**Microservices Architecture for an E-commerce Application**

**Task :**

**Building a Microservices Architecture for an E-commerce Application with .NET Core, Ocelot Gateway, PostgreSQL, Kafka, and Clean Architecture.**  
**1. Introduction**

This report provides an overview of the microservices architecture developed for the product and order services with auth service for authentication and authorization and API gateway.

The following sections detail the architecture, implementation, and key features of this microservices-based e-commerce solution.

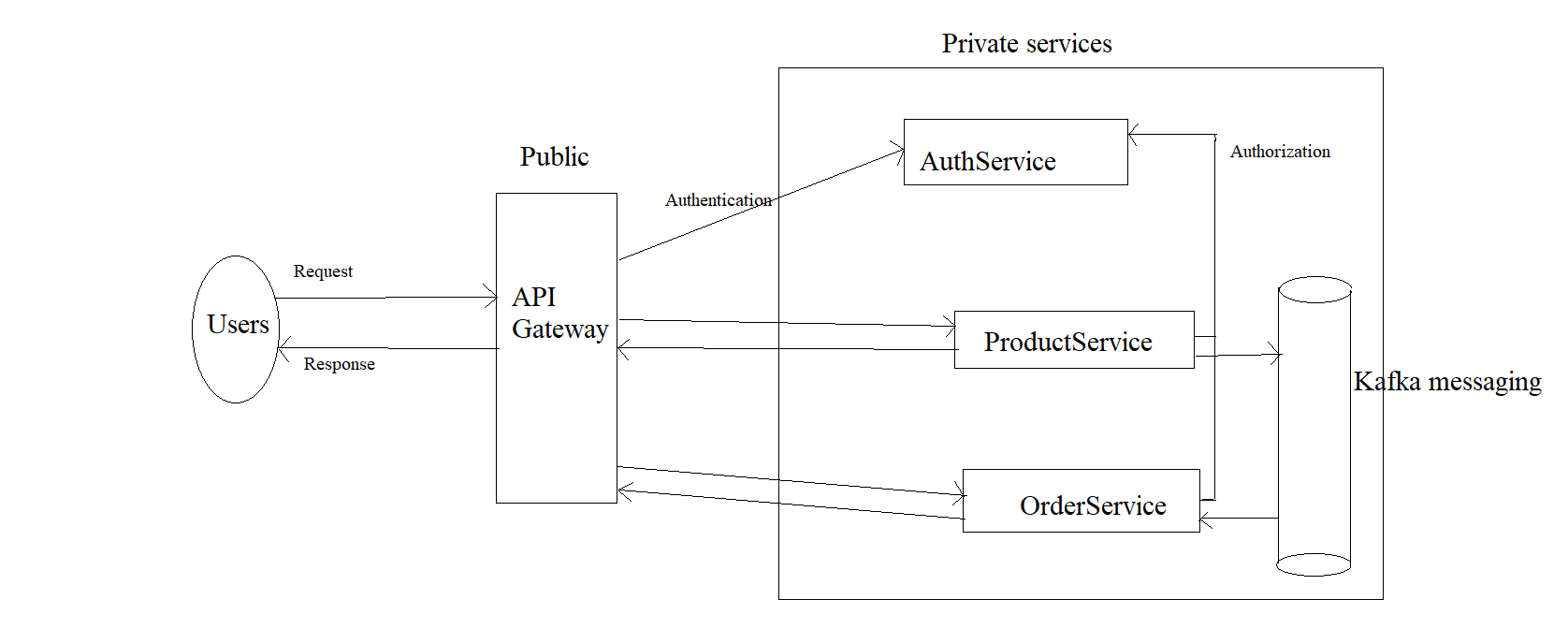


Figure 1: Project Architecture

* The application comprises two microservices: a product microservice and an order microservice. These microservices communicate with each other through an Ocelot gateway, ensuring efficient routing and management of client requests.
* Data persistence is achieved using a PostgreSQL database.
* Moreover, Kafka is employed as the message broker, facilitating seamless inter-service communication.
* Authentication and authorization is done through auth service.

**2. Microservices Overview:**

The developed microservices architecture consists of four main components:

1. Product Service: Responsible for managing product-related functionalities.
2. Order Service: Handles order management tasks.
3. Authentication Service: Manages user authentication and authorization.
4. Gateway: Acts as an entry point for client applications to access the microservices.

**3. Service Architecture:**

The microservices are built using clean architecture principles, emphasizing separation of concerns and modularity. Each service follows a similar architectural pattern consisting of the following layers:

1. Domain Layer: This layer encapsulates the core business logic and domain models. It remains independent of any external dependencies and focuses solely on representing the business concepts.
2. Interface Layer: The interface layer defines contracts and interfaces for interacting with external systems or databases. It abstracts away the implementation details, facilitating flexibility and testability.
3. Application Layer: This layer houses the application-specific logic, including services and DTOs (Data Transfer Objects). Services orchestrate the interactions between different layers and components, while DTOs facilitate data exchange between layers and services.
4. API Layer: Implemented as ASP.NET Core Web APIs, the API layer exposes the microservices' functionalities to client applications. It serves as the entry point for external requests and handles communication with clients.

**4. Challenges Faced:**

* Difficulty in structuring the project from scratch due to the complexity of microservices architecture and adherence to Clean Architecture principles.
* Implementation challenges encountered while developing the authentication service and defining its flow within the microservices ecosystem.
* Challenges in writing unit tests for services and repositories, ensuring comprehensive test coverage while maintaining code quality and readability.
* Configuration complexities arose while setting up Kafka for both publishing and consuming messages, requiring careful consideration of topics, partitions, and serialization formats.
* Scheduled dedicated time slots outside of office hours for development and coding tasks, ensuring uninterrupted focus on the project.

1. **Overcoming Challenges**

* To overcome difficulties in structuring the project, thorough planning and research were conducted initially. Breaking down the application into smaller modules and defining clear boundaries between microservices helped in organizing the project effectively.
* Addressing challenges related to the implementation of the authentication service involved searching over internet, going through multiple blog post and took help form Chatgpt and also breaking down the authentication flow into smaller, manageable steps.
* For unit testing I took reference from my other task and also I read multiple blog post in internet.
* Kafka Configuration is most challenging task for me, I never configured Kafka by myself before this, and I took reference for couple of YouTube videos and Chatgpt.
* Also configuring Publisher and Consumer is challenging but I took help form internet and run docker compose file for Kafka, which I downloaded form internet.