

TQS: Product specification report

Gonalo Ferreira [107853], Joo Carlos Santos [110555], Matilde Teixeira [108193], Sara Almeida [108796]

V2024/04/24

1	Introduction	1
1.1	Overview of the project.....	1
1.2	Limitations.....	2
2	Product concept	2
2.1	Vision statement.....	2
2.2	Personas	4
2.3	Main scenarios	2
2.4	Project epics and priorities.....	6
3	Domain model.....	7
4	Architecture notebook.....	8
4.1	Key requirements and constrains.....	8
4.2	Architetural view	9
4.3	Deployment architerture	11
5	API for developers.....	12
6	References and resources.....	12

1 Introduction

1.1 Overview of the project

This is a project developed in the context of TQS course (Testes e Qualidade de Software), for which the main goals are the implementation of a software quality assurance strategy and the implementation of continuous testing, integration and delivery. To achieve these aspects, the proposal was to build a Minimal Viable Product (MVP) system, with core functionality, architecture and implementation that is coherent with the previously mentioned principles.

With this in mind, our aim is to build the DETICafe, a system to support a bar service for university students, mainly DETI ones. The innovation of the system is centralized in the order process: when placing orders, the client does so via a kiosk and then they wait for the order's number to appear on another dashboard, where they can see if the order is ready to be picked up.

To make this happen the system will have 4 different interfaces: one for the clients to place order (kiosk one), one for the cook to see the order's products, their ingredients and possible costumization and select when it's ready for the waiter to pick it up, one for the waiter to sinalize the order's dashboard that the order is ready for customer deliver and one for the clients to know if their order is preparing or ready for them to pick it up.

This service will be a digital transformation of a cafe and will be very helpful in its the management by organizing and streamlining the preparation and delivery of orders. It will make students life easier in the way

they can previously order, for example, their lunches in an easy way but also help bar workers by giving them a system that organizes the orders for them and ends up shortening waiting lines.

1.2 Limitations

Our system will feature a kiosk interface allowing the client to select the order's items. Once the order is placed, it appears on the kitchen dashboard in a queue-like view, providing the order's ingredients for the cook to prepare. When the order is ready, the cook will signalize that it is ready for pick up. The waiter responsible for the product's pick-up will signal on the platform that the product is ready. By doing this, the corresponding number of order will appear in the order's dashboard and the client will know that he needs to pick it up.

This type of systems require payment processing but we won't be implementing this feature, not only because we lack time for it, but also because it is a bit off topic and unnecessary for the achievement of the main goal of this project.

2 Product concept and requirements

2.1 Vision statement

The aim of our system is to provide an easier and faster order and delivery process for students in university, specifically DETI students, who don't always like communicating with others.

Our system will function in a way that the customer makes the order at the kiosk interface, and when they pay, the order is sent to the kitchen interface. There, the cook can see the order's specific requests and possible alterations in order to cook it. When the status is done, the cook sends a signal to the waiter dashboard for him to pick it up. When the order is at the pickup spot, the waiter changes the status in the order display interface, so that the customer can pick it up.

The main difference from other similar well-known products is the fact that our system will make life easier, not only for the bar workers but especially for DETI students who don't have the time to wait in waiting line for their order. Using our application they will be able to previously place their order and only get them when they are ready for pick up. Also during the ordering process, this is separated into different sections, enabling a customer to order from a cafeteria, pastery or even restaurant, etc., making the process easier and quicker.

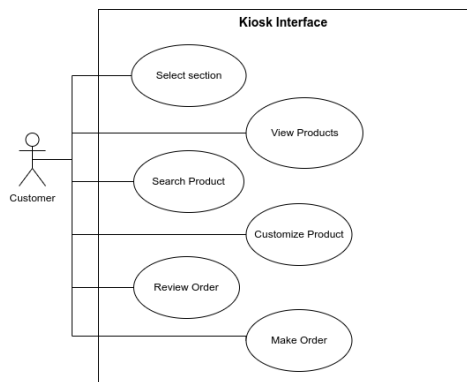


Figure 1- UML Case Diagram Customer Actions in Kiosk Interface

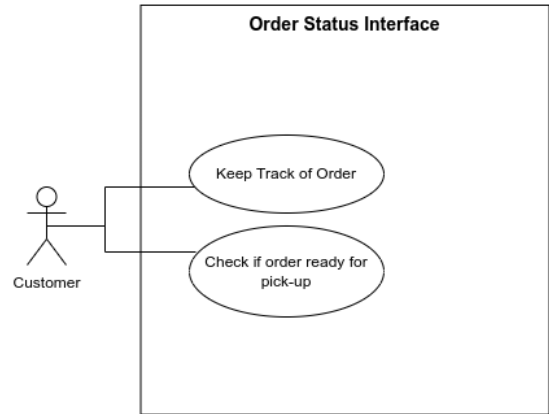


Figura 2- UML Case Diagram Customer Actions in Order Status Interface

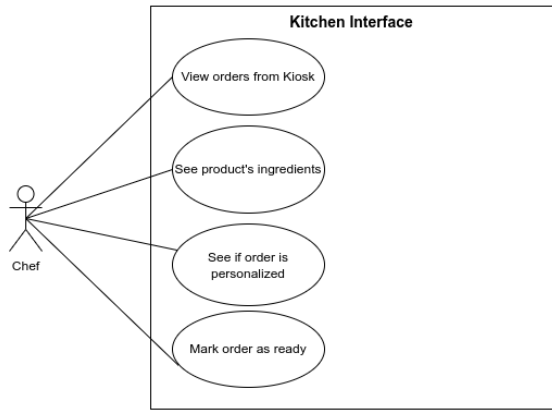


Figure 3- UML Case Diagram Chef in Kitchen Interface

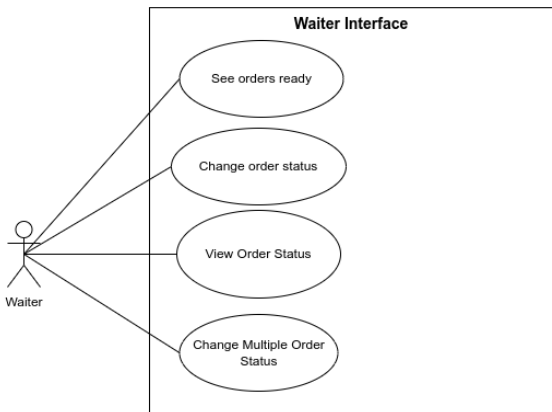


Figure 4- UML Case Diagram Waiter in Waiter Interface

2.2 Personas and scenarios

Personas



D. Alberta, 58, Waitress at DETICafe

Fifty-eight-year-old D.Alberta is a very nice lady that works at the DETICafe and is very cherished by all the DETI students. She is a grandmother of 2 young adults, both in college, so she is very familiar with this age, she loves all her costumers and her job.

D.Alberta is still a very agile and active lady but sometimes the lines of students waiting for their orders are big and it is a bit difficult to manage all the orders and deliver them tho whom indeed ordered them.

Motivation: D.Alberta would like a very simple and intuitive system to help her manage the waiting lines, so she would keep track of the orders that were already delivered and those that were not yet and are still being prepared.



Luís, 29, Cook at DETICafe

Twenty-nine-year-old Luís is one of D.Alberta's sons. Since he was little he used to go with his mom to work during school breaks and always dreamed of working with her and helping her at DETICafe. Therefore, when he grew up he decided to take a cooking course so he would be in charge of the kitchen at DETICafe and make the best menu for the students.

For many years, D.Alberta would write the orders in a paper and take to the kitchen. However, in times where there are a lot of costumers coming to the bar at the same time this method would turn out to be not so efficient. Sometimes

the papers would even get lost or unordered.

Besides, sometimes students ask for personalized menus, without some of the ingredients or more of another some, and the method they used would sometimes make him mistake those.

Luís and D.Alberta have also been looking for a way to make life easier to students and shorten waiting lines by having a system where these could order online.

Motivation: Luís needs a more efficient way to keep track of the orders, specially the order in which they were made and those who have customized ingredients requests. He would also like to have a view of which ingredients each order requires so it would be easier for him to quicker serve his costumers.

**Pedro Santos, 23, LECI student**

Pedro is a determined student and focused on his field of study. He was always interested in electronics and computing from an early age and decided to pursue a career in Computer and Informatics Engineering. He is a quiet and reserved person, but very dedicated to his studies and academic projects. Despite his busy schedule, Pedro values moments of pause and relaxation during the day. He sees the launch break as an opportunity to recharge and socialize with his course mates. The DETICafe is the ideal place for Pedro, as it offers a variety of quick and practical snack options, perfect for breaks between classes.

At the DETICafe, Pedro usually meets other students on his course and exchanges experiences about the projects they are developing. He also takes advantage of the moment to review class material and discuss doubts with colleagues. Furthermore, the relaxed atmosphere of the DETICafe allows Pedro to relax and unwind before returning to academic activities.

Motivation: Pedro has a short break between classes and needs a fast way to choose and buy some food, he doesn't want to wait in a line since he hasn't much time.

**Ana, 21, LEI student**

Ana is a dedicated and enthusiastic student, passionate about technology and programming since a young age. He is an extroverted and sociable person, always willing to help colleagues with projects and group work. Ana highly values education and is committed to achieving her goal of becoming a successful software engineer.

In addition to her studies, Ana also enjoys maintaining a balanced lifestyle. She understands the importance of taking breaks and resting during the day, and one

of these breaks is lunchtime.

The DETI (Department of Electronics, Telecommunications, and Informatics) bar is the ideal place for Ana, as it offers a variety of quick and healthy meal options at affordable prices. It's a place where she can meet friends, relax, and recharge for the rest of the day of studying.

Motivation: Ana wants to have lunch at the DETICafe and she needs a simple way to see the menu, select what to eat and make the request. Ana also needs to know when food is ready to pick it up.

2.3 Scenarios

1. **Ana Silva places an order** - Ana wants to order a cheese and ham croissant and a coffee at DETICafe. For that she opens our application, and she starts by choosing the pastery section. She already knows what she wants to order so she goes for the croissant and selects it. After this she selects the cafeteria section, searches for a coffee, chooses that she wants it large and selects it. Then she finishes her order and receives a receipt with the order number.
2. **Pedro Santos orders a sandwich with customized ingredients** - Pedro needs to order his lunch previously because he only has 30 minutes to eat so he goes to our application and selects the restaurant section. Then he selects the sandwich menus and scrolls to see all the sandwiches that the DETICafe offers. He decides to choose the tuna sandwich, but he is allergic to tomato, so he selects the "personalize" button and takes the tomato out in his order. Then he goes to the drinks section and chooses a water and selects it. Then he finishes his order and receives a receipt with the order number.
3. **Pedro Santos gets his order**- Pedro goes to DETICafe to get the order he placed. He arrives and he sees that his order is ready at the dashboard, so he goes to the counter with his receipt and D.Alberta delivers him his order.
4. **Luís receives an order at the kitchen dashboard and prepares it** - At the kitchen dashboard, Luís receives Pedro's order and see the ingredients of the sandwich he ordered and an indication that he wants it without tomato. He prepares the desired sandwich and when he is finished, he selects it in the interface and clicks the "ready" button to notify D.Alberta that she can go pick up the order from the kitchen to deliver it.
5. **D. Alberta gets an order and delivers it** - At the counter dashboard, D.Alberta receives a notification saying the order 102 is ready. She goes get that order to the kitchen and changes the state of the order in the dashboard to ready.

2.4 Project epics and priorities

Epics and respective user stories

For this system implementation, our team is following an Agile methodology (with weekly sprints), oriented by the concretization of the epics. The order for this development process was decided based on the importance of each epic for the final goal of the whole application, and the necessary requirements for the general workflow. We want to follow a test-driven approach, to ensure quality of testing and use.

1- Week 1 (10/05/24-15/05/24)

Kiosk Interface

- (Epic) As a customer, I want to select the section I want to order from, so that it is easier to see the available products from there.
- (Epic) As a customer, I want to explore all the items available at the DetiCafe in the initial page, so that I can choose what I want for lunch.
- (Epic) As a customer, I want to search a specific product so that I can order faster and in an easier way.
- (Epic) As a customer, I want to customize my sandwich, so that I can have a personalized sandwich without eggs since I'm allergic.
- (Epic) As a customer, I want to review my order before finalizing it, so that I am sure that I'm ordering everything right.
- (Epic) As a customer, I want to receive my order number, so that I can keep track of my order in the order display interface.

2- Week 2 (16/05/24-22/05/24)**Kitchen Interface**

- (Epic) As a cook, I want to view incoming orders from the kiosk interface, so that I can start preparing them as soon as I can in the order they were placed.
- (Epic) As a cook, I want to see the ingredients needed to prepare the incoming order, so that I don't forget anything and deliver the best product to my client.
- (Epic) As a cook, I want to see if the customer personalized the ingredient of the order, so that I don't mistake the order and deliver the wanted product to my client.
- (Epic) As a cook, I want to mark orders as ready, so that the waiter knows that they can pick them up.

3- Week 3 (23/05/24-)**Waiter Interface**

- (Epic) As a waiter, I want to see orders that are ready for pickup in the kitchen, so that I can go get them from there.
- (Epic) As a waiter, I want to change the status of an order to ready, so that I inform the customer that their order is ready for pickup.
- (Epic) As a waiter, I want to view organizedly the orders that are ready and the ones that aren't done yet, so that I can see which orders do I need to be ready to pick up.
- (Epic) As a waiter I want to efficiently change the status of multiple orders, so that the waiting line doesn't get upset.

Order Status Interface

- (Epic) As a customer, I want to be able to keep track of my order to see if it's still preparing, so that I can quickly go to the bathroom and not miss it.
- (Epic) As a customer, I want to see when my order is ready for pick up, so that I can pick it up and enjoy it faster.

3 Domain model

In our system, we have identified six primary entities: **Customer**, **Order**, **Staff**, **Order Details**, **Product**, and **Section**. Each entity serves a distinct role within the system.

- **Customer**: Represents individuals who place orders. Each customer may have multiple orders associated with them.
- **Order**: Captures details of each order placed by customers, including the order details and status of the order.
- **Staff**: Represents the employees who handle orders. They can be waiters or the cook chef. Staff members are associated with the orders they process.
- **Order Details**: This entity provides specific details about each product within an order, which means the modifications made by the customer.
- **Product**: Represents the items available for purchase. Each product may belong to a specific section, such as 'Restaurant' or 'Cafe.'
- **Section**: Describes different categories or locations within which products are sold.

Our chosen database for this project is MySQL, selected for its reliability and foundation for persistence. By utilizing this database model, we aim to create a system where customers can easily

place orders, which are then efficiently processed by staff members. The inclusion of the Order Details entity allows for precise tracking of each order's contents and any modifications made by the customer.

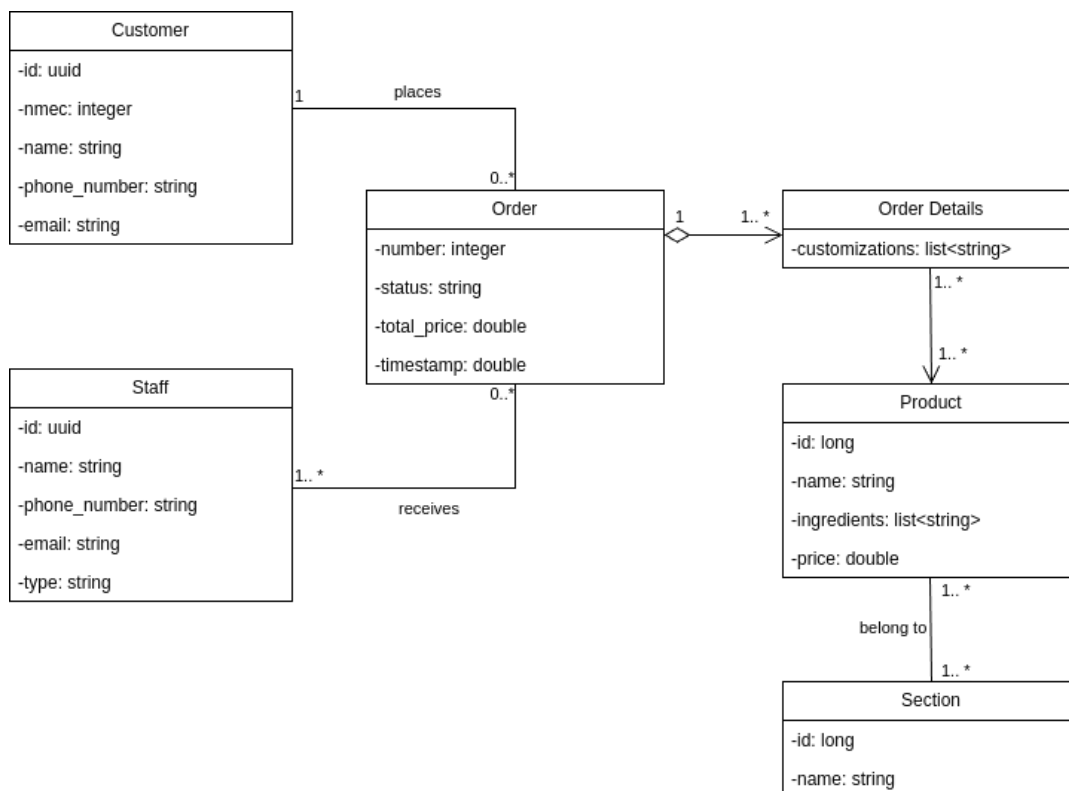


Figure 5- System's Domain Model

4 Architecture notebook

4.1 Key requirements and constrains

The architecture choices made were derived from the key requirements identified within the application:

- Non-dependency from external services, legacy-systems, providing a more straightforward approach, by simplifying the deployment of the service
- To efficiently prepare and deliver orders, efficiency is a must, to streamline orders. The system needs to be able to handle concurrent orders, and update its information in real time, to provide a better and seamless approach to the general user, the customer, as well as to the chef and waiter.
- To maintain an uninterrupted process, it is essential to implement mechanisms for fault tolerance and reliability, such as redundancy, failover systems, and error handling strategies, to help mitigate a possible failover of the system.
- In relation to concurrent orders, the system should implement additional performance optimization strategies like a fast-caching and easy-to-use mechanism, such as Redis.
- The following system will require the existence of 4 different interfaces to streamline the preparation and delivery of the different orders meant to be done by different workers with specific roles (kiosk, kitchen dashboard, waiter dashboard, order status dashboard). It's essential to ensure smooth operations to assure proper communication and integration between these interfaces while ensuring each interface meets its functional requirements.
- The system will be accessed through the kiosk interface, a physical kiosk, that will pass the order's request to the kitchen dashboard. When the order is ready, the chef will need to select that the order

is ready for pick-up. This will trigger the waiter dashboard that will, when the waiter selects that the order is ready to be delivered, showcase the order number on the client dashboard.

- It is essential to ensure security when the client is making an order. It is necessary to implement access control for the client and the different positions within the cafe, as well as a mechanism for secure data processing.
- As the system is meant to be used by a cafe, used by university students, scalability is crucial. To handle increasing user loads and potential future expansions (e.g., additional campus locations), the architecture must be designed to accommodate expansion.
- To ensure data integrity and consistency since data is shared between environments it is crucial to implement appropriate data validation, synchronization, and transaction management mechanisms.

4.2 Architecture view

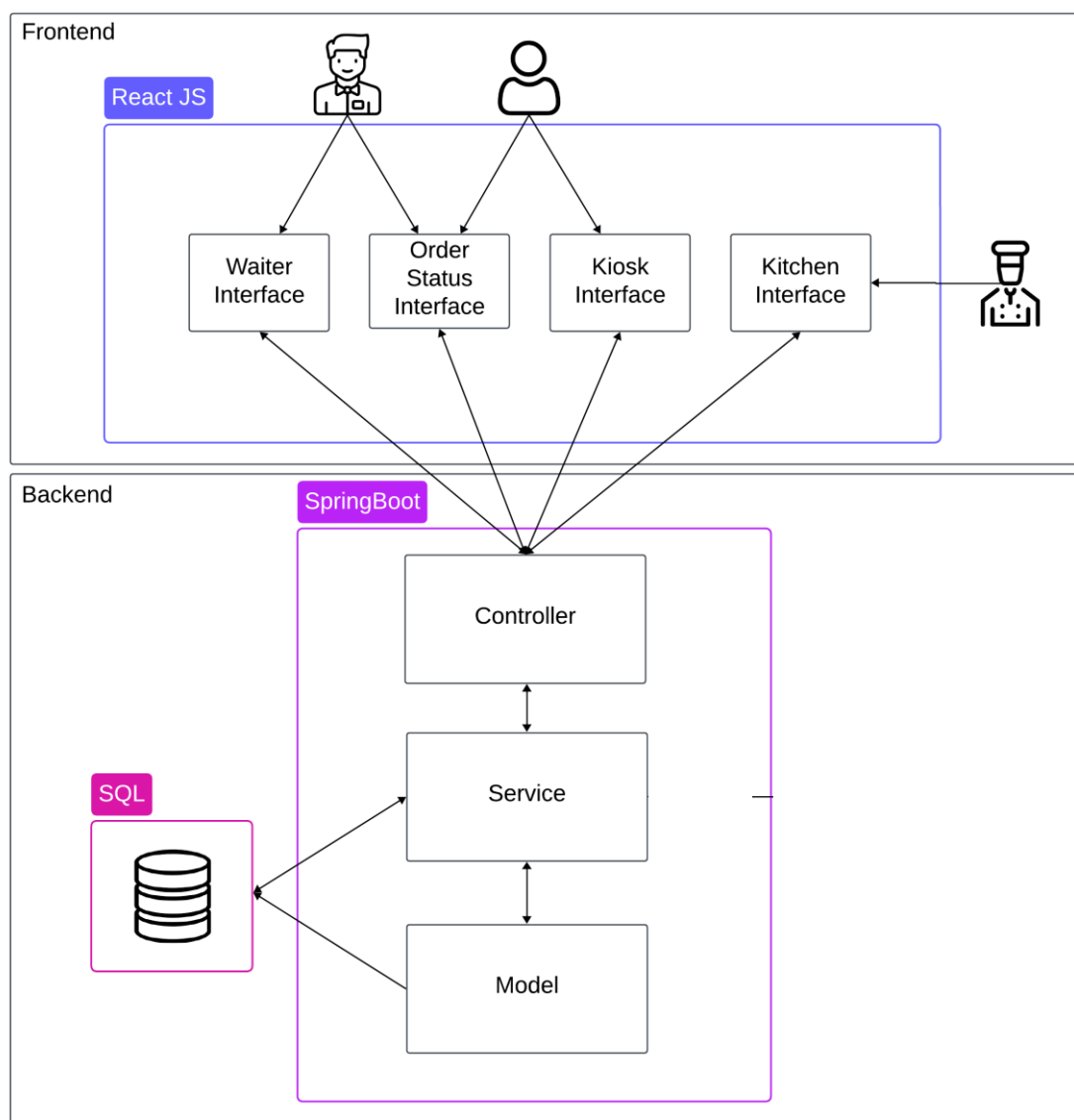


Figure 6- System's Architecture Diagram

As it is presented on the image above, the architecture of the system is based on two distinct layers: the front-end layer and the back-end layer:

Front-end:

For the front-end layer, we have chosen to use ReactJS due to its component reusability and the team's expertise in this technology. ReactJS will enable us to build a dynamic and responsive user interface, significantly enhancing the user experience.

Back-end:

In the back-end layer, we have opted for Spring Boot as the primary framework. Spring Boot offers simplified configuration and facilitates rapid development of robust and scalable web applications.

Within the Spring Boot application, we have three main components:

- Controllers: Responsible for handling HTTP requests from clients and directing them to the appropriate services.
- Services: Implementing the business logic of the application and interacting with controllers and data models.
- Models: Representing the data entities of the application and interacting with the MySQL database and Redis for storage and retrieval of information.

Technology Choice:

We have chosen MySQL as the database due to its reliability and the relational nature of the application's data. Redis was selected as the caching engine for its speed and ease of use, which will contribute to the overall performance of the application.

Additional Considerations:

In addition, we have implemented security measures such as authentication and authorization to protect the application against unauthorized access. We are continuously evaluating and optimizing the architecture to ensure adequate scalability and performance, particularly regarding the intensive use of Redis as a caching mechanism.

To clarify the interactions between the different components, we made a sequence diagram that is shown below.

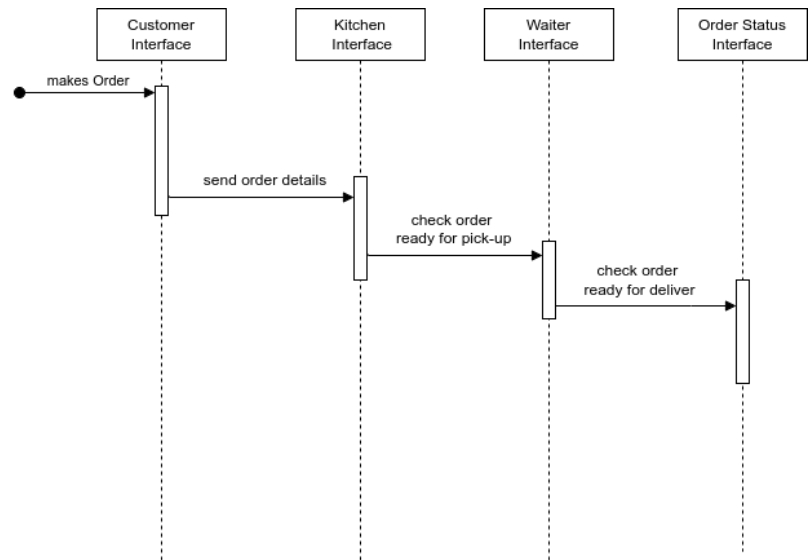


Figure 7- Sequence Diagram for System's Interactions

4.3 Deployment architecture

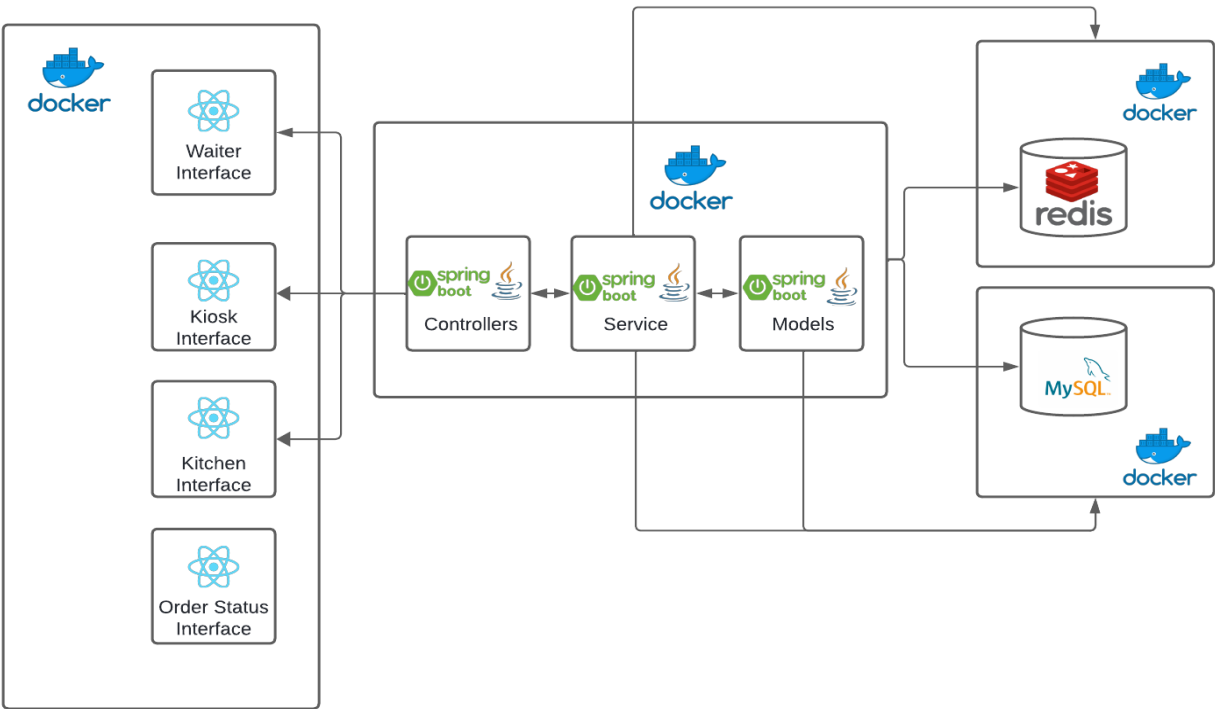


Figure 8- System's Deployment Diagram

The deployment process was executed using Docker Compose. To achieve this, MySQL and Redis databases were loaded using containers on the provided machine, along with containers for the backend and frontend. This approach was aimed at simplifying the deployment process. Consequently, the continuous deployment process has been streamlined, requiring only a restart of the services and rerunning them.

5 API for developers

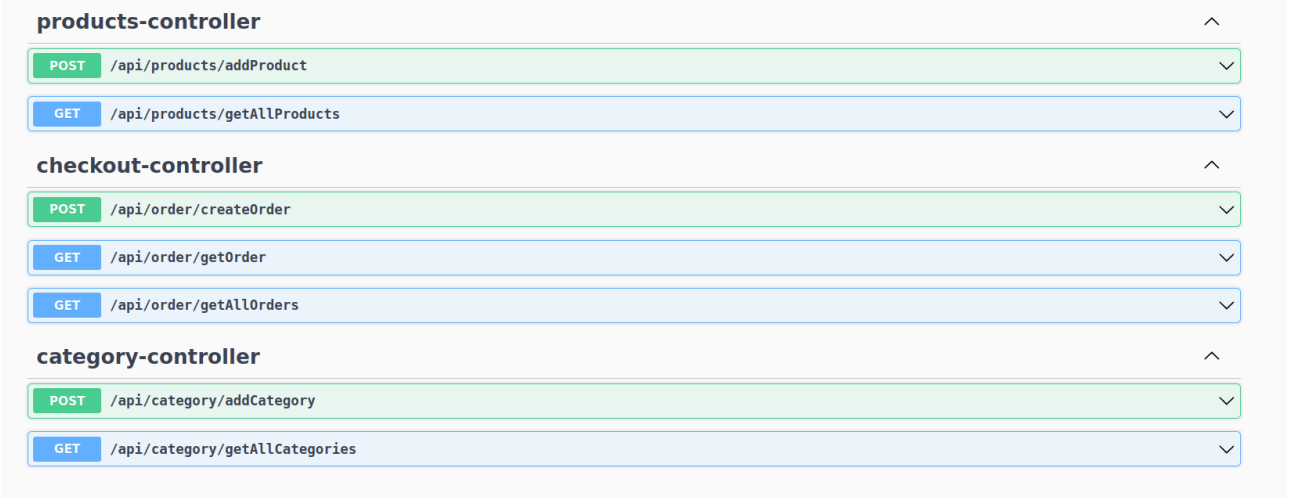
Our API is organized with our main entities. We have an endpoint for the product entity, order entity and category entity.

In our product endpoint we can make a post with the product name, the list with the name of the ingredients, the price and the category. After the validation of those parameters a new product is added to our database. Our get endpoint returns a list of all products in our database.

In our Order endpoint we can post a new order by giving it a list of orderDetailsDTO. We have 2 endpoints for the GET methods, one like our product GET that returns all orders, the other one get the order by the Id provided.

For the Category endpoint, the POST method creates a new category with the name of the category being provided, and the GET method returns all categories.

To access the API information consult swagger: localhost:8080/swagger-ui/index.html



products-controller		^
POST	/api/products/addProduct	▼
GET	/api/products/getAllProducts	▼
checkout-controller		^
POST	/api/order/createOrder	▼
GET	/api/order/getOrder	▼
GET	/api/order/getAllOrders	▼
category-controller		^
POST	/api/category/addCategory	▼
GET	/api/category/getAllCategories	▼

Figure 9- API documentation

6 References and resources

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

<https://app.diagrams.net/>

<https://start.spring.io/>