TQS: Product specification report

João Dourado [108636], José Mendes [107188], Miguel Figueiredo [108287], Vítor Santos [107186]
v2023-04-18

1 Introduction

1.1 Overview of the project

Spring Canteen is a chain of restaurants designed to revolutionize the perception of fast food, by delivering high quality healthy meals in record speed. Customers in a hurry can quickly get a chicken salad with low carbs and not feel guilty after a great meal.
Spring Canteen is an integrated IT solution that provides a restaurant with the ability to efficiently handle different kinds of tasks regarding the management of orders between the staff members and the clients. Spring Canteen will meet these requirements through the implementation of various platforms:

- **Spring-Kiosk** - A customer portal in which the clients can submit their orders. This platform should provide the customer with menu options and handle the payment processing. It should also give the customer the ability to choose between payment methods for the order, namely selecting between paying at the kiosk or at the counter. At the end of the menu selection process, it should be presented to the user the option to upgrade the order priority status, for an additional fee.
- **Spring-Desk** – A platform responsible for providing the staff members with services to handle administrative tasks including employee authentication and payments at the counter. Additionally, there should be presented to the employees' different service queues, a normal and a priority queue where priority orders should take precedence regarding meal preparation.

The staff members will also be able to manage the different waiting lines according to the order's readiness and delivery confirmation.

- **Spring-Boards** - A digital signage solution that concisely communicates to the client that his/her order is being prepared and promptly inform him/her of when it is ready for pickup at the counter. The different orders will appear by the sequence of the order placement, within the different priority queues.

This project aims to implement an MVP compliant with the prior usage scenarios, while applying software enterprise architecture patterns. The development of the solution will follow collaborative agile practices and a Software Quality Assurance (SQA) strategy. Software engineering practices like continuous testing, continuous integration and continuous delivery will be embraced with the aim of reducing manual intervention and saving precious time, promoting collaboration and trust among the software development team, and streamlining the development lifecycle to rapidly deliver high-quality software releases.

1.2 Limitations

We handled the different queues for incoming orders by using Java ArrayBlockingQueues. This queue uses a static array under the hood, which means size is fixed. When the first queue (the one which stores idle orders) is full and brand-new orders are created, these orders are left **dangling,** as they are not added to the in-memory queue. We currently have no mechanism to deal with dangling orders, and all we can do to prevent it from happening is setting the array capacity to a large value. In practice, unless employees are slacking and letting orders pile up, the idle orders queue will never fill up and dangling orders won't exist.

Spring Canteen also has an unimplemented feature, related to the kiosks themselves. During domain modelling, we identified the importance of assigning unique identifiers to each kiosk terminal, enabling us to detect when a specific kiosk stops functioning. While this idea holds potential for future development, it is not essential for delivering a valuable MVP. Automating the detection of kiosk terminal failures is beneficial, but it is not critical for the initial implementation.

2 Product concept and requirements

2.1 Vision statement

Traditional fast-food restaurants tend to have a bottleneck regarding the speed and the number of customers which they can serve. The origin of this bottleneck lies in the cashier counter where one must wait in line, ponder the order while looking at those awful advertising posters with little to no information and place the order, before even having the meal start being cooked.

The number of requests the restaurant can attend per hour is limited to the sum of the number of orders each member of the staff working on the payment counter can attend. This person may be efficient, then saving a lot of time and maximizing the number of orders but the opposite can also happen which would cause a loss on revenue not only because there are customers waiting but also because more customers will not come foreseeing the hassle of having to wait on such a busy restaurant. However, in both prior cases the tally of placed orders would be much inferior to what our system allows for.

By placing multiple kiosks with an intuitive user interface where a customer can select and pay his/her order we automate most of the placement order process and eliminate the existence of a middleman while maximizing revenue and commanding the staff efforts to focus on the primary task: preparing orders.

Following this approach the restaurant manager can always add more machines if he thinks the order output is still not enough which would always be cheaper than having to pay for a new payment cashier and the corresponding employee that would be using it.

It's important to highlight that our system isn't just constituted by these kiosks but by the whole infrastructure that allows the order to travel from the kiosks to an internal order management system which can then be viewed by the customer through a digital signage platform and managed by the restaurants employees that are required to confirm the fulfilled status of the order.

The following image showcases all customer use cases. The user can essentially make their order by choosing between a list of pre-existing menus. At the end of the order placement process, the user may or may not chose an option for their order to be part of a priority queue. The preparation of orders of the priority queue takes precedence over those on the normal queue. Then the user can opt to pay directly on the kiosk or on the payment desk. In this case the payment will be handled by the respective employee.

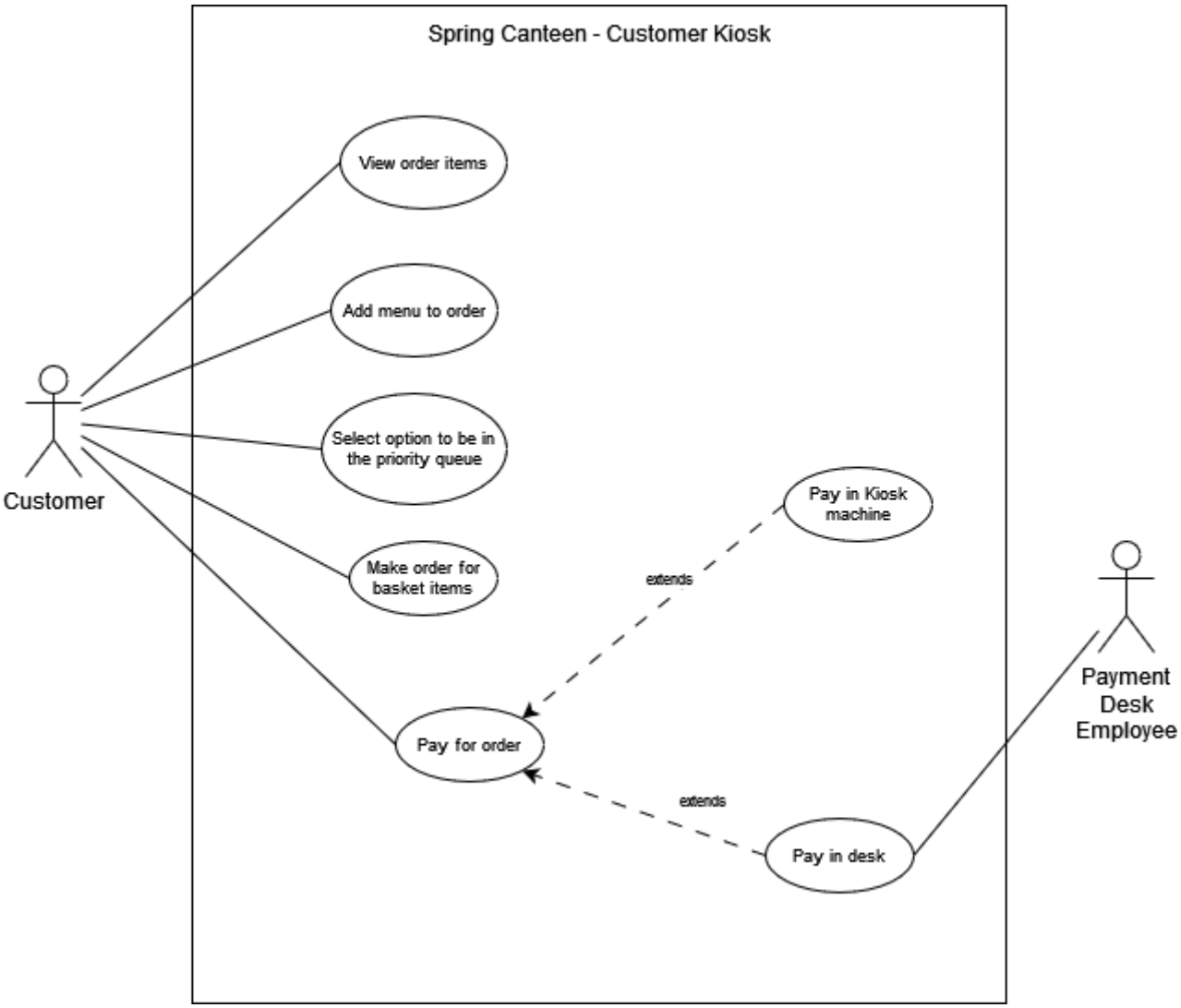


Figure 1: Customer Kiosk Use Case Diagram

The following image showcases all use cases related to the payment of orders at the counter despite being made on the kiosks. For this to happen the payment desk employee must be able to view which orders have not yet been paid. Then, the employee can collect the payment of the customer and confirm the order so that it can appear to the cooks (which will then start preparing the order).

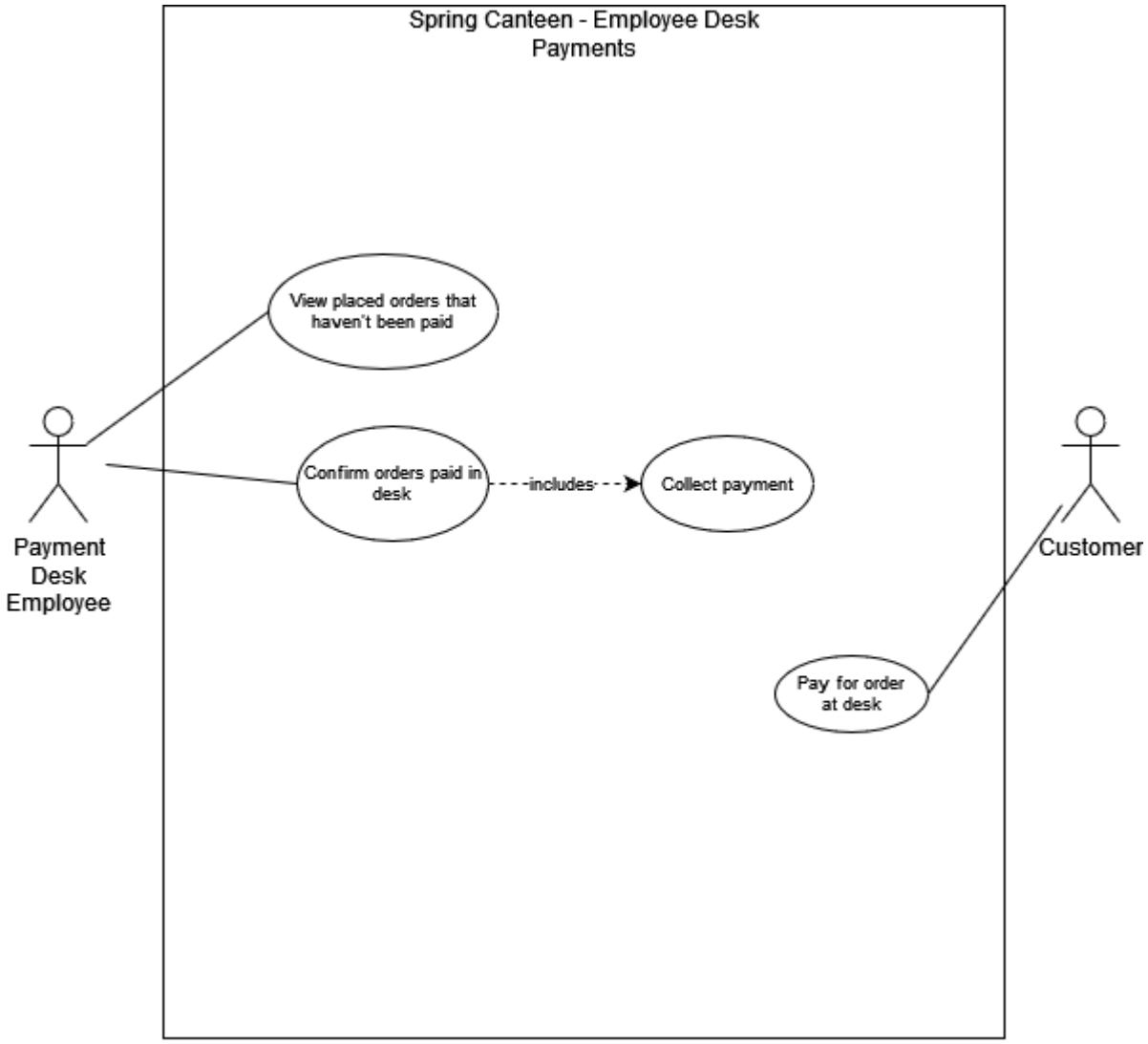


Figure 2: Employee Desk Payments Use Case Diagram

The following image showcases all use cases related to the preparation of the orders in the kitchen. The cook can change the status of an order when they start or finish preparing it. To do this he must be able to see all processed orders with the corresponding menus, ingredients, and priority status. In the last case another employee of Spring Canteen should pick it up to deliver it to the customer.

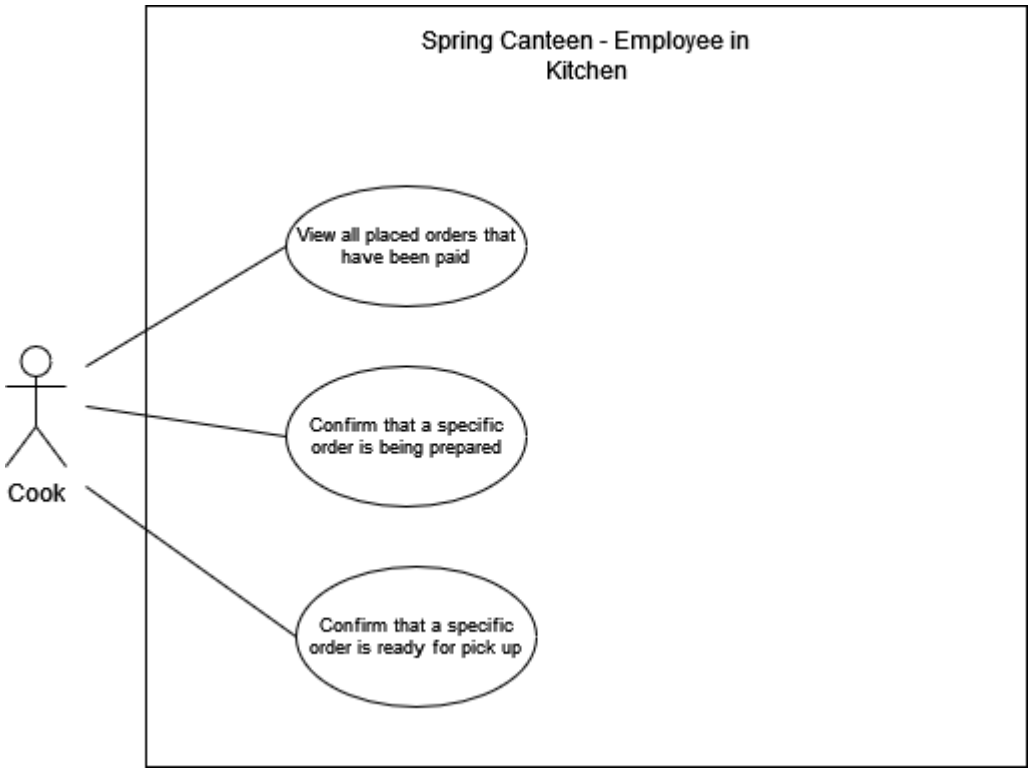


Figura 3: Cook Use Case Diagram

The following image showcases all use cases related to the pickup and delivery of orders at the counter. A different employee will be responsible for checking whether orders are being prepared or if they are already ready. In the last case, the employee working on the desk will see on their screen the respective order moving from “preparing” column to “ready” column. Then the employee should pick up the order, announce its number and deliver it to the customer. After this, the pickup desk employee should mark that order as delivered on the system so that it disappears from their page as well as the customer digital signage page.

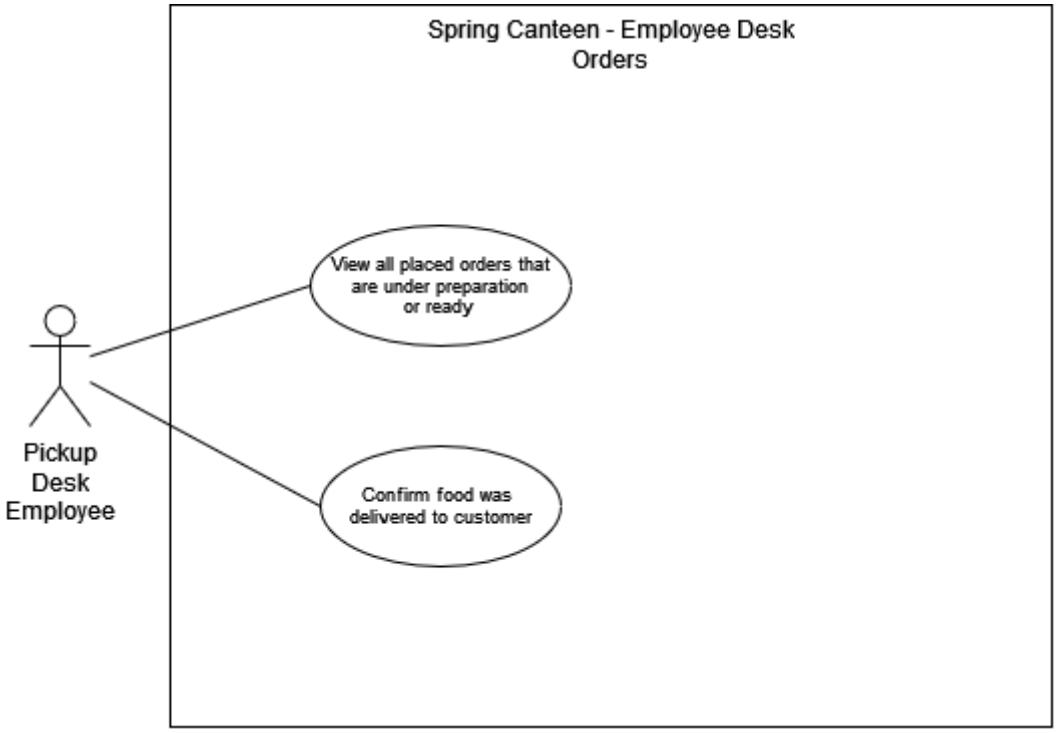


Figure 4: Employee Desk Orders Use Case Diagram

2.2 Personas and Scenarios

As customers of fast-food restaurant chains with and without kiosk and take-away services we have insight into the general workings of the business area, including customer and employee profiles along the most common tasks conducted by each.

2.2.1 Personas

1. Customer

Jorge is a twenty-five-year-old emergency room (ER) nurse, obtaining his degree 4 years ago. He typically works long night-shifts and has little time for his personal life, which often limits his diet to unhealthy fast-food. Jorge is not very tech savvy, but he knows an efficient system when he sees one, being highly observing of the number of steps and time he takes just to make his habitual order.

His preferred fast-food chain's kiosk platform suffers from a notable efficiency problem with too many redundant steps, slowing him down when time is critical. Another fast-food restaurant he often visits when the other is full to the brim doesn't have a *kiosk* platform at all. Long waiting lines and missed orders are no stranger to him there.

Combined with his professional life, this has taken a significant toll on his overall health. Jorge feels stuck in this unhealthy loop, trying but failing to find healthier replacements with take-away and fast delivery solutions, which often suffer from delays and lacking digital platforms.

2. Restaurant Staff

Carla is a thirty-one-year-old fast-food restaurant employee with five years of experience and some tech knowledge. Carla handles hundreds of orders per day, as her restaurant is one of the most active in town. She wants to handle orders as fast as she can to hopefully get that one promotion, but the very way the restaurant manages orders is flawed: all orders are received and handled physically at the counter. At peak hours, many potential customers would turn back because of long, curving waiting lines.

Given this environment, she's under extreme pressure to handle orders as fast as possible, which has only helped towards giving her stress. All Carla wants is to focus on the food orders and deliver exceptional results for that promotion she's been yearning for the past few years.

2.2.2 Scenarios

1. Customer Order Placement

a. Without Kiosk at some restaurant

It's 12:20 and Jorge is on his 30-minute lunch break. With no chance to place an order by telephone in the busy morning and his typical restaurant having overflowing attendance lines, he resorts to one of his favourite fast-food restaurants, which is limited to counter attendance and drive-through. After waiting 13 minutes in line, he finally makes his order.

Jorge then waits for 6 minutes, standing a few meters away from the delivery counter while his order is not ready. The digital counter display shows Jorge's order number, and, with an aching back, he takes his food and goes find a seat. However, he's forced to eat outside by the sidewalk as he can't seem to find a seat to have his meal. 22 minutes have passed and now Jorge has just 8 minutes to eat and get back to the hospital.

b. *With kiosk at Spring Canteen*

It's 13:00 and Jorge is just now going to lunch. Usually, he lunches at an earlier hour but today work extended past that hour. Now Jorge is facing a problem due to at this time of day all restaurants being overflowing with people with huge and messy lines. With no chance to lunch at his usual place he resorts to Spring Canteen to take an order, eat, and leave in no time. After he enters the facility, he goes to one of the available kiosks, chooses the desired menu and pays with his debit card. Following the payment, he gets the receipt with his order number. Now he just needs to wait for his number to appear at the digital signage platform to receive his order. A few minutes later, he gets his order, and he is now ready for lunch.

2. Food preparation pipeline

a. Without Kiosk at some restaurant

Carla, who's been on a morning shift for the past 4 hours, attends to Jorge's order, rushing to note it down on the monitor in front of her. Navigating the numerous food and drink categories on screen, Carla selects every corresponding consumable item. After the customer pays for his order, Carla forwards it to the kitchen staff and issues the receipt for the customer containing an identification number and his order items. Now Carla needs to attend to the next customer in line. She rushes through it because the line is picking up. However, in the meantime, Carla is called for delivery duty, taking finished orders from the kitchen to the counter. The line continues to get bigger and bigger but there's nothing Carla can do.

b. *With kiosk at Spring Canteen*

Jorge uses the kiosk to place his order and to make the related payment. After this, the order is automatically processed and already appears in the cook's screen with all menus and ingredients necessary. When the time arrives for the order to be prepared the cook starts preparing it. As soon as the order is ready to be delivered to the customer the cook notifies Carla (who is now working at Spring Canteen and only has the responsibility to handle food pickup/delivery at the counter) that immediately picks up the finished order and delivers it to Jorge. Then, she marks the order as delivered making it disappear from the customer digital signage platform. Now that she only has one responsibility to focus on she is now stress free.

3. Customer Priority Order Placement

a. Scenario A

Today is a very busy day for Jorge. Unlike other days he only has 10 minutes to lunch. Knowing this, he chooses not to go to a restaurant because he knows he won't be able to place his order, find a seat and lunch in between that time. That's why he opted to just eat an hamburger and a bag of chips from the vending machines across the street.

b. *Scenario B - With kiosk at Spring Canteen*

Today is a very busy day for Jorge. Unlike other days he only has 10 minutes to lunch. Knowing this he chooses to go to Spring Canteen. He goes to one of the kiosks, chooses its menu and before paying selects the option to pay an extra fee to have priority in order preparation. Then the order is processed and is shown to the cooks as a priority order. Because of this, after finishing the current order, the cooks start preparing Jorge's order. When the order is finished Carla picks it up and delivers it to Jorge that can now eat a proper meal before going back to work.

2.3 Project epics and priorities

In the implementation of our solution, we will use an Agile Methodology with weekly sprints (1 week each), making a total of 4 sprints. The priorities are set by the order of the listed epics, top being the highest priority and bottom being the lowest priority.

1. Order in Kiosk

User Story 1 - Select Items:
As a Costumer
I want to select items from the kiosk
So that I can customize and tailor my order to my preferences

User Story 2 - Pay Directly on the Kiosk:
As a Costumer
I want to pay directly on the kiosk
So that I can complete my transaction conveniently

User Story 3 - Pay on the Desk:
As a Costumer
I want to choose to pay on the desk
So that I can pay with money

User Story 4 - Cancel or Modify:
As a Costumer
I want to the option to cancel or modify my order before finalizing the payment
So that I can make changes as needed

User Story 5 - Customize Order:
As a Costumer
I want to the option to customize the ingredients in the items of my order
So that I can change the amount of ingredients I get on my order

User Story 6 - Receive Confirmation:
As a Costumer
I want to receive confirmation of my order
So that I can have proof of my purchase

User Story 7 - Place Order:
As a Costumer
I want to place my order with all my selection
So that I can have my food be cooked and receive it for eating

User Story 8 - Place Order in a Priority Queue:
As a Costumer
I want to place my order in a priority queue by paying an additional fee
So that I can receive my order more quickly

2. Kitchen cooks able to see orders and start preparing them

User Story 1 - View Incoming Orders:
As a kitchen cook
I want to view incoming orders in real-time
So that I can start preparing them promptly

User Story 2 - View orders by priority:
As a kitchen cook
I want to prioritize orders based on the queue they are in
So that I can ensure timely preparation and delivery for clients who paid for priority

User Story 3 - Update Order Status:
As a kitchen cook
I want to update the order status (preparing, ready, etc.) for visibility to the rest of the staff
So that I can indicating what I am working on, keeping the kitchen team organized

3. Desk employees receiving payments at the desk

User Story 1 - Accept Payments:
As a desk employee
I want to accept payments from customers directly at the desk
So that I can guarantee that costumers conveniently settle their bills in person

4. Desk employee able to confirm orders were delivered to customer

User Story 1 - Confirm Orders:
As a desk employee
I want to confirm and mark orders as delivered to costumers
So that I can maintain accurate records of what orders are still being worked on

5. Digital signage screens for customers to see the state of their order

User Story 1 - Manage Order Status on Digital Signage:
As a staff member
I want to manage the status of orders on the digital signage
So that I can indicate to the customer the status of their order from preparation to delivery

3 Domain model

The principal entities present in the domain model are the **KioskTerminal** through which clients can make their orders and the employees that prepare and change the status of those orders. There are various types of employees ranging from the cooks (**EmployeeRole.COOK**) to the ones that handle the payments (**EmployeeRole.DESK_PAYMENTS**) and the ones that deliver the order to the customers at the counter.

Throughout its entire preparation the order passes through a series of stages: waiting for payment when payment is requested to be in desk instead of kiosk directly (**OrderStatus.NOT_PAID),** waiting to be prepared just after the payment is processed (**OrderStatus.IDLE**), being prepared by the cooks (**OrderStatus.PREPARING**), being ready for pickup (**OrderStatus.READY**) and delivered at last (**OrderStatus.DELIVERED**).

Each order aggregates the menus chosen by the client. Each of these menus contains one or more main dish options, which in turn has multiple different ingredients, and one or more drink options to pick from. Menus should be completely customizable in terms of the ingredients used for cooking it – a customer might want to add extra pieces of a particular ingredient or remove pieces of another based on their preference.

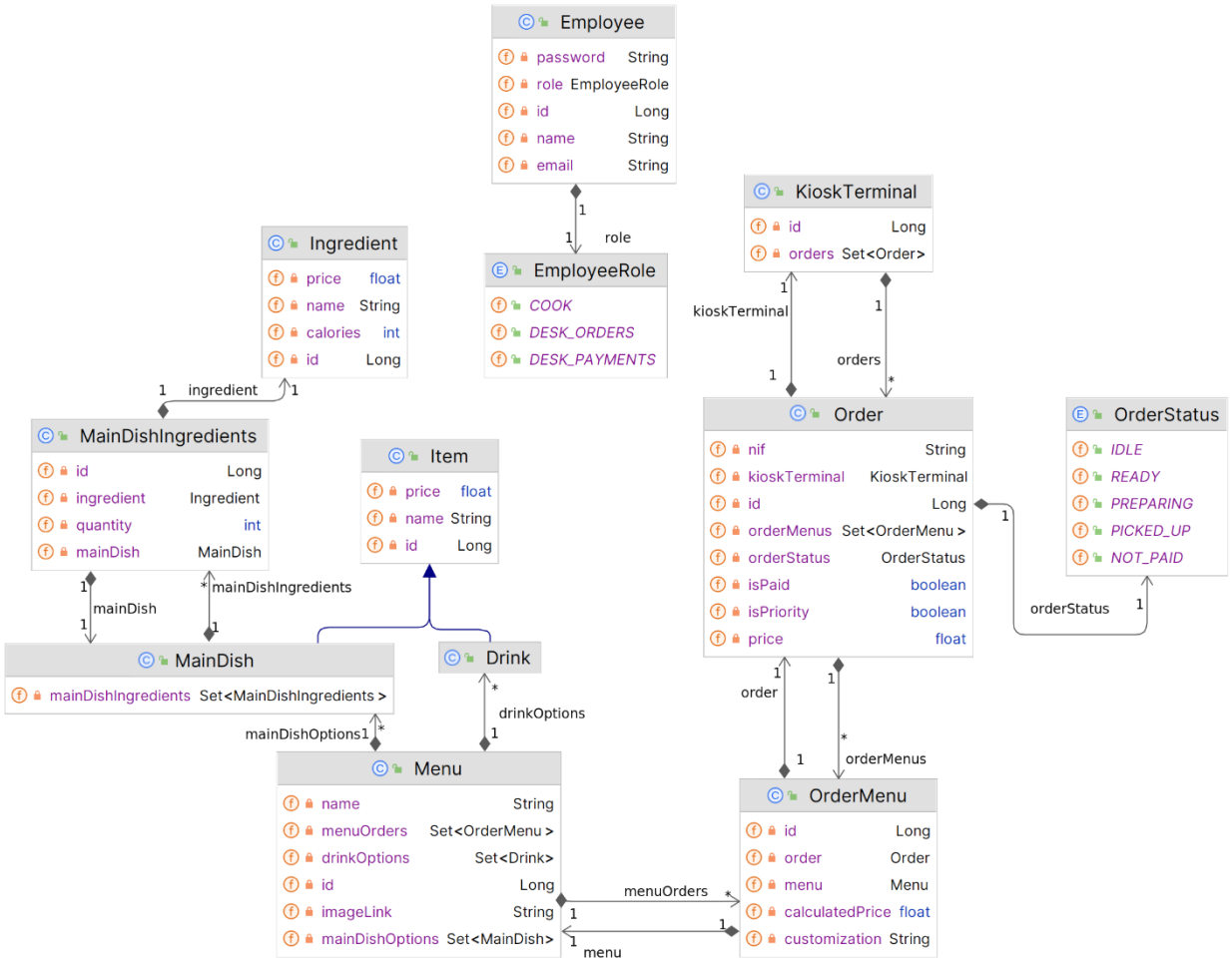


Figure 5: Domain Model

4 Architecture notebook

4.1 Key requirements and constrains

Here are some key requirements and constraints dictating our architectural choices:

- Our system has multiple user-interfacing platforms, and most of them require a stream of real-time information in the form of orders, so that those orders get processed as soon as possible.
- Our system only supports a web interface for placing orders, but we might expand and support a mobile interface and it should not require too much code modification.
- The placement order system should have a design that supports its usage by a user in a kiosk with a big screen.
- Our system should support at least 16 order requests per minute.
- User authentication and authorization must be implemented so that each employee can only access the information they're intended to and can only change the order status according to their role.

4.2 Architecture view

We designed the following system architecture, based on the key requirements and constraints we outlined:

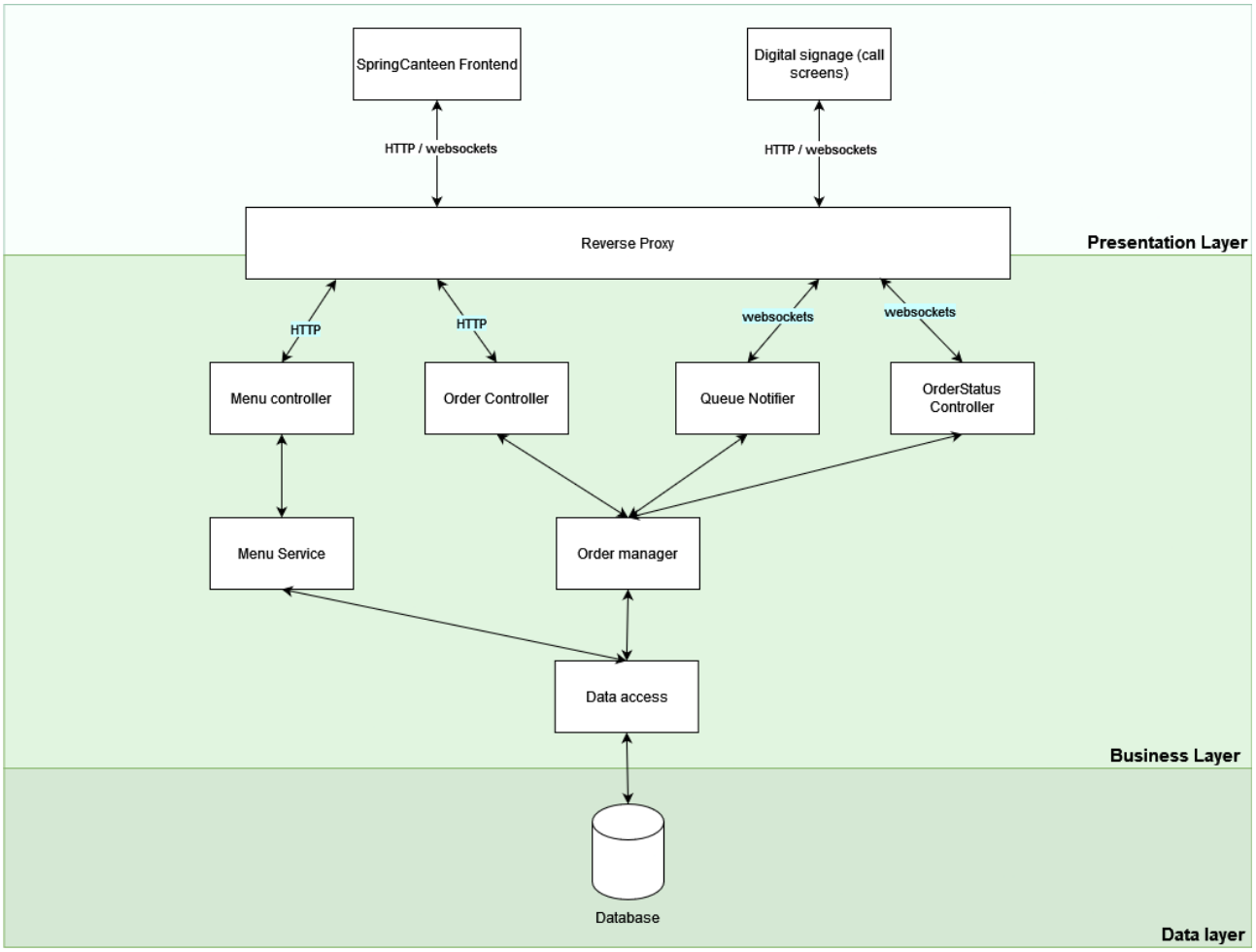


Figure 6: Architecture Diagram

Our system will have multiple user-interfacing platforms:

- **SpringCanteen Frontend –** where users and employees interact with the SpringCanteen services. Users can place orders with all its constraints (priority orders, customizing menus, paying on the desk or kiosk) and employees can fully operate (receive payments, confirm cooking/ready/delivered orders) through this module.

- **Digital Signage (Call Screens)** - where users can see the status of orders, to assess whether their order is ready for pick up or still being cooked. They can see a list of orders that are being cooked, and a list of orders that are ready for pick up, with an identifying number (order number). This interface has very little user interaction (only requires initial authentication for security reasons) and does not send any information to our backend server, it simply receives data through Websockets.

For this reason, we will have a REST API that supports all these interfaces, and if we wanted to add a mobile interface, we would not need to change existing code.
Since we want our clients to receive their food expeditiously, we use WebSockets to provide a real-time stream of data to interfaces that require that data.

This is the expected behavior of the business layer components, starting from boundary components that directly send data to users, and finishing with service-level components that process incoming data:

- **Order Controller**: receives requests from customers that want to place new orders and receives requests from employees handling payments on the desk, such as updates to an order's status (**NOT_PAID** to **IDLE**).
- **Menu Controller:** where users access existing menus and all their information, such as ingredients, main dish options and drink options.
- **Queue Notifier**: this is the component responsible for sending users the current state of orders when they first connect through websockets (subscribe event), and sending users updates of any new orders created or order status updates.
- **OrderStatus Controller:** this component is responsible for receiving order status updates from users, such as **IDLE** to **PREPARING**, **PREPARING** to **READY** or **READY** to **PICKED_UP**. It forwards these status updates to order manager.
- **MenuService**: service-level component responsible for fetching requested menus.
- **Order Manager**: a centralized component to handle all operations related to orders, such as updating their status (triggered by employees through websockets) and receiving new orders.

4.3 Deployment architecture

For deployment, we divided our system into 5 different containers:

- **Database:** our PostgreSQL database.
- **Backend:** our Spring Boot backend service.
- **Kiosk:** our frontend that handles all user interaction, built with React using Typescript, with Vite as a development server and build tool (we build static pages when deploying to production).
- **Digital-signage:** our second frontend, also built with React, Typescript and Vite.
- **Traefik:** our proxy server. Uses Traefik with two different servers, listening in port 80 and 8080 respectively. Port 80 listens to requests going into **Kiosk** and **Digital Signage** user interfaces, and port 8080 hosts a Traefik dashboard for monitoring and setting up Traefik itself.

We used continuous deployment as a practice (we explore this topic in depth in the QA Manual) using **Docker Compose** for repeatable and consistent deployments. We have two different configuration files – one for development, which has hot-module reloading for backend and frontend containers; and one for production, which does not expose any ports apart from the reverse proxy ports for users to access our system, builds frontend static pages and hides security critical information, such as our JWT signing key and our database password.

The image below shows a detailed view of all modules of our system with the technologies used to build each one.

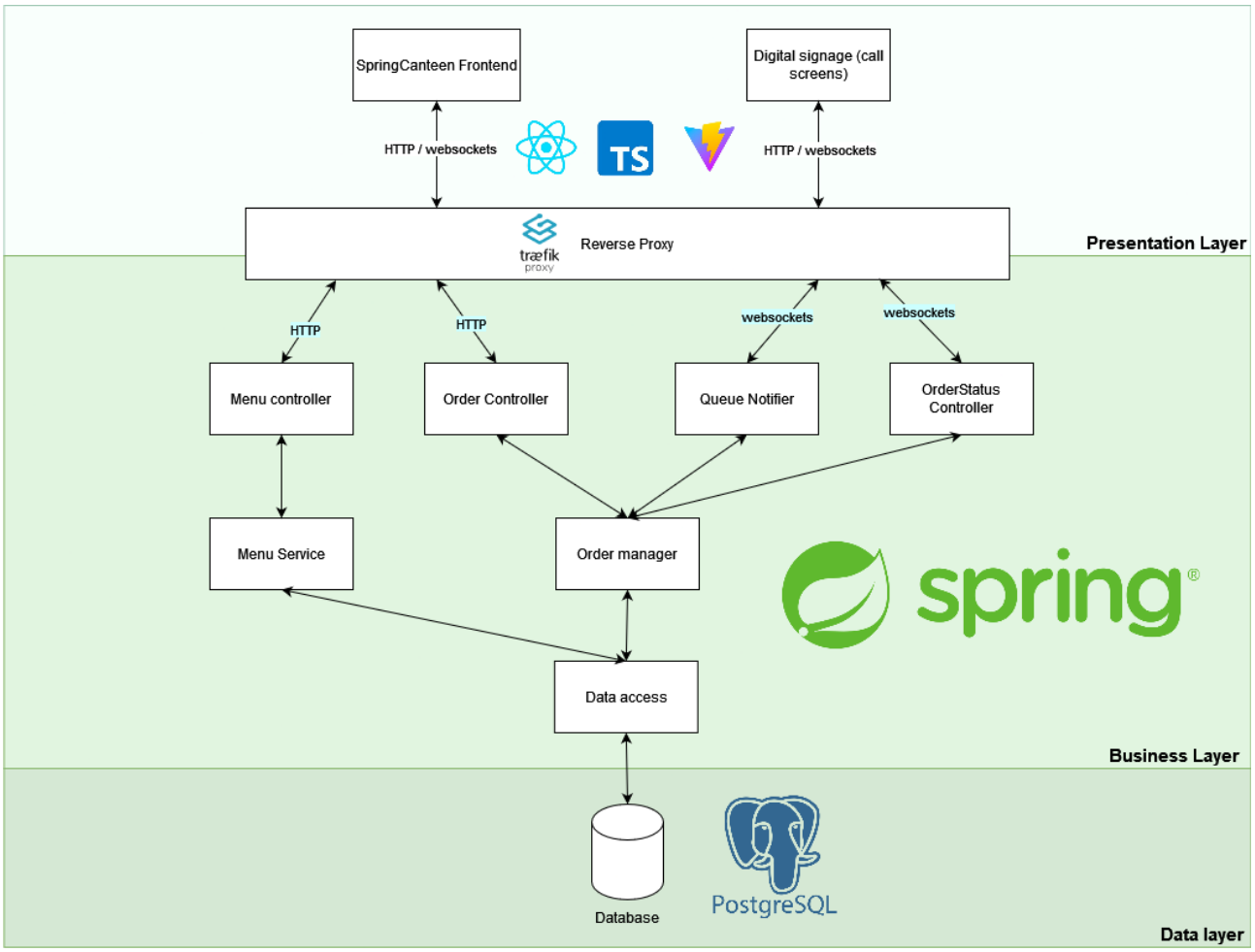
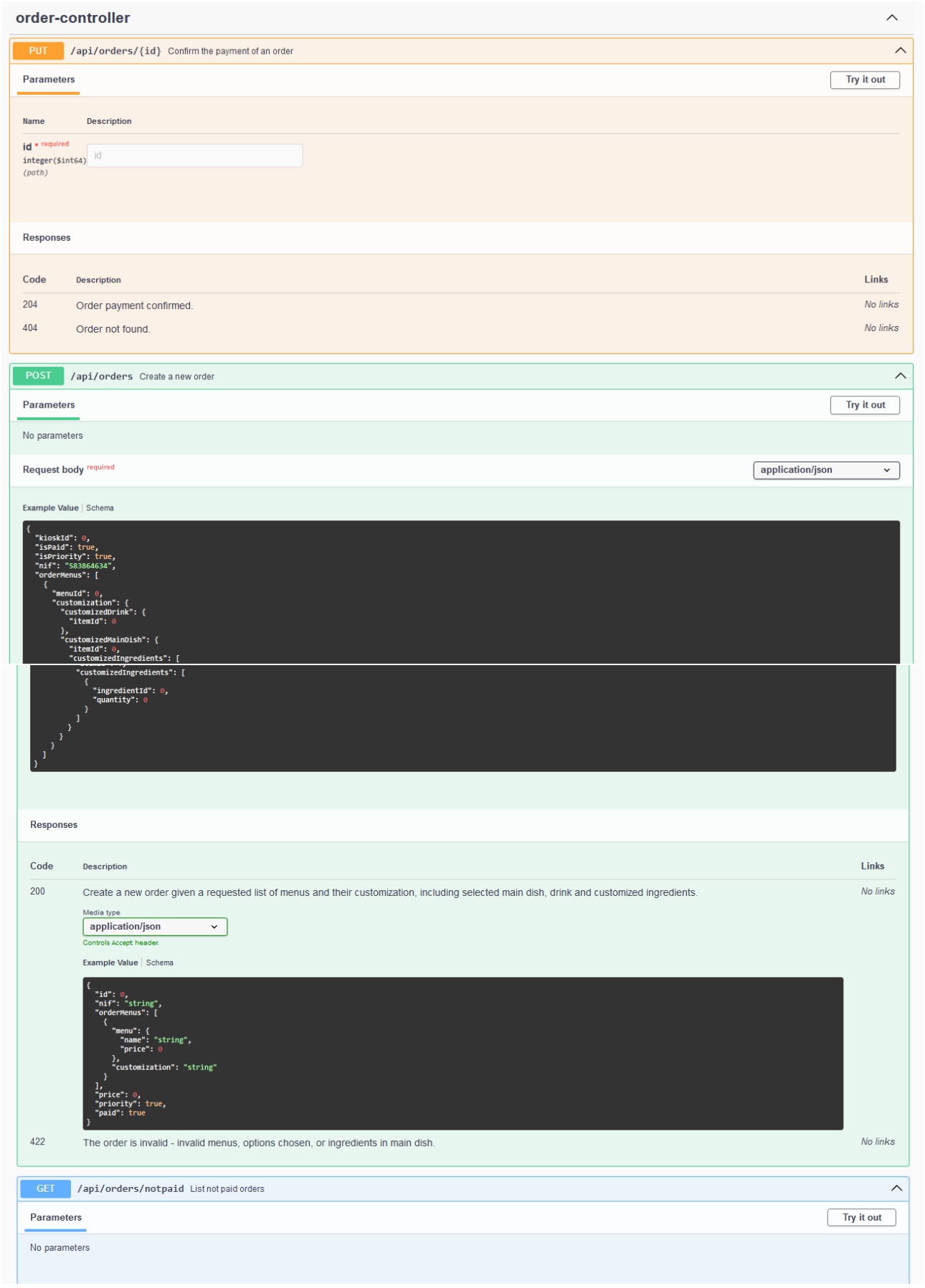


Figure 7: Deployment Architecture Diagram

5 API for developers

The API documentation was hosted with Swagger. Aside from the REST endpoints documented with Swagger, there are also Websocket messaging endpoint:

Endpoint Path	User's behavior	Example Input	Description
/app/order_updates	publisher	{ "order_id": 12 }	Updates the given order to its next status, and updates queues accordingly. Also broadcasts changes to websocket listeners.
/topic/orders	subscriber		Endpoint through which new orders or order updates are broadcasted to all websocket subscribers
/user/topic/orders	subscriber		Endpoint through which the current orders in each queue are sent to the user upon websocket subscription event


order-controller
PUT /api/orders/{id} Confirm the payment of an order
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
204
Order payment confirmed.
No links
404
Order not found.
No links
POST /api/orders Create a new order
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"kioskId": 0,
"isPaid": true,
"isPriority": true,
"nif": "583864634",
"orderMenus": [
{
"menuId": 0,
"customization": {
"customizedDrink": {
"itemId": 0
},
"customizedMainDish": {
"itemId": 0,
"customizedIngredients": [
"customizedIngredients": [
{
"ingredientId": 0,
"quantity": 0
}
]
}
}
}
]
}
Responses
Code
Description
Links
200
Create a new order given a requested list of menus and their customization, including selected main dish, drink and customized ingredients.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"nif": "string",
"orderMenus": [
{
"menu": {
"name": "string",
"price": 0
},
"customization": "string"
}
],
"price": 0,
"priority": true,
"paid": true
}
422
The order is invalid - invalid menus, options chosen, or ingredients in main dish.
No links
GET /api/orders/notpaid List not paid orders
Parameters
Try it out
No parameters

Responses

Code	Description	Links
200	List of all orders that have not been paid yet. Media type: application/json Controls Accept header. Example Value | Schema	*No links*

```
{
  "id": 0,
  "nif": "string",
  "orderMenus": [
    {
      "menu": {
        "name": "string",
        "price": 0
      },
      "customization": "string"
    }
  ],
  "price": 0,
  "priority": true,
  "paid": true
}
```

authentication-controller

POST /api/auth/signup Create an account.

Parameters | Try it out

No parameters

Request body required — application/json

Example Value | Schema

```
{
  "username": "string",
  "email": "string",
  "password": "string",
  "role": "COOK"
}
```

Responses

Code	Description	Links
200	Account created successfully. Media type: application/json Controls Accept header. Example Value | Schema	*No links*

```
{
  "id": 0,
  "username": "string",
  "email": "string",
  "token": "string",
  "refreshToken": "string",
  "userRole": "string"
}
```

POST /api/auth/signin Authenticate an account and log in.

Parameters | Try it out

No parameters

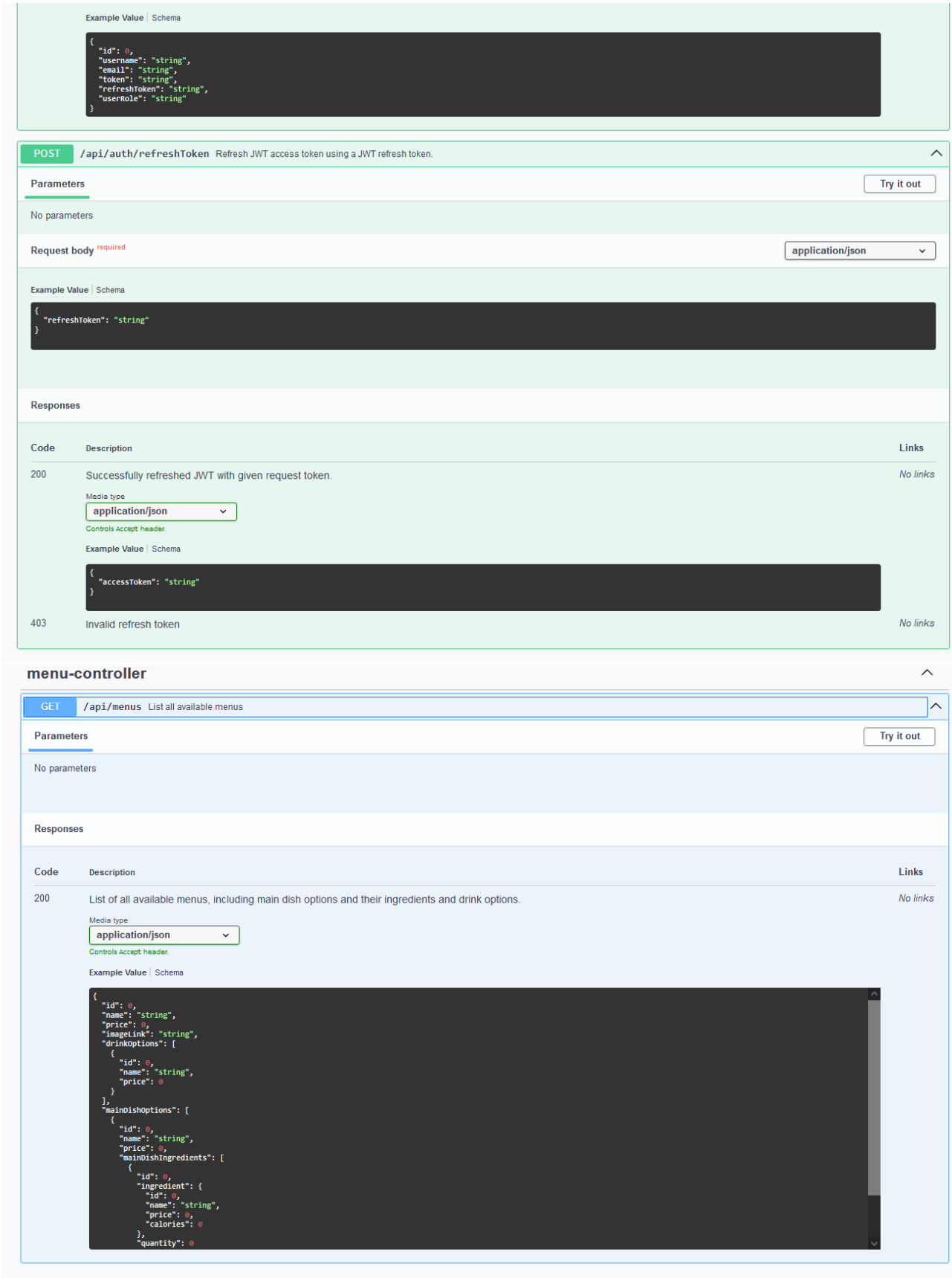
Request body required — application/json

Example Value | Schema

```
{
  "email": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	Successfully authenticated and logged user in. Media type: application/json Controls Accept header.	*No links*


Example Value | Schema
{
"id": 0,
"username": "string",
"email": "string",
"token": "string",
"refreshToken": "string",
"userRole": "string"
}
POST /api/auth/refreshToken Refresh JWT access token using a JWT refresh token.
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"refreshToken": "string"
}
Responses
Code
Description
Links
200
Successfully refreshed JWT with given request token.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"accessToken": "string"
}
403
Invalid refresh token
No links
menu-controller
GET /api/menus List all available menus
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
List of all available menus, including main dish options and their ingredients and drink options.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"price": 0,
"imageLink": "string",
"drinkOptions": [
{
"id": 0,
"name": "string",
"price": 0
}
],
"mainDishOptions": [
{
"id": 0,
"name": "string",
"price": 0,
"mainDishIngredients": [
{
"id": 0,
"ingredient": {
"id": 0,
"name": "string",
"price": 0,
"calories": 0
},
"quantity": 0

6 References and resources

Project Resources

Resource	URL / Location
Git Repository	https://github.com/notjoao1/TQS_SpringCanteen
Jira Project	https://springcanteen.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog
Video Demo	https://www.youtube.com/watch?v=ugFbt7i12fo
QA Dashboard	https://sonarcloud.io/summary/overall?id=notjoao1_TQS_SpringCanteen
CI/CD Pipeline	https://github.com/notjoao1/TQS_SpringCanteen/tree/main/.github/workflows