Project Plan

Course Project Group information

Tamara Meghla

- 50391183
- tamara.meghla@tuni.fi

Mohammad Hassan

- 50457535
- mohammad.hassan@tuni.fi

Talha Manj

- 050365148
- talha.manj@tuni.fi

Group name: Vegabond

GitLab repo URL: webarch-2021 / Vegabond · GitLab (tuni.fi)

Working during the project

Each member went through the project document and understood the requirements. The flow of project was already defined in the project documentation. We have drawn the architecture, and started implementing it. RabbitMQ was new to us, so each of us read the recommended documentation on it.

We used GitLab Issue for assigning tasks and Issue board for tracking the work status. Besides that, each of us commited to help other than their own task. We used GitLab board as suggested. We have also used teams and whatspp as our communication channel during the project.

Issue Board URL: Development · Boards · webarch-2021 / Vegabond · GitLab (tuni.fi)

- **Aqib:** Responsible for dockerization, server B implementation, RabbitMQ and documentation.
- **Talha:** Responsible for server A implementation, RabbitMQ connectivity, docker Images and documentation.
- **Tamara:** Frontend implementation, connection with server and documentation.

Promised Hours

- Aqib > 4 hours per week
- Talha > 4 hours per week
- Tamara > 4 hours per week

Actual Time spent

- Aqib > 6 hours per week
- Talha > 6 hours per week
- Tamara > 3 hours per week and last 2 weeks 6 hours per week.

System Details

The system architecture consists of a front-end application, server A & B back-end servers, RabbitMQ message broker, Swagger APIs for communication and Mongo DB database to store the data.

In the frontend we have implemented an UI for ordering sandwiches among the sandwich list(6 different Sandwiches). After placing an order, a customer can search the order by order ID and check it is received or not. If the order is not placed, the it will show a message: Not Found. Otherwise, after receiving order it will show received. Sandwich APIs are used to keep up the communication between the front-end application and the back-end server. RabbitMQ maintains a task queue for both the server ends perspective.

System Architecture

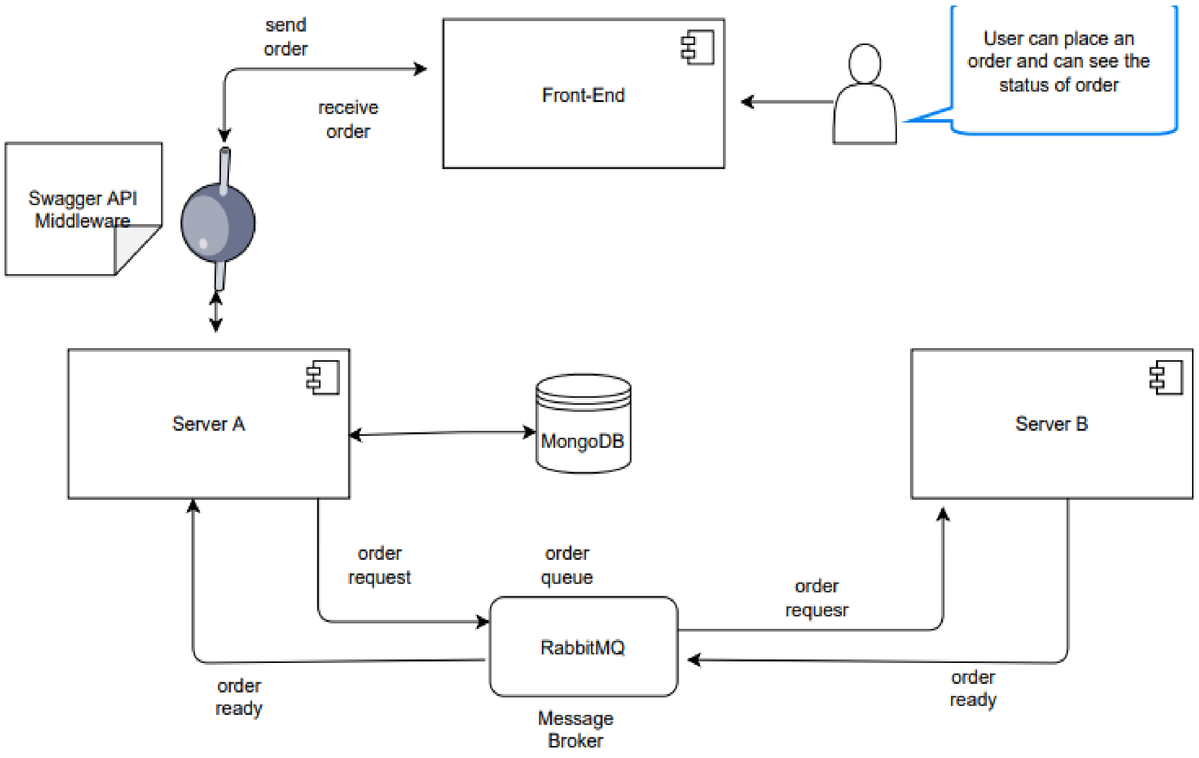

Figure 1: System Architecture

Used Tool & Technologies

For development, we have used:

- GitLab

- ❖ NodeJS
- ❖ ReactJs
- ❖ Docker
- ❖ React-Bootstrap
- ❖ MongoDB

Libraries:

- ❖ Express
- ❖ Amqplib
- ❖ Swagger-tool

Play with the system

- Step 1: start mongo DB database
- Step 2: run backend with docker-compose up
- Step 3: run server A with node index.js
- Step 4: run frontend with npm start

Learning during the project

We have learned a lot from this project, especially about docker and using RabbitMQ as a broker. These two technologies were new to all our group members. And implementation of Swagger API was also something new to us. After working on this project we realize that how docker helps to maintain the scalability of a project by putting every component in a container. Although the outcome was not perfect but this project gives us a chance to learn about some new technologies and their uses.