TQS: Product specification report

***Daniel Pinto [89341], Diogo Andrade [89265], João Dias [89236], Rodrigo Oliveira [90514]***

v2020-06-05

[1 Introduction 1](#_Toc39943375)

[1.1 Overview of the project 1](#_Toc39943376)

[1.2 Limitations 1](#_Toc39943377)

[2 Product concept 2](#_Toc39943378)

[2.1 Vision statement 2](#_Toc39943379)

[2.2 Personas 2](#_Toc39943380)

[2.3 Main scenarios 2](#_Toc39943381)

[2.4 Project epics and priorities 2](#_Toc39943382)

[3 Domain model 2](#_Toc39943383)

[4 Architecture notebook 3](#_Toc39943384)

[4.1 Key requirements and constrains 3](#_Toc39943385)

[4.2 Architetural view 3](#_Toc39943386)

[4.3 Deployment architeture 4](#_Toc39943387)

[5 API for developers 4](#_Toc39943388)

[6 References and resources 4](#_Toc39943389)

# Introduction

## Overview of the project

As the final group project in the TQS course, students were asked to specify and implement a medium-sized project, comprising a multi-layered, enterprise web application, more specifically an online marketplace, and a Software Quality Assurance (SQA) strategy, applied throughout the software engineering process.

The application we decided to create is called MovieGoodies and it is, in short, an online marketplace for merchandise related to movies. We intended the application to provide a centralized service for individuals and small businesses without a prominent online presence to sell their movie related merchandise and in turn also provide buyers with a more comprehensive catalog of merchandise where they can find what they’re looking for without having to trawl several websites.

## Limitations

Due to the limited scope and timeframe in which to develop this project, any kind of actual integration with payment methods had to be scrapped. We meant to include integration with PayPal and implementing safeguards for the payment process but had to forego that in benefit of other more key features.

A more in-depth movie database integration was also in order but the time needed to implement it ended up forcing us to abandon most of it.

# Product concept

## Vision statement

As mentioned previously, MovieGoodies is an online marketplace for merchandise related to movies. Our target users are both people looking to buy, and sell, merchandise related to movies. More specifically, the aim was to provide a one stop shop for buyers to look for merchandise, and a simplified approach to selling, where we were aiming to bring the buyers directly to relevant sellers of movie merchandise, thus giving sellers without an online platform an opportunity to reach a bigger client base.

The problem we were seeking to solve was the way sellers of movie memorabilia tend to be individuals or small businesses, who do not see a significant advantage to their business in creating their own online sales platform. Many seek online marketplaces like eBay or OLX to advertise their products for sale, but these platforms are poorly organized in some cases or just too generalized to cater for a niche market like this one, making it hard for the sellers to reach the clients and for the clients to find specific products.

Initially we thought of a simpler approach to the problem, letting everyone who wanted to setup an account in order to sell products but we found that this would be too hard to moderate in a real life scenario so we took a register and verification approach where sellers are only able to post any listings after they apply for it in our platform and are approved by the system admin.

## Personas

*Persona 1*

Diogo is a 26-year-old man who works in a clothing store in Glicinias Plaza. Diogo lives alone in a small apartment in Aveiro.

*Persona 2*

Ana is a 21-year-old woman studying biology in the University of Aveiro. Currently she is studying and working from home and has online classes.

*Persona 3*

Rui is a 28-year-old man who lives in Lisbon with his girlfriend. Rui runs a movie merchandise store in the center of Lisbon.

## Main scenarios

*Persona 1*

One of Diogo’s many passions is cinema. When he is able, he likes to go to the movie theater and watch the latest projects. Being as passionate as he is, Diogo finds himself buying a lot of movie merchandise to the point of almost not having space to store it. To solve this problem, he often tries to sell pieces that he doesn’t wear no more, but this process can be tedious as he has to find someone interested to buy them.

**Motivation:** Diogo would like to sell the merchandising pieces he owns in a market with interested buyers.

*Persona 2*

Ana is a movie affectionate, and she loves the Marvel universe. She owns some Marvel merchandising and is always looking to find new items. However, the process of finding new items is sometimes difficult as she needs to look in various websites.

**Motivation:** Ana would like to easily find new marvel merchandising.

*Persona 3*

Aside from the physical store he also makes sales online on the store website.

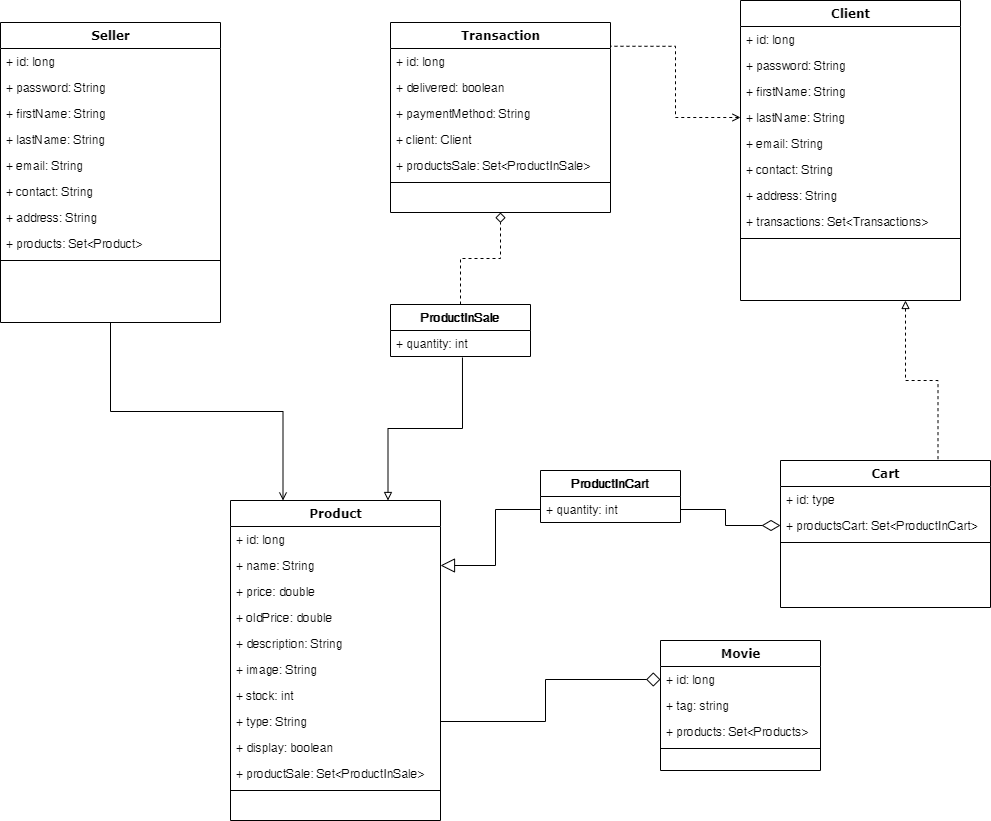
Right now, most of Rui’s sales are made in the physical store, however Rui would like to expand his business. He knows that selling movie merchandise is not as profitable as it would be in other countries like the USA, therefore, expanding his business online is the best option. With his website not having the best visibility to the world, Rui tries to sell in more casual sales websites. However, since millions of items can be found in this websites, Rui’s items often get lost not giving him the best visibility either.

**Motivation:** Rui wants to make his merchandise visible in a website dedicated to movie merchandise.

## Project epics and priorities

N/A

# Domain model



# Architecture notebook

## Key requirements and constrains

The system must be prepared to be deployed in systems running consumer grade operating systems such as Windows, MacOS or Linux.

To minimize performance hits, it should consume as little data as possible when operating online, making it possible to run in constrained situations such as using a mobile data plan.

All parts of the application are deployed on one docker image to improve adaptability between machines and eliminate possible hitches.

The footprint of the application is not big enough to justify running different components on different machines. One package is enough to harbor all running parts of the application in tune with each other.

## Architetural view

The backend was implemented with Spring Boot/Java EE technology. The database solution in use is H2 due to its high speeds, light footprint, and simple implementation with Spring Boot. The implemented REST API has all the needed endpoints for CRUD actions within the database that are needed to run the site. All actions on the application send requests to the created REST API which then makes the desired alterations in the database.

## Deployment architecture

Uma imagem com captura de ecrã

Descrição gerada automaticamente

# API for developers

From our REST API, a developer can obtain access to many facets of our application that are present in the website, in a programmatic manner. You can access the list of products available for sale, their details, create buyer accounts and even make orders through the API. With the right authorization, the endpoints that are behind authentication can also be used programmatically, as long as the requests are being made with the credentials of a valid seller account. These actions include posting new products for sale or even processing sales.



# References and resources

<https://www.petrikainulainen.net/programming/spring-framework/integration-testing-of-spring-mvc-applications-forms/?fbclid=IwAR1s6LQJ5lQyRlSrY1uTbbNzDh7mTR6TdsZRk6u3i4E3ZwhpvBjnrdX9YOM>

<https://reflectoring.io/spring-boot-web-controller-test/?fbclid=IwAR2ctqLbfyjBsGtYgshqQH8ZiLBNXoawjPjs482mSE8jHjYXdZF2Noux2xU>