

# TQS: Product specification report

*João Andrade 107969, Pedro Pinho 109986, Tomás Victal 109018*  
v2024-06-3

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview of the project	1
1.2	Limitations	1
<b>2</b>	<b>Product concept</b>	<b>2</b>
2.1	Vision statement	2
2.2	Personas	2
2.3	Main scenarios	2
2.4	Project epics and priorities	2
<b>3</b>	<b>Domain model</b>	<b>2</b>
<b>4</b>	<b>Architecture notebook</b>	<b>3</b>
4.1	Key requirements and constraints	3
4.2	Architectural view	3
4.3	Deployment architecture	4
<b>5</b>	<b>API for developers</b>	<b>4</b>
<b>6</b>	<b>References and resources</b>	<b>4</b>

## 1 Introduction

### 1.1 Overview of the project

The main objective of this project is to familiarize us with having a defined team and project structure, where the features are divided into different user stories and are assigned to different team members. With this we cultivate knowledge of working as a team while keeping an organized backlog. Not only that but also integration of QA and CI/CD methods that allow us to seamlessly have continuous integration and deployment.

Our product is called Restaurante Moles Ovos and aims to make communication between restaurant staff easier while also providing customers with information and booking options. This product focuses on removing the need for traditional restaurant communication methods by providing a better and easier UX for both our customers and employees. Providing a digital signage system in a restaurant's kitchen, mobile application for the waiter and a web application for our customers, facilitating orders, bookings, requests communication to the kitchen and delivering food faster to the tables.

## 1.2 Limitations

- **Registration:** There is currently no way for a user to register themselves. If a customer/employee wants to login he must use one of the already created login credentials
- **Booking Management:** While a user can create and check his bookings he has no extra control over them, such as updating/canceling them.

## 2 Product concept and requirements

### 2.1 Vision statement

Our system is meant to be used in a restaurant environment, where through our applications the waiters will be able to communicate back and forth with the cooks, making it easy for the waiters to share the customer's orders with the cooks, and for the cooks to easily be able to tell the waiters when the food is ready. The orders made by the waiters will be sent to the kitchen to a digital signage board where the cook's can see them.

With our system we plan to remove the need to use pen and paper to annotate the customer's orders, removing the possibility of forgetting/losing the order. Not only that but this also allows the waiters to be more efficient, not needing to move back and forth between the tables and the kitchen. We also implement a system that allows the waiters to check for the stock of certain menu items, making it easy for them to know which items can be ordered or not.

In addition to the applications for the staff we also have a website for the user to be able to check for available times and make reservations.

### 2.2 Personas and scenarios

#### PERSONAS

##### Zé

**Background:** Zé has been a cook in a restaurant for a long time. Zé dislikes confusion in the kitchen and hates it when people don't know what to do, making the process in the kitchen slower and causing problems.

**Occupation:** cook

**Idade:** 52

**Goals:** Zé would like a faster and more efficient way of receiving the orders from the clients so that it's easier to coordinate the situation in the kitchen to, in turn, make it faster. Zé also wants an easy way to alert the waiters that the food is ready without having to call them or let the food get cold.

##### Maria

**Background:** Maria just got started as a waiter in a big restaurant. She has quite bad handwriting and forgets very easily making it so that the other staff gets mad at her for this. She also really dislikes going back to a table to tell the clients that the food they ordered has ran out.

**Occupation:** Waiter

**Idade:** 33

**Goals:** Maria wants to get an easier way to tell the cooks the orders that were made without causing trouble with forgetting them or bad handwriting. She also wishes there was a way to know the stock of the menu items so that she never has to go back and tell the customers that they need to choose something else. When the food is ready she would also like an easy way to

be notified when the food is ready so that she doesn't have to go back and forth to the kitchen all the time.

### **Joana**

**Background:** Joana is a first year university student. Even though her lunch break is quite short, when she knows that the canteen is closed she likes to go to nearby restaurants in her limited time.

**Occupation:** University Student

**Idade:** 22

**Goals:** Joana would like to be able to check the menu of the restaurants near her university and make a reservation so that she can be sure she has time to eat in her lunch break.

## **SCENARIOS**

### **Scenario 1 - Zé**

Quickly glances over the digital signage to know exactly what is left to do and updates its state to PREPARING.

### **Scenario 2 - Zé**

After having cooked a meal he updates the request to COMPLETE and a notification is sent to our waiter.

### **Scenario 3 - Maria**

Checks food and quantity requested by the customer and indicates if it is available or not.

### **Scenario 4 - Joana**

Analyses possible slots to book her reservation on a day that has yummy food.

## **2.3 Project epics and priorities**

To better organize our project, and guarantee that most functionalities are ready by the set deadline, we focused on creating 8 epics and splitting them throughout different sprints.

These epics are split into four main categories (ordered from most to least priority):

- Repository configuration
  - This epic contains some tasks to make the basic configuration of the multiple repositories.
- The Other Epic are described in the table below:
  - Use cases in the table are the project epic.
  - User stories were transformed into tasks of the Epic.

Use case (episode of use related to some actor's objective. → defined by the Analyst)	User stories (a unit of work to assign to the development team that reflects some useful increment from the business perspective. → defined collectively)	Acceptance criteria (specific situations to verify the expected functional behavior)
Fazer Um pedido à cozinha (Empregado de mesa)	<b>Empregado de mesa consegue ver stock</b> Sendo Empregado de mesa(Maria), gostaria de poder ver os pratos do menu disponíveis na cozinha de modo a poder fazer o pedido.	<b>Cenário 1</b> Sendo que sou um empregado de mesa Quando procuro por um prato Então posso verificar se esse prato está disponível
	<b>Empregado de mesa escolhe pratos para pedido</b> Sendo Empregado de mesa(Maria),gostaria de escolher os pratos do menu de modo a colocar os pratos no pedido.	<b>Cenário 1</b> Sendo que sou um empregado de mesa Quando procuro por um prato E o prato está disponível Então posso adicionar o prato ao pedido  <b>Cenário 2</b> Sendo que sou um empregado de mesa Quando procuro por um prato E o prato não está disponível Então não posso adicionar o prato ao pedido E posso continuar a procurar  <b>Cenário 3</b> Sendo que sou um empregado de mesa Quando já tenho pratos no pedido Então posso remover pratos do pedido
	<b>Empregado de mesa faz o pedido</b> Sendo empregado de mesa(Maria),gostaria de poder enviar o pedido para a cozinha de modo a os cozinheiros serem informados sobre o novo pedido.	<b>Cenário 1</b> Sendo que sou um empregado de mesa Quando tenho pratos adicionados ao pedido E todos os pratos estão disponíveis Então posso enviar o pedido para a cozinha
Receber notificação pedido pronto (Empregado de mesa)	<b>Empregado de mesa recebe notificação</b> Sendo empregado de mesa(Maria),gostaria de receber uma notificação da cozinha de modo a saber que o pedido está pronto.	<b>Cenário 1</b> Sendo que sou um empregado de mesa Quando o pedido está pronto Então recebo uma notificação a dizer que está pronto  <b>Cenário 2</b> Sendo que sou um empregado de mesa Quando ocorre um problema com o pedido Então recebo uma notificação a informar do problema
Aceitar reservas dos clientes (Empregado de mesa)	<b>Empregado de mesa vê reservas dos clientes</b> Sendo empregado de mesa(Maria),gostaria de ver reservas feitas pelos clientes de modo a poder gerenciar as reservas.	<b>Cenário 1</b> Sendo que sou um empregado de mesa Quando procuro as reservas pendentes Então consigo ver a informação sobre estas
	<b>Empregado de mesa aceita/recusa reserva</b> Sendo empregado de mesa(Maria),gostaria de poder aceitar ou recusar uma reserva feita pelo cliente de	<b>Cenário 1</b> Sendo que sou um empregado de mesa Quando acedo a uma reserva pendente

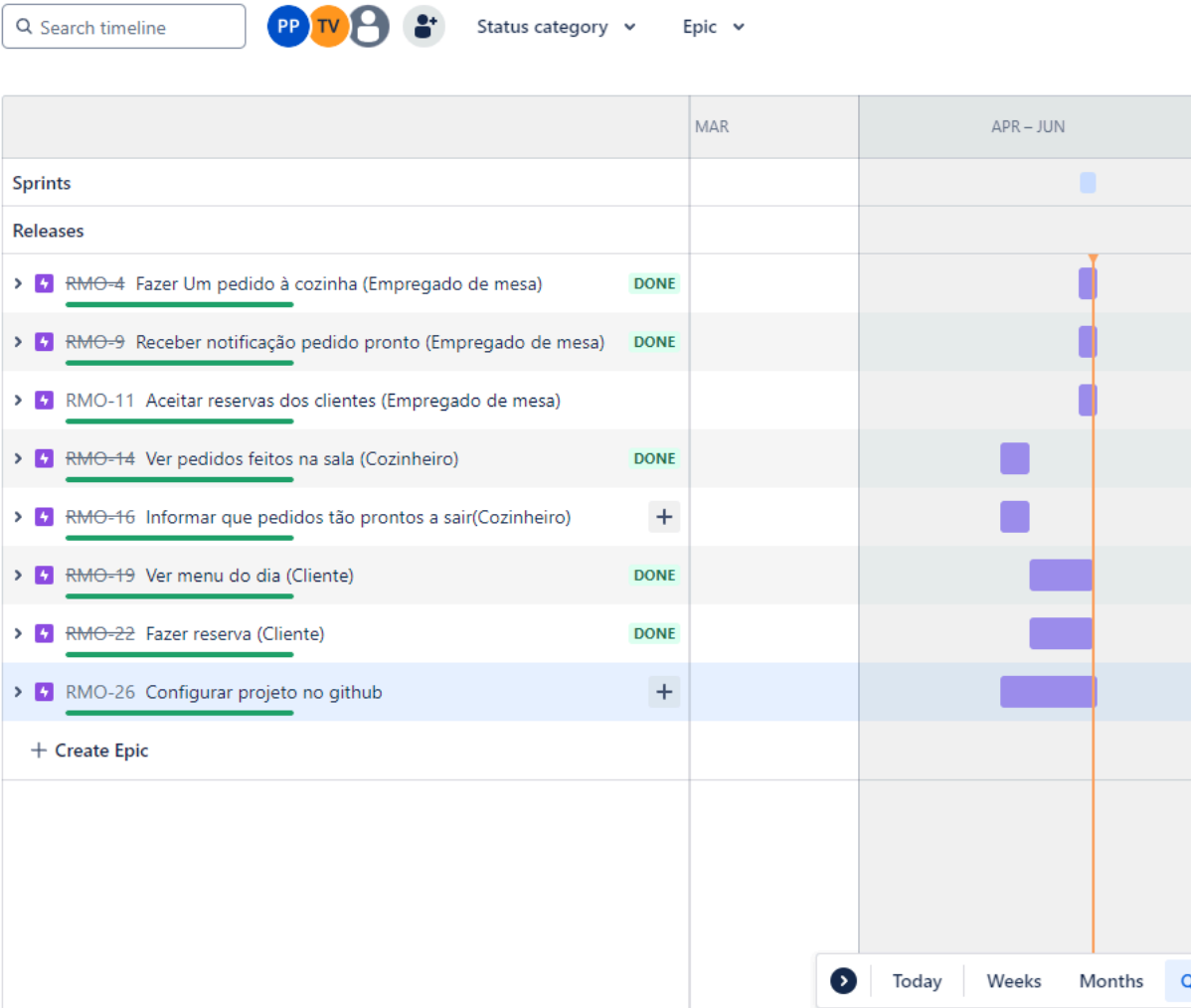
	modo a informar o cliente sobre a reserva. (mensagem customizada ao recusar)	<p>E é possível fazer a reserva Então consigo aceitá-la</p> <p><b>Cenário 2</b> Sendo que sou um empregado de mesa Quando acedo a uma reserva pendente E não é possível fazer a reserva Então consigo recusá-la</p>
Ver pedidos feitos na sala (Cozinheiro)	<p><b>Cozinheiro vê pedidos feitos</b> Sendo cozinheiro(Zé), gostaria de ver todos os pedidos feitos pelos empregados de mesa num sistema de digital signage de modo a ter informações detalhadas do pedido.</p>	<p><b>Cenário 1</b> Sendo que sou um cozinheiro Quando recebo um pedido Então devo poder ver a informação sobre este na digital signage</p>
Informar que pedidos tão prontos a sair(Cozinheiro)	<p><b>Cozinheiro vê todos os números de pedido</b> Sendo cozinheiro(Zé), gostaria de poder ver uma lista com todos os números de pedido de modo a poder informar que o pedido está pronto a sair.</p>	<p><b>Cenário 1</b> Sendo que sou um cozinheiro Quando acedo à lista de pedidos Então consigo interagir com os pedidos</p>
	<p><b>Cozinheiro informa que pedido está pronto</b> Sendo cozinheiro(Zé), gostaria de poder selecionar o pedido e informar que está pronto ou ocorreu um erro para sair da cozinha de modo que o empregado de mesa seja informado que o mesmo está pronto.</p>	<p><b>Cenário 1</b> Sendo que sou um cozinheiro Quando um pedido está completo Então consigo informar o empregado</p> <p><b>Cenário 2</b> Sendo que sou um cozinheiro Quando ocorre um problema com o pedido Então consigo informar o empregado</p>
Ver menu do dia (Cliente)	<p><b>Cliente vê o menu do dia</b> Sendo Cliente(Joana), gostaria de poder ver o menu do restaurante de modo a saber os vários pratos que estariam disponíveis no dia.</p>	<p><b>Cenário 1</b> Sendo que sou um utilizador Quando acedo ao website do restaurante Então devo poder ver o menu do dia</p>
	<p><b>Cliente vê as informações de um prato</b> Sendo Cliente(Joana), gostaria de poder ver as informações de um prato do menu do restaurante de modo a ver os seus ingredientes e modo preparo (ver se o prato é frito, cozido ou assado)</p>	<p><b>Cenário 1</b> Sendo que sou um utilizador Quando acedo à página de um prato Então devo poder ver a sua informação</p>
Fazer reserva (Cliente)	<p><b>Cliente Ver horários disponíveis</b> Sendo Cliente(Joana), gostaria de ver os horários disponíveis para fazer reserva de modo a escolher o melhor horário para as minhas necessidades.</p>	<p><b>Cenário 1</b> Sendo que sou um utilizador Quando acedo à página dos horários Então devo poder ver os horários disponíveis para fazer reserva</p>
	<p><b>Cliente escolher horário</b> Sendo Cliente(Joana),gostaria de escolher um horário</p>	<p><b>Cenário 1</b> Sendo que sou um utilizador</p>

	para fazer reserva de modo a ter uma reserva pendente nesse horário.	Quando escolho um horário E o horário está disponível Então devo poder fazer uma reserva
	<b>Cliente ver status da reserva</b> Sendo Cliente(Joana),gostaria de verificar o status da reserva de modo a saber se a reserva está pendente, recusada ou aceite.	<b>Cenário 1</b> Sendo que sou um utilizador Quando tenho uma reserva feita Então posso verificar o estado da minha reserva


Epic Stories and Sprint Board






Projects / Restaurante Moles Ovos


Timeline








Exemplo Epic Story


 RMO-26

 1   ... 

 **Configurar projeto no github**



In Progress ▾

 Actions ▾

























Description


Para ter tudo a funcionar e pronto para o desenvolvimento do projeto, será necessário organizar e configurar tudo corretamente antes de começarmos.

Child issues

Order by ▾ ... +

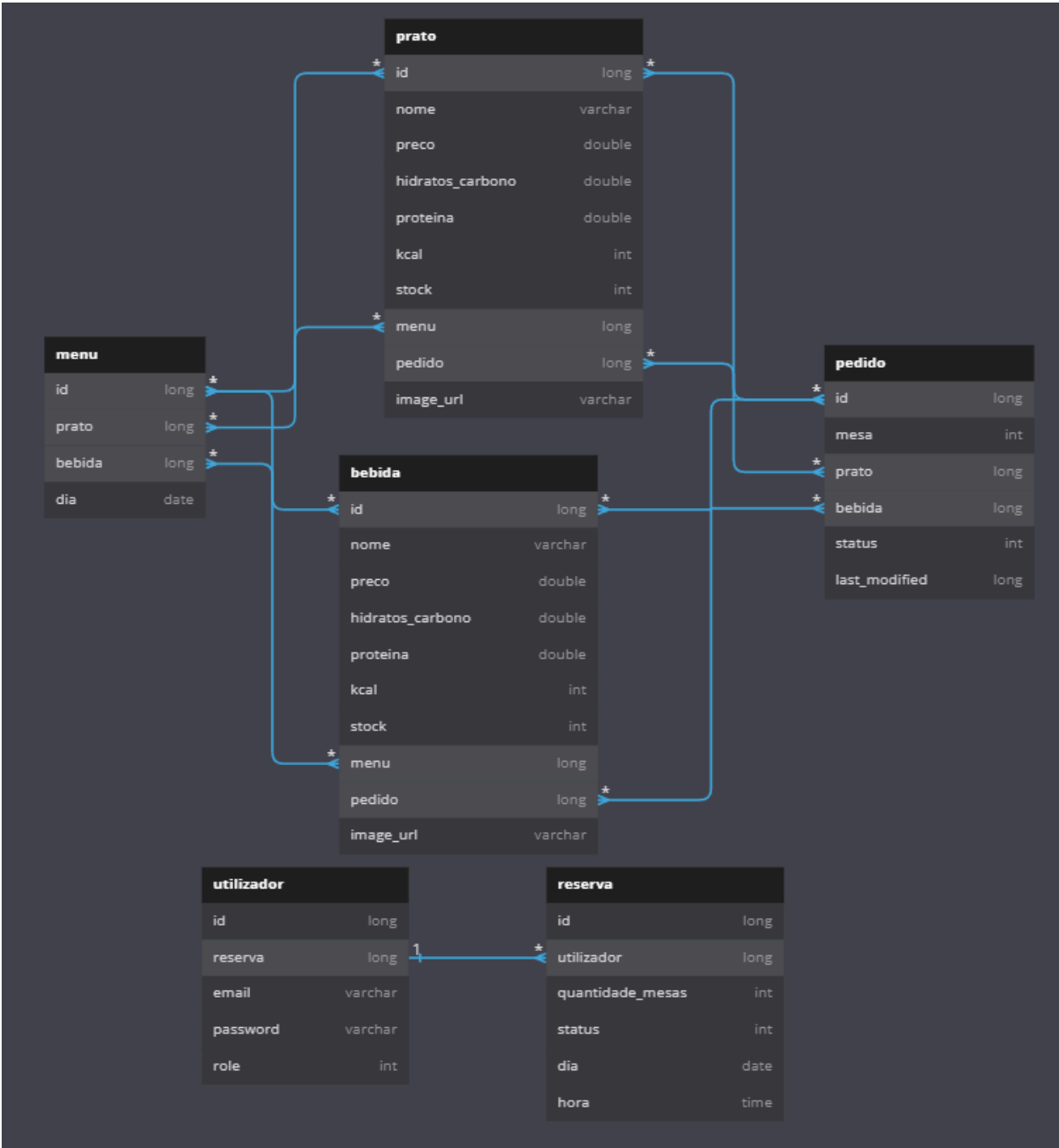
100% Done

 <b>RMO-47</b>	(frontend) install daysy ui	 		DONE ▾
 <b>RMO-27</b>	Adicionar repositórios	 		DONE ▾
 <b>RMO-28</b>	Adicionar dockers	 		DONE ▾
 <b>RMO-29</b>	Adicionar QA	 		DONE ▾
 <b>RMO-30</b>	Adicionar workflows	 		DONE ▾
 <b>RMO-53</b>	Adicionar proteção de branches no gith...	 		DONE ▾



Add a comment...

### 3 Domain model



### 4 Architecture notebook

#### 4.1 Key requirements and constraints

Due to the nature of our system, a small delay of 5 seconds between a request being shown on our digital display is not going to have a massive impact on the restaurant's flow due to it having being considered a non-fast-food restaurant, but a restaurant with an average of 150 daily customers.



This made it so choosing between web sockets and restful API calls was an easy task due to the following reasons:

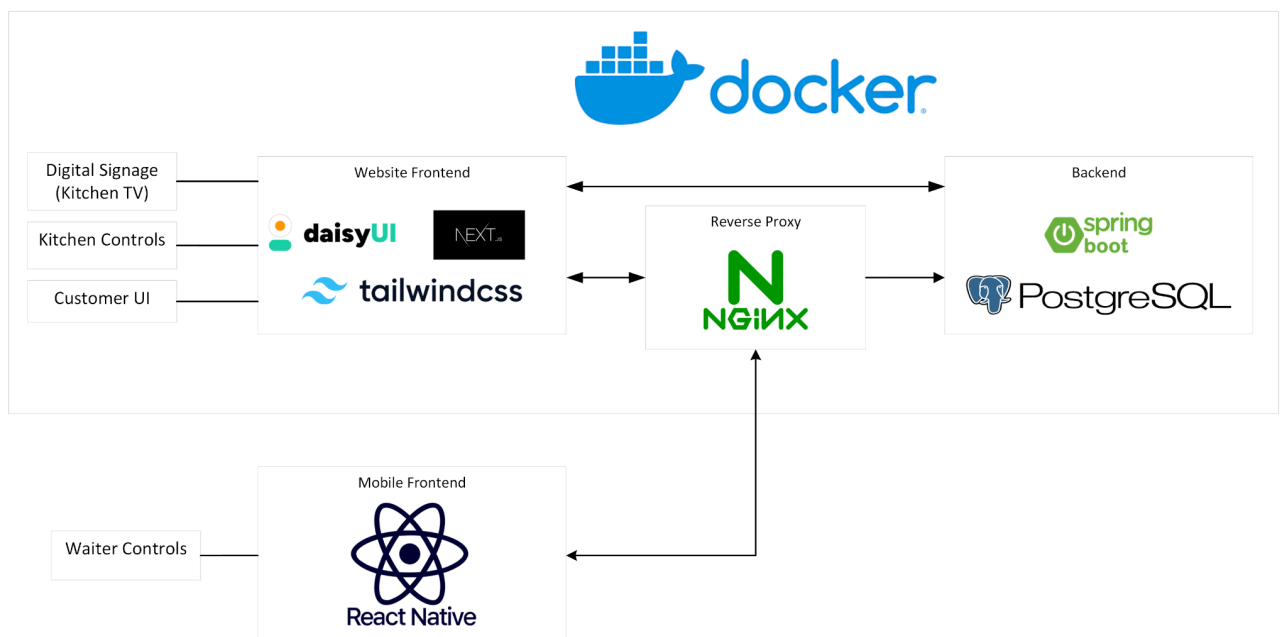
- Latency is not a concern (if there is a 5 seconds delay, no one will notice it)
- Resource efficient compared to websockets
- Simplicity of implementation

In the case of the waiter use cases we will need that these are executable with a mobile device like a phone or tablet, to make this possible we decided to build an android app using a mobile framework based on React Native.

To make all the services accessible with one port we will need to use Nginx as a reverse proxy.

## 4.2 Architecture view

Our architecture solution is divided into 3 main parts: the backend developed in java with springboot and using postgresSQL as the database, the Web Frontend developed in javascript with Next JS and the Mobile Frontend developed with React Native. The objective is to have both frontends use the information from the backend by fetching it with REST api calls. Both frontends should have access to the same information but we decided to create a mobile one since it would be more convenient for the waiter.



## 4.3 Deployment architecture

For deployment we utilize docker to dockerize our system and NGINX as a reverse proxy. Both the web frontend and the backend are inside the same docker compose allowing us to use docker's internal dns to make some calls since Node JS allows for server side rendering of pages. With this system we can easily deploy our system and customize it using environment files and variables.

## 5 API for developers

Every single endpoint, besides the authentication one, are mostly documented within our swagger docs, which can be accessed at <http://deti-tqs-01.ua.pt/swagger-ui/index.html>.

The most well documented endpoint is the Dish API as it has schemas and example values. To make the rest comply with that small changes would need to be done.

The only bits that aren't well documented are endpoints that require authentication. Even though it blocks access it does not explicitly say that it is a forbidden status.

Request API		
PUT	/api/requests/{id}	Update a given request
GET	/api/requests	Get list of all requests
POST	/api/requests	Create requests
GET	/api/requests/preparing	Get list of all preparing requests
GET	/api/requests/pending	Get list of all pending requests
Booking API		
PUT	/api/bookings/confirm/{id}	Confirm booking
PUT	/api/bookings/cancel/{id}	Cancel booking
GET	/api/bookings	Get all bookings from specific user
POST	/api/bookings	Post booking on selected date
GET	/api/bookings/pending	Get all pending bookings
GET	/api/bookings/availableSlots	Get available slots by date

<b>Authentication API</b>	
<b>POST</b>	<code>/api/authentication</code>
<b>Menu API</b>	
<b>GET</b>	<code>/api/menu</code> Get today's menu
<b>Dish API</b>	
<b>GET</b>	<code>/api/dishes/{id}</code> Get dish by ID
<b>Beverage API</b>	
<b>GET</b>	<code>/api/beverages/{id}</code> Get beverage by ID

## 6 References and resources

### JWT security

<https://www.toptal.com/spring/spring-security-tutorial>

<https://medium.com/@tericcabrel/implement-jwt-authentication-in-a-spring-boot-3-application-5839e4fd8fac>

<https://github.com/natanaelsc/spring-boot-3-jwt-security/tree/main/src/main/java/br/com/security/config>

### SL4J logging

<https://stackoverflow.com/questions/21881846/where-does-the-slf4j-log-file-get-saved>

### Http vs websockets

<https://blogs.windows.com/windowsdeveloper/2016/03/14/when-to-use-a-http-call-instead-of-a-websocket-or-http-2-0/>

<https://www.geeksforgeeks.org/difference-between-rest-api-and-web-socket-api/>

### Next js

<https://nextjs.org/docs>

### Expo

<https://docs.expo.dev/>