Project 6

**Overview**

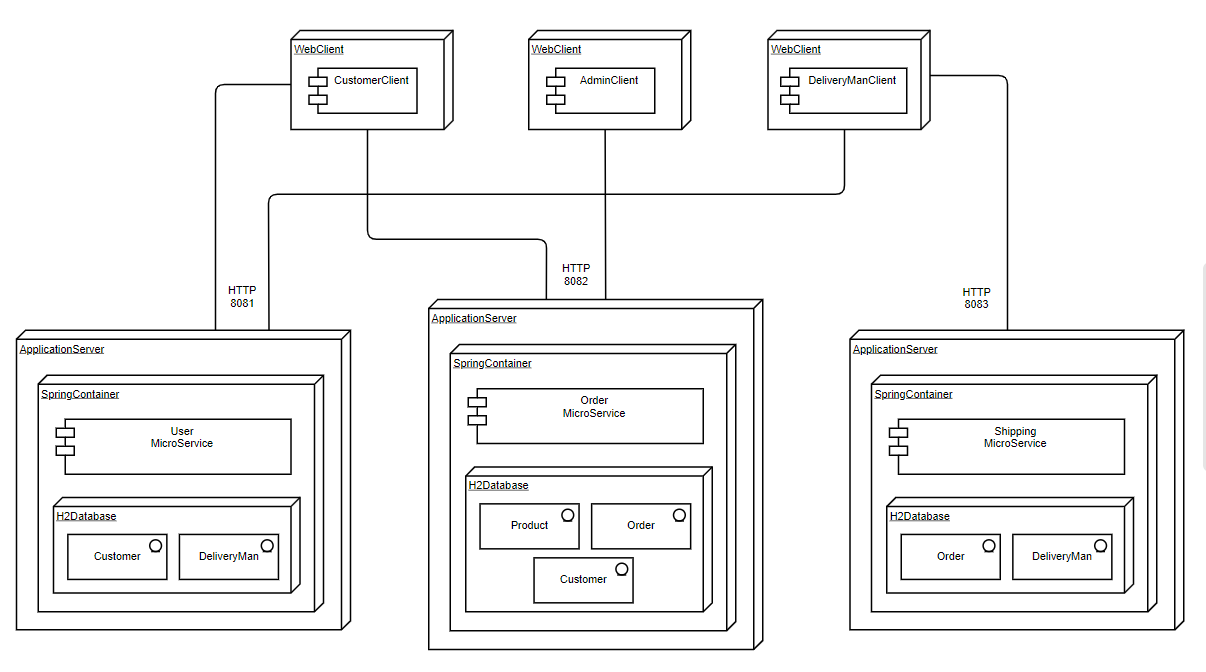
The goal of the project is to build a food delivery application backend; the actors interacting with the system are customers, admins and delivery men; the backend is based on the microservices architecture and uses Apache Kafka middleware for inter-service communications.

Microservices which compose the application are the user service, the order service and the shipping service; services can crash at any time and lose their state; the system must also provide faults tolerance and recover capabilities.

**Target system**

Services are designed to be used by three different web clients, one for each actor; each client can use a subset of functionalities of all the services to achieve application goals.

Services expose REST endpoints to the clients; each service is loosely coupled with the others, and it uses a private database which includes only service required data structures.



Deployment diagram of the system

We decided to develop the application following the fully event-driven microservices architecture: each service notifies events to other interested services using only (Kafka) messaging component.

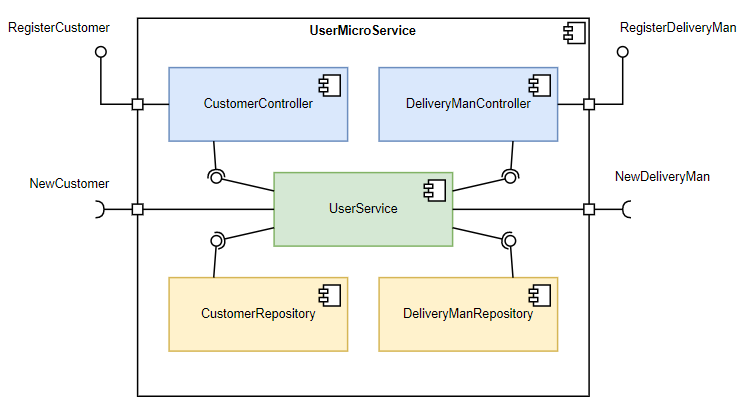
In this way each service can cache required data of other services, loosing space but increasing system response time, because each service calls the DBMS as well, but it does not need to also call other service first.

Kafka messaging QoS is set to exactly once delivery semantics in order to properly recover on service crash, following exactly one distributed backend execution; replication factors are set to 2 to provide faults tolerance to topics, and all of them have 2 partitions to show Kafka messaging scaling capabilities.

**Implementation details**

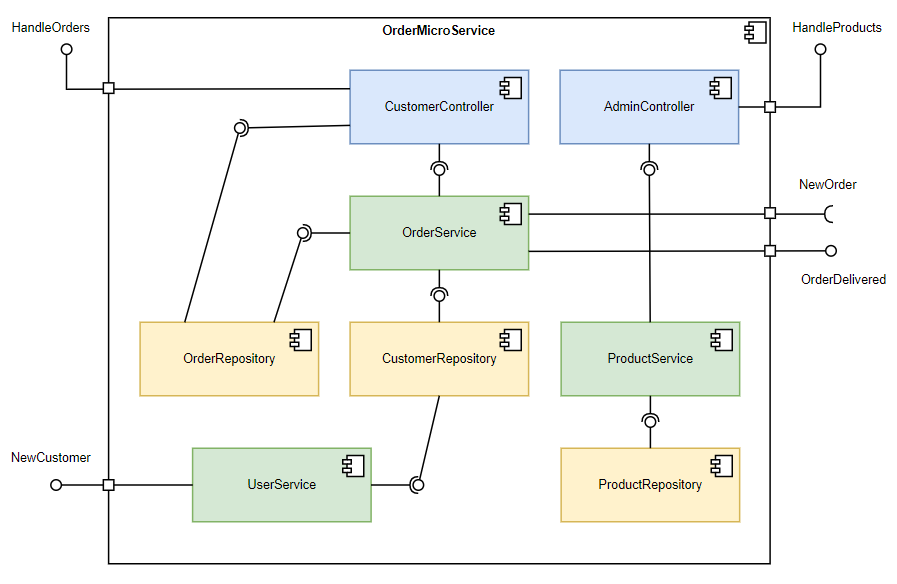
The system has been implemented using Spring BOOT framework, JPA, Hibernate and Spring Kafka for binding the middleware; the database system has been developed by exploiting an in memory H2 database system.

**User microservice**

* Provide registrations functionalities to customers and delivery men
* Notify new customers to order service and new delivery men to shipping service, for user validation purpose

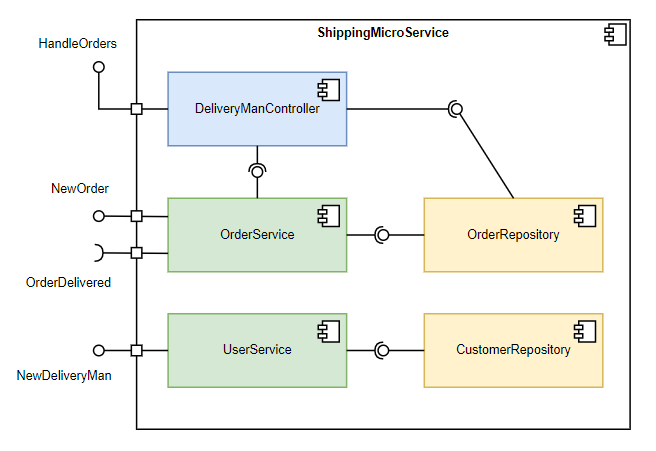
Components diagram of user microservice

**Order microservice**

* Allows customers to place an order or to check all their orders
* Allows admins to add/delete products available for purchase
* Notify to shipping service when a new order has been placed
* Listen from shipping service for delivery completion events, in order to update order states for the customers to observe it

Components diagram of order microservice

**Shipping microservice**

* On new order assigns a delivery man to it
* Allows delivery men to signal an order delivery event, which is then notified to order service

Components diagram of order microservice