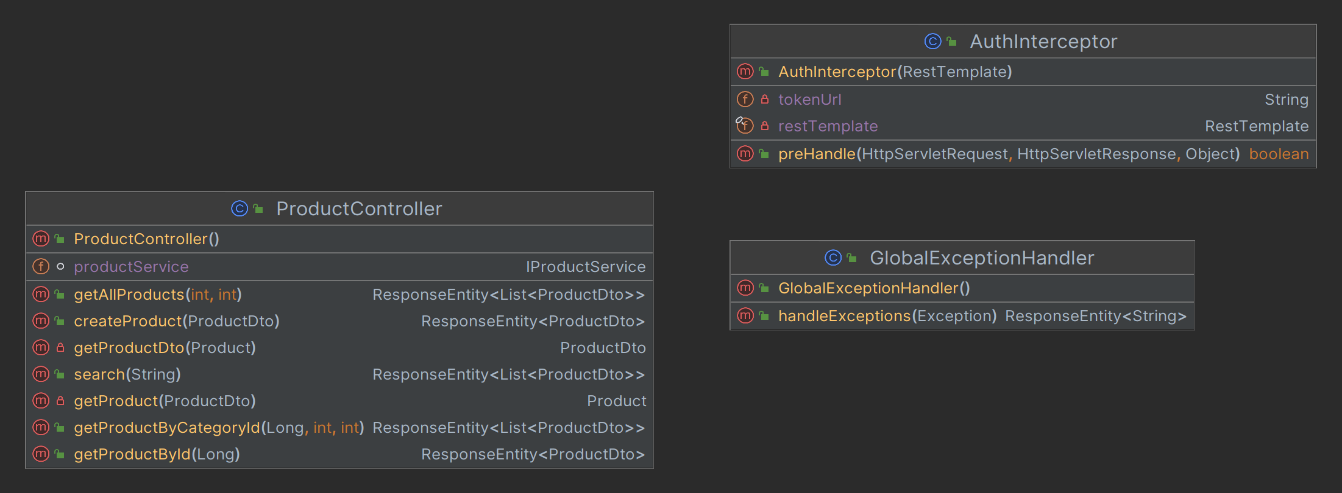


**Figure 3.2.2**: Product Catalog JPA Repository



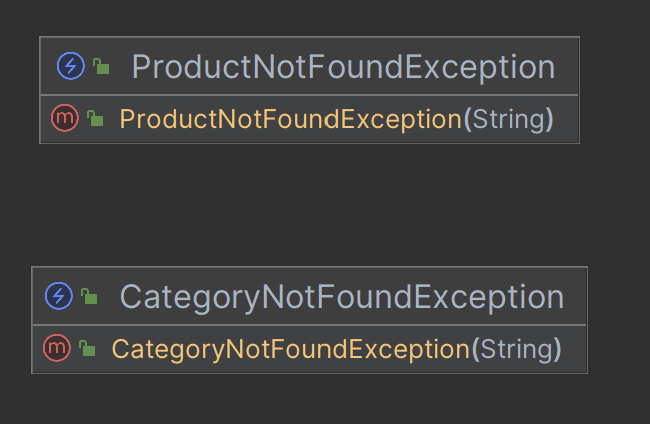
**Figure 3.2.3**: Product Catalog ElasticSearch Repository

* + **Controllers**



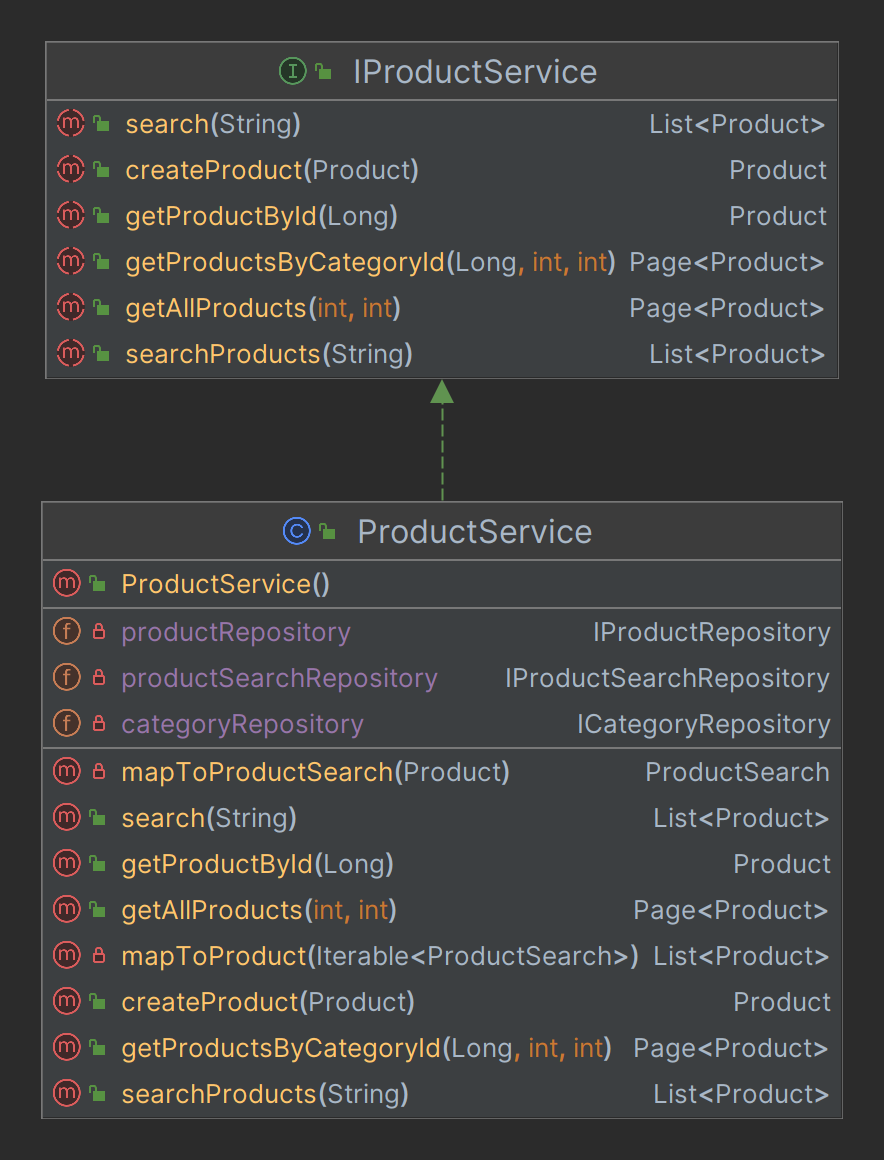
**Figure 3.2.4**: Product Catalog Controller

* + **Exceptions**



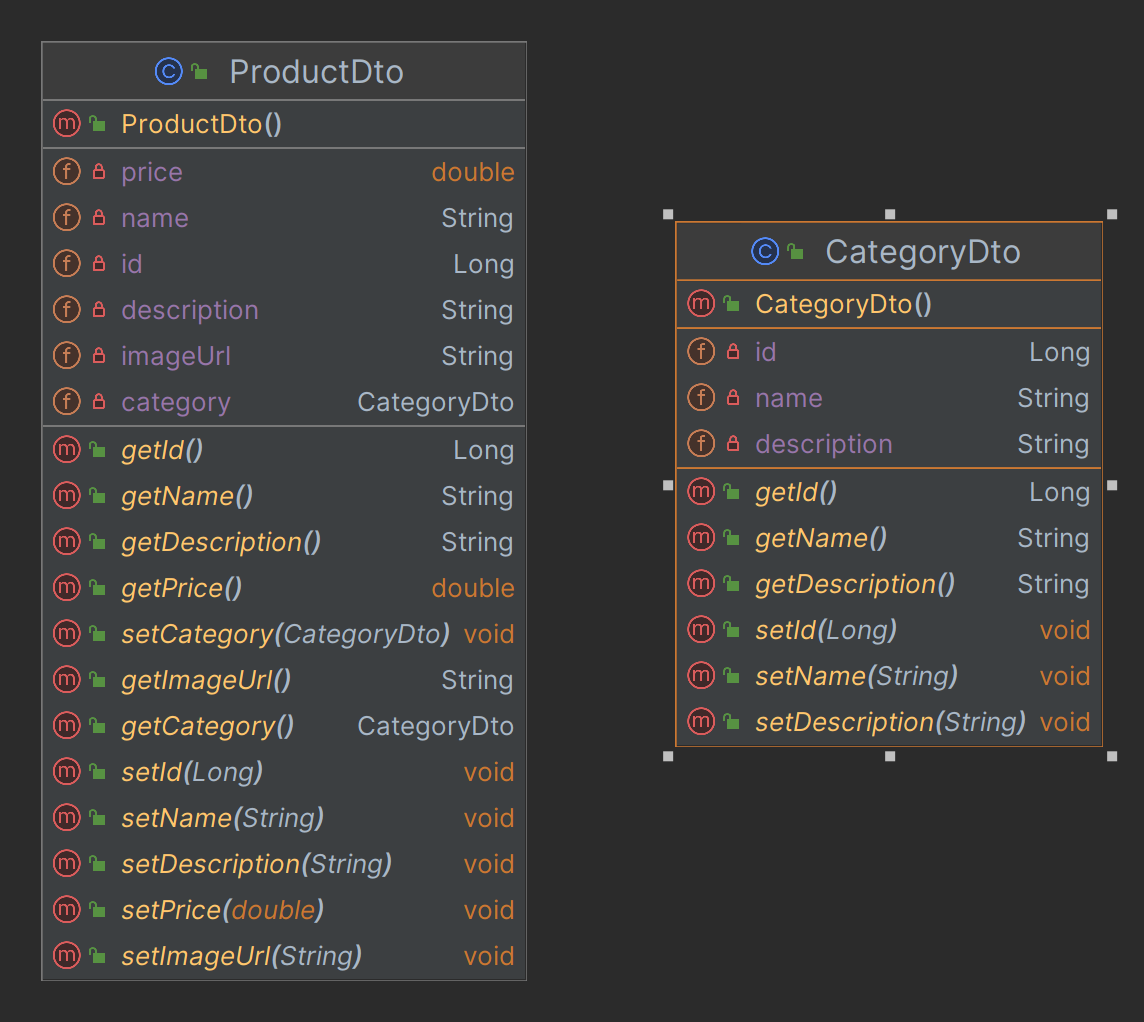
**Figure 3.2.5**: Product Catalog Exceptions

* + **Services**



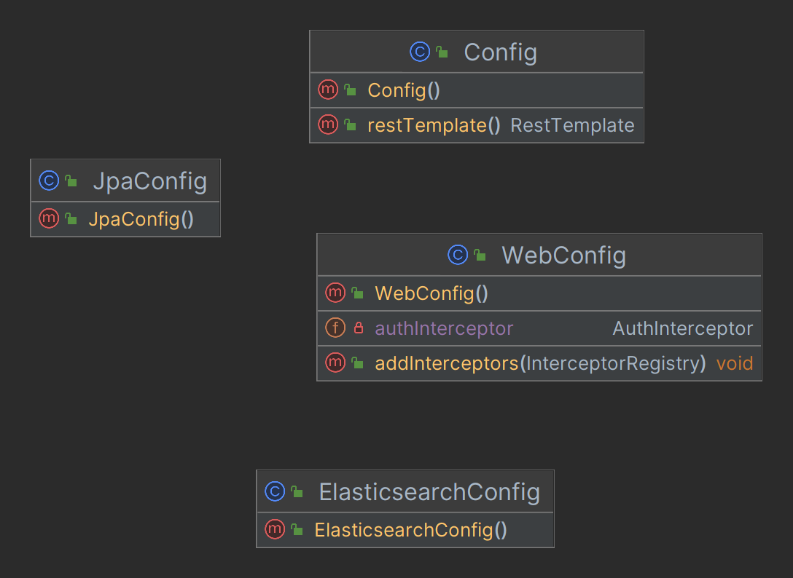
**Figure 3.2.6**: Product Catalog Service

* + **Dtos**



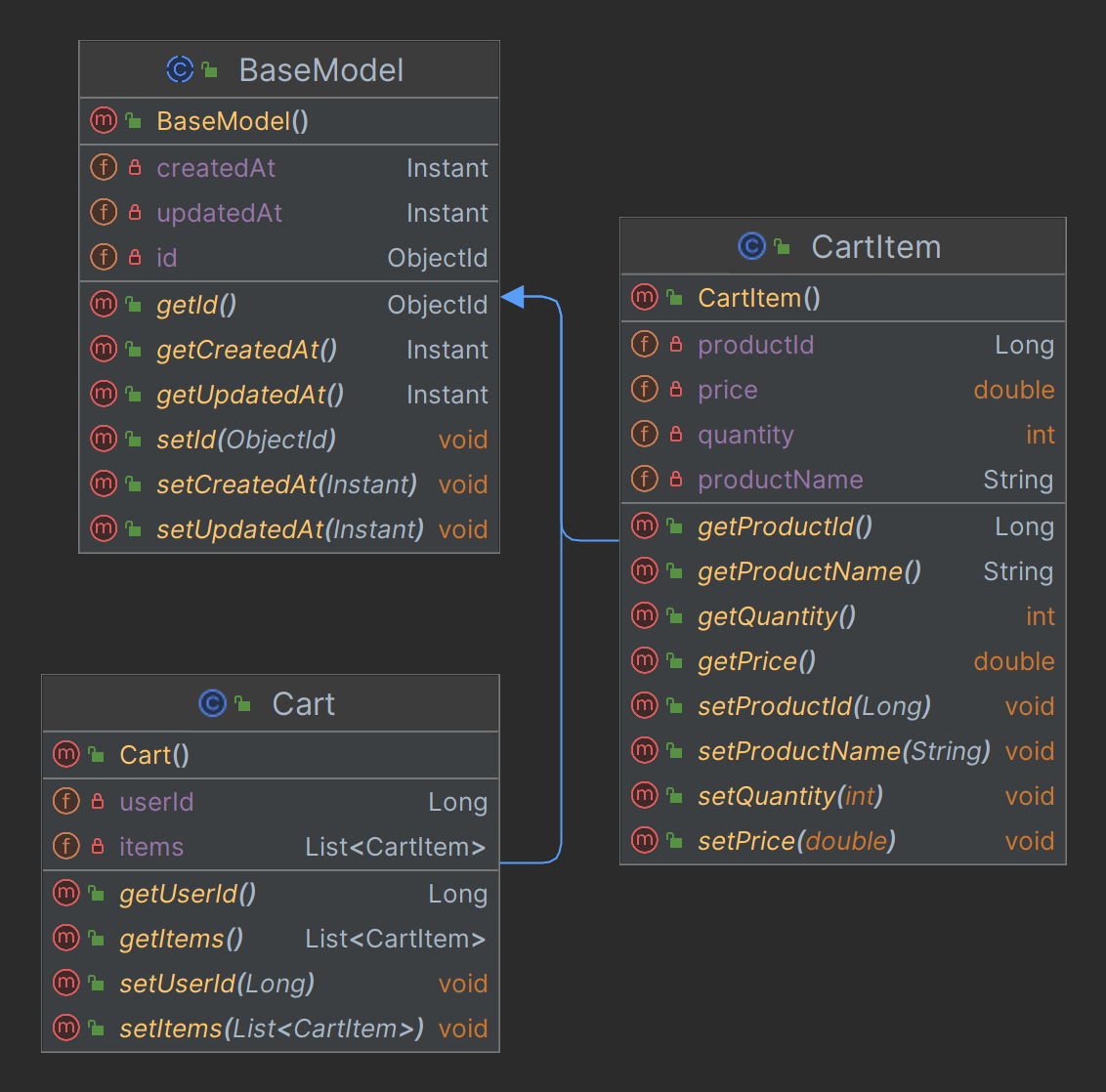
**Figure 3.2.7**: Product Catalog Dtos

* + **Configs**



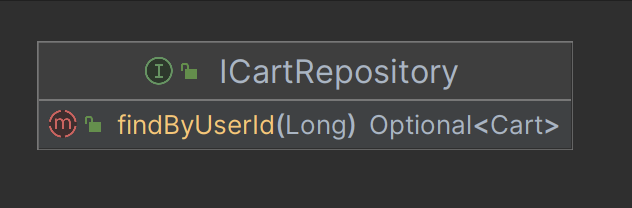
**Figure 3.2.8**: Product Catalog Configurations

1. **Cart Service**
   * **Models**



**Figure 3.3.1**: Cart Models

* + **Repositories**



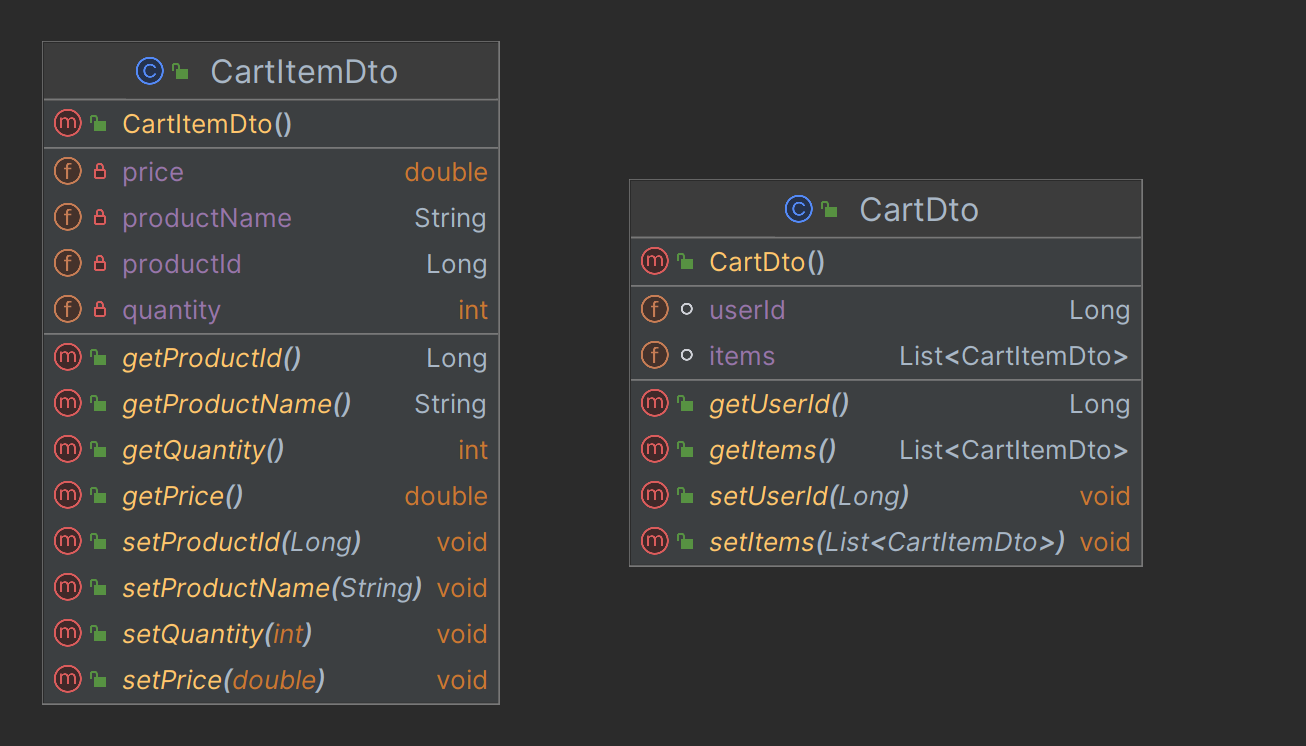
**Figure 3.3.2**: Cart Repository

* + **Services**



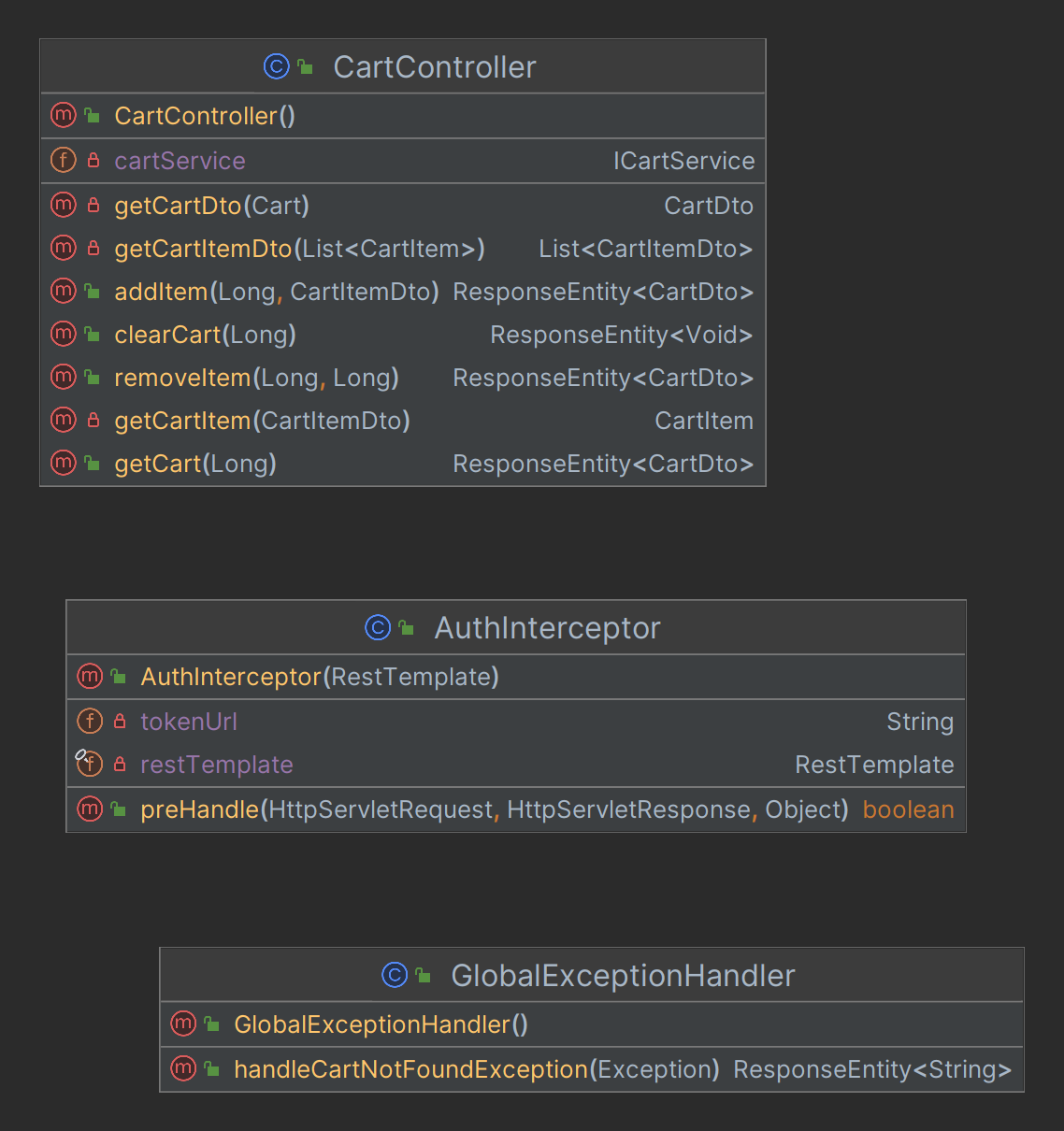
**Figure 3.3.3**: Cart Service

* + **Dtos**



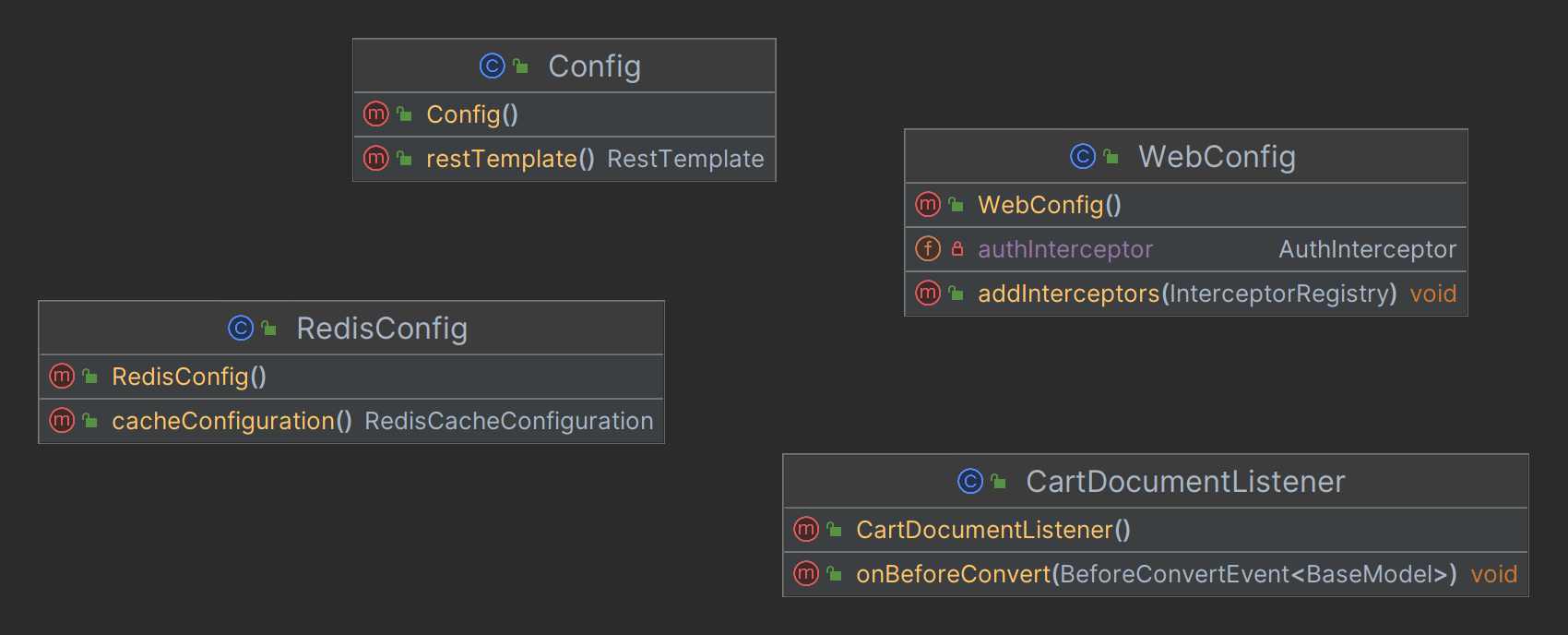
**Figure 3.3.4**: Cart Dtos

* + **Controllers**



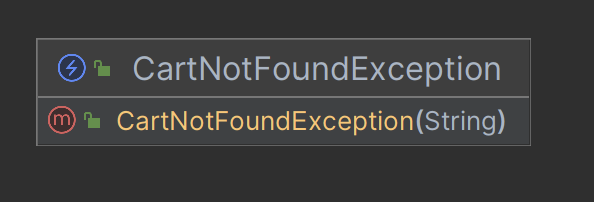
**Figure 3.3.5**: Cart Controller

* + **Configs**



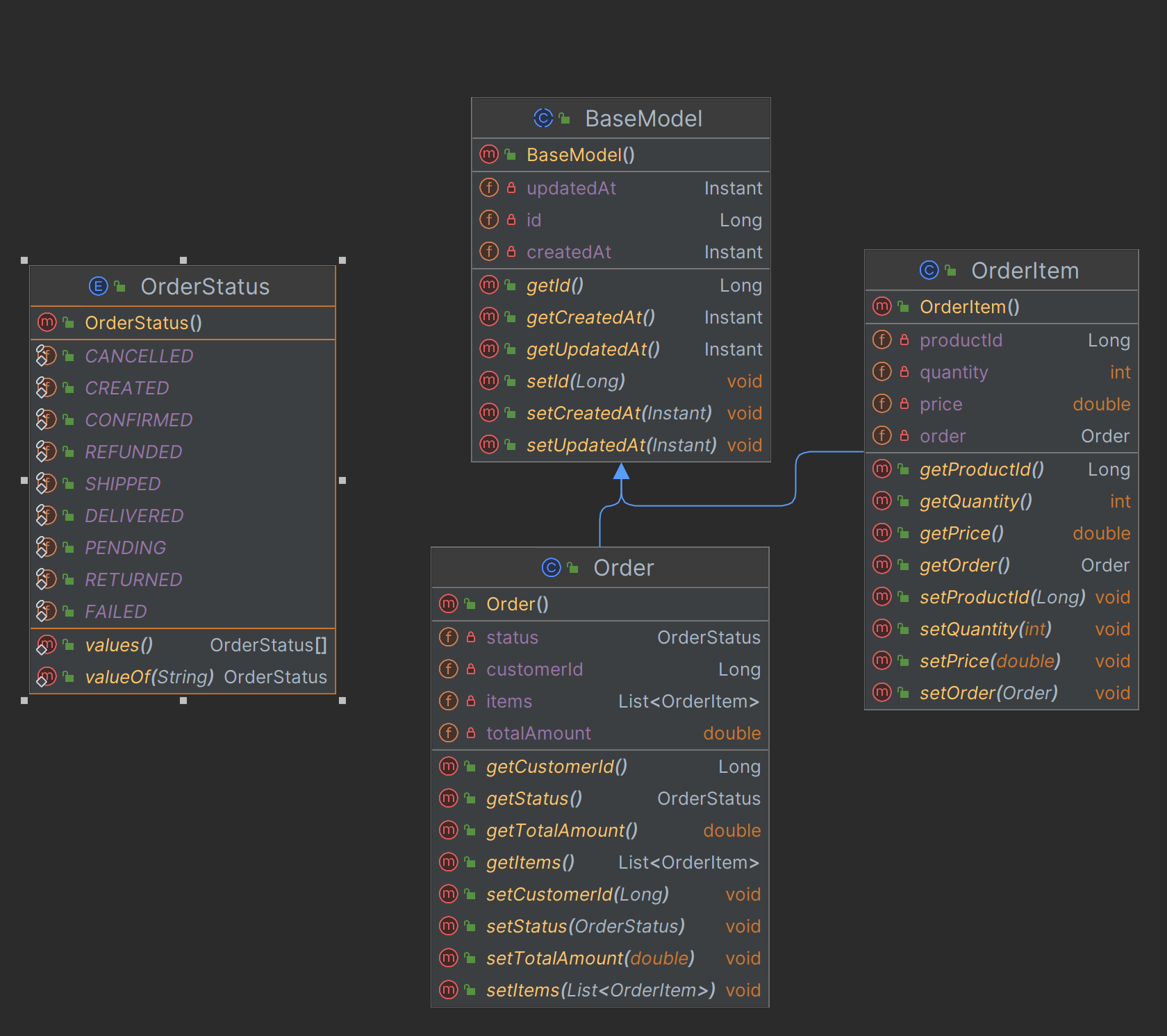
**Figure 3.3.6**: Cart Configuration

* + **Exceptions**



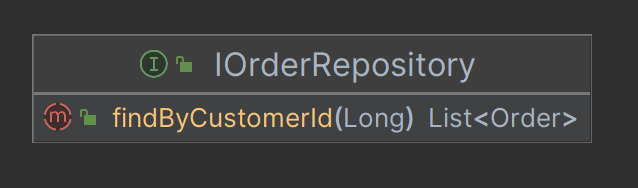
**Figure 3.3.7**: Cart Exceptions

1. **Order Management Service**
   * **Models**



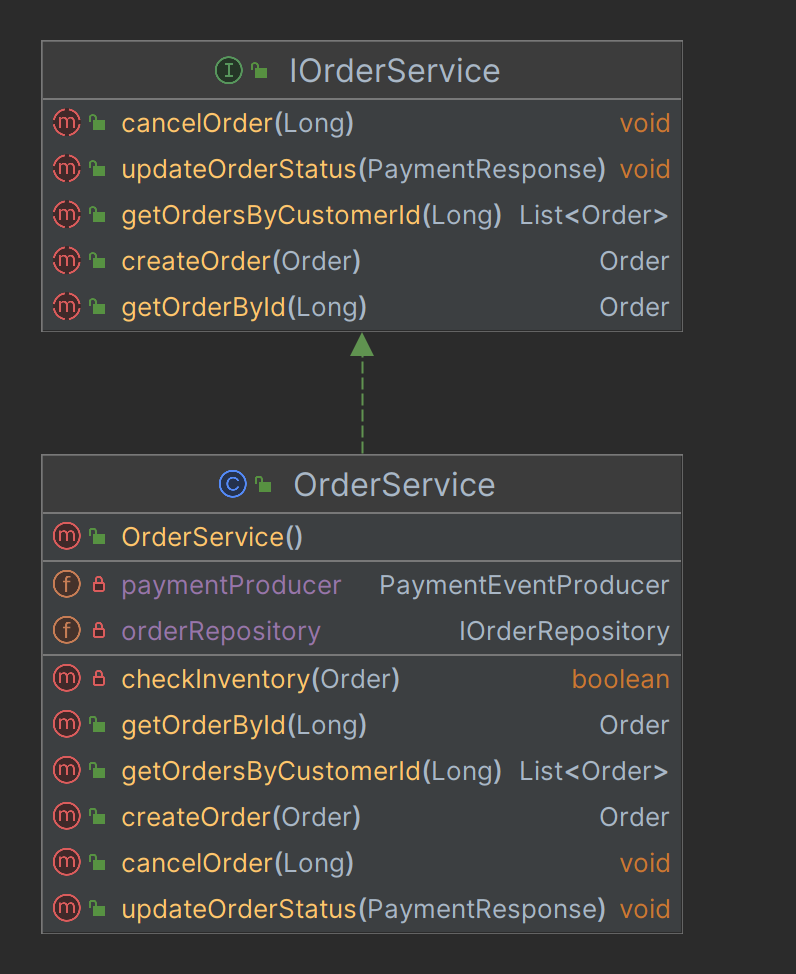
**Figure 3.4.1**: Order Management Models

* + **Repositories**



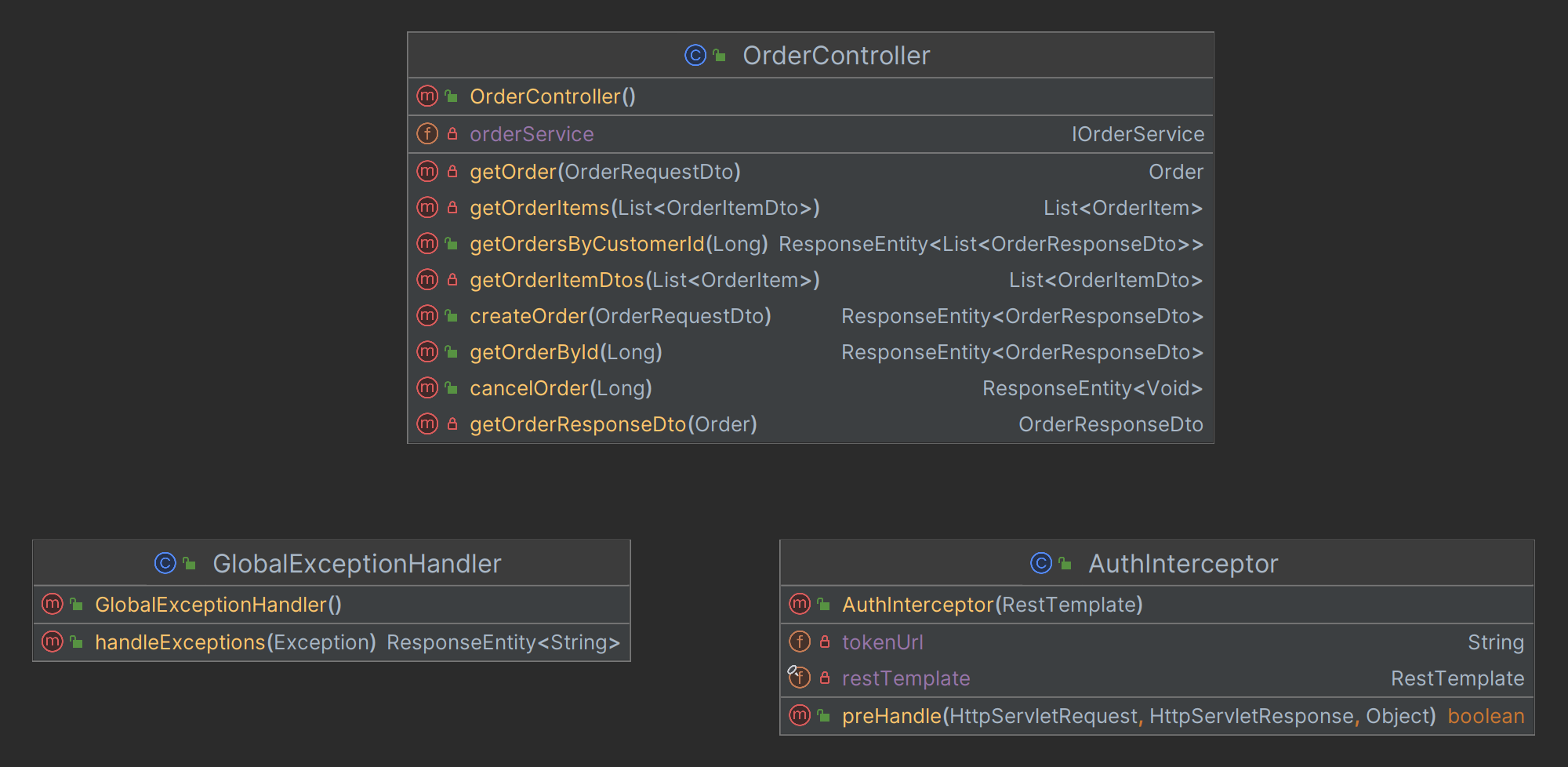
**Figure 3.4.2**: Order Management Repository

* + **Services**



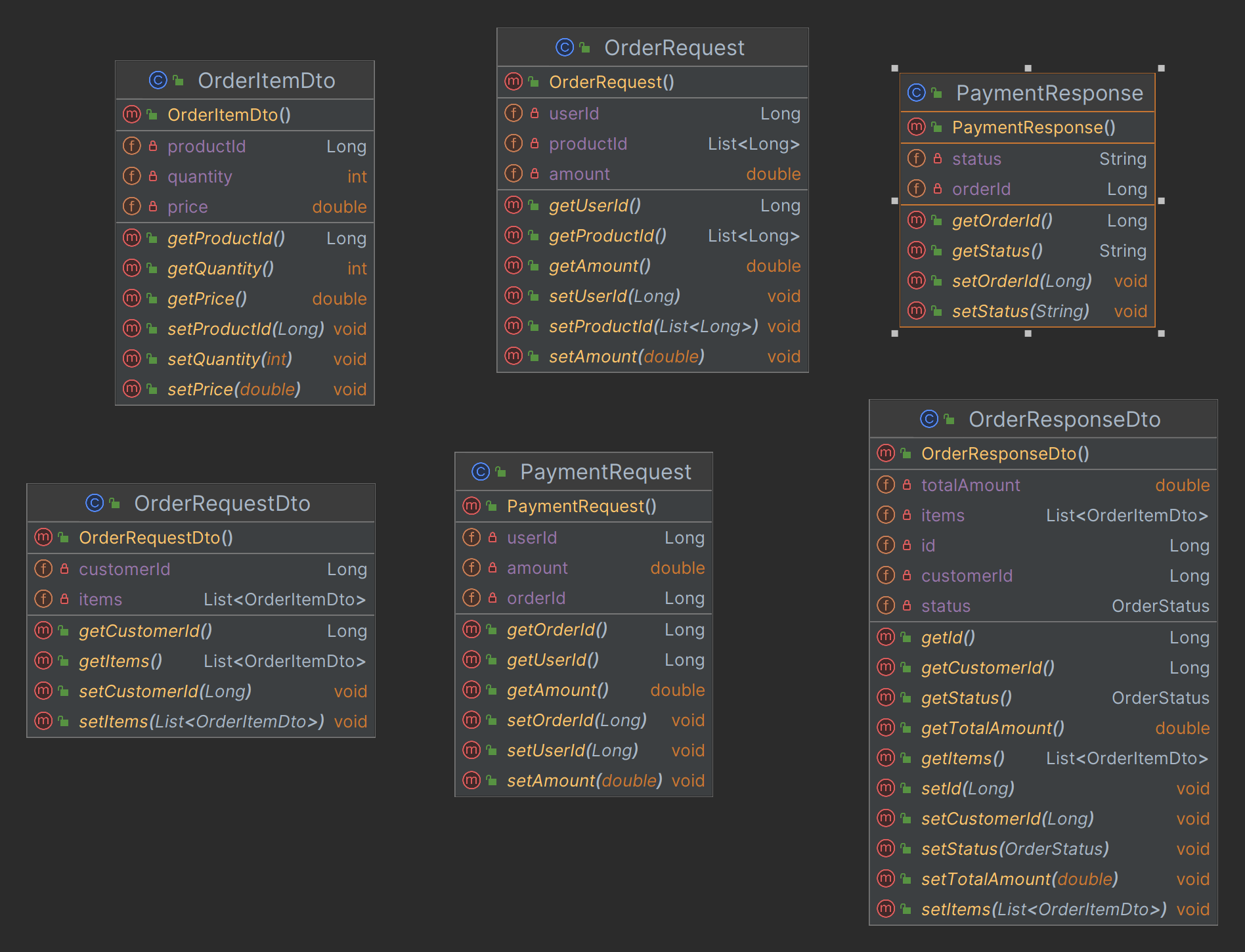
**Figure 3.4.3**: Order Management Service

* + **Controllers**



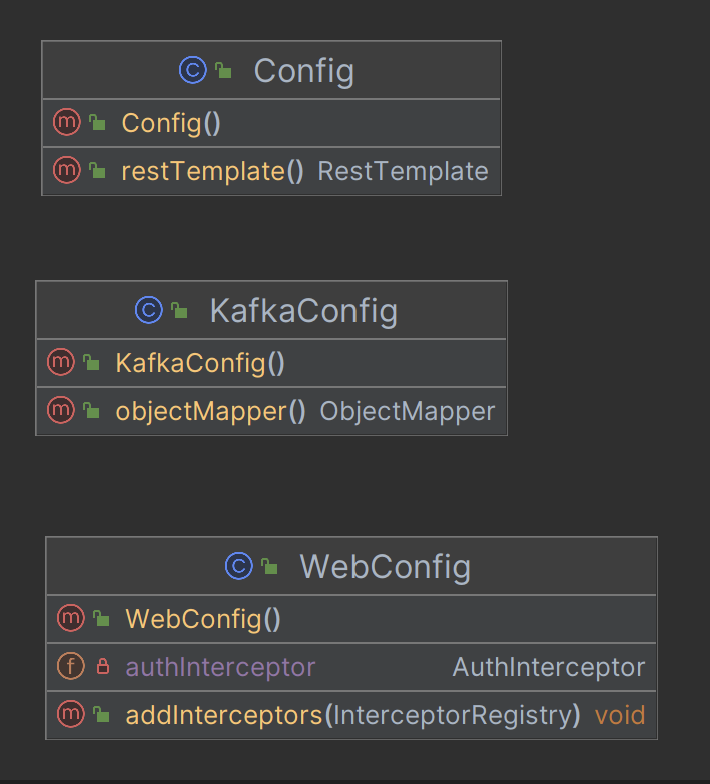
**Figure 3.4.4**: Order Management Controller

* + **Dtos**



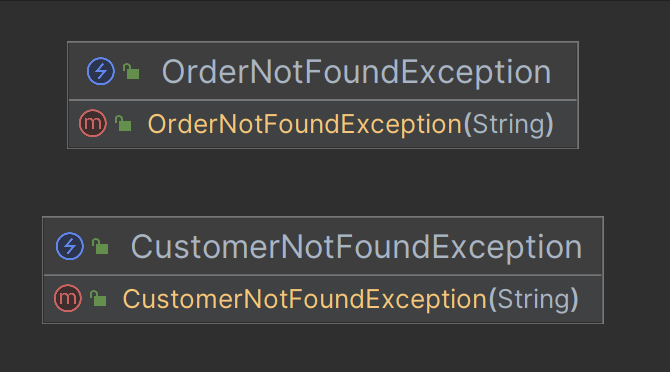
**Figure 3.4.5**: Order Management Dtos

* + **Configs**



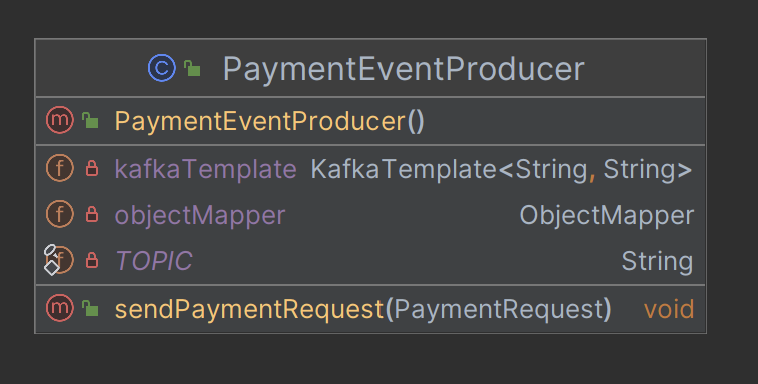
**Figure 3.4.6**: Order Management Configurations

* + **Exceptions**



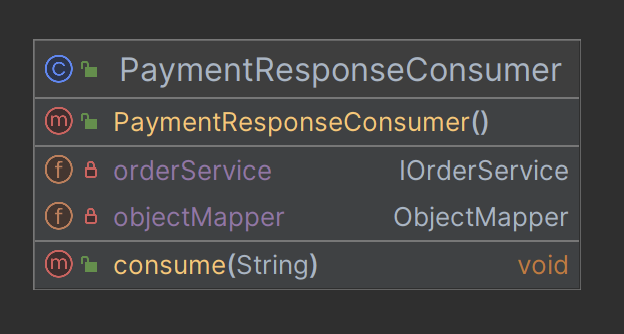
**Figure 3.4.7**: Order Management Exceptions

* + **Producers**



**Figure 3.4.8**: Order Management Kafka Producer

* + **Consumers**



**Figure 3.4.9**: Order Management Kafka Consumer

1. **Payment Service**
   * **Models**



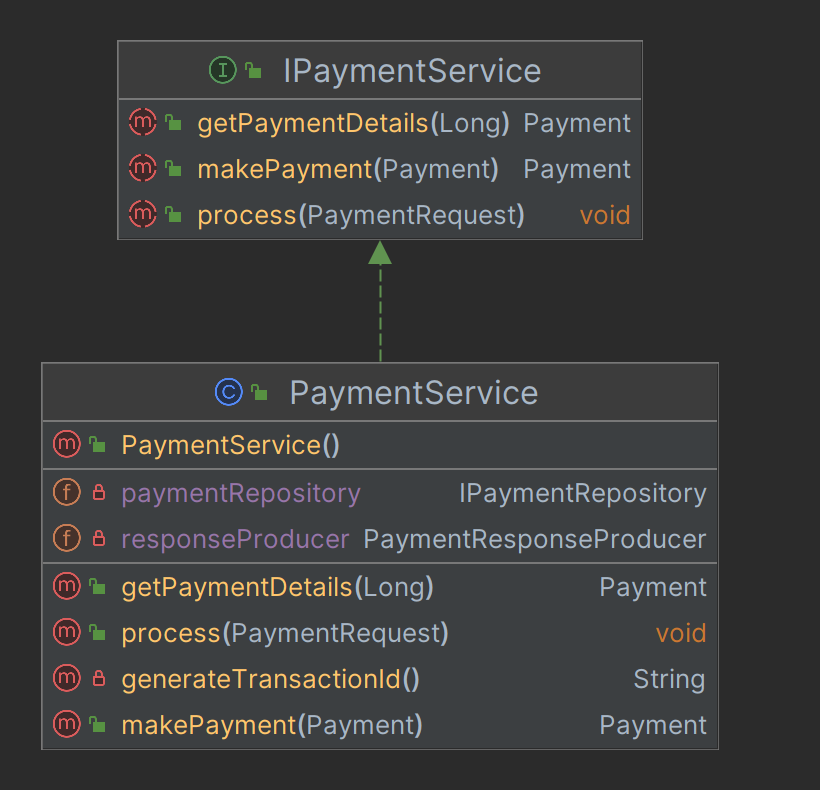
**Figure 3.5.1**: Payment Models

* + **Repositories**



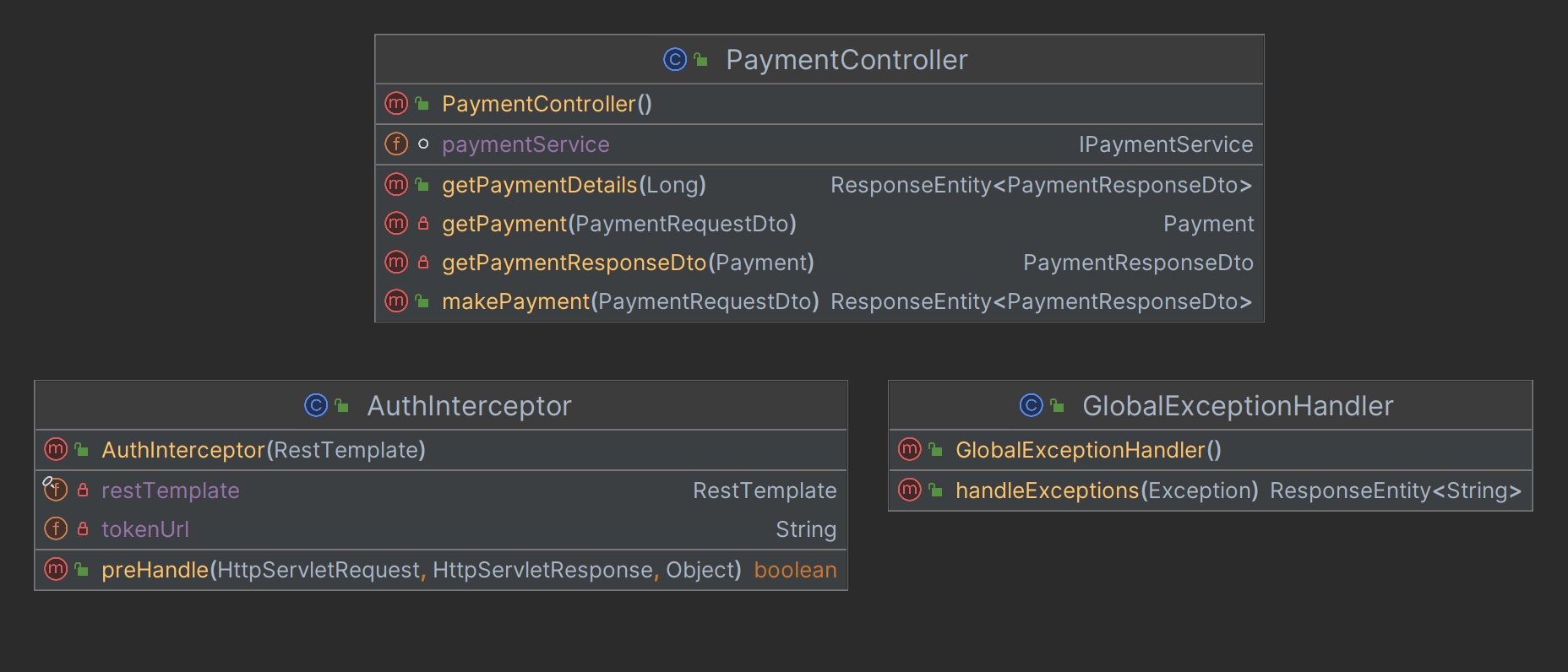
**Figure 3.5.2**: Payment Repository

* + **Services**



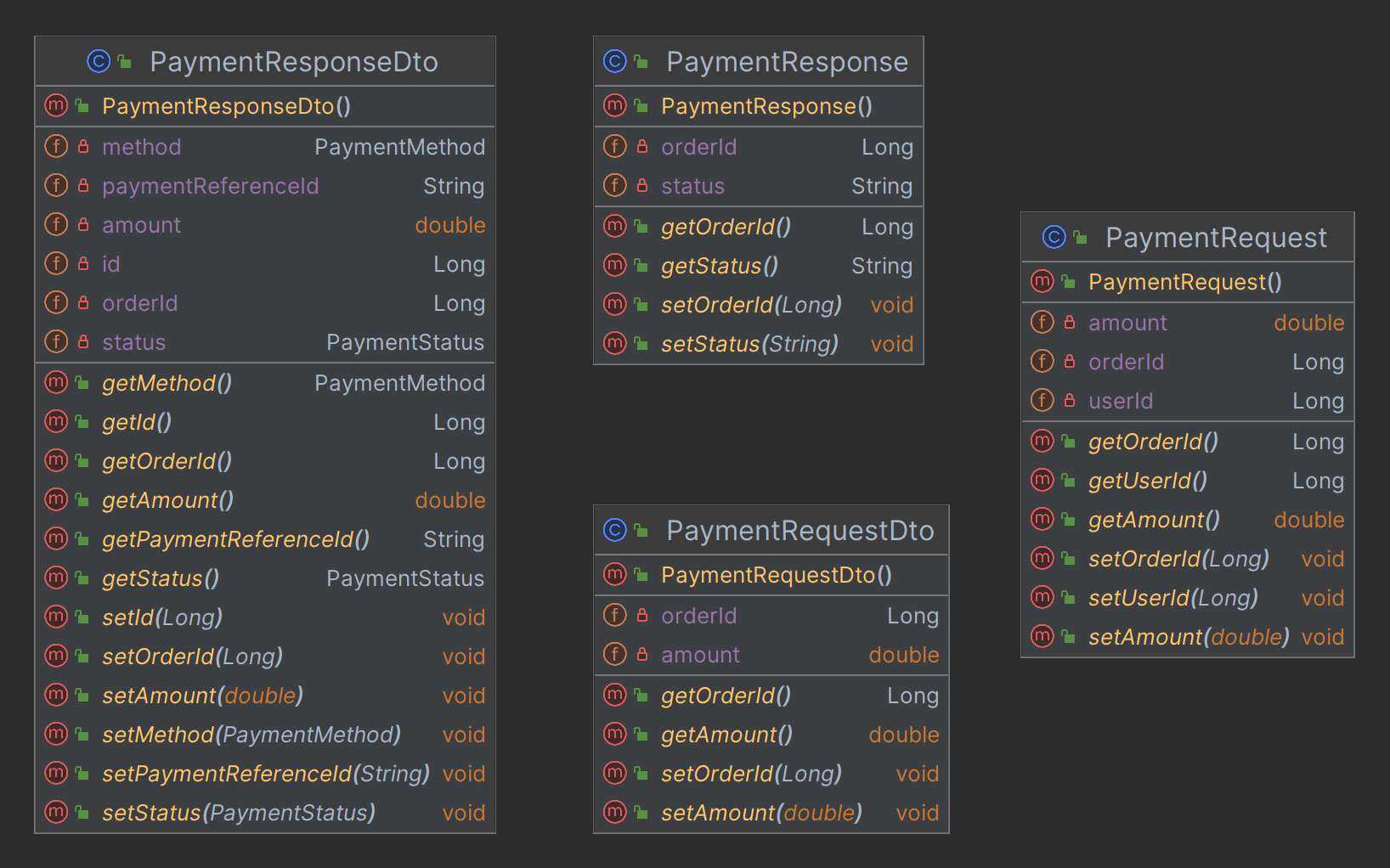
**Figure 3.5.3**: Payment Service

* + **Controllers**



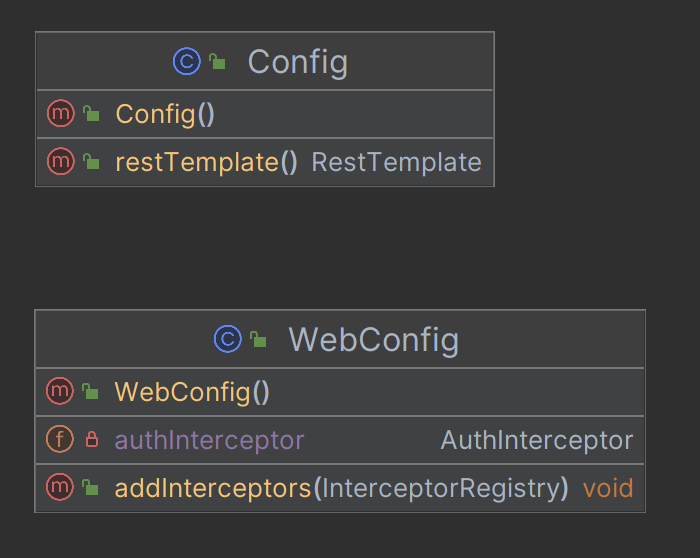
**Figure 3.5.4**: Payment Controller

* + **Dtos**



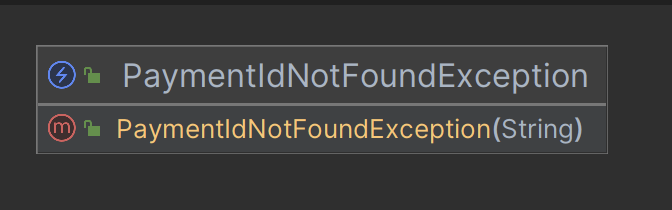
**Figure 3.5.5**: Payment Dtos

* + **Configs**



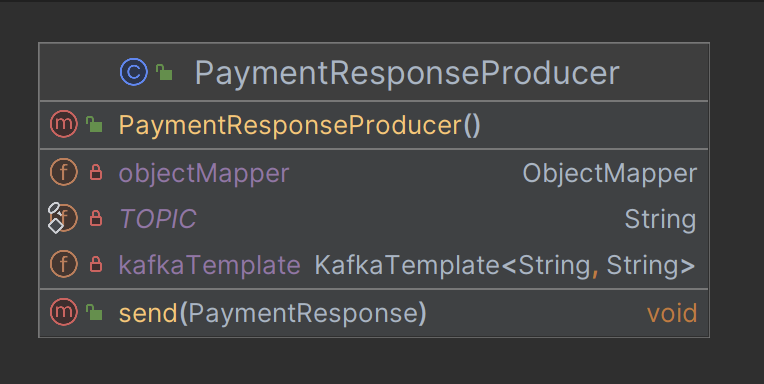
**Figure 3.5.6**: Payment Configurations

* + **Exceptions**



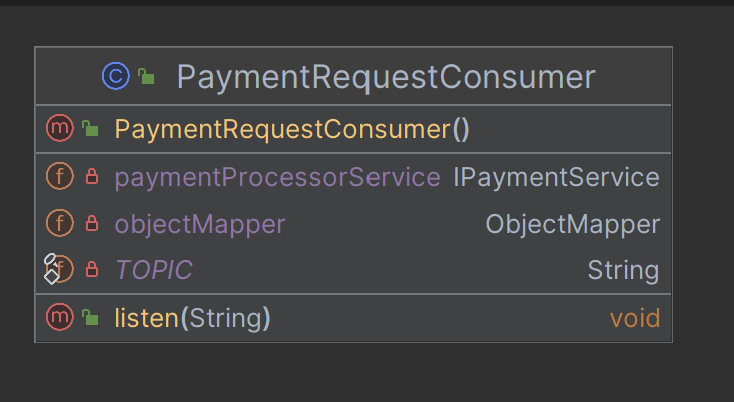
**Figure 3.5.7**: Payment Exceptions

* + **Producers**



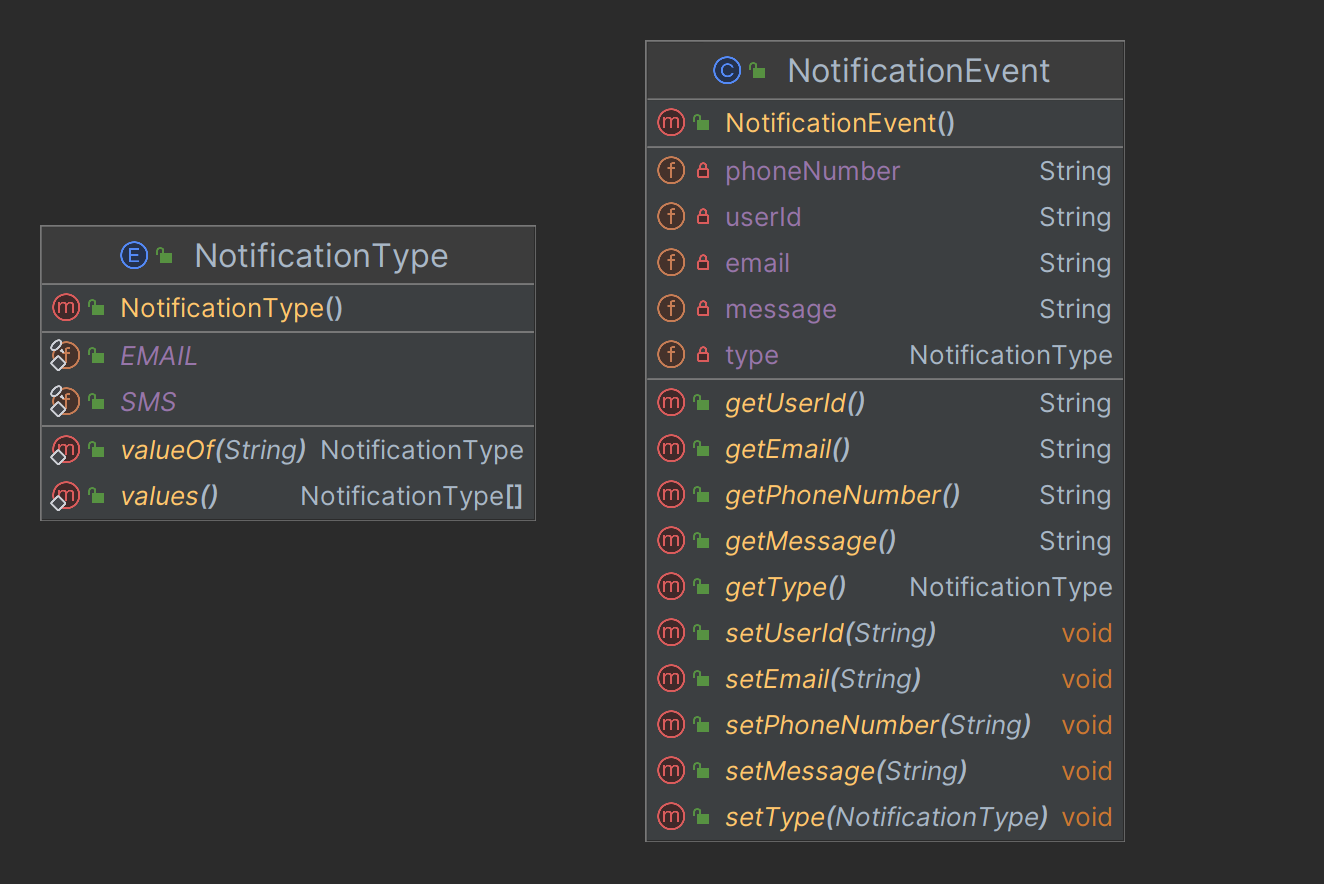
**Figure 3.5.8**: Payment Kafka Producer

* + **Consumers**



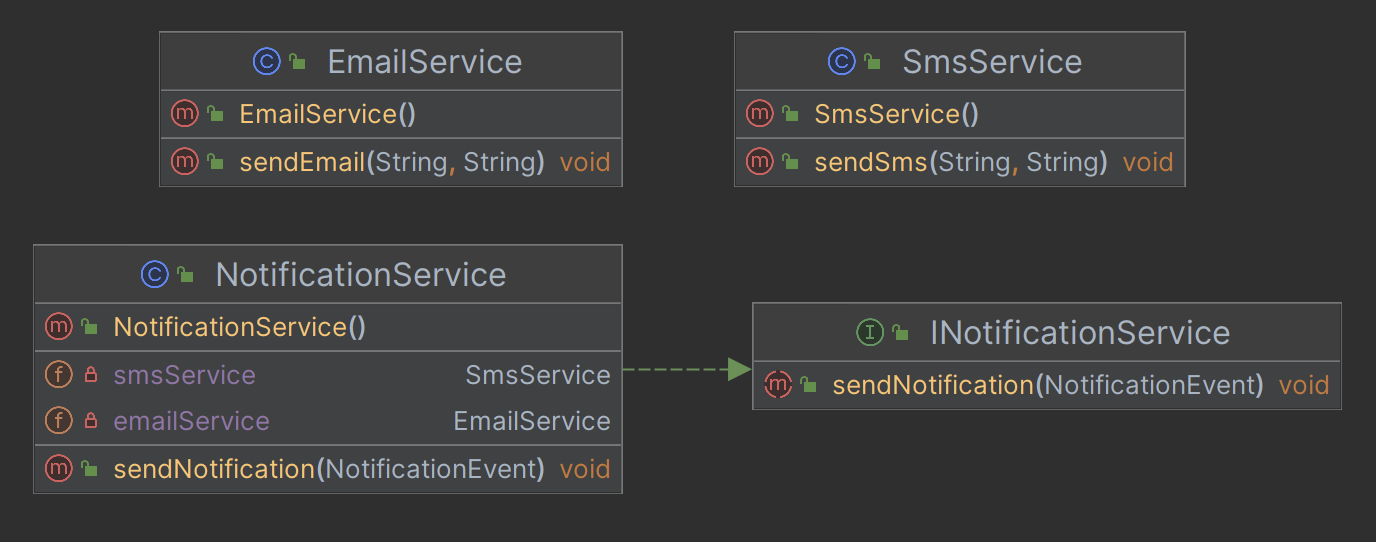
**Figure 3.5.9**: Payment Kafka Consumer

1. **Notification Service**
   * **Models**



**Figure 3.6.1**: Payment Models

* + **Services**



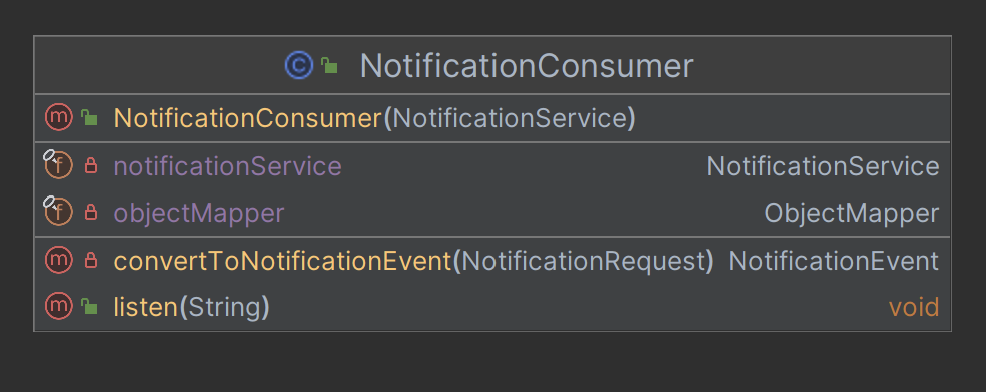
**Figure 3.6.2**: Payment Service

* + **Dtos**



**Figure 3.6.3**: Payment Dtos

* + **Consumers**



**Figure 3.6.4**: Payment Kafka Consumer

### Database Schema Design

1. **User Management Service**
   * Tables

users

* id (PK)
* state
* created\_at
* updated\_at
* username
* email
* password
* fullName

sessions

* id (PK)
* state
* created\_at
* updated\_at
* token
* user\_id (FK to users.id)
  + Foreign Keys
* sessions(user\_id) refers users(id)
  + Cardinality of Relations
* Between users and sessions -> 1:m

1. **Product Catalog Service**
   * Tables

products

* id (PK)
* created\_at
* updated\_at
* name
* description
* price
* image\_url
* cateogory\_id (FK to categories.id)

categories

* id (PK)
* created\_at
* updated\_at
* name
* description
  + Foreign Keys
* products(cateogory\_id) refers categories(id)
  + Cardinality of Relations
* Between products and categories -> m:1

1. **Order Management Service**
   * Tables

orders

* id (PK)
* created\_at
* updated\_at
* status
* total\_amount
* customer\_id

order\_items

* id (PK)
* created\_at
* updated\_at
* price
* product\_id
* quantity
* order\_id (FK to orders.id)
  + Foreign Keys
* order\_items(order\_id) refers orders(id)
  + Cardinality of Relations
* Between order\_items and orders -> m:1

1. **Payment Service**
   * Tables

payments

* id (PK)
* created\_at
* updated\_at
* amount
* method
* order\_id
* payment\_reference\_id
* status

### Feature Development Process

Pick One key feature - Talk about its development process, implementation and performance optimisation / metric optimisation achieved…

For example, ‘Book a seat’ feature in developing ‘BookMyShow’ app

Elaborate the request flow to backend

* 1. API Request Payload
  2. Service which picks the request
  3. Flow of MVC architecture

Explain the performance improvement / metric optimization achieved.

For example,

* Used Cache to reduce API Response time by X seconds…
* Optimized Query Response time by using Indexing…

Benchmarking of response time without the optimisation and post the optimisation

### Deployment Flow

Explain how the deployment will work via AWS (Describe the below) -

* EC2
* VPC
* Security Groups
* RDS
* Cache
* Managed Infra / Elastic Beanstalk

Use diagrams, images to explain better

### Technologies Used

Kafka, MySQL, Springboot, Cloud etc…

* For each key technology used in building the project,
  + Detail and describe each of them
  + Elaborate how they can be used in real life
  + Provide example of real-life applications using them

Use diagrams, images to explain better

**Tip** - Use the internet to improve your project but DO NOT PLAGIARIZE - Include proper references if you are quoting articles from the internet

### Conclusion

The Conclusion should include some key points as elaborated below -

* Key Takeaways: Highlight the important concepts and technologies learned from doing the Project
* Practical Applications : Significance of technologies with their real-world applications
* Limitations : Limitations of the technologies, cost implications and suggestions for improvement

## References

Include the websites or works or the list of works referred to in a text or consulted by you for writing this report

1. Name of the Website, Date and time of referring to the Website, Name of the Author, Title/Topic
2. Author Name, Title / Topic, Research Paper Name / Book Name, Year of Publication

Format Guidelines

1. Detailed and Elaborate report of 40 pages at least is expected
2. Margins - Every page of your document must meet the margin requirements of 1.25 inches on the left and right, and 1 inch on the top and bottom.
3. Font:
   1. Style: Times New Roman,
   2. Font Size:14 (For Headings), 12 (For body text) in black colored text.
   3. All text must be the same justification, like left justified or fully justified.
4. Line Spacing:
   1. Body of the text: 1.5
   2. List of Tables/graphs/charts/bibliography: single line.
5. Alignment:
   1. Title page: Centre
   2. Chapter Heading: Centre
   3. Subheading: Left
   4. Body of the text: Justify
6. Titles: All titles and subtitles should be in bold. All tables/graphs/charts/figures should have appropriate titles.
7. Numbering of the tables, charts, graphs should be in the following fashion: Second table/graph/chart in the second chapter should be numbered as Table/graph/chart no. 2.02; where the first digit stands for chapter no. and digits after (.) stands for number of table/graph/charts in that chapter. Same numbering should be followed for all other chapters.