



universidade
de aveiro



TQS: Product Specification

Made by:

Guilherme Amorim, 107162;

Rafael Ferreira, 107340;

José Gameiro, 108840;

Ricardo Quintaneiro, 110056

Contents

1. Introduction	3
1.1 Overview of the project	3
1.2 Limitations	3
2. Product Concept	4
2.1 Vision statement	4
2.2 Personas	4
2.3 Main scenarios	5
2.4 Project epics and priorities	6
3. Domain Model	7
4. Architecture notebook	9
4.1 Key requirements and constraints	9
4.2 Architectural view	9
4.3 Deployment architecture	11
5. API for developers	12
6. References and resources	13

1. Introduction

1.1 Overview of the project

BeautyPlaza is a renowned elite beauty salon with a wide variety of products and services. Any client may book an appointment for services like hair care and washing, styling, combing, coloring, waving or hair straightening; Skin care, manicure and pedicure, and many, many others.

Due to the complicated nature of running, working and using such an intricate environment, we created the **BeautyPlaza Platform**. Consisting of five interconnected modules, the **BeautyPlaza Platform** tries to solve most of the problems faced by all the actors.

Two client applications, one web and one mobile, will allow customers to choose and schedule the desired service according to their wishes and needs. An administrator platform that allows swift handling of all the requests, as well as the management of client and personnel schedules. As well as a call screen, where users may see the current status of the queue, as well as when they get called in.

1.2 Limitations

- Web platform:
 - Reserve a service
 - Choose data
 - Choose speciality
- Mobile platform:
 - <Same as Web Platform>
- Admin platform:
 - Call next person
 - Assign an employee
- Call screen:
 - Visualise Reservations' Status

2. Product Concept

2.1 Vision statement

BeautyPlaza aims to revolutionize the beauty industry by providing a seamless platform that caters to the diverse needs of both clients and staff within beauty salons. Our goal is to enhance the overall experience by streamlining appointment scheduling, optimizing staff allocation, and facilitating transparent communication between all stakeholders.

The BeautyPlaza Platform serves as a comprehensive solution for managing the complexities inherent in beauty salons. It enables clients to effortlessly schedule appointments for a wide range of services, including hair care, skin care, manicures, and pedicures.

Our system facilitates efficient staff management by allowing them to view appointments, input specialties, and ensure optimal allocation for each client. Staff members are required to create accounts for seamless integration into the system, while administrators have the authority to manage employees.

Furthermore, a dedicated module provides real-time visibility into current and upcoming reservations and their status, (if they have already begun, are running late, have finished, etc), enhancing customer experience and operational efficiency.

2.2 Personas

Persona #1 - Guilherme

Name: Guilherme Cativeiro.

Age: 28.

Occupation: Lumberjack.

Background: Guilherme Cativeiro, a lumberjack for six years, meticulously organizes his life, maintaining a detailed schedule of weekly and monthly events. He likes to schedule a session to get his hair and beard cut one month in advance.

Persona #2 - Ângela

Name: Ângela Soares.

Age: 53.

Occupation: High School Teacher.

Background: Ângela teaches History and English in a small High School in Batalha, and she likes to get her nails done and curl her hair once a week, but she doesn't like that she has to go to 2 different places that can be far away from each and don't have availability on the same day.

Persona #3 - Bruno

Name: Bruno Duarte.

Age: 35.

Occupation: Hairdresser and barber.

Background: Bruno, works in a barber shop, where the reservations are not easy to manage because customers schedule a time slot but then they arrive late and, consequently, he has to work more hours than usual.

2.3 Main scenarios

Scenario #1 - Client making a reservation to a **simple** service

As a client, Guilherme wants to use the website to make an appointment to cut his hair on Saturday, 8th June 2024.

Scenario #2 - Client making a reservation to a **multiple** service

As a client, Ângela wants to use the website to make an appointment to get her nails done and curl her hair, on Friday, 7th June 2024.

Scenario #3 - Client waiting for his turn to be attended

As a client, Guilherme arrives at the hair saloon 5 minutes before his appointment and wants to watch the digital screen to check if the reservation is on time and which slot he will be attending.

Scenario #4 - Member of Staff calls next customer

As a member of staff, Bruno wants to allow the next customer in the queue to come to his chair so that he can perform his duties as a hairdresser. For that, he goes to a staff portal and clicks to call the next customer who is waiting for the service he is providing.

2.4 Project epics and priorities

Week 1:

- Define the product concept, personas and main scenarios
- Define epics
- Create product resources (repository and project backlog management system)
- Software/system architecture proposal

Week 2:

- Define the system architecture and start building SQE tools/practices
- Write product specification
- Start the UI prototype using Figma

Week 3:

- UI prototypes for the core user stories using target technologies
- Establish SQE strategy

Week 4:

- Implement key user stories
- Write QA manual
- Establish CI pipeline
- Make a 'live' call screen

Week 5:

- Develop user stories based on access and data persistence
- Fully develop API
- Set up a CD technology pipeline

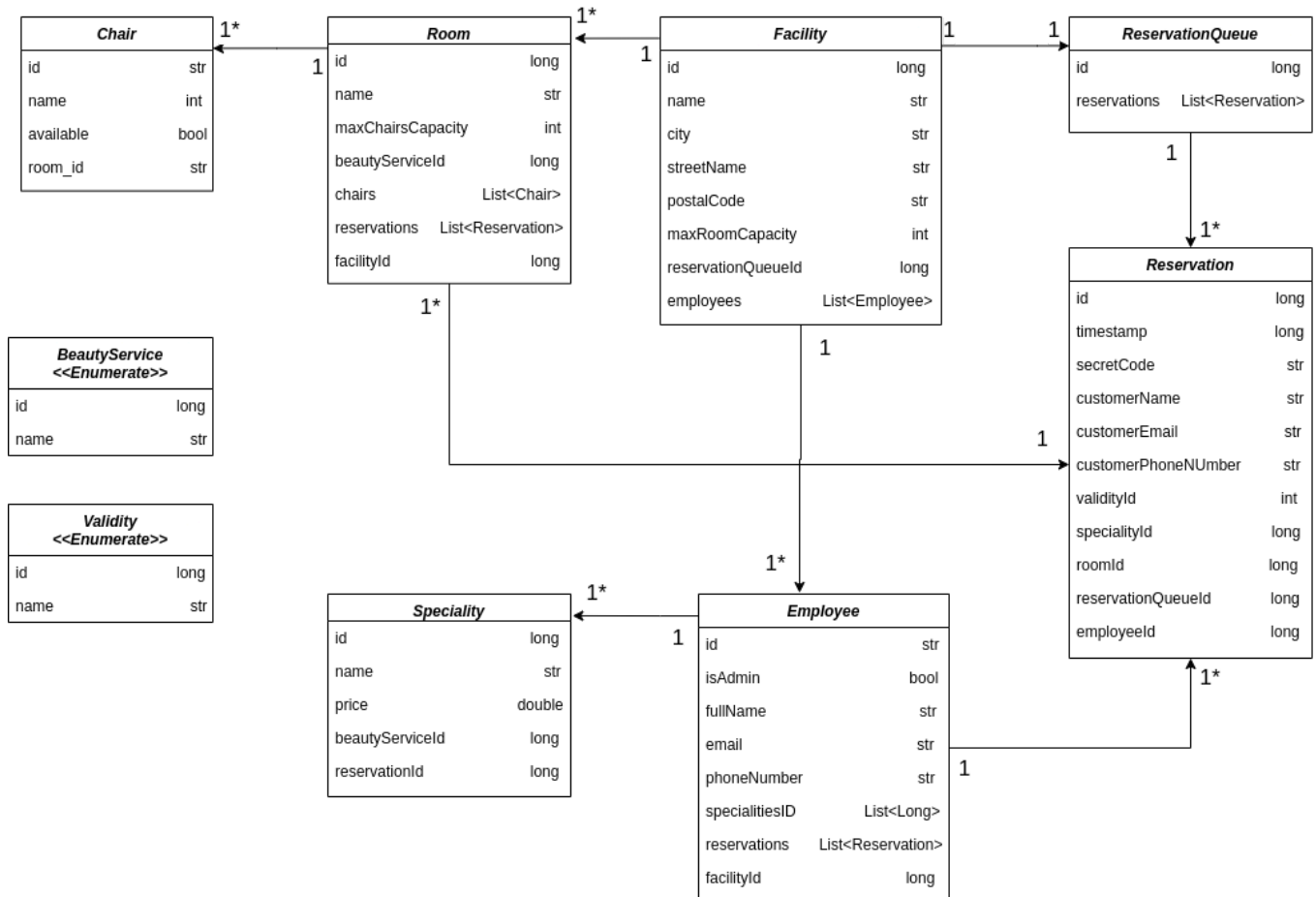
Week 6:

- Stabilize the Minimal Viable Product
- Deploy the MVP in a server
- Finish product Specification Report
- Perform non-functional tests and systems observability

Week 7:

- Final improvements for the last delivery
- Prepare the presentation of the project

3.Domain Model



Our domain model is composed by 9 entities:

- **Chair:** with the attributes id, name, available, room_id, it has a one to many relation with the Room entity, because a room can have multiple chairs. We define this entity with the purpose of when a customer checks in the facility and gets called by a customer he needs to go to a chair to get its service completed;
- **Room:** it has the following attributes: id, name, maxChairsCapacity, beautyServiceId, a list of chairs, a list of reservations and a facility Id. It has 2 one to many relations with the chair and reservation entity, because this entity can have multiple chairs and

reservations. And it also has another one to many relation with the facility entity cause a facility can have multiple rooms;

- **Facility:** this entity has the attributes id, name, city, streetName, postalCode, maxRoomsCapacity, reservationQueueId and a list of the employees that are working in that facility. It has two one to many relations with the Room and Employee entity, and a OneToOne relation with the ReservationQueue cause each facility has its own queue of reservations;
- **Reservation Queue:** it has the attributes id and a list with reservations, so it has a one to many relation with the Reservation Entity. We created this entity with the purpose of when the customers check in the facility it automatically saves their reservation in a queue so that it can be displayed in the digital signage;
- **Reservation:** it has the following attributes id, timestamp (the date that the customer selected), secret code (to get the reservation details), customerName, customerEmail, customerPhoneNumber, validityId, specialityId, roomId, reservationQueueId and employeeId;
- **Employee:** this entity has the attributes id, isAdmin, fullName, email, phoneNumber, a list with all the specialities Id's, a list with all the reservations assigned, a the id of the facility where the employee works;
- **Speciality:** it has the following attributes id, name, price, beautyServiceId and reservationId. An employee can have multiple specialities and a reservation has only one speciality associated;
- **Validity:** its an enumerate created it the different statuses that a reservation can have, which can be PENDING (the customer created a new reservation but it didn't made the checkin in the facility), CHECKED_IN(the user checked in in the facility for his/her's reservation), OCCURRING (the designated employee has called the customer and the reservation is now decorring), WAITING_PAYMENT(the reservation has ended and now all that's left is for the user to confirm the payment), PAYMENT_CONFIRMED(the reservation has ended);
- **BeautyService:** also an enumerate that we added that have the different areas for each speciality, that are, BASIC_HAIRDRESSER, COMPLEX_HAIRDRESSER, MAKEUP, DEPILATION, MANICURE_PEDICURE and SPA.

4. Architecture notebook

4.1 Key requirements and constraints

The project's architecture was designed with two key concepts in mind: responsiveness and performance, and future support for a mobile app. These considerations guided the selection of the architectural model and technologies. The team focused on ensuring that the system would be robust for long-term use, capable of handling high performance without issues, and adaptable to future enhancements, such as the integration of a mobile application. By prioritizing these aspects, the team aimed to future-proof the architecture, ensuring its scalability and longevity.

Additionally, while the current product does not operate with databases, backend, and frontend in separate locations, the architecture was designed with flexibility in mind. The possibility of future separation of these components was considered, leading to the adoption of a three-layered architectural approach. This approach not only allows for potential future distribution but also ensures modularity and ease of maintenance. The team also anticipated the need for future integrations with a mobile app, ensuring that the architecture could support multiple user-interfacing platforms, including web, mobile devices, and possibly larger screens. This forward-thinking design ensures that the system can efficiently adapt to various conditions and integrations, providing a solid foundation for future growth and enhancements.

4.2 Architectural view

Our architecture has three main components, each plays a critical role in ensuring the smooth operation and functionality of the platform:

1. Database

At the core of the system lies the main database, powered by PostgreSQL. This database serves as the centralized repository for storing crucial data related to clients, staff,

appointments, services, and more, ensuring data integrity and accessibility across the platform.

2. Backend

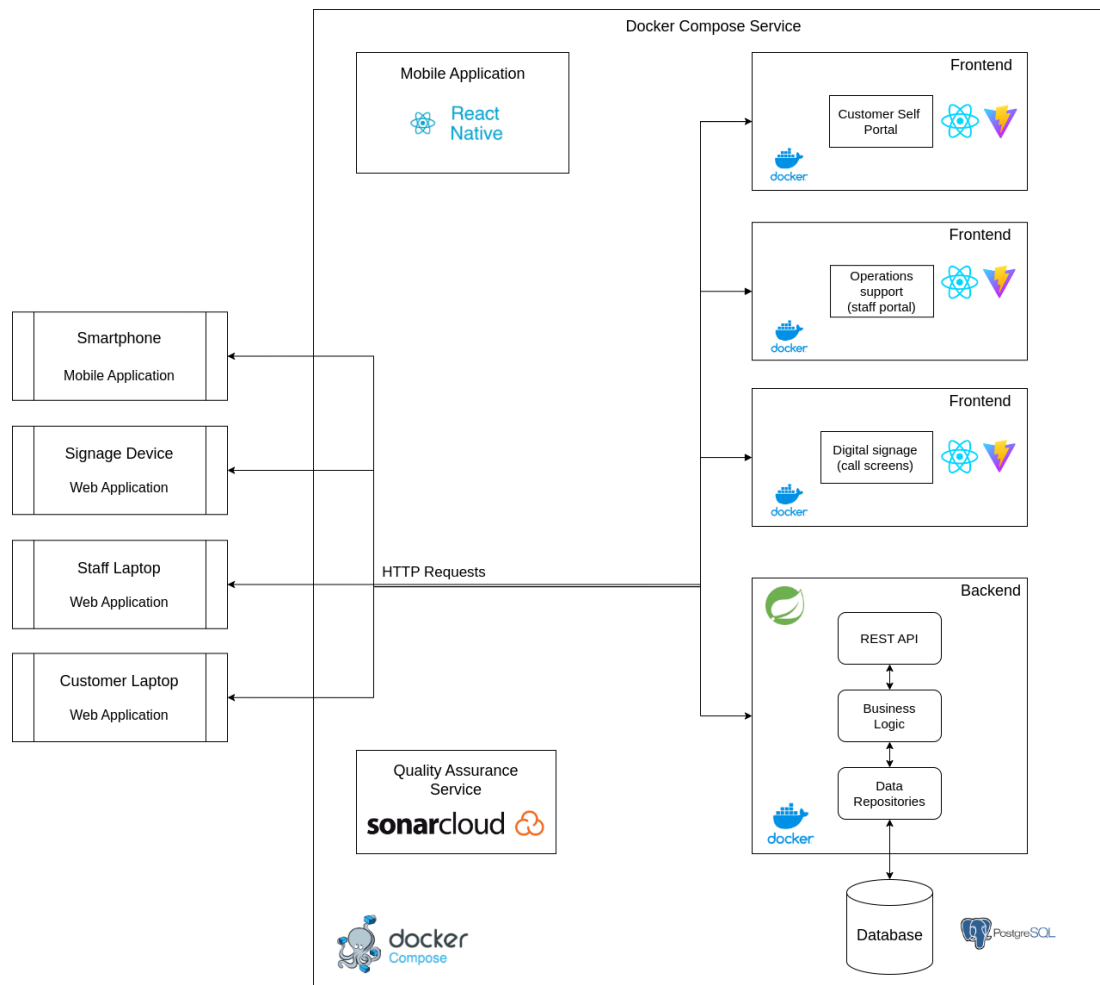
The backend of the BeautyPlaza system is developed using Spring Boot, a robust framework for building enterprise-grade applications. It consists of three key components:

- **REST API:** facilitates communication between the frontend and backend, allowing seamless data exchange and integration;
- **Business Logic:** Houses the core business logic of the application, including appointment scheduling, staff management, and client engagement;
- **Data Repositories:** Manages interactions with the main database, handling data retrieval, storage, and manipulation.

3. Frontend

The frontend encompasses four elements, each tailored to specific user roles and functionalities:

- **Customer Self Care Portal Web Application:** empowers clients to schedule appointments and view reservations through an intuitive web interface developed with React + Vite.
- **Operations Support Staff and Admin Portal:** provides staff and administrators with tools to manage appointments, allocate resources, and handle administrative tasks efficiently, utilizing React + Vite.
- **Digital Signage Screen:** displays real-time information on current reservations, enhancing customer experience and operational visibility within the salon environment. It is developed also using React + Vite.
- **Mobile Application:** Offers clients the flexibility to access BeautyPlaza services on the go, allowing appointment scheduling and account management through a user-friendly mobile interface developed with React Native.




4.3 Deployment architecture

Our deployment architecture is equal to the architecture explained above.

5.API for developers

Our API is documented using Swagger, ensuring that you have access to clear and comprehensive documentation. Swagger provides a user-friendly interface to explore and understand our API endpoints, making it easier for developers to integrate and work with our services efficiently.

Reservation Operations pertaining to reservations in the system.			^
POST	/api/reservation/pay/{reservationID}	Pay a reservation	▼
POST	/api/reservation/checkin/{reservationID}	Check-in a reservation	▼
POST	/api/reservation/add	Create a new reservation	▼
GET	/api/reservation/{reservationID}	Get a reservation	▼
GET	/api/reservation/employee/{employeeID}	Get reservations by employee ID	▼
GET	/api/reservation/customer/{email}	Get reservations by customer email	▼
GET	/api/reservation/code/{code}	Get a reservation by secret code	▼
GET	/api/reservation/all	Get all reservations	▼
Facility Operations pertaining to facilities in the system.			^
PUT	/api/facility/admin/update	Update a facility	▼
POST	/api/facility/admin/add	Create a new facility	▼
GET	/api/facility/{id}	Get a facility	▼ 
GET	/api/facility/all	Get all facilities	▼
DELETE	/api/facility/admin/delete	Delete a facility	▼
Room Operations pertaining to rooms in the system.			^
PUT	/api/room/admin/update	Update a room	▼
POST	/api/room/admin/add	Create a new room	▼
GET	/api/room/{id}	Get a room by ID	▼
GET	/api/room/search	Search for a room	▼
GET	/api/room/all	Get all rooms	▼
DELETE	/api/room/admin/delete	Delete a room	▼

Chair Operations pertaining to chairs in the system. ^	
POST	/api/chair/admin/add
GET	/api/chair/{id}
GET	/api/chair/all
DELETE	/api/chair/admin/delete

Employee Operations pertaining to employees in the system. ^	
POST	/api/employee/admin/delete/{employeeId}
POST	/api/employee/admin/add
GET	/api/employee/all

Speciality Operations pertaining to specialities in the system. ^	
POST	/api/speciality/admin/create
GET	/api/speciality/{id}

6. References and resources

<https://junit.org/junit5/>

<https://rest-assured.io/>

<https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html>

<https://tailwindcss.com/docs/installation>

<https://nextui.org/docs/guide/introduction>

<https://docs.getxray.app/display/XRAYCLOUD/Global+Settings%3A+API+Keys>

<https://github.com/SonarSource/sonarcloud-github-action>