

TQS: Product specification report

Gonçalo Ribau [119560], Bernardo Lázaro [119230], Rodrigo Costa [119585], Diogo Pinheiro [119907]

v2025-12-19

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

1 Introduction

This project is developed in the context of the TQS (Software Quality and Testing) course, whose main objective is to teach students how to design, implement, test and deploy software systems following modern Software Quality Assurance (SQA) and DevOps best practices.

1.1 Overview of the project

MEGA, or Multisport Equipment Gear App, is a sport gear rental app that provides the users with the ability to rent out their unused sport equipment for a fee, to many types of people in a customized system that addresses many of the problems in the tiring task of finding cheap, high quality gear to practice/perform with.

In this project we will develop an MVP to this system and test it thoroughly in order to minimize problems.

1.2 Project limitations and know issues

The external services, showed in the architecture in 4., were not possible to incorporate, since, besides being annoying to implement, do not make much of an impact in the MVP for the app. With this in mind, those services were both made to their simplest forms, static. Also, notifications, as we wanted it, wasn't implemented either due to time constraints and irrelevance for the MVP. Many

features considered insignificant for the MVP, and for evaluation of this course in general, have been given up on, in trade for completeness of work and refinement of main features.

1.3 References and resources

This project relies on a modern stack of open-source technologies and Software Quality Engineering (SQE) tools to ensure robustness and maintainability.

- **Core Frameworks and Libraries:**
 - **Backend:** Built with **Spring Boot** (Java), utilizing *Spring Web* for REST APIs, *Spring Data JPA* for database abstraction, and *Lombok* to reduce boilerplate code.
 - **Frontend:** Developed using **React** to provide a responsive user interface.
 - **Database:** Uses **PostgreSQL** for production data persistence and **H2 Database** for in-memory testing.
- **Quality Assurance and Testing Tools:**
 - **Unit & Integration Testing:** **JUnit 5** and **Mockito** are used for isolated logic testing.
 - **BDD/Acceptance Testing:** **Cucumber** is implemented to verify user stories against acceptance criteria, ensuring alignment with business requirements.
 - **Static Analysis & Coverage:** **SonarQube** enforces code quality gates, while **Jacoco** monitors code coverage, ensuring it meets the project's 80% threshold.
 - **Load Testing:** **k6** is used to assess system performance under load.
- **DevOps and Project Management:**
 - **CI/CD:** **GitHub Actions** manages the continuous integration pipeline, enforcing the "Definition of Done" before merging pull requests.
 - **Containerization:** **Docker** and **Docker Compose** are used to orchestrate the deployment of the application and database services.
 - **Management:** **Jira** and **Xray** are utilized for project tracking and test management.

2 Product concept and requirements

2.1 Vision statement

As mentioned before, the project is a peer-to-peer sports gear rental app that aims to provide fast cheap sports gear to regular customers and passive income to owners of said gear.

We plan on developing a smart search system so users can search for sport, date and location primarily, with a dynamic listing function with images, price, description, seasonal window and location. Some other important general features include a trust system with ratings and comments, and a payment system.

Our service is similar to GeerGarage, but for sports, and including Group Booking, feature that doesn't exist in this competitor. Other "competitors" are either marketplaces, for very niche products or were shut down for lack of use.

To build the foundation of the project, we essentially looked up popular peer to peer marketplaces, for example Vinted, and gathered some cool practices they have on their websites/apps.

2.2 Personas and scenarios

2.2.1 Scenario 1: Alberto – "Turning Dust into Income"

Persona: Alberto Pais (The Owner); **Core Value:** Passive income with security and control.

The Situation: It is a Tuesday evening. Alberto has just returned from coaching his youth football team. As he parks his car in the garage, he squeezes past his high-end mountain bike and a stack of surfboards he hasn't used since last summer. He feels a bit guilty that this expensive gear is gathering dust, but he is hesitant to sell it because he might want to use it again. He wants to monetize it but is worried about strangers damaging his prized possessions.

The Interaction: He takes out his phone and opens MEGA. He decides to list the surfboard. He snaps a few photos right there in the garage and uploads them.

1. **Seasonal Control:** Since he knows he might want to surf in July, he sets the availability to "Weekends only" and blocks out his own vacation dates in July using the **Seasonal Availability** calendar.
2. **Safety First:** He toggles the "Require Condition Checklist" option and sets a buffer time of 48 hours between bookings so he has time to inspect the gear.

The Outcome: Two days later, while at work, he gets a notification. A renter with a 5-star rating wants the board for the weekend. Alberto checks the renter's profile and accepts with one tap. He feels secure knowing the platform handles the deposit and condition checks.

2.2.2 Scenario 2: Eva – "The Friday Escape"

Persona: Eva Martins (The Spontaneous Renter); **Core Value:** Speed and location-based convenience.

The Situation: It is Friday at 2:00 PM. Eva is finishing up her remote work for the week. Her friends text her: "The weather is perfect, let's meet at Barra beach in an hour for a paddle session." Eva is excited but realizes she doesn't have a paddleboard, and the rental shops at the beach are usually sold out on sunny Fridays or have long queues.

The Interaction: She doesn't have time to wait for an owner to "approve" a request. She opens MEGA and:

1. **Search:** Filters by "Stand Up Paddle" and her current location.
2. **Instant Access:** She applies the "Instant Book" filter to see only items that don't require manual approval.
3. **Booking:** She finds a board listed by a nearby owner, reviews the specs, and books it immediately.

The Outcome: She receives an immediate confirmation with the pickup address (3 streets away). She picks up the board on her way to the beach. By 3:30 PM, she is in the water with her friends. No queues, no stress.

2.2.3 Scenario 3: Sara – "The Guaranteed Group Trip"

Persona: Sara Coutinho (The Planner); **Core Value:** Logistics management and reliability.

The Situation: Sara is organizing a hiking and cycling trip to Gerês for her group of 4 friends, scheduled for two months from now. She is the "planner" of the group and is stressed about logistics. She needs to ensure that when they arrive, there are exactly 4 mountain bikes available. She cannot risk arriving and finding only 3 bikes, or finding that the shop is closed.

The Interaction: She sits down at her laptop with her digital calendar open.

1. **Future Planning:** She searches for bikes in Gerês for specific dates weeks in advance.
2. **Group Logistics:** Instead of booking one by one, she uses the **Group Booking** feature. She selects 4 bikes from the same owner (or compatible nearby owners) and adds them to a single cart.
3. **Verification:** She checks the total price, including insurance for the whole group, and submits the request.

The Outcome: The system confirms availability for all 4 items simultaneously. She pays once. She receives a "Group Trip Dashboard" link which she shares with her friends, so everyone knows the gear is secured. She can relax until the trip.

2.2.4 Scenario 4: João – "The Low-Cost Trial"

Persona: João Ribeiro (The Beginner); **Core Value:** Affordability and guidance.

The Situation: João has been invited to play Padel for the first time. He is a student on a tight budget and sees rackets online costing €150+. He doesn't want to invest that money without knowing if he even likes the sport.

The Interaction: He opens MEGA looking for a cheap solution.

1. **Discovery:** He searches for "Padel Racket" and sorts by "Price: Low to High".
2. **Trust:** He finds a racket for €5/day. Being a beginner, he doesn't know if it's a good brand, so he reads the **Reviews** left by other renters to ensure it's not broken or too heavy.
3. **Local Pickup:** He sees the owner is a fellow student at his university campus.

The Outcome: He requests the rental for a single day. He meets the owner on campus, plays his game, and returns it. He spent €5 instead of €150, satisfying his need to "try before committing".

2.3 Project epics and priorities

Our development roadmap prioritizes the "Critical Path" of the application. The goal is to establish a functional skeleton early, allowing for immediate testing of the core business logic before introducing complexity. The implementation is divided into three logical iterations:

Iteration 1: The Core Marketplace Loop (Functional Foundation)

Target Epics: 1 (Discovery), 2 (Management), 3 (Booking).

- **Objective:** To enable the fundamental interaction between the two key actors: an Owner listing an item and a Renter finding and requesting it.

- **Justification:** These Epics represent the minimum requirements for the system to exist. By implementing **Search (Epic 1)**, **Equipment Listing (Epic 2)**, and the **Basic Booking Flow (Epic 3)** first, we ensure the backbone of the application is stable. This allows testing to begin immediately on the most frequently used features (availability checks and state changes) without being blocked by peripheral features.

Iteration 2: Value Differentiators & Security

Target Epics: 4 (Payments), 6 (Group Booking), 8 (Authentication).

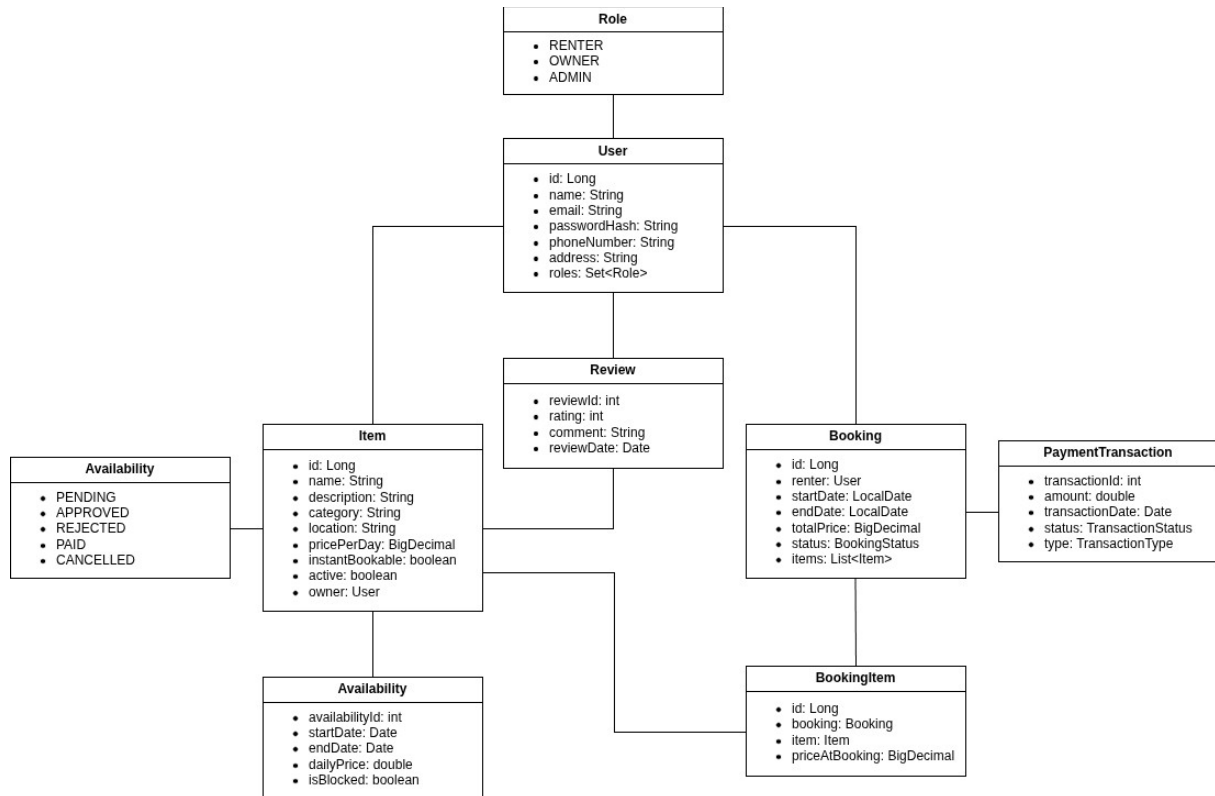
- **Objective:** To transform the prototype into a viable product by securing access, monetizing transactions, and implementing MEGA's unique selling point.
- **Justification:** Once the core loop works, we introduce:
 - **Group Bookings (Epic 6):** Our main competitive advantage (Persona Sara's need), allowing complex logistics.
 - **Authentication (Epic 8):** Robust security, role management, and user profiles are integrated here to secure the booking process.
 - **Payments (Epic 4):** Closing the loop by handling deposits and transactions.

Iteration 3: Trust, Operations & Polish

Target Epics: 5 (Reviews), 7 (Notifications), 9 (Administration).

- **Objective:** To enhance user retention, trust, and operational oversight.
- **Justification:** These are "quality of life" and trust features. **Reviews & Condition Checks (Epic 5)** are critical for long-term trust but require completed bookings to function. **Notifications (Epic 7)** streamline communication, and the **Admin Dashboard (Epic 9)** provides the necessary oversight for platform managers to monitor the system once it goes live.

3 Domain model



The domain model represents the core business entities present in our app. The system will be centered around three primary pillars: Users, Items and Bookings.

- **User**: Acts as both Renter and Owner, possessing specific **Roles** (RENTER, OWNER, ADMIN) and credentials.
- **Item**: Represents sports gear with attributes for description, price, and state (active, instantBookable). It belongs to a specific Owner (**User**).
- **Booking & BookingItem**: To enable Group Bookings, reservations are split into a **Booking** (transaction details, dates, total price) and **BookingItems** (list of specific items).

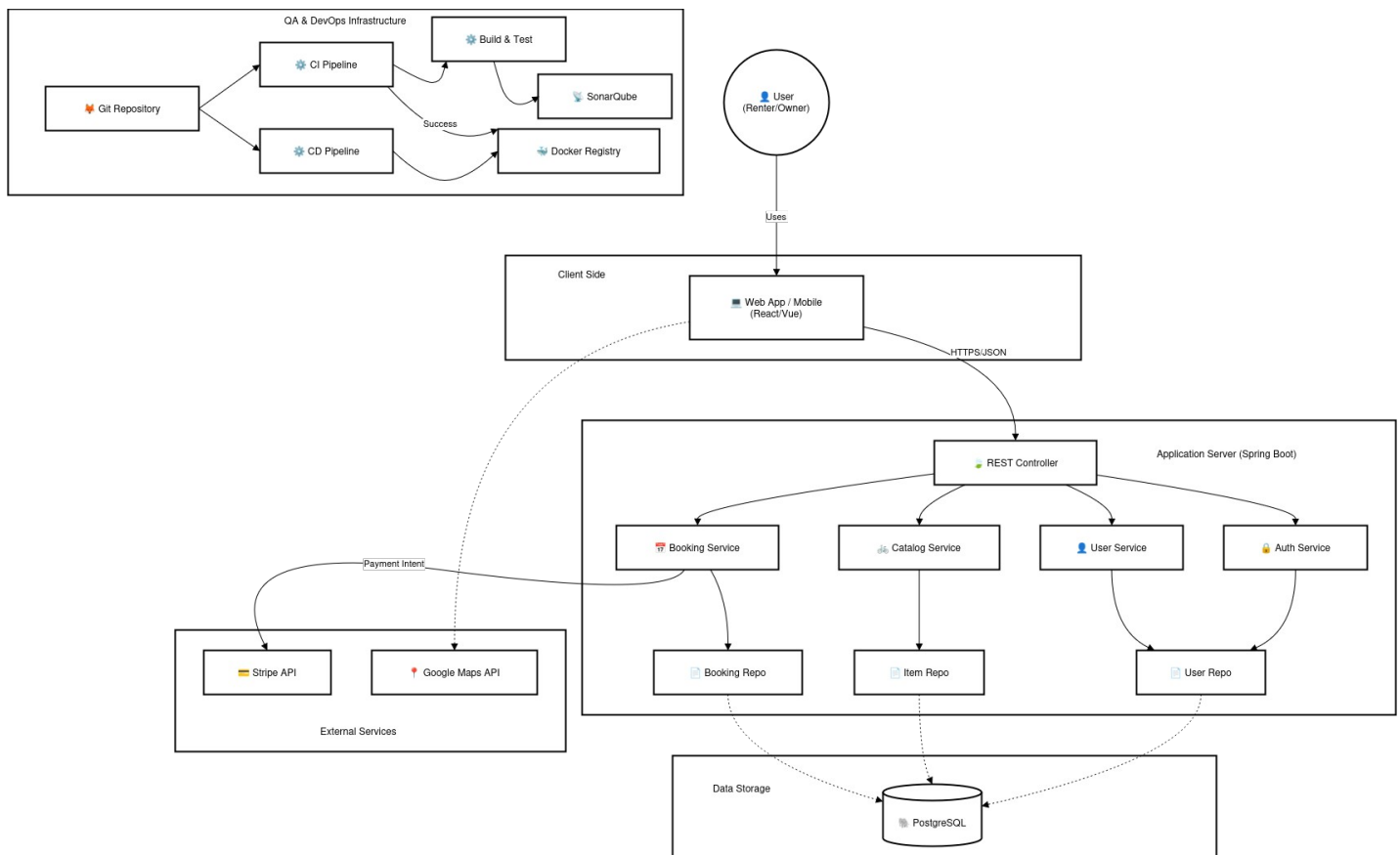
4 Architecture notebook

4.1 Key requirements and constraints

The system, as a requirement, will be designed with testability in mind, so we have opted for loose coupling for effective mocking. Besides that, Usability and Responsiveness should be cared for as well so we plan on making a dynamic frontend with real-time interaction on many fields like searching and booking. Data Integrity and Security are also top priority, as we will keep consistency on the business logic and plan on implementing a role-based access system.

With that, some defined constraints include being mandatory the system as a web application based on PostgreSQL for databases and docker for containerization.

4.2 Architecture view



The MEGA application utilizes a standard Client-Server Layered Architecture. This ensures separation of concerns and ease of implementation and maintainability.

- **Presentation Layer:** A React based web app.
- **API Layer:** Exposes RESTful endpoints.
- **Business Logic Layer:** Encapsulates the core logic of the system.
- **Data Access Layer:** Manages the interactions with the PostgreSQL database.

Advanced app design issues:

- **Integration with External Services:** Instead of building complex functionalities from scratch, to maintain the MVP lightweight and deliver the core features, we opted for integrating external services (Mapping for location and Payment for, well, payments).
- **Complex Workflows:** The Group Booking, for example, introduces a complex workflow that will be managed with a transactional state machine.
- **Deployment:** Whenever a new version is pushed to the main branch, the Docker containers are rebuilt and redeployed.

4.3 Deployment view (production configuration)

The application will be designed to be deployed using Docker containers to ensure consistency between development, testing and production environments. It will, essentially, consist on 3 containers, each hosting a different part of the project. Containers for:

- **Database:** A container for the PostgreSQL Database so we have data persistence.
- **Backend service:** To host the business logic and have it up and running in sync with the other parts. A Java/Spring Boot Application, dependent on the Database.

- **Frontend Service:** To give a visual for the project, sending API requests to the Backend container's address. Built in React (Node.js/Nginx)

5 API for developers

payment-controller ^	
POST	/api/payments/pay
item-controller ^	
GET	/api/items
POST	/api/items
GET	/api/items/{id}
DELETE	/api/items/{id}
POST	/api/items/{id}/block
PATCH	/api/items/{id}/price
POST	/api/items/{itemId}/reviews
GET	/api/items/owner/{ownerId}
booking-controller ^	
POST	/api/bookings
PATCH	/api/bookings/{id}/accept
PATCH	/api/bookings/{id}/decline
PATCH	/api/bookings/{id}/status
GET	/api/bookings/owner/{ownerId}
GET	/api/bookings/renter/{renterId}
auth-controller ^	
POST	/api/auth/login
POST	/api/auth/register

Disclaimer: This documentation from Swagger is present in “<http://deti-tqs-17.ua.pt:8080/docs>”.

In our System, we have, as said before, the three main pillars of the project (User, Item and Booking), so our endpoints revolve around them. Item's endpoints, as seen in the picture above, serve to manipulate the items for rent in the application. The Booking's endpoints, contrary to the previous example, serve to manipulate the bookings, or, the RENTED items in the application. Authentication, as simple as it sounds, is merely to register and log into the app.