deti universidade de aveiro departamento de eletrónica, telecomunicações e informática

TQS: Product specification report

Alexandre Lopes [88969], André Amarante [89198], João Nogueira [89262], Tiago Melo [89005]
v2020-05-14

1 Introduction

1.1 Overview of the project

1.2 Limitations

2 Product concept

2.1 Vision statement

reCollect is a platform that allows users to sell their old collections, from simple things such as figurines sold some years ago in stores to more rare items like limited editions of a certain product (book, game, etc.). Here, the buyers can find items they are interested in and then finish their collections or even start a brand new personal collection!

This platform is born due to how hard it is to find rare items and "relics" on the internet, mainly items and collections that people built as kids (but not limited to) and want to bring back the collector they have inside and relive some old memories. From the sellers' perspective, many people have such items stored and want to get rid of them, whether because they want to free some space or just because those items are not useful or don't mean anything anymore. So, many times, those collectibles end up in the trash (which not only creates more waste but also does not return to people the investment they made in those products)… So, to solve both buyers and sellers problems, reCollect comes as a place focused on collectors that want a centralized place where they can find and get collectibles more easily; as well as on those people that have some old collections stored, getting dust, and want to make some money out of them.

2.2 Personas

Francisco Faria

Francisco is a twenty-four-year-old man from Ílhavo and is finishing a Master's Degree in Marketing at the University of Aveiro and working in a part-time job as a writer in a games blog. He lives alone and is a big fan of movies and comic books. When he was a kid he always stayed fascinated by the superheroes and their adventures in the comics he was reading and since then he has been building a big collection of heroes comics and their figurines as well that their parents bought him. However, as he grew up, some stores around their area closed and their parents had to save money due to his dad losing his job, which made it impossible for their parents to continue to spend money in his collection.

Now, as Francisco is now independent and has a job, he would like to spend some of his savings to finish and continue his collection, mainly because he heard of some limited editions of comics that were sold when he was fifteen. However, Francisco has been searching in many websites around the Internet and is having hard times to find the items he wants and, when he finds them, most of them are still new and very expensive. Since he just wants to finish his collections, used items in good conditions would also fit.

Motivation: Francisco would like to have an online place where he could find affordable collectibles, secondhand or not, in good conditions in order to finish his collections

Alexandra Ramos

Alexandra is a twenty-seven-year-old woman from Portimão and has been working as a researcher in an Aquaculture Institute in the Algarve since she finished her Masters in Marine Biology at the University of Algarve. When she has time for it, she likes hiking and riding her bike around the Ria Formosa. Five years ago she met João, his boyfriend, who came from The Azores to study at the University of Algarve as well. João is two months from finishing his studies and has not yet a fixed and stable job, as he jumps from part-time jobs to part-time jobs. Alexandra, however, has received an invitation to work in a project linked to marine biodiversity conservation in the Azores and the couple is planning to move there once João obtains his degree. Since they don't live together and are planning to buy a house soon, Alexandra thinks this is a good opportunity to start their life as a couple in João's homeland.

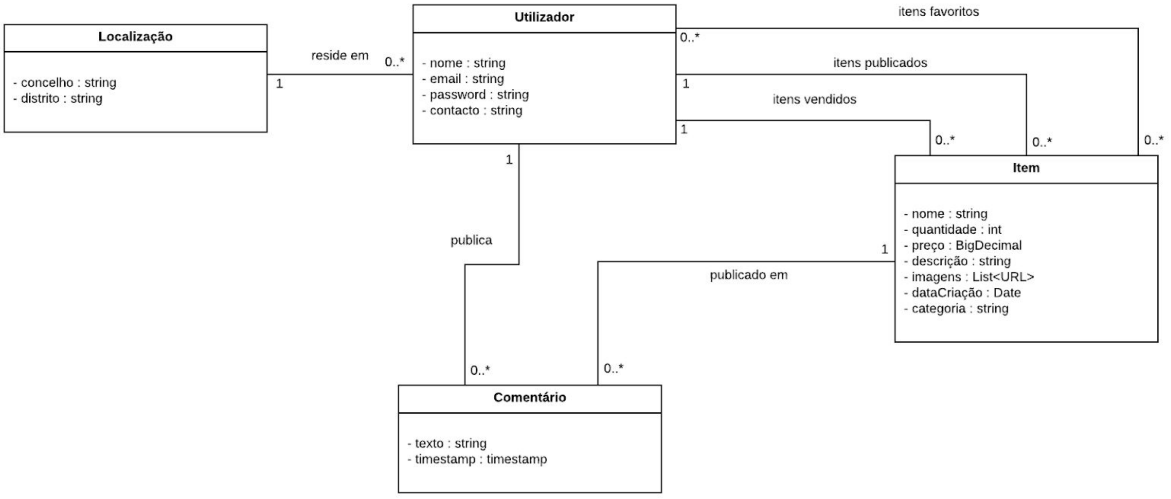
As Alexandra sets things up to move along with João to the Azores, she finds some old collectibles stored that she bought with her savings when she was younger and realizes that she can't take it all to the house in the Azores as she will be sharing it with João. So, Alexandra is a little confused about what she should do with those materials: concerned about nature and the environment as she is, she does not want to throw them in the trash, so she decided to sell those items to get back some of the money spent on them, but does not know anyone who could be interested in her items.

Motivation: Alexandra would like to find a platform where she could find people that enjoy collecting old items that can be interested in buying some old collections she has been storing since a kid and is not interested in them anymore.

2.3 Main scenarios

2.4 Project epics and priorities

3 Domain model



4 Architecture notebook

4.1 Key requirements and constraints

At the time of writing, the main constraints conceived for reCollect's system architecture consist of:

- Since it aims to provide service to everyday consumers, good system performance is a key requirement. In other words, no client should be kept waiting for a page to load for more than X seconds.
- Given that there are currently no hardware dependencies planned, there is not a particular key requirement to ensure strong microservices compatibility. We are also currently not expecting mobile support, since at the time of writing the goal is to provide a single user-interfacing medium: a web app.
- In order to ensure high availability and maximize uptime, the system must be robust. Furthermore, since long-term maintenance is also a goal, this requirement is of utmost importance.
- The system should also be scalable. To meet undefined customer needs, such as posting a variable number of items or an undefined number of clients being served simultaneously, we need to ensure the system is able to scale without being too technically expensive.
- We are currently not expecting random peak user loads, hence dynamically handling system performance for such occasions is not a priority.
- The system must ensure there is no unauthorized data access. There can be no data leaks and users must be able to trust the platform.
- Users must be authenticated to post items to the platform but not to browse already available and posted items.
- Being a client-server application, reCollect must provide a User Interface that will be accessed on the client's behalf and provide a server that will be unknown to the user, operating on a Virtual Machine provided by UA.
- Given that the application will run on a typical system, no complex deployment overhead should be necessary.

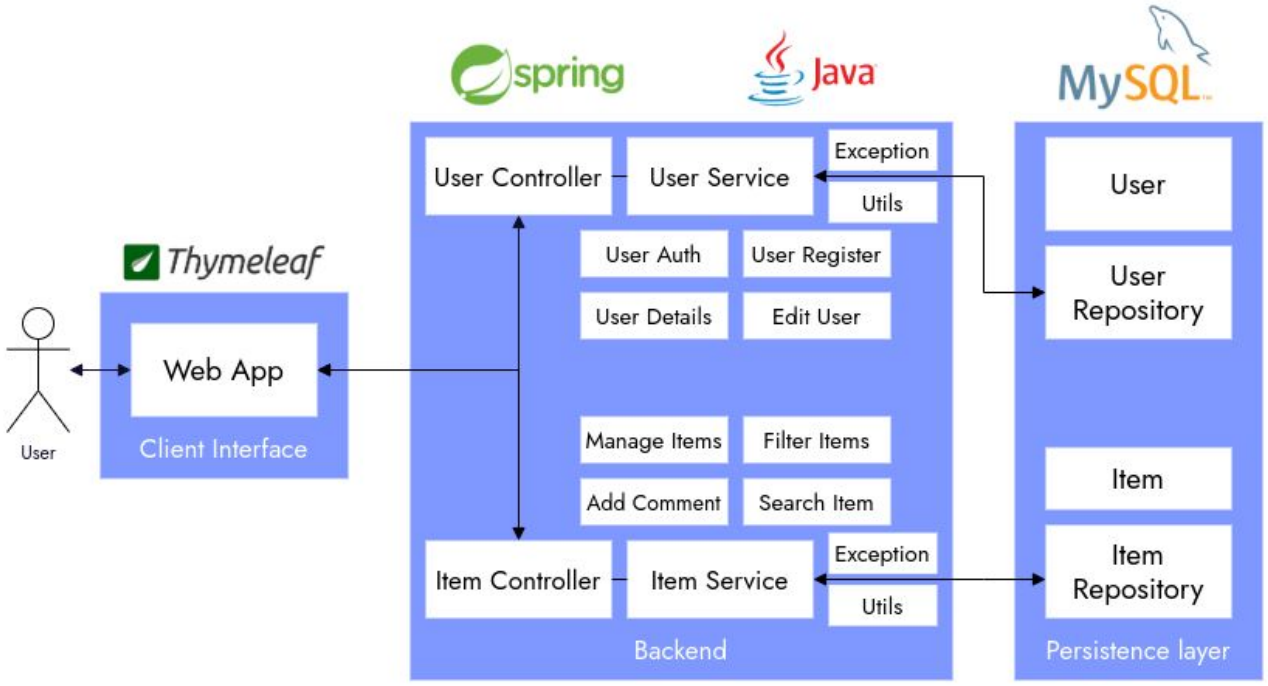
- Being a Spring Web App, the system architecture must comply with Spring Framework standards.

4.2 Architectural view

The software solution should follow the MVC architecture pattern. In other words, the Web App should comply with the standards demanded by such. These include: the Model, responsible for handling and managing data, the Controller that should expose and receive data to/from users or other developers and the View component that should provide a means of interaction between the user and the system. The architecture of a generic Spring based solution can be broken down into 6 main modules:

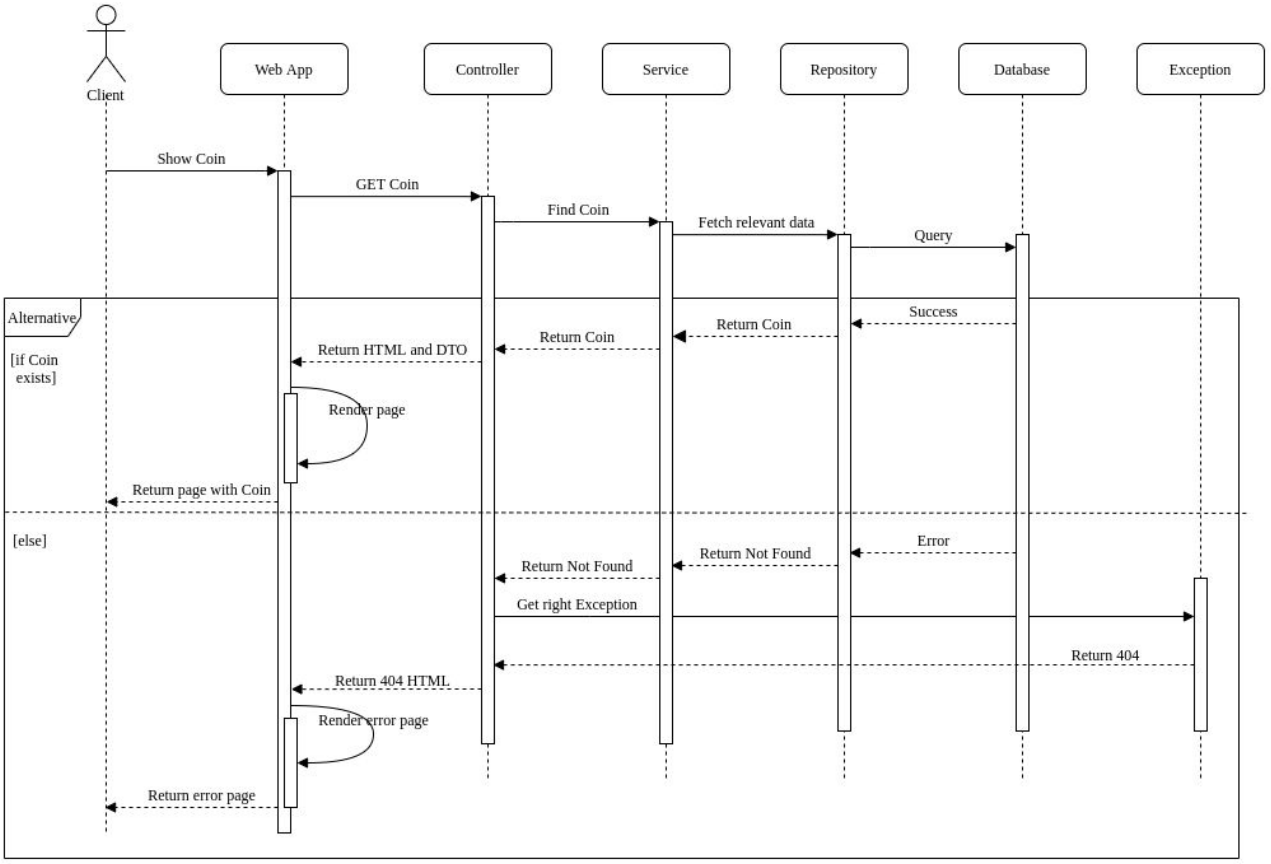
- Entity - contains the relevant POJO's for the context of the application. In this case, it will contain the User and Item classes, which will be stored persistently.
- Service - contains the classes that will handle the business logic.
- Repository - contains the interfaces needed to interact with the persistence layer.
- Controller - the controllers necessary to map endpoints and build the API.
- Exception - contains the exceptions that can be thrown and error messages.
- Utils - misc classes needed to perform different operations that might be necessary.

As to the modeling of entities, our system aims to provide service to two main models. The user, which is a client that is allowed to browse and publish products on the platform; and the item. The item consists of an abstract class that encapsulates the common attributes that different collectibles own, such as title, description, price, number of units and so-forth. This type of architecture will prove to be extremely useful since it provides a convenient layer of abstraction through polymorphism. Each collectible item will inherit and extend this class and add its own attributes, for instance, for a Coin maybe it is relevant to add the date it remounts to, as well as the press, edition or country. This means a direct implementation of each class that will be supported on the platform. However, after discussion, we have concluded this would not be an overly daunting task, since at the time our platform aims to serve only the mainstream collectible items.
Our architecture follows this structure with a couple of tweaks. Since we will have 2 core entities in our application, we have decided to split the system into 2 cores: one responsible for managing users that register themselves in the app and are able to publish products and another one responsible for managing the items that are stored in our application. Each entity is mapped and managed by its own Controller which interacts with a Service that will have the implementation of the features that regard to a specific entity. At the time of writing, for the user side these include user authentication and registry as well as managing user details; whereas for the product side of the system, the main features to be implemented include managing (such as retrieving and saving) items, filtering and searching for products by category or name and adding comments to published items. It is worth mentioning that both cores will interact with the MySQL database using their own specific repository interface.



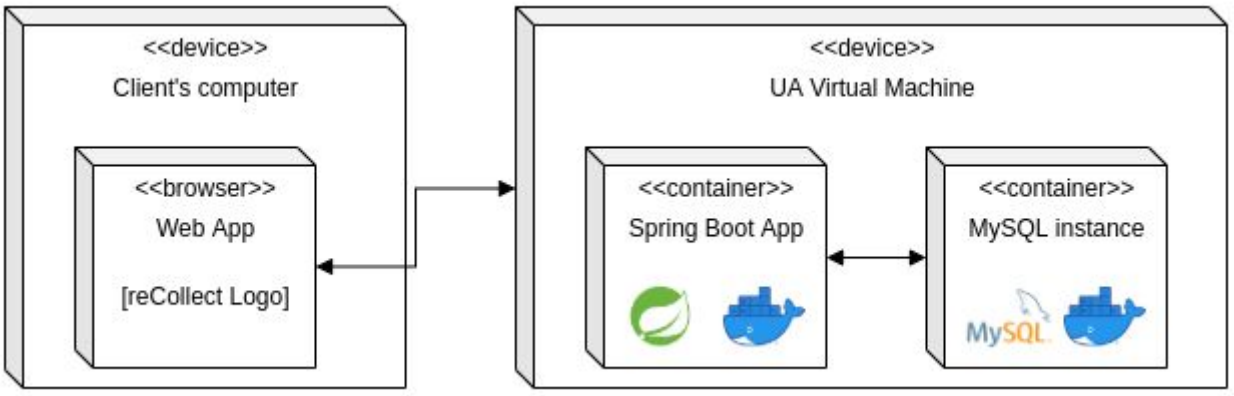
A normal workflow of the application would consist of a client wanting to see information on the website or to add new information to the website. That being said, since the primary way of interacting with the system will be through the web application, a user will start off by making a request via the website. In other words, the website will lead the user to the right endpoint depending on the information they want to see. This request will be mapped to an endpoint by the controller module which will take into account the type of HTTP request and the endpoint name. If the request isn't mapped to any available endpoint, a call to the Exception module is made in order to properly handle the error. Afterwards, the Controller will make the appropriate call to the Service module, which is responsible for the business logic. The service layer will then interact with the Repository in order to fetch the relevant information that is to be made available or to persistently store new information. Depending on the type of request, calls to Utils can be made for intricate operations or handling misc situations. Once the request reaches the Repository, a proper call to the Database is made using an interface and the Entity object is retrieved. The database will consist of instances of Entity POJOs. If the database operation was successful, then its output is returned to the Service, which returns it to Controller, which returns it to the Web App, that will render an appropriate HTML page using Thymeleaf. If, on the other hand, the object requested is not found or the database operation is unsuccessful, then another interaction with the Exception module is made, in order to properly handle the error.

To understand better how these modules interact with each other, let us consider the sequence diagram regarding a Use Case where our client wants to see information on a given product. Let us call this product Coin.



4.3 Deployment architecture

In order to deploy the application, the system will be split across Docker containers, which will communicate with each other in the VM production environment. In later stages, this deployment will be done continuously and automatically.



5 API for developers

6 References and resources