Beep Me

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 30 Janeiro 2022
Students:	97953 - Andreia Portela 98411 - Ricardo Ferreira 98607 - Diana Oliveira 100850 - Miguel Marques
Project abstract:	Beep Me is a web and mobile application that allows the consumer to follow orders progress, the restaurants to innovate the approach to their relationship with the consumer and the shopping center administrators to make better management of the food court.

Index

1. Introduction

In the scope of the IES course, we decided to develop a web and a mobile application that will manage orders and deliveries of food in shopping center food counters in order to adapt the lifestyle to the Covid-19 pandemic and improve the consumer experience, as well as benefit the shopping center restaurants.

The Beep Me application eliminates the need to use physical devices in the consumer-restaurant relationship, in addition to significantly reducing the crowding of people at certain points in the food court, thus reducing possible contagion chances.

Furthermore, the adhesion by restaurants to use the application allows the administrative part of the shopping center to have access to relevant information and important statistics, thus benefiting the shopping center.

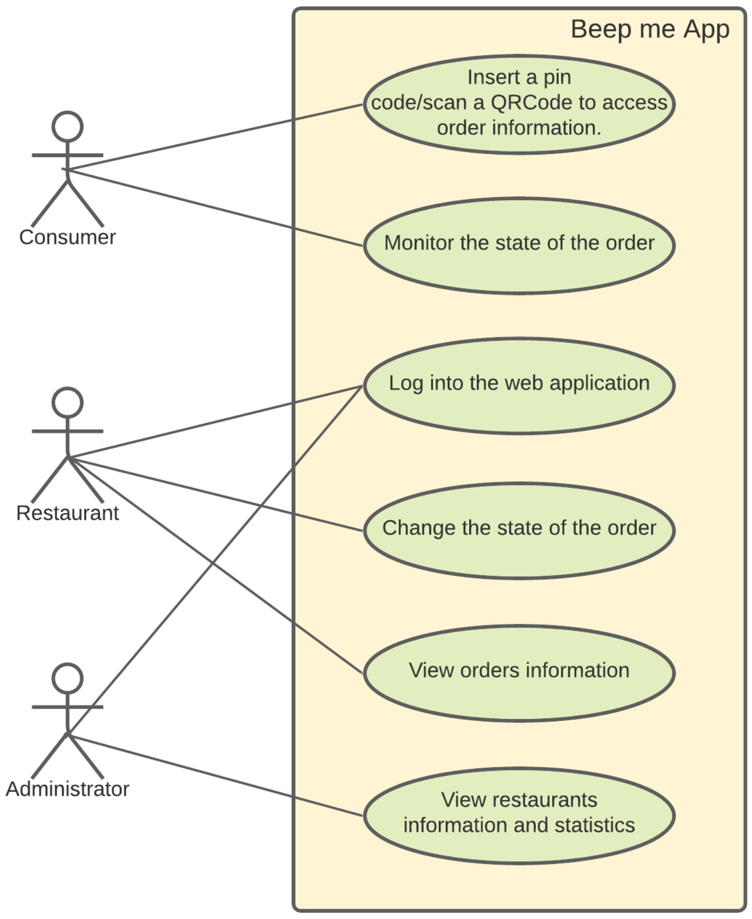
2. Product concept

Vision statement

Our product will prevent customers in shopping malls' food courts from waiting for their food near the restaurant that will serve it, therefore reducing the crowds of people. Furthermore, the use of physical mechanisms to alert the consumer that the order is ready for delivery becomes unnecessary, an aspect designed to avoid multiple contact with these devices in times of a pandemic. The product will also ease up the communication between restaurant's sections (notably the counter and the kitchen) and client-restaurant. In addition, the managers of the shopping centers will be able to access different types of data and study the statistics of the food court.

The application will have three main parts:

- **Consumer side**: where the client can scan a QRCode or insert a pin code present on their receipt and the phone will receive a notification when their order is ready.
- **Restaurant side**: where orders are registered automatically and it is possible to manage the state of the order throughout its execution (this is useful to alert the client that their order is ready or even delayed).
- **Administration side**: where analysts will receive data about which restaurants work better, the average time of delivery per restaurant and affluence hours. This will help food court administrators improve the food court functioning.


Beep me App
Consumer
Restaurant
Administrator
Insert a pin code/scan a QRCode to access order information.
Monitor the state of the order
Log into the web application
Change the state of the order
View orders information
View restaurants information and statistics

Our product is similar to some restaurants' pager systems, however it is more hygienic, has a bigger range (no need to be close to the restaurant), makes unnecessary spending money on equipment and it has the added bonus of being able to collect data.

For the development of our product, we will take inspiration from past experiences with these pagers, but also interview some of our peers to get their opinion on the matter.

One of the main focuses that the product has is to help people in this time of the Covid-19 pandemic, as mentioned above. This is because the methods currently used are not concerned with people's safety in the face of this virus.

Personas

Consumers

Maria is an environmental engineering student at the University of Aveiro. She is 21 years old and as she is in her final year of graduation, she is very focused on her studies, spending half of her free time studying. As a result, she has very little time to shop for groceries and eat. Therefore Maria eats out a few times a week, wasting a lot of time while waiting for her food (time she could spend buying groceries for the week or studying at a table).

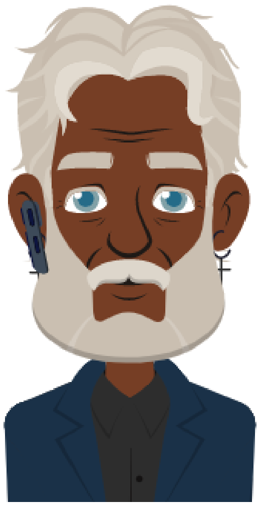
José is a minimarket manager and he’s 54 years old. He recently underwent surgery on his left foot for an injury while playing football in his senior league. Even though he is on a good path to recovery, it is still very painful for José to stand up for long periods of time. Therefore, José tries to sit down whenever possible and avoids standing in lines.

Restaurants

Luciana is a 26-year-old woman who works at the ‘A Gula do Prego’ restaurant in the commercial center of Aveiro. She is in charge of registering and delivering orders over the counter. Whenever an order is ready, Luciana is forced to call the order number several times, making it almost voiceless at the end of the day.

Alfredo is a 36-year-old apprentice cook. After years working as a mechanic, Alfredo decided to change his profession and got a job at the restaurant ‘Alicarius’ in Aveiro. He loves his job, but sometimes he makes a mistake in a request or the food burns. In addition to harming the restaurant, it delays orders, which catches customers off guard. Whenever possible, he goes to the counter to let the customer know that the order is late, but it would be much more efficient for him if he could let them know that without leaving the kitchen.

Administration



Carlos is one of the managers of the Aveiro shopping center and works in the department of events organization. He is 42 years old and has worked in the same department for 10 years, having achieved his position as head of department by being perfectionist, careful and meticulous in every event he prepares.

Sofia works in the profit management department at the Aveiro shopping center. She is 31 years old and has worked in this job since she was 27 years old. At the end of every month, Sofia is responsible for making an analysis of the restaurants in the shopping, in order to find out if there are significant changes in the profit that each of them obtained and if it's worth continuing to invest in all of them.

Main scenarios

Scenario 1 - prototype 1 consumer

Maria has a very important exam on Wednesday. After spending the entire Monday studying, she decides to head to the mall for a bite to eat. Maria needs something ready quickly, so she heads to a fast food restaurant with an average waiting time of 15 minutes (which she can see in the app). After placing her order, she opens the app 'Beep Me', scans the QR code on the receipt she received and goes away to shop for groceries at the minimarket next door. After about 13 minutes, Maria receives a notification from the app, informing her that her order is ready to be picked up.

Scenario 2 - prototype 2 consumer

José decides to go to the Aveiro shopping center for lunch. He is torn between the 'Senõr Pizza' or the 'Maravilhas no Prato' restaurants. Both have a waiting time of 25 minutes, so José sees in the app the one who is least busy this day. The elect is the first. After ordering his favorite pizza, José scans the QR code on the receipt in the app and looks for a table where he can sit. He takes advantage of the waiting time to rest his leg and respond to some emails. When his pizza is ready, the app notifies him so he can pick it up.

Scenario 3 - prototype 1 restaurant

Luciana is working at the counter. At the beginning of her shift, she logs onto the app. Later, she receives an order and registers it in the app. Luciana prints the receipt, containing a QR code,unique to the order. Once the customer scans this code, the order appears on her screen as 'Preparing'. When the kitchen delivers the order, Luciana moves the order from 'Preparing' to 'Done'. This sends a notification to the customer. When the customer comes to pick up the order, Luciana passes the order to 'Delivered', thus eliminating the order record from her screen.

Scenario 4 - prototype 2 restaurant

Alfredo is working the night shift at 'Bifes e Bifanas'. At lunchtime, the restaurant is busier and he lets the steak he was cooking burn. Alfredo enters the app, clicks on the corresponding order and marks the order as 'Delayed', which will notify both the customer and the workers at the counter.

Scenario 5 - prototype 2 administration

Carlos was in charge of organizing an event to celebrate Christmas. Since the other floors of the Aveiro shopping center are already being used, there is no other alternative but to use the food court's floor. The event lasts for 3 hours, with 3 intervals of 10 minutes. Carlos needs to know what is the best day to do the event, taking into account the hours and days of the week when the food court is busiest, thus trying to avoid large crowds in the place. To find out this information, Carlos uses the 'Beep Me' app and looks for statistics regarding the hours of affluence.

Scenario 6 - prototype 1 administration

The end of November has arrived and Sofia needs to do an analysis on which restaurants have made the most profits this month, since every December the Aveiro shopping center offers an amount in value to the restaurant that sold the most during the current year. In addition, as we are nearing the end of the year, Sofia needs to know if there are any restaurants whose profits are well below average and which it is not beneficial to keep in the shopping center. To do this, Sofia opens the 'Beep Me' app and searches for the information she wants in the general statistics tab.

3. Architecture notebook

Key requirements and constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The fact that restaurants already have their own management system is a problem that needs to be resolved and taken into account in the development of our platform.
- In extreme situations, our backend will have to deal with a very large number of requests, since, in a large shopping center, the number of restaurants and customers using our platform will also be large, which means that the backend will have to respond to a large amount of response requests.
- On the client side, it is good for the client to use a device that has a camera built in to be able to read the QRCode that is made available to access the order status in the client app. It is not mandatory, because a pin is also available to cases where the client doesn't have access to a camera.
- We need to have 2 different “sides” to the web app, meaning that we need to have a side to the restaurant and a side to the administration. We also need a mobile app for the consumer side. These different “sides” will be well apart so that, for example, the restaurant doesn't have access to the administration side, and the consumer does not have access to either the restaurant and administration sides. The separation administration-restaurant is mainly made by a login page.
- The devices that are using our platform need to have access to the internet.
- The DBMS needs to be able to handle large amounts of data, and needs to handle requests to that data quickly.

Architectural view

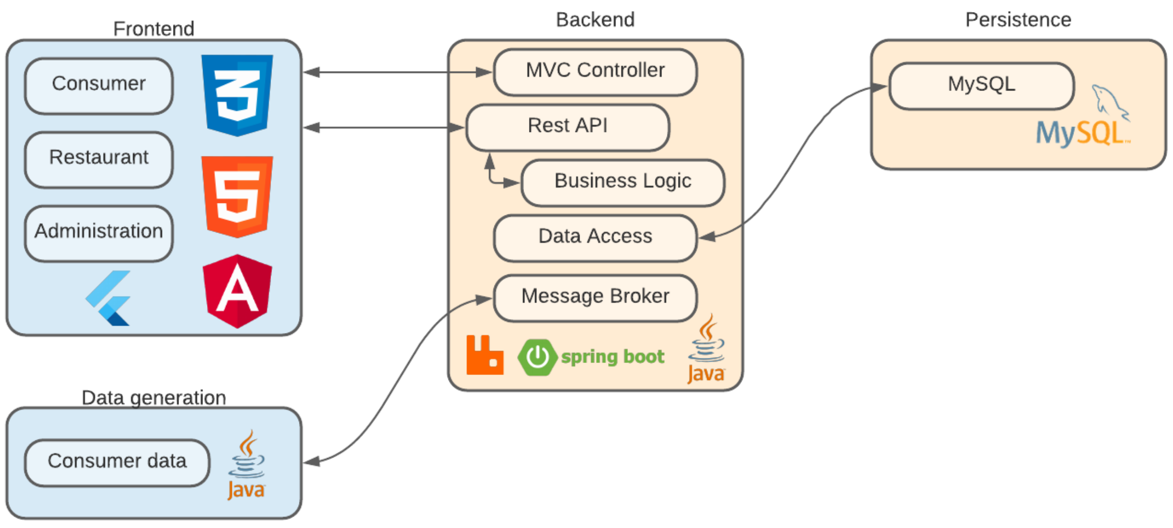
In terms of architecture, we opted for a multilayer architecture.

This architecture has a central backend that will do all of the heavy work. The backend will be built on top of Java using the Spring Boot framework to manage data access, business logic and also the http request and replies.

The frontend, that will display the information to the end user will have three segments: the consumer side, the restaurant side and the administration side. The restaurant and administration sides will be developed using Angular, HTML and CSS and they can be accessed via the web; on the other hand, the consumer side is focused on mobile access, being then developed in Flutter.

In terms of data storage, we will use Spring Data to work as an interface between the backend and the database. For the DBMS we will use MySQL because we think it is easier to integrate with Spring Boot and it is the best option for handling relational data.

In terms of message broker, we chose to use RabbitMQ since it’s easy to implement and works exactly as we’d like it to.



Module interactions

The interaction between the Restaurant/Administration module and the system starts with accessing the web application through the login mechanism. A connection is made to the Backend, which asks for the verification of the credentials in the database, with the credentials being validated as valid or invalid. In this credential check, there is also a check on which type of user is accessing the application: restaurant or admin; the appropriate screen for the given user is displayed.

In the case of the restaurant, once the login is successfully completed, a screen appears where the user can choose between the kitchen and the counter. In the "kitchen" section, orders that are in "in preparation" mode are displayed and can be delayed if necessary. In the "counter" section, the user has access to three different containers, each one representing a state in which an order can be: in preparation, ready for delivery or delivered. The user can move orders between containers as the state of the order changes. It is also possible for the user to cancel orders that are in the “in preparation” phase (if a client changes their mind or something similar).

In the case of the administrator, after logging in with the correct credentials, a screen appears with two separate graphs: the first contains the contribution, in total orders, of each restaurant by stacking each restaurant with their contribution in percentage, making the restaurants with more orders appear larger and the restaurants with lower amounts of orders smaller. The second chart, shows for all restaurants the current delivered and orders that have not been delivered yet, whilst being able to filter by either one of them.

By pressing the “restaurants” button in the navigation bar, we are sent to a page that shows all restaurants that are being managed. When we select any one of them, the page shows two charts, one on each side. On the left, we have a chart that shows the current state of that restaurant's orders, as seen in the restaurant section above. On the right, we have a chart that has the orders for that restaurant grouped by day and hour.

The interaction between the consumer module and the system starts with accessing the mobile application, where they can continue with the insertion of a pin code or scanning a QRCode. When the user opens the app he will find the home page with all the orders that he has scanned (if any), or a page with no orders. In order to scan a QRcode the user has to click the plus icon in the bottom right of the screen. When the user clicks on the icon, he will go to a new page where the camera is open and waiting to find a QRcode. When a QRcode is read, the app will fetch, through the API provided by the server the order related to that code, and the server will return with all the relevant information about that order. With this information the app will take the user to the homepage where a new order, if the fetching returned an existing order, is present. To "refresh" the state of the orders, the app fetches timely from the server all the information related to that order using the order code. When an order is ready, a popup shows up on the screen letting the user know that that order is ready for pickup. This works because the app is listening on a queue named "notifications" that is present on our message broker that when an order's state changes to ready, the server publishes a message with that order's information.

In terms of data generation, our goal is to simulate the behavior of the real orders that are created when the client makes an order at the counter. To do so, we developed a java program that creates an order based on the time of day and the restaurant in case. If the time of day is less agitated, an order will be less likely to be created. We assigned a probability to each restaurant to make sure that bigger and more popular restaurants would have more orders than the others. The orders are published using a message broker. This works by publishing an order in the message broker each time one is created. On the server side, it will be listening to new orders being added in a specific queue. When an order is added, the server will "pick it up" and save it in the database. To make all of this happen, we had to integrate the data generated program in a Spring Boot application to be able to publish each order created to the message broker.

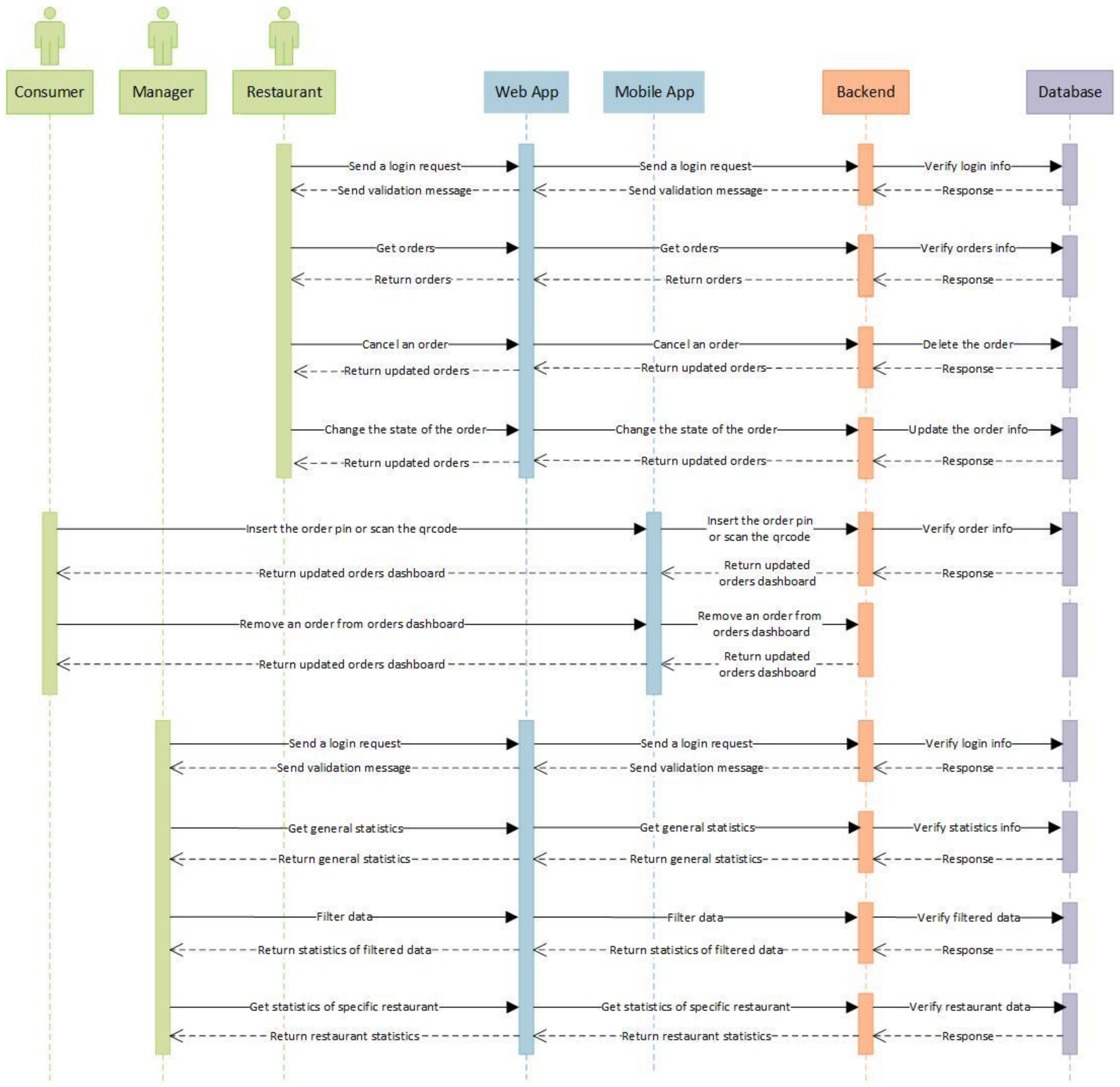
All of the modules present in our system are deployed in our virtual machine. In there are the 5 modules of our system, which one with his own docker container.

The main container is the container of the main Server. Is through this container that almost all the modules communicate. To create this container we used the maven/spring command **./mvnw spring-boot:build-image** that creates an image of that specific maven project.

Then we have 2 containers that are created in the same docker compose file, that are the MySQL container, that contains the database management system, and the RabbitMQ message broker.

The Frontend is also in a container, in this case in a container that is running nginx. The image is built using a Dockerfile located in the angular project directory. In that Dockerfile we programmed it to at first create an npm container, that generates all the modules necessary for the angular app and then builds the project generating the "artifacts". Then these files that were created are copied over to the nginx container, the environment where the server is run.

The Data Generation container is built the same way as the main Serve, we use the maven/spring command **./mvnw spring-boot:build-image** to create the image.

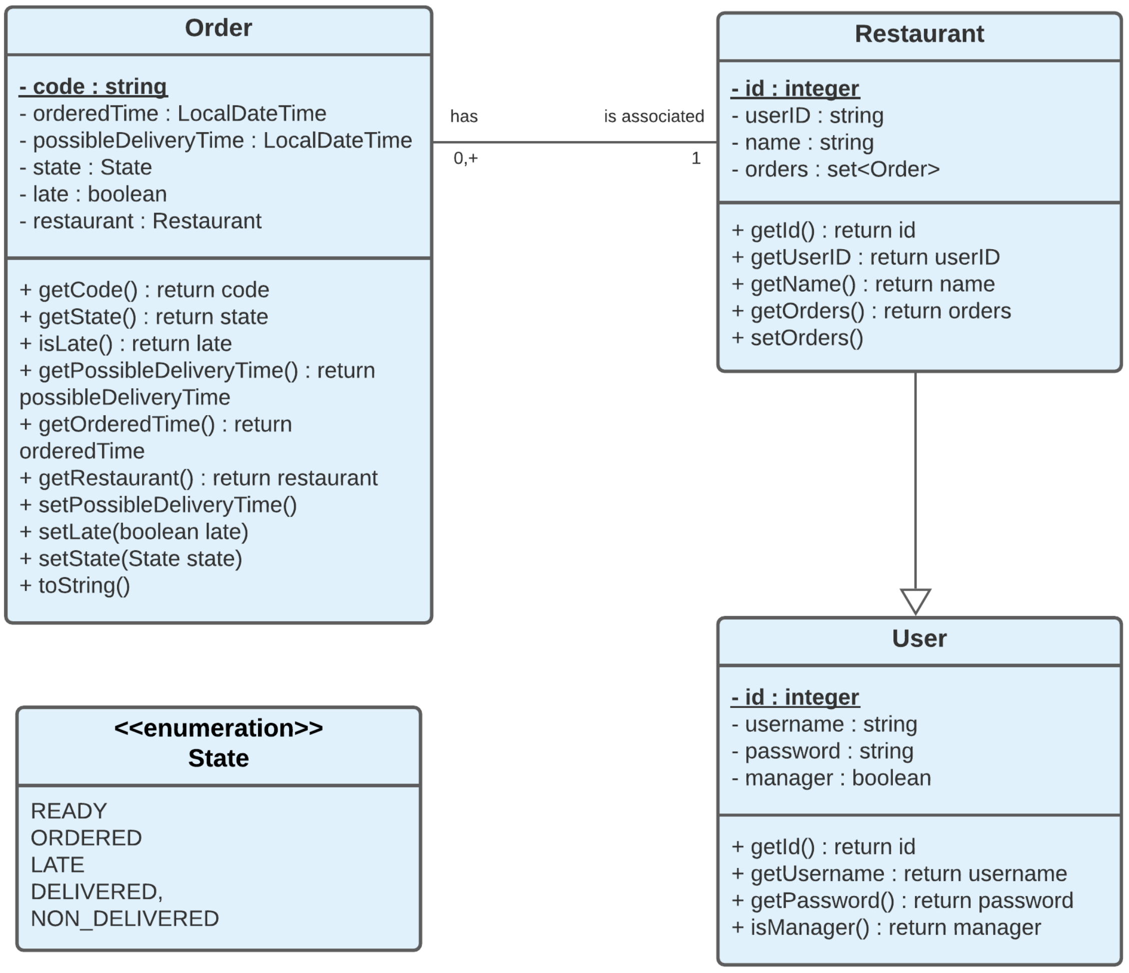


Consumer
Manager
Restaurant
Web App
Mobile App
Backend
Database
Send a login request
Send validation message
Verify login info
Response
Get orders
Return orders
Verify orders info
Response
Cancel an order
Return updated orders
Delete the order
Response
Change the state of the order
Return updated orders
Update the order info
Response
Insert the order pin or scan the qrcode
Insert the order pin or scan the qrcode
Verify order info
Return updated orders dashboard
Return updated orders dashboard
Response
Remove an order from orders dashboard
Remove an order from orders dashboard
Return updated orders dashboard
Return updated orders dashboard
Send a login request
Send validation message
Verify login info
Response
Get general statistics
Return general statistics
Verify statistics info
Response
Filter data
Return statistics of filtered data
Verify filtered data
Response
Get statistics of specific restaurant
Return restaurant statistics
Verify restaurant data
Response

4. Information perspective

There are two types of users that we want to store in our database: restaurant type and manager type. The difference is made through the boolean attribute “manager” in the class "User". If the value of this attribute is FALSE, a new instance will be created in the "Restaurant" class; this instance may have zero or more orders associated. However, an order can only be associated with one and only one restaurant.

An order has an attribute that must change over time: the state. This attribute can take on multiple values: READY, ORDERED, LATE, DELIVERED, NON_DELIVERED.


Order
- code : string
- orderedTime : LocalDateTime
- possibleDeliveryTime : LocalDateTime
- state : State
- late : boolean
- restaurant : Restaurant
+ getCode() : return code
+ getState() : return state
+ isLate() : return late
+ getPossibleDeliveryTime() : return possibleDeliveryTime
+ getOrderedTime() : return orderedTime
+ getRestaurant() : return restaurant
+ setPossibleDeliveryTime()
+ setLate(boolean late)
+ setState(State state)
+ toString()
has
0,+
is associated
1
Restaurant
- id : integer
- userID : string
- name : string
- orders : set<Order>
+ getId() : return id
+ getUserID : return userID
+ getName() : return name
+ getOrders() : return orders
+ setOrders()
<<enumeration>>
State
READY
ORDERED
LATE
DELIVERED,
NON_DELIVERED
User
- id : integer
- username : string
- password : string
- manager : boolean
+ getId() : return id
+ getUsername : return username
+ getPassword() : return password
+ isManager() : return manager

5. References and resources

Since a lot of us had never used Angular Framework before, we followed an Angular Tutorial for Beginners offered by Google in order to understand the basics of Angular. We also read Angular Documentation to understand how to communicate with the backend via http.

For the development of the diagrams that accompany this report, it was necessary to research the material provided to students previously at U.C. “Modelação e Análise de Sistemas”.