Service based software architecture and Cloud Computing

A Cloud Native Microservice System
for Ticket Booking

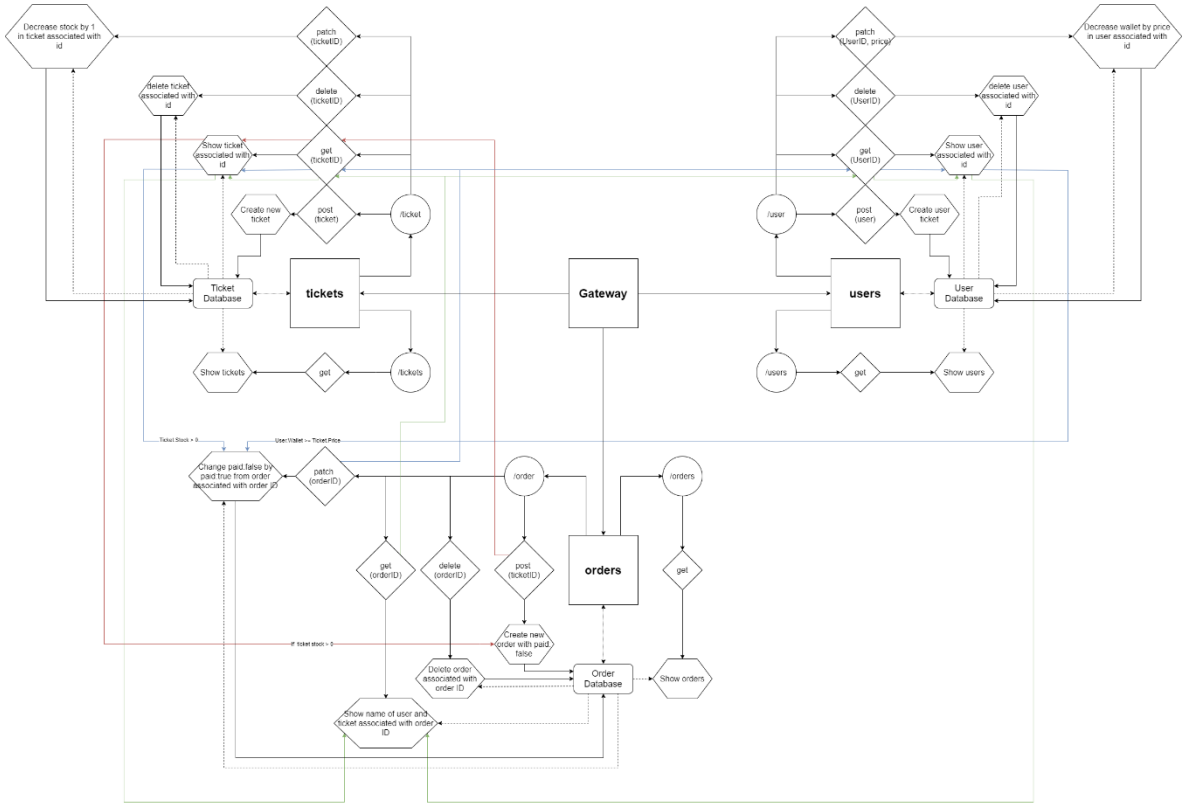
Cyril Canillas - 19129148
Marc Delrue - 19129063
Sébastien Tan - 19129054
Aymeric Vernholles - 19129125

Architecture and Design pattern

Our application uses nodeJS with the express package, MongoDB for the database.

Our application consists of 3 micro-services: Ticket, Order and User, and a gateway. Each micro-service has its own database. The first one contains the tickets, which have an ID, a name, a price and a stock. The second one contains the purchase orders, and is made up of an ID, a userID which is the ID of the user placing the purchase, a ticketID which is the id of the ticket that wants to be purchased, and a paid status, which defines whether the purchase has been paid or not. The last one contains the users, and has an ID, a name and a wallet.

The gateway let us access the multiples services with a single address.



Above is a diagram of the application architecture.

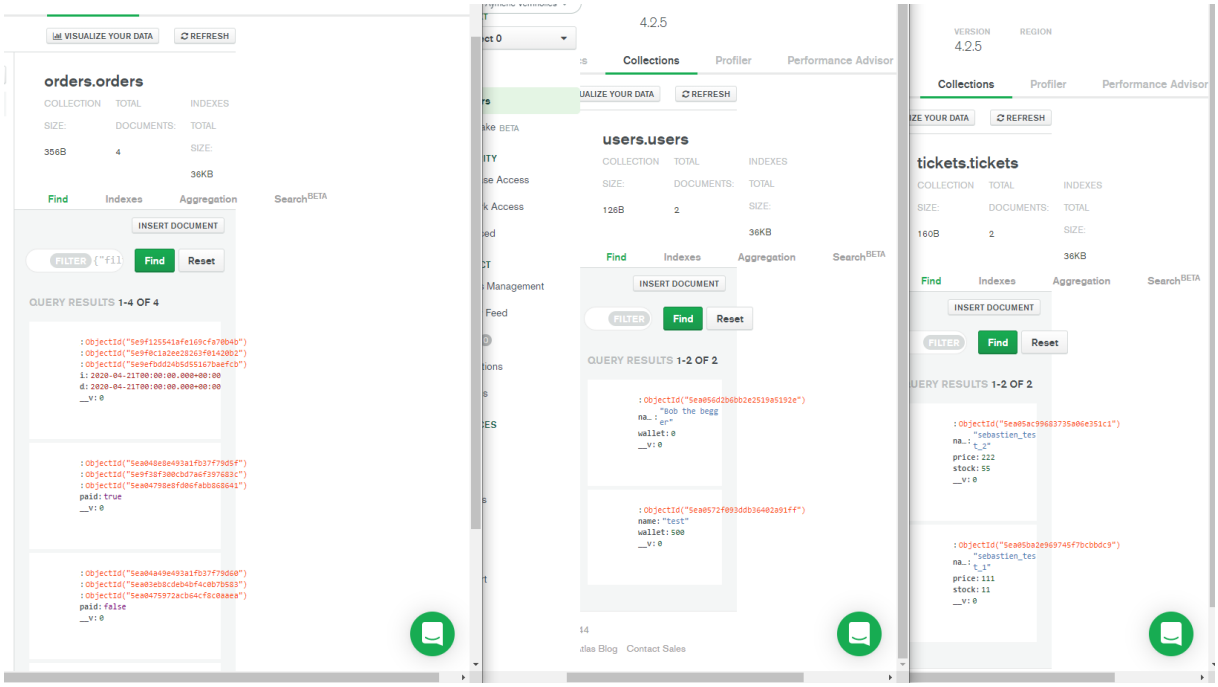
You can create, view, remove money or delete an account through the Users micro-service. You can create, view, reduce stock or delete a ticket using the Tickets micro-service. And finally, you can place an order, which checks if there is stock left before, display purchase orders, delete a purchase order and finally validate a purchase order, which reduces the stock of the associated ticket and withdraws the user's money.

Deployment guide

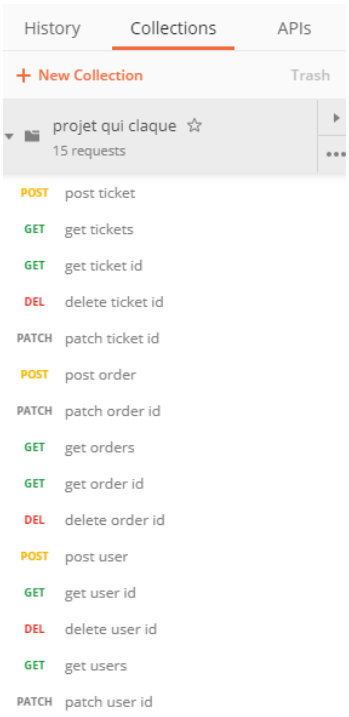
To deploy our application, you need to install nodeJS, clone the repository and launch launch.sh (which launch every micro-service, and the gateway). You also need to have access to our databases located in orders/orders.js, tickets/tickets.js and users/users.js or modify the files to put your own databases.

Testing results

Our 3 databases:



Our postman:



Link

Github : https://github.com/Setanas/service-based-software-architecture_assignment-1