TQS: Product specification report

Ana Alexandra Antunes [876543], Bruno Bernardes [876543]

v2025-11-25

[1 Introduction 1](#_Toc213750830)

[1.1 Overview of the project 1](#_Toc213750831)

[1.2 Known limitations 1](#_Toc213750832)

[1.3 References and resources 2](#_Toc213750833)

[2 Product concept and requirements 2](#_Toc213750834)

[2.1 Vision statement 2](#_Toc213750835)

[2.2 Personas and scenarios 2](#_Toc213750836)

[2.3 Project epics and priorities 2](#_Toc213750837)

[3 Domain model 3](#_Toc213750838)

[4 Architecture notebook 3](#_Toc213750839)

[4.1 Key requirements and constrains 3](#_Toc213750840)

[4.2 Architecture view 3](#_Toc213750841)

[4.3 Deployment view 3](#_Toc213750842)

[5 API for developers 3](#_Toc213750843)

[This report is the main source of technical documentation on the project, clarifying the functional scope and architectural choices. Provide concise, but informative content, **allowing other software engineers to understand the product**.

Tips on the expected content placed along the document are meant to be removed!

You may use English or Portuguese; do not mix.]

# Introduction

## Overview of the project

<contextualize the objectives of this project assignment in the scope of the TQS course>

<introduce your application/product: brief overview of the solution concept. What is it good for? For whom? Introduce the name of the product if it has one>

O projeto é desenvolvido no âmbito da unidade curricular **Teste e Qualidade de Software (TQS)**, que tem como objetivo aplicar princípios de engenharia de software orientados para qualidade.

A aplicação desenvolvida consiste numa plataforma de **aluguer de fatos e peças de vestuário formal**, permitindo que utilizadores disponibilizem ou aluguem peças de forma segura e prática.

## Project limitations and know issues

 <explain the known limitations, especially the **features that were planned/expected but not implemented** (and why…)

**To be reviewed and completed by the end of the project** >

## 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>

Entre os recursos utilizados no projeto destacam-se:

* **Spring Boot** (backend)
* **React + TypeScript** (frontend)
* **PostgreSQL** (base de dados)
* **Stripe API** (pagamentos e cauções)

# Product concept and requirements

## Vision statement

<functional (black-box) description of the application: Which is the high-level/business problem being solved by your system? Which are the key features you promise to address it?>

<if needed, clarify what was planned/expected to be included but was changed to a different approach/concept >

<optional: how is your system different or similar to other well-known products?>

<optional: additional details on the process for the requirements gathering and selection (how did we developed the concept? Who helped us with the requirements? Etc)>

A plataforma tem como propósito p**roporcionar um sistema seguro e acessível que permita o aluguer rápido e eficiente de fatos e peças formais, garantindo confiança e simplicidade para Owners e Renters.**

O sistema resolve diversos problemas como a falta de necessidade de comprar fatos de unico uso, a rentabilização de roupas que ninguém compra em loja e o consumo sustentável de roupa.

O serviço deve ser capaz de listar e publicar peças, pesquisar, filtrar, reservar com datas, permitir pagamentos, impor cauções, avaliar utilizadores e roupa, monitorar utilizadores e gerir disputas.

## Personas and scenarios

<“Personas are fictional people. They have names, likenesses, clothes, occupations, families, friends, pets, possessions, and so forth. They have age, gender, ethnicity, educational achievement, and socioeconomic status. They have life stories, goals and tasks. Scenarios can be constructed around personas, but the personas come first. They are not ‘agents’ or ‘actors’ in a script, they are people. Photographs of the personas and their workplaces are created and displayed. […] It is to obtain a more powerful level of identification and engagement that enable design, development, and testing to move forward more effectively”. Adapted from Grudin, J. and Pruitt, J., 2002, June. Personas, participatory design and product development: An infrastructure for engagement. In Proc. PDC (Vol. 2).

Sample personas: [secção 4.1, neste artigo](http://www.mdpi.com/1424-8220/18/4/1285) (open access)] >

<Develop one or more representative scenarios for each persona. You don’t need to include all possible details. Pick the main scenarios, related to the core value of the system.>

<The scenarios tell the story of the Personas in their lives, doing their daily/professional activities that are relevant to find the points of contact with the system under specification.

Scenarios are somewhat similar to use cases (they have a goal and tell a story), but, unlike use cases, they capture a larger process, with activities that may not use the software. Scenarios don’t require a “template”, like the usual use cases description.>

Sample: [secção 4.2 neste artigo](http://www.mdpi.com/1424-8220/18/4/1285) (open access)] >

### **Persona 1 – João Ribeiro (Renter)**

**Idade:** 32  
**Profissão:** Consultor empresarial  
**Localização:** Porto  
**Nível Digital:** Médio

João participa frequentemente em conferências e eventos corporativos, onde necessita de uma apresentação profissional. Considera pouco sustentável comprar fatos caros que usa poucas vezes. Procura eficiência, rapidez e confiança no processo.

**Motivações:** ter acesso a fatos elegantes sem comprar; processo simples.  
**Frustrações:** tamanhos inadequados; baixa disponibilidade.  
**Objetivos:** descobrir peças rapidamente; entrega e devolução sem complicações.

**Persona 2 – Marta Faria (Owner)**

**Idade:** 29  
**Profissão:** Criadora de conteúdos  
**Localização:** Braga  
**Nível Digital:** Elevado

Marta tem um guarda-roupa extenso com peças formais usadas raramente. Vê na plataforma uma oportunidade de gerar rendimento e ao mesmo tempo promover moda sustentável.

**Motivações:** rentabilizar peças; ajudar outros utilizadores.  
**Frustrações:** danos nas peças; atrasos nas devoluções.  
**Objetivos:** publicação fácil; controlo de disponibilidade; proteção das peças.

### **Persona 3 – Pedro Duarte (Owner)**

**Idade:** 35  
**Profissão:** Costureiro / dono de loja de moda masculina  
**Localização:** Coimbra  
**Nível Digital:** Médio

Pedro possui vários fatos premium na sua loja que não têm muita saída. O aluguer permite rentabilizar o inventário com um risco controlado.

**Motivações:** rentabilizar fatos premium.  
**Frustrações:** comportamentos irresponsáveis de renters.  
**Objetivos:** garantir alugueres seguros com controlo de reputação.

### **Persona 4 – Carolina Mendes (Admin)**

**Idade:** 38  
**Profissão:** Gestora de Operações  
**Localização:** Lisboa  
**Nível Digital:** Elevado

Carolina é responsável por manter a plataforma segura, justa e estável. Supervisiona disputas, comportamentos suspeitos e incidentes operacionais.

**Motivações:** garantir segurança; resolver disputas de forma imparcial.  
**Frustrações:** falta de evidências; fraudes; necessidade de intervenções manuais.  
**Objetivos:** aplicar regras, gerir utilizadores e manter a credibilidade do sistema.

### **Cenário 1 – João aluga um fato para uma conferência**

**Ator:** João (Renter)  
**Motivação:** precisa de um fato para um evento profissional sem o comprar.  
**Ação:**

1. Acede à plataforma.
2. Pesquisa por fatos e filtra por tamanho e estilo.
3. Seleciona um fato de Pedro.
4. Analisa fotos e descrição.
5. Reserva a peça para um período de 7 dias.
6. Realiza o pagamento.
7. Recebe e utiliza o fato.
8. Devolve dentro do prazo.
9. Avalia o Owner.

**Resultado:** João obtém uma peça adequada sem custos elevados e com um processo simples.

### **Cenário 2 – Marta publica novas peças no catálogo**

**Ator:** Marta (Owner)  
**Motivação:** rentabilizar peças formais pouco usadas.  
**Ação:**

1. Inicia sessão no painel de Owner.
2. Carrega fotos da peça.
3. Adiciona detalhes como tamanho, cor, estilo e ocasião.
4. Define preço e disponibilidade.
5. Publica no catálogo.

**Resultado:** a peça fica imediatamente disponível para aluguer.

### **Cenário 3 – Pedro gere pedidos de aluguer**

**Ator:** Pedro (Owner)  
**Motivação:** garantir que renters são responsáveis.  
**Ação:**

1. Recebe um pedido de aluguer.
2. Analisa avaliações e reputação do renter.
3. Aprova o pedido.
4. Organiza entrega com a transportadora.
5. Confirma o estado da peça após devolução.

**Resultado:** aluguer concluído de forma segura e sem danos.

### **Cenário 4 – Disputa sobre dano numa peça**

**Ator:** Owner e Admin  
**Motivação:** proteger o valor das peças e garantir justiça.  
**Ação:**

1. Owner reporta dano após devolução.
2. Admin analisa fotos, histórico e evidências.
3. Admin decide aplicar caução, penalização ou compensação.
4. Renter é notificado.

**Resultado:** disputa resolvida de forma clara, documentada e imparcial.

### **Cenário 5 – Admin monitoriza atividade**

**Ator:** Admin  
**Motivação:** manter a plataforma livre de abuso ou fraude.  
**Ação:**

1. Revê relatórios e sinais de comportamento anómalo.
2. Toma medidas quando necessário (avisos, suspensões, bloqueios).

**Resultado:** plataforma permanece estável, segura e credível.

## Project epics and priorities

**[**Apresentar um plano indicativo para a implementação incremental da solução ao longo de várias iterações/releases, explicando as funcionalidades a atingir por [*epics*](https://www.atlassian.com/agile/project-management/epics-stories-themes) ]

Os principais épicos são:

1. **Gestão de Utilizadores**  
   Registo, login, papéis, perfis, reputação.
2. **Gestão de Peças (Catálogo)**  
   Publicação, edição, remoção, fotos, filtros e pesquisa.
3. **Reservas e Logística**  
   Datas, códigos de entrega, confirmações, devoluções.
4. **Pagamentos e Cauções**  
   Pagamentos, reembolsos, retenção de caução, integração Stripe.
5. **Avaliações e Reputação**  
   Avaliação de Owners, Renters e peças.
6. **Gestão de Disputas**  
   Evidências, decisões, penalizações.
7. **Administração da Plataforma**  
   Monitorização, controlo de utilizadores, revisão de incidentes.
8. **Dashboards e Estatísticas**  
   Ganhos, histórico, indicadores e recomendações.

# Domain model

<which information concepts will be managed in this domain? How are they related?>

<use a logical model (UML classes) to explain the concepts of the domain and their attributes, not a entity-relationship relational database model>

# Architecture notebook

## Key requirements and constrains

<**Identify issues that will drive the choices for the architecture** such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions? Are there integrations with external systems? Is the system to be offered in different user-interfacing platforms (web, mobile devices, big screens,…)?

For a more systematical approach:

* Note the collection of [Architectural Characteristics](https://learning.oreilly.com/library/view/fundamentals-of-software/9781098175504/ch04.html) the software architect should be aware
* [Identify architectural characteristics](https://learning.oreilly.com/library/view/fundamentals-of-software/9781098175504/ch05.html) that are relevant for your project (will drive the key design decisions). Note the [case study](https://learning.oreilly.com/library/view/fundamentals-of-software/9781098175504/ch05.html" \l "ch-identifying-ss) and the explicit characteristics related to users and extensibility. This will support later non-functional tests.

## Architecture view

→ Discuss architecture planned for the software solution: what are the main building blocks? [include a diagram](https://learning.oreilly.com/library/view/fundamentals-of-software/9781098175504/ch23.html) (a logical view, such as a package or block diagram). Avoid implementation technology or deployment references, but protocols/standards can be included.

→ refer to the [architecture style](https://learning.oreilly.com/library/view/fundamentals-of-software/9781098175504/part02.html) applied, if any

explain how the identified modules will interact. Use a sequence diagram to clarify the interactions along time, when needed

→ discuss more advanced app design issues: integration with Internet-based external services, data synchronization strategy, distributed workflows, push notifications mechanism, distribution of updates to distributed devices, etc.>

## Deployment view (production configuration)

[Explain the planned organization of the solution in terms of production configuration (deployment). Note the implementation technologies in the diagram, e.g.: place the PostgreSQL symbol in the Database, etc.]. Indicate the existence of containers (Docker), IP addresses, and ports, etc.

This part will be completed when deployments are actually in place.

# API for developers

[Explain the API organization and main collections in general terms. Details/documentation of methods should be included in a hosted API documentation solution, such as Swagger, Postman documentation, or included in the development itself (e.g., Maven site).

Be sure to use [best practices for REST Api design](https://learning.oreilly.com/library/view/mastering-rest-apis/9798868803093/html/617304_1_En_7_Chapter.xhtml). Keep minda REST API applies a resource-oriented design (APIs should be designed around resources, which are the key entities your application exposes, not actions)