Final Iteration - Project Specifications

ShopAholytics

Course: IES - Introdução à Engenharia de Software

Date: Aveiro, 25th January 2022

Students: 97505 | Alexandre Serras
98008 | Gonçalo Leal
98679 | João Farias
97636 | Vasco Regal

Project Abstract:

ShopAholytics is a web-based application that offers analytics related to shoppings. It allows managers and marketing teams to study client behavior: rush hours, most/less visited stores and other statistics related with shopping traffic.

Apart from statistics with past data, **ShopAholytics** also presents realtime information associated with customer flow.

To simulate a real-world situation, we can use an infrared sensor connected to a Raspberry Pi.

1. Index

2. Introduction

The present document will describe client requirements and project structure, within the scope of the IES course.
ShopAholytics aims to help shops and shopping centers managers to plan their selling strategies more efficiently based on traffic statistics calculated by this web-application.
Personas, Use Cases, Main Scenarios will be explored in the following chapters in order to prove the application concept. To conclude the project specifications, the chosen architecture will also be described.

3. Product Concept

3.1 Vision Statement

Our system is meant to be used in an informative way not only for the shopping users, but also for shop managers and shopping responsibles. The system provides real time data about almost all shopping information, for example free parking spots, shops and shopping occupations, shop details…
With this product, we can help enterprises and the shopping by itself with marketing aspects and we can provide information to people about shopping and free park spots occupation, which will reduce too much traffic.
Our goal is to provide a tool that makes managers' lives easier and provides enough data for business analytics. If well used it is a powerful tool, capable of giving advantage to shopping centers that use it.

3.2 Personas

Mariana is a 22 year old law student born in Porto that currently lives in Lisbon. In spite of always living in big cities, she hates spaces with a lot of people, especially shops. This is a bit contradictory, because her favorite shop (*Primark*) always has a lot of people.
In her whole life, she frequently used to go with her parents and her young brother to spend time in the biggest shopping in his homeland.
Even living in a new city, her passion about shopping doesn't disappear, but since the Covid-19 pandemic began and consequently shop restrictions, her passion is becoming lower.

António is a 37 year old shop manager that works in *Fórum Aveiro* for 20 years. He is married and has 3 sons, often going with them on outdoor activities.
He is a curious and a focused person, always trying to increase sales and to captivate people to get in the shop. For this process he looks for the shop's analytics, such as most and less saled products, people average affluence.
Despite his effort counting how many people are in the store each moment, this is almost an impossible mission.
To analyse shop statistics, he likes to store this data in his laptop in order to make graphics, but it is too much work and learning.

Armindo is a 52 year old shopping manager from Colombo Shopping Center since 2000. He is married and has 2 sons (1 boy and 1 girl).
There are two main characteristics that his teammates use to describe him: focused and curious. In his job, he's always trying to get answers to the daily situations, and doesn't stop until reach an answer.
For shopping maintenance, he is looking for statistics, such as free parking spots, most busy areas/shops, most busy days, etc...

3.3 Main Scenarios

- Mariana checks how many people are in the shopping and how many parking spots are available . Mariana opens the application, no login needed, and gets this info in the main page.
- Mariana checks if his favorite store is open and how many people are there - Mariana opens the application, no login needed, goes to the main page and sees if his favorite store is open and other information related to the store.
- António wants to know which were the most busy days in his stores last month - António opens the application, logs in, goes to “Statistics” and gets the results.
- Armindo has a budget to spend on the park, but he is not sure whether that is a good investment or not - Armindo opens the application, logs in, goes to “Statistics” and checks the occupancy percentage statistics to help him in his decision

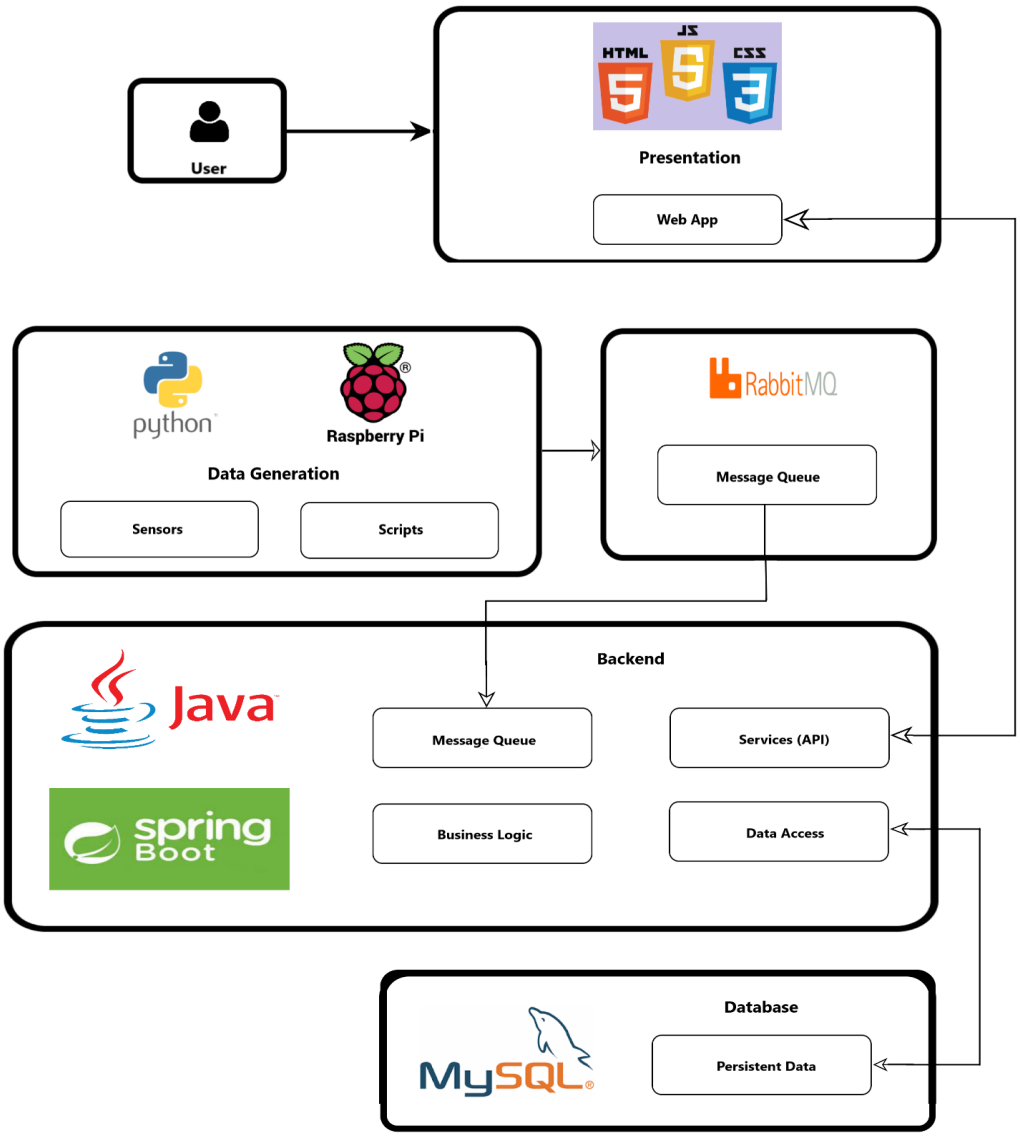
4. Architecture notebook

4.1 Key requirements and constraints

- The users should be able to access the Web-App from remote PCs or Mobile Devices with internet connection
- The shopping managers must be able to add stores, providing the store's name, business hours, capacity , location and store manager's email.
- The store managers need to register on the platform in order to access it. This is possible because when the shopping manager registers the store manager, the store manager receives an email with your provisional password. This includes providing personal information, such as a password, birth date and gender.
- The Web-App should be continuously available.
- The API should be be continuously available.
- Both the shopping and store managers must have access to statistics and charts.
- When the shopping or a store hits its maximum capacity its manager should receive a notification
- The sensors must register entrances and exits from the parking lot, shopping and store
- The Web-App should be protected with authentication and authorization, meaning that each manager only has access to his assigned stores or shopping and the people from outside cannot access the data.
- One random person can access the index page of a shopping.

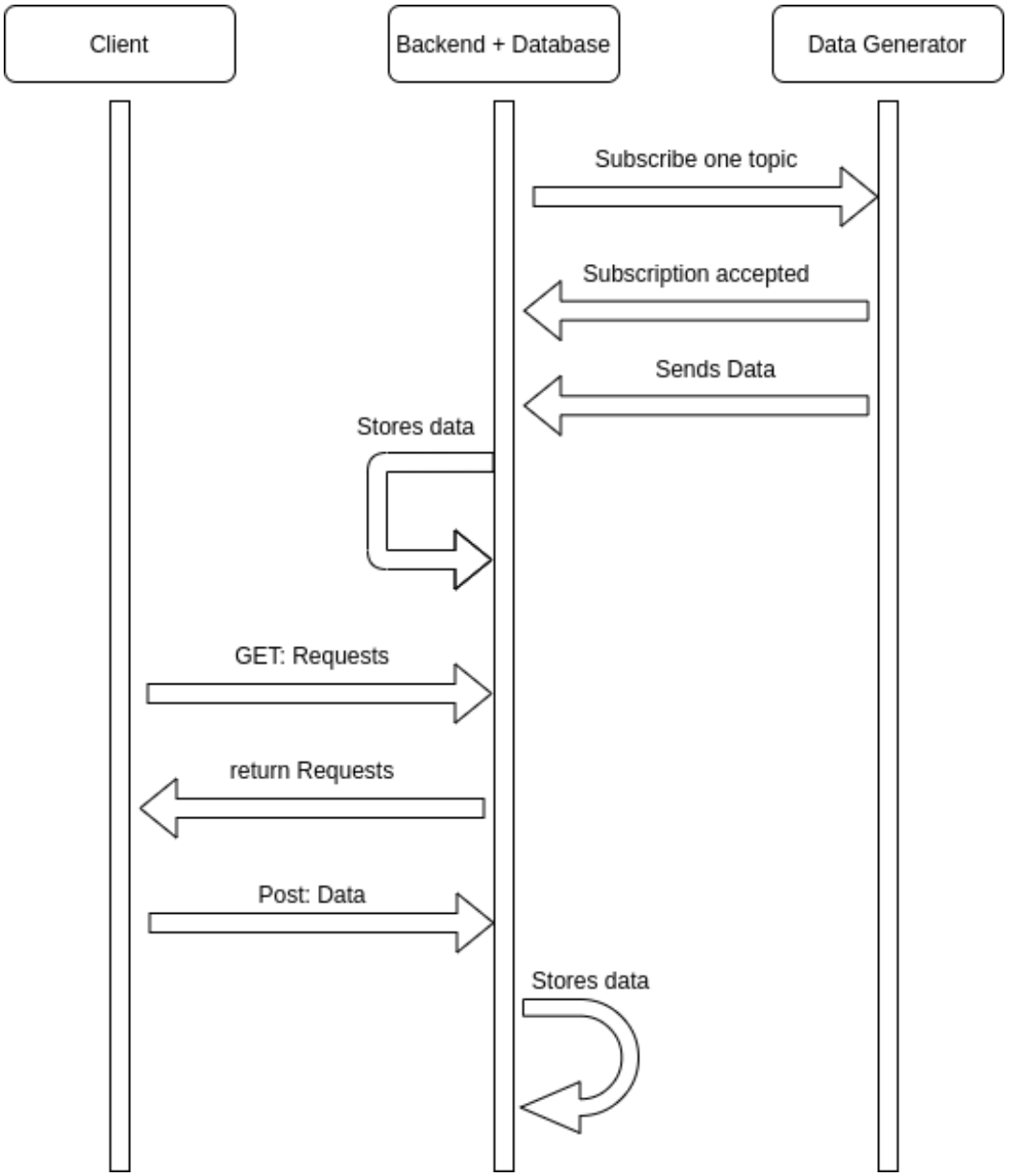
4.2 Architecture view

The architecture of the project is mainly composed of four parts: the data generation layer, the backend layer, the presentation layer and the database layer.
The data generation layer will generate all the necessary information about the shopping, send it through message queues using RabbitMQ to the backend, which after receiving the
messages, processes them and saves them to the database.
The database will be developed in mySQL, in order to be able to store the necessary information.
The client side, developed in HTML + CSS + JavaScript, will communicate through the API to obtain the necessary information.The backend will be developed using Spring Boot with Spring Security, being able to communicate with the client side through Web sockets.



4.3 Module interactions

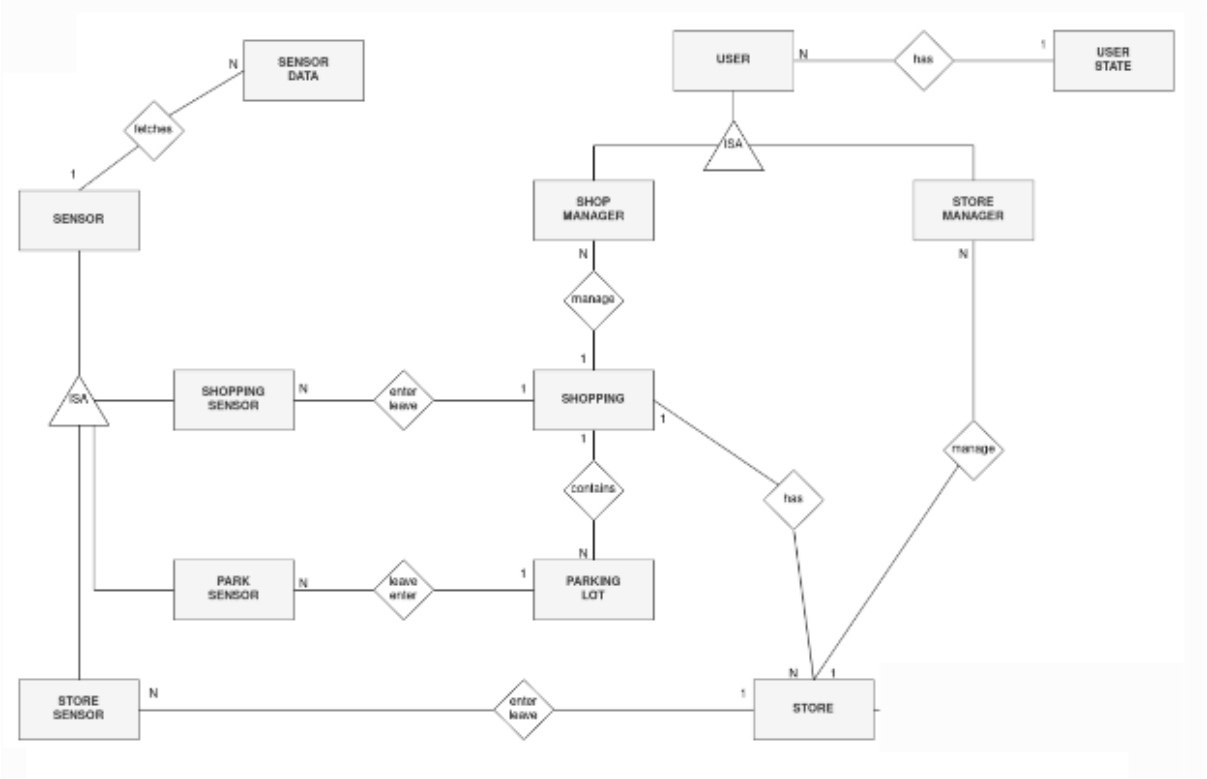
A typical interaction between the different modules consists of accessing the Web App , providing the user login. The client side calls the API and the backend checks whether the login is valid or not using the information stored in the database. The Web App then checks the user's type and displays the appropriate dashboard to the given user. If the user is a shopping manager, he is able to see all information about it. The sensors' module generates data for each park/shopping/store movements, which then sends it to the backend module through the RabbitMQ message queue. The backend updates the states. Example of possible basic interactions.

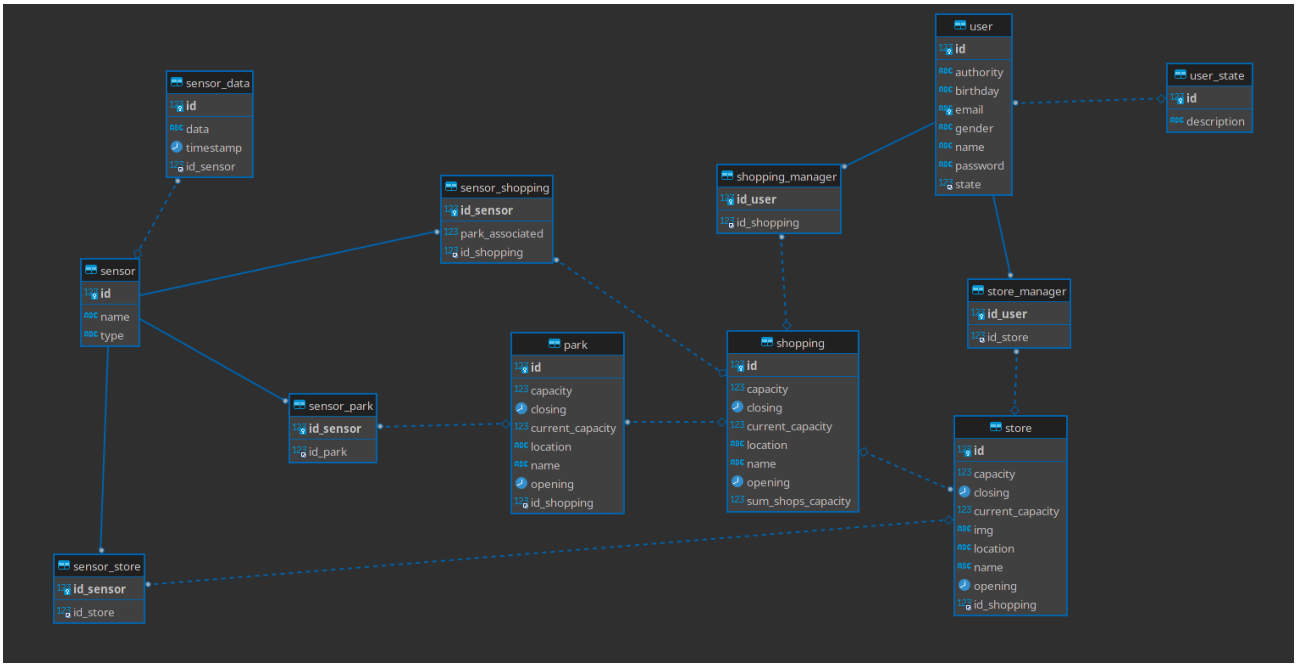


5. Information perspective

The next two diagrams represent all steps to reach the database domain of ShopAholytics's application database; it was obtained with Dbeaver.

This is used to store all information related to ShopAholytics.





6. API Documentation

All documentation of the API is available here.

7. Authentication

The authentication was implemented with JWT and ROLES.
All endpoints from mq are public because we have problems with the deployment so i can't fix the issues so we choose to put all public
The follow images shows the endpoints which role can access.

```
public static final String SHOPPING_MANAGER = "ROLE_SHOPPING_MANAGER";
   public static final String STORE_MANAGER = "ROLE_STORE_MANAGER";

   // ENDPOINT MANAGEMENT
   public static final String LOG_IN_URL = "/auth/login";

   public static final String[] PUBLIC_ENDPOINTS = {
       "/api/shoppings/Shoppings",
       "/api/shoppings/Shopping",
       "/mq/**"
   };


   public static final String[] SHOPPING_MANAGER_PROTECTED_ENDPOINTS =
{
       "/api/parks/*",

       "/api/users/addUser/{pid}",
       "/api/users/addUserState",
       "/api/users/Users",
       "/api/users/UserStates",
       "/api/users/deleteUser/{id}",
       "/api/users/User",

       "/api/storemanagers/addStoreManager/*",
       "/api/storemanagers/StoreManagers",
       "/api/storemanagers/StoreManagerShopping/{id}",
       "/api/storemanagers/deleteStoreManager/{id}",
       "/api/storemanagers/updateAcceptStoreManager/{user}",
       "/api/updateBlockStoreManager/{user}",
```

```
        "/api/stores/addStore/{pid}",
        "/api/stores/Stores",

        "/api/shoppingmanangers/*",

        "/api/shoppings/updateShopping",
        "/api/shoppings/deleteShopping/{id}",

        "/api/sensorsstore/*",

        "/api/sensorsshopping/*",

        "/api/sensorspark/*",

        "/api/sensors/deleteSensor/{id}"
    };
```