

# Grid: Product Specification Report

**Diogo Silva [89348], Pedro Oliveira [89156], Pedro Escaleira [88821], Rafael Simões [88984]**

v2020-05-143

<b>Introduction</b>	<b>1</b>
Overview of the project	1
Limitations	3
<b>Product concept</b>	<b>3</b>
Vision statement	3
Personas	5
Main scenarios	7
Project epics and priorities	9
<b>Domain model</b>	<b>10</b>
<b>Architecture notebook</b>	<b>12</b>
Key requirements and constraints	12
Architecture view	12
Deployment architecture	15
<b>API for developers</b>	<b>16</b>
<b>References and resources</b>	<b>18</b>

## 1 Introduction

### 1.1 Overview of the project

In the course of TQS, we were taught to prioritize testing over coding, and how to adopt a **Test-driven Development** (TDD) strategy when writing code and the various features a large-scale project might include.

While we had started to get used to TDD in a smaller simpler project, for this assignment, we will use the same strategy for a larger scale project. This will force us, not just to prioritize tests when developing, but also to get used to the workflow when working on a group project.

Seeing as the group is largely interested in games, we had the idea of building an e-commerce platform for game retail. In this context, the main use case involves a user putting out their game key at a certain price for other clients to buy. We also thought it interesting to add the feature of being able for a user to put his game key for auction: the user would set up a start and end date, as well as the starting bid; other clients would then see this and add their own bid. This operation will put us above other, similar platforms, like Steam or GoodOldGames, as they do not allow for auctions to occur in their platforms.

This platform only works on the basis of clients trusting the sellers. In order to inspire this confidence, we added the feature of being able to rate the seller: when you buy the game key in question, you may add a review and a score that accurately represents the experience with the vendor. Of course, these reviews extend to games as well: wanting to let other people know their thoughts on a certain game promotes the development of a good community in our service, which leads to the return of customers.

Of course, in order to have a good community, we need a system that ensures healthy, non-toxic discourse. We decided then to also setup a Report system, where you may report a review that uses inappropriate language or may be directly insulting other clients. This concept also extends to users, as they may be selling faulty game keys, or conning other users. Under this system, if the users get too many reports, they get a Warning notification, as they are clearly doing something wrong, maybe even illegal; if they keep the behavior, they will then get banned.

Finally, we have decided the name of our product to be **Grid**. A short but impactful name that references back to the 80's synthwave style.



Img 1.1. The Grid Marketplace main Logo

## 1.2 Limitations

Unfortunately, a project this size came with some compromises. Most notably, one of the most important features in these types of e-commerce systems is not only the ability to review a user or game, but being able to **report users** or malicious and hateful reviews; this would help to control the way the community would behave, as there'd be an effective way to combat these illintended users, who may leave bad reviews (a process known as “*review bombing*”) on games not to justifiably critique them, but just to lower their overall score; or even by putting out keys for sale they don't actually own. Our idea was for these reports to be filled by users and automatically sent over to admins so that they could analyze the situation and take the correct action. So while definitely important, due to time constraints, we were not able to implement this reporting feature.

Another important feature we had to leave out was what we had referred to as **Admin Statistics**. The plan was to have a page dedicated to showing the administrators of our website all types of useful information about the continuous update of data on the platform. These informations would include, but not be limited to, on a per day basis, the amount of new sales created, number of reviews written and number of transactions processed.

Although we're distraught about the lack of time to implement these capabilities, it should also be stated that these were left out as a group decision after analyzing the course's main objectives and goals, and indeed we did compensate by focusing more of our efforts on testing and finalizing all other user stories originally proposed during the conception of this work.

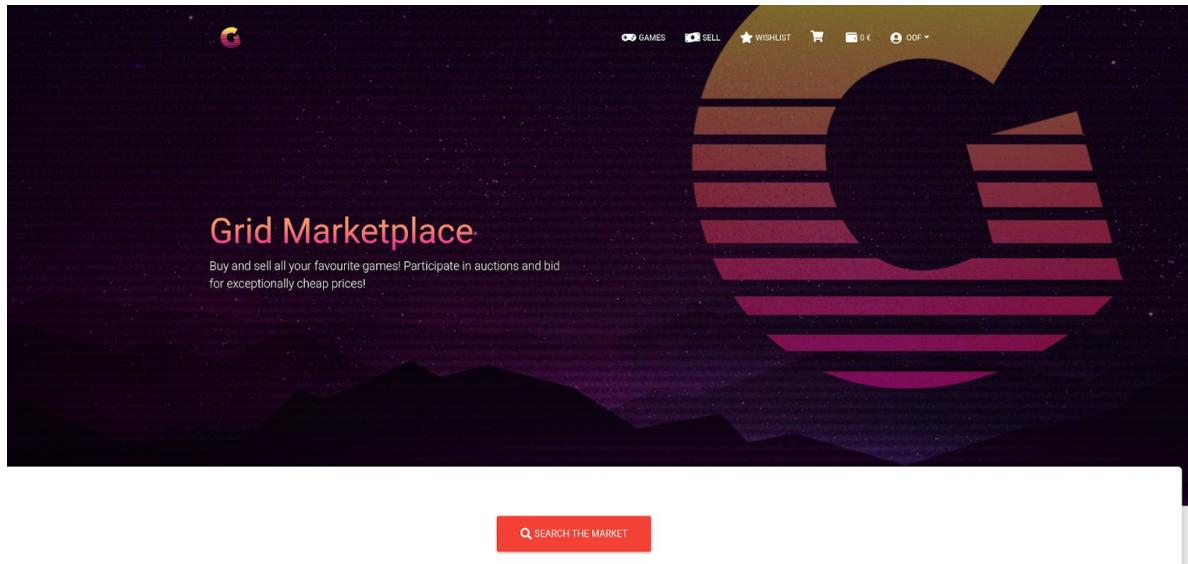
# 2 Product concept

## 2.1 Vision statement

We envisioned the **Grid Marketplace** as an extensive and appealing e-commerce platform for the sale and purchase of *digital video-game keys*. To clarify, these *keys* have been the de-facto form for video game distribution since the mid 2010's having replaced the sales of physical discs and cartridges due to their commodity of not forcing a user to dislocate to an actual retailer to proceed with their purchases. Indeed nowadays, users simply need to open one of the various online game key shops, such as Steam or Epic Games Store and get a key that unlocks the game to be downloaded. This practice has, funnily enough, also replaced the purchasing of physical copies of games, as the custom right now is for game cases to be simple CD cases with a single key code to unlock the game on one of the various game library platforms (which most online shops double down as).

With the upbringing of game keys the practice of reselling games went on a downfall. Physical retailers such as Game or CEX offer to trade-in game CDs for in-store credit or money, but when it came to online retailers and game keys this practice started to disappear. This is where our platform comes in. We give users a place to **sell keys** they own and might not want to use (either by being a duplicate key for a game they might

already own, or due to that key having been a gift, or even in the case they purposely bought the key during a sale in order to then resell it for a profit).



Img 2.1. The Grid Marketplace Homepage

The core purpose of our service is, therefore, to **allow users to sell their unwanted keys** and to give users a place to look for and **purchase games for a lower price** they might have when using an official online retailer (which sell keys directly acquired from the game's publishers, hence being limited to selling them at full retail price).

It should be noted that this concept isn't entirely new. Other platforms in the same vein do exist, such as [G2A](#) and [Kinguin](#). Both these platforms, alongside others, allow users to resell and buy keys directly from other users and they both offer their advantages and disadvantages. Whilst the core functionalities may be the same we also do believe our platform stands out by offering a cleaner interface than either of the aforementioned, a companion mobile-app that allows for the browse and the ability to create and participate in auctions for game keys.

The process for defining the features and user stories our service would need to have boiled down to an initial meeting the entire team had where we discussed several possible project ideas and proposals. One of our team members threw this idea in the air and the group came to the agreement that it would be something they'd want to try to create. This idea was then refined as stories and features were thrown into the table while others were taken out for conflicting with others or even the core idea of the platform. It should be noted that our group is composed of possible stakeholders as half of our team consists of avid gamers and users of similar platforms (with one of them having been assigned the role of Product Owner) and as such coming up with viable requirements wasn't hard. Harder was defining what we thought we would have time to implement (and properly test).



Img 2.2. Example Use Case Diagram

## 2.2 Personas

To help with the better understanding of our end users and their requirements, the following personas were devised in order to, broadly speaking, exemplify the types of people that would have an interest in our service

Firstly, there's **Pedro Simões**, a university student, currently in his second year of a tough Mathematics course. Pedro enjoys playing video games and always had all his life. He loves fantasy and prioritizes story above all else. Video games, however, aren't his only hobby as he always enjoys Tabletop gaming as well. Unfortunately this latter taste is rather expensive, and being a university student his funds are rather limited. Usually he can only afford to purchase games when there's a big sale going on (such as the Steam Summer/Winter sale), but these don't happen too often, and coincidentally when they do they tend to coincide with his exam season, where he tends to be away from his PC, hence not being able to make purchases during these times. Physical stores tend to also hold big sales, even during periods when he's not overburdened with work, however, living in a tiny student flat gives him very low storespace to put game cases in. With this being said, **Pedro has stated that he wishes for a way to purchase cheap games, all year around, without having to actually physically store said games.**

**Miguel Gonçalves** is another game enthusiast who's finishing his Bachelor's degree in Economics. Being at 23 years of age and coming from a rather wealthy family, he has been subscribed to an online service that gives him 10 new games each and every month. Despite being a university student he has quite a bit of disposable income, and as such he also enjoys purchasing games on the regular, especially new AAA games that have just come out. This has the rather unfortunate downside of him ending up with multiple duplicate

keys for the same game. These are quite literally, useless to him, and as such **he would love for there to be a way to dispose of these keys whilst also turning a profit** to get more money to purchase even more games.

Furthermore, we have **Nicolau Silva**, a 30 year old married family man that quite enjoyed collecting games during his twenties, but, due to his job as a lawyer in a firm, no longer has the time or the interest to keep pursuing this hobby. However, and mostly out of habit and due to brand loyalty, he kept buying the latest games from his favourite developers. This ended up in causing him to have quite a bunch of keys that he has never used, nor does he intend to. In his younger years he was quite the game connoisseur, able to determine a fair market value for a game just by looking at it. Nowadays he's lost his touch and has no clue of what the state of the industry is. His husband keeps bugging him to get rid of the games he's not using as they could use the extra money since their 2 younger kids are just now starting to go to preschool. This, and his current lack of knowledge on the fluctuating prices of games, leads him to wish for **a form to sell his games**, perhaps **by setting a lower bound price and letting potential buyers make the offers themselves**, a-la auction style.

Lastly there's **Mafalda Meopito**, a 24 year old woman who graduated from design school last year. She still lives with her parents and younger, 17 year old brother, despite having started working for a small design firm just a couple of months ago. Mafalda's brother is quite the gamer, who loves and plays on both PC and PS4 and has even participated in some professional tournaments (or so he claims). She, however, has never been that fond of games, thinking of them as a waste of time and preferring other recreational activities such as hanging out with her friends, going to festivals and concerts, alongside other social activities. She loves her brother quite a bunch, and as such enjoys giving him gifts occasionally. She knows a gift he'd like the most would probably be a video game, and she also knows her brother is quite fond of Action and RPG games and that his preferred platforms are, as aforementioned, the PS4 and PC. She is desperate for an **easy and quick way to search and find games, filtering by parameters such as platform and genre, read information and reviews about said game**, in order to make sure the game is half decent, **and possibly following through with a purchase**, in order to make her brother happy. Due to his brother's fondness for games, people also tend to think she's an avid gamer, and as such she sometimes receives game keys as gifts that she never uses. As such she would really **like for a way to get rid of the game she was gifted and also get some money in return**.



Img 2.3 Our 4 personas. From left to right we have Pedro Simões, Miguel Gonçalves, Nicolau Silva and Mafalda Meopito

## 2.3 Main scenarios

In this section we'll be including some example scenarios for some of our main features. Despite these being good representations of our system's value, it should be noted that the extent of our system is not limited to these next descriptions. These scenarios will be made from the perspective of one of the four personas mentioned in the prior section.

### Pedro Simões wants to buy a cheap game

Pedro just finished his exams and he could really use a way to vent off. Since he wasn't able to go to any parties or hang out with his friends during his exam season, and by cooking his own meals everyday, he actually managed to save some money. Some money, however not much, and it's still not enough to buy the new game by his favourite AAA developer that came out 2 months ago. He looked on the two major online game PC storefronts, Steam and Epic Games Store, however, on both platforms, the game was being sold for full-retail price (i.e 69,99€). Saddened he takes to the web and ends up in the **Grid Marketplace**. He decides it'd be worth a shot to check it out and **searches for the game he wants by inputting its name** in the search page. To his surprise, he finds out that there are several people selling it, and the reported **Best Offer** is being sold for just 23,22€ ! He immediately creates an account, choosing not to associate any credit card to his account, since it's the first time he's utilizing the platform and doesn't know whether to trust it fully or not yet. He adds the game to the shopping cart and proceeds to the checkout, filling the **temporary credit card** (a card whose information will be used only for this purchase and won't be stored by the system) form and following through with the purchase. He **receives the key** and is happy about being now able to play this game he's been wishing for so long.

### Miguel Gonçalves wants to sell one of his duplicate keys

Miguel is hanging out with his friends and is about to withdraw some money from his card to enjoy a night out, when he realizes that he has no money in it. He phones his parents who tell him he's been spending way too much, and as such, they're cutting his monthly allowance. Bummed out he turns to his friends and states that he won't be able to go out with them. One of his friends, who knows of his unusual collection of multiple keys for the same games, asks him if he won't simply and quickly sell one of his duplicates in order to get enough money to enjoy himself. Miguel asks how he could accomplish this and it's at this point that his friend tells him about the **Grid**. In just a few minutes he **creates an account, registers his credit card and puts one of his game keys for sale** at an unusually low price (just the amount he needed to go out). Not long goes by before his game has been sold and he receives the money.

### Nicolau Silva wants to sell a game on auction

Nicolau's sons are going to preschool in 1 weeks and, as such, he and his husband are going to have to buy a bunch of school materials such as pencils, pens, markers, scissors, backpacks, alongside others. They, however, are a bit short on money since we're reaching the end of the month and their salary won't come in until the first of the next month. Nicolau's husband points out that Nicolau has quite a few game keys he has never even used, and he knows this because, to his dismay, a couple of them were gifted by him. Nicolau immediately goes online to try to find a way to sell his keys, ending up in the **Grid Marketplace**. He quickly sees that there are several auctions going on and that there are active biddings occurring. **Knowing**

**he has a few days before he needs the money** (worst case scenario he needs to go shopping for materials 5 days from now), and wanting to get the most money possible, **he sets up a couple of auctions for some of his game keys, setting a reasonable starting price and setting the end date to be exactly 4 days from now.** Plenty of time for several bidders to participate and make offers and give him just enough time to use the money to get the required school supplies.

#### **Mafalda Meopito wants to find a decent game to buy**

Mafalda's brother's birthday is coming up and as the good sister she is she wants to buy him a video game (since she knows he's quite an avid gamer). She, however, doesn't know anything about games, nor whether a game is good or not. She does have a friend however who also enjoys games and has been using the **Grid Marketplace** to get her keys since, as she says, is an easy to use marketplace where you can buy games for really cheap prices. With this knowledge in hand she starts browsing form games, quickly noticing she can **filter games by genre, platform and name**. Knowing her brother plays a lot of PS4 and PC and really likes Action and RPG games she inputs these parameters and analyzes the shown games. There's quite a few so she doesn't really know which to buy. This is when she notices **each game has a Grid Score and several User-Made Reviews**. She starts reading these and this ends up influencing which game she wants to buy her brother. Unfortunately she has to go to work, and doesn't have time to proceed with the purchase. During her commute to work by train, however, she downloads the **Grid Mobile App, finds the game she had looked at before and buys it**, all before exiting the train!

#### **Pedro Simões wants to review a seller that ripped him off**

Pedro has been using the Grid Marketplace for a while now, and he's had several good experiences buying games at rather cheap prices. One day however, he sees one he really wants for a ludicrously low price. **He does notice that the user who's selling the game has a pretty low Grid Score of 2.** But his eyes are fixed on the low price and he decides to take the risk. He proceeds to payment and receives the key, which was advertised as being available to be redeemed on the Steam platform. As he's attempting to do so, however, it fails and Steam gives out a notice that the key is not correct. Mad, he goes into his profile and starts **writing a review about this seller, which will be displayed on that user's public profile, notifying other users** that this seller is a potential scammer and should not be trusted. He also hopes that one of the admins sees the review and decides to investigate further, **maybe even banning that user from the platform.**

#### **Nicolau Silva wants to review a game that he really enjoyed**

Nicolau has been buying a bunch of games on the Grid Marketplace having made this platform his de-facto way to purchase games. One of these games is a very recent, fairly unknown indie game that has been struggling to be noticed by the gaming community. Even when talking to his friends, none seem to know about the existence of this game, going as far as saying that if it's not known then it's probably not even that good. This leaves Nicolau rather distraught. Whilst **checking his profile he notices he has the ability to review the games he's bought**, and with this in hand he decides it'd be more than fair, almost civic duty even, to leave a positive review about that game, so that maybe other users pondering whether its worth a shot or not might be persuaded in favour of trying it out. He does this and **now that review is forever attached to that game for everyone browsing to see.**

## 2.4 Project epics and priorities

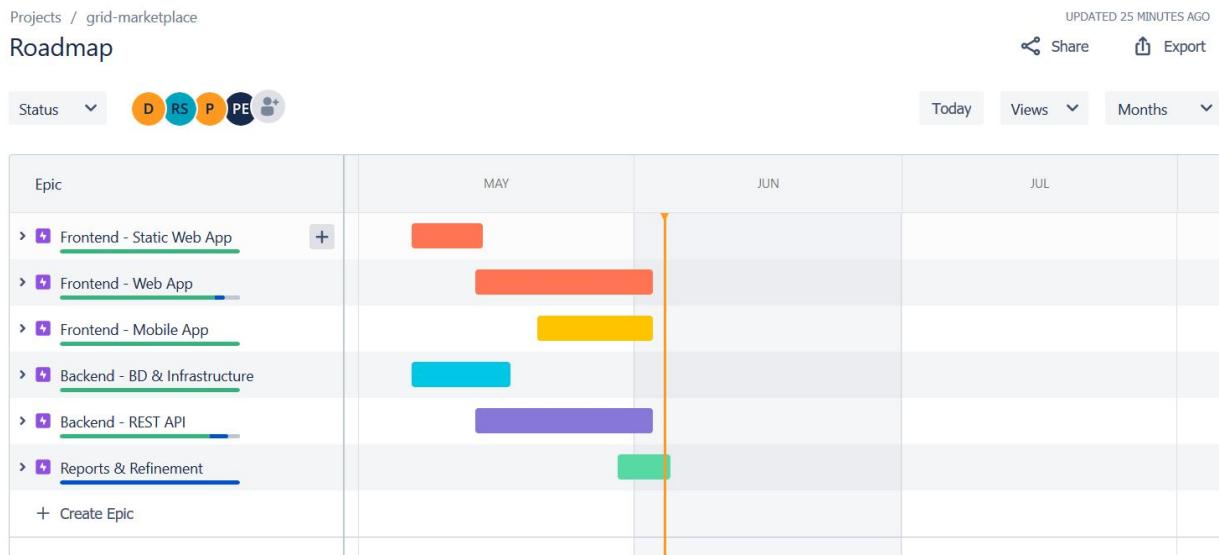
Right from the bat we decided to implement an AGILE methodology, participating in weekly SCRUM Meetings, defining Sprints and all other characteristics of this practice, mixed in with the Kanban *To Do, Doing, Done board*. As such, on our first meeting our group divided all features that had to be done into several User Stories. These User Stories were then transformed into workable tasks, and these tasks were grouped into **Epics**.

*"Epic are large bodies of work that can be broken down into a number of smaller tasks (called stories)."*

-[Max Rehkopf](#)

Such states the official Atlassian AGILE Coach documentation, and in fact this is what they represented in our project. In total we had 6 epics:

- **Frontend - Static Web App**
  - Created due to the fact that in the first week of the project we had to present a useable, static version of our web-app in order to showcase some use cases (yet to be integrated with the REST API implementations), purpose of our project and overall style of our product.
  - This epic contained tasks related to the creation of several static web pages that could give the impression of a finished working project for purposes of presentation
- **Frontend - Web App**
  - Representative of the actual implementation of our Web App. Contained several sub tasks each relating to the completion of one of our defined User Stories. Each task was only considered done when the page, implementation with the API and tests were properly conducted.
- **Frontend - Mobile App**
  - Our companion Mobile App was thought of as a simplified version of the Web-App that also connected to our API and allowed a user on mobile to easily interact with our platform. As such it deserved an epic of its own since most tasks from the Web App wouldn't be transposable to the Mobile App and new tasks would have to be devised
- **Backend - BD & Infrastructure**
  - This epic refers to all tasks related to the definition and creation of all infrastructures needed for the implementation of the project. These include but are not limited to, BD structure, BD population scripts, CI/CD pipeline definition, amongst others.
- **Backend - REST API**
  - In this epic we grouped tasks related to the creation and proper testing of the endpoints necessary for our service's functioning.
- **Reports & Refinement**
  - This final epic included tasks such as the completion of this and the QA Manual reports, debugging of possible errors, minor performance improvements and other quality of life tasks.



Img 2.4 Our project's JIRA Roadmap graph displaying all of our epics, their completions and end dates

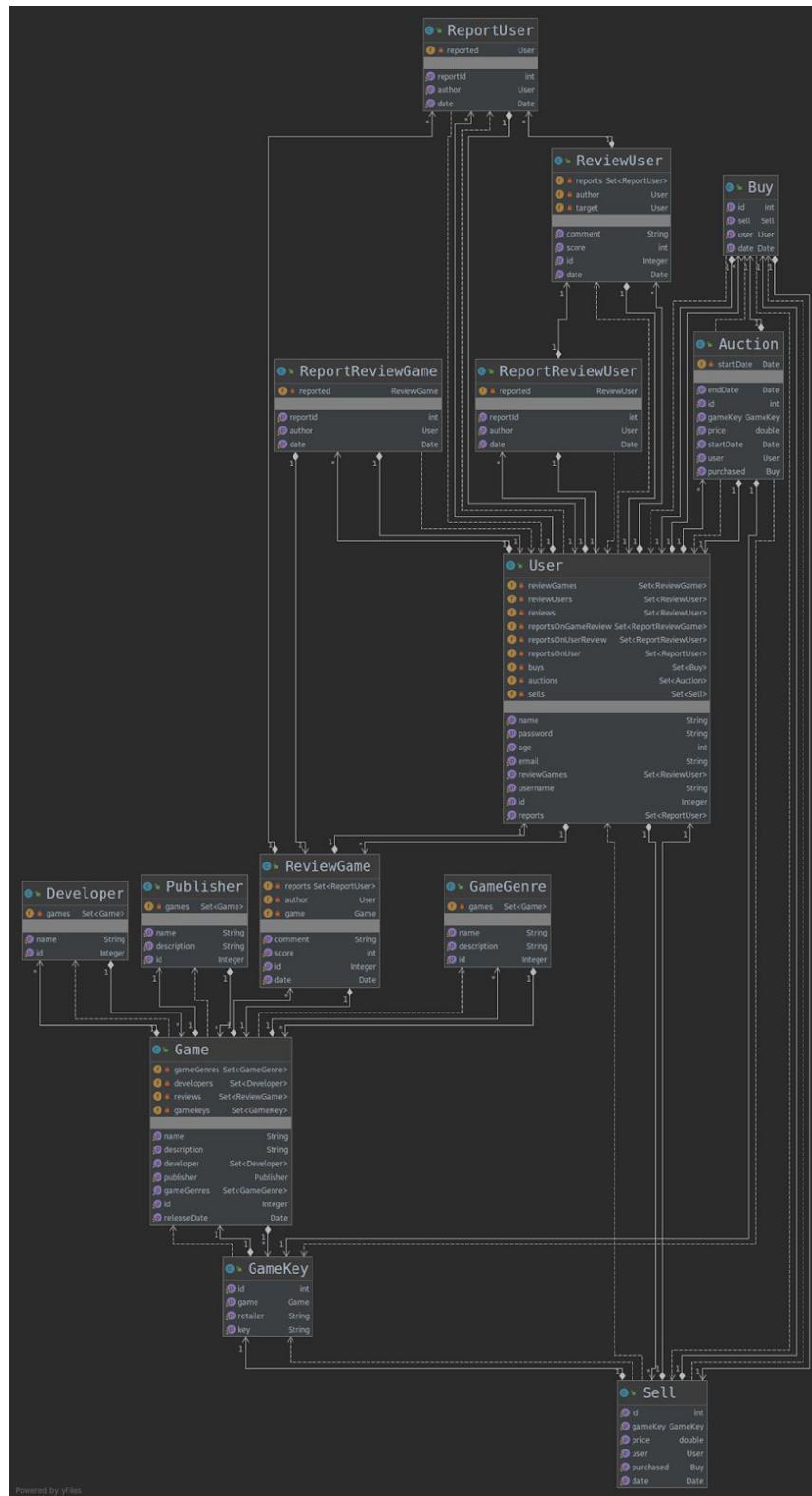
Relative to our priorities it should be noted that we as a group had a discussion as to which features had more or less priority in terms of bringing value to the system and representing its core usage and ideas. With this we had a clear idea of which User Stories to prioritize and which to leave on the backlog. Overall we ended up being able to implement basically all of the features we had planned, except for a few indicated in the Limitations section, but for the curious, we started by prioritizing the User Stories relating to Login, Sign Up, See Games, See Games Info, Buy and Sell.

### 3 Domain model

So as we have explained before, our main entities will be Users and Games, most other entities coming from one of these two.

Focusing first on the Games entity, it's made by a team of developers, and published by a specific Publisher, who have to be their separate entities as we would not just want to associate the entity with a simple String object. The game must also have information about their genres, be it Action, Adventure, RPG, so on and so forth. Each game will have a list of various Gamekeys that are currently in the market, Gamekeys being another entity themselves.

Now focusing on the User entity, it represents what our clients can do. The clients may buy, auction, or sell game keys; they may review other users and games, or report any client or review to an administrator. All of these actions are identified as different entities in our databases.



Img 3.1 Our backend's class diagram.

## 4 Architecture notebook

### 4.1 Key requirements and constraints

We also utilized our first meeting as a group to decide upon the technology stack that we would be utilizing. We wanted our platform to be fast, easily testable, allowed for the ease of deployment and was reliable.

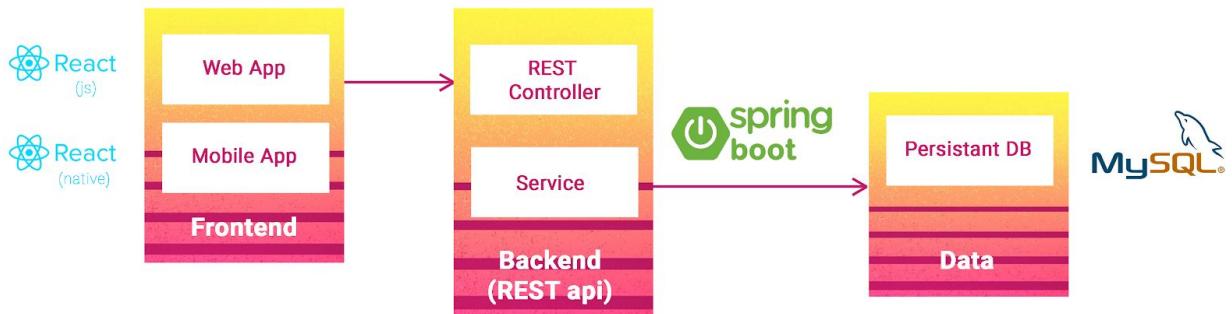
Starting with the frontend part of our system we wanted it to be available on all major browsers and be easily adaptable for mobile. We also wanted to be able to reutilize components throughout several pages both to allow for a quicker development with smaller implementation time, as well as to maintain a consistent style between the entire frontend. For the mobile app, in specific, we wanted to be able to reutilize the integration code between our web-app and our API service, since the web app would start development before the mobile app, hence being able to reuse code would give us a good base to start development of the app with. We also wanted it to be available both on Android and iOS devices in order to not discriminate any possible users.

For our backend we wanted it be something we were familiarized with, but above all it should provide us with a good suite of testing tools in order to assure the quality of our service. This was vital for the robustness of our API and overall integrity of our entire service. Knowing we would be dealing with user accounts we also had to make sure our backend framework allowed for the safe handling of passwords and confidential user data.

Finally we knew that we would be dealing with a big amount of games (over 1000) and as such we needed a persistent DBMS that would be capable of handling the storage of thousands of game information, users, sells, reviews, amongst others. This DBMS should also allow for the indexing of rows in order to allow for the quick querying, especially when searching for games by parameters. We also knew our model was relational, and as such we would need a relational DBMS to maintain relations between our entities (such as, for example, the relation between a user and his owned games) Last, but certainly not least, this DBMS would need to provide *transactions* to make sure that when a user buys a game, that game is removed as a listing and is added to the users account, alongside other types of similar interactions which would also require a consistent state to be maintained.

### 4.2 Architecture view

Taking into account all of the aforementioned requirements, below is presented a diagram that shows, from a high-level view, the technologies and frameworks that our service has been built upon.

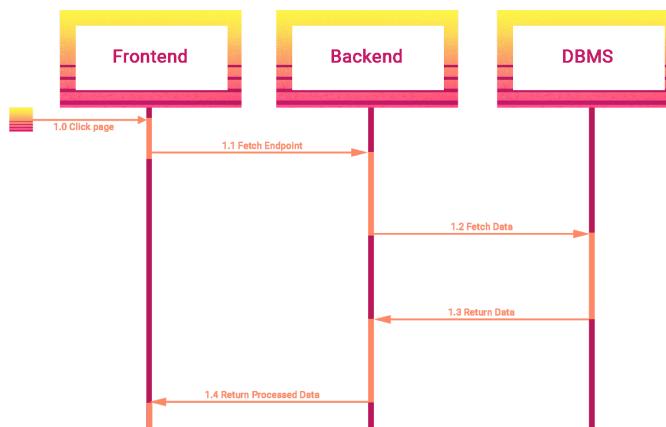


Img 4.1 A simplified view of our architecture

On the **frontend** side we opted to go with the JS Framework we're most familiar with - **React JS** for the web app and **React Native** for the mobile app. These tools allow us to quickly build our platforms with a high degree of fidelity and control through the usage of reusable components and pre-built modules. Alongside this, React Native also allows for the reutilization of previously made code for communication with the REST API made using React JS, and can be used to build apps that are OS agnostic (i.e apps that work on both Android and iOS).

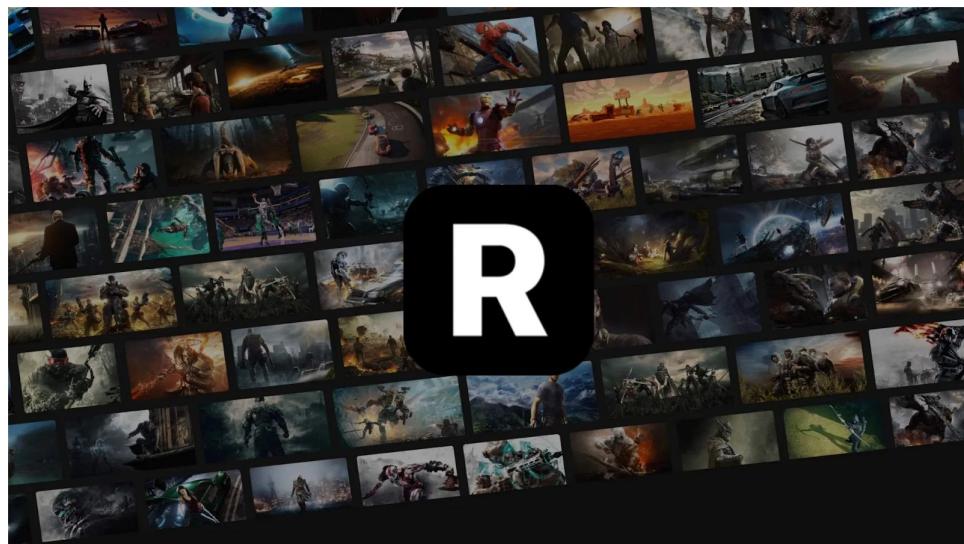
As for our **backend**, we opted to go with the **Spring Boot Framework**, as per the course's directives. This permits the reutilization of previously acquired knowledge from both this and prior courses in order to build a robust API capable of serving our platforms. Besides this, using Maven as our dependency controller allows us to import several types of modules and testing tools such as JUnit and Selenium to our project with ease. This gives us a rather large suite of testing tools at our disposal to assure the quality of our product. We also found out about **SpringSecurity** which can be used to manage user information and authentication.

For our DBMS we chose **MySQL** over all other possibilities. We did this because it ticked every box in our requirements, as well as being a very lightweight, easy to use, well documented and widely utilized DBMS that we're rather familiar with.

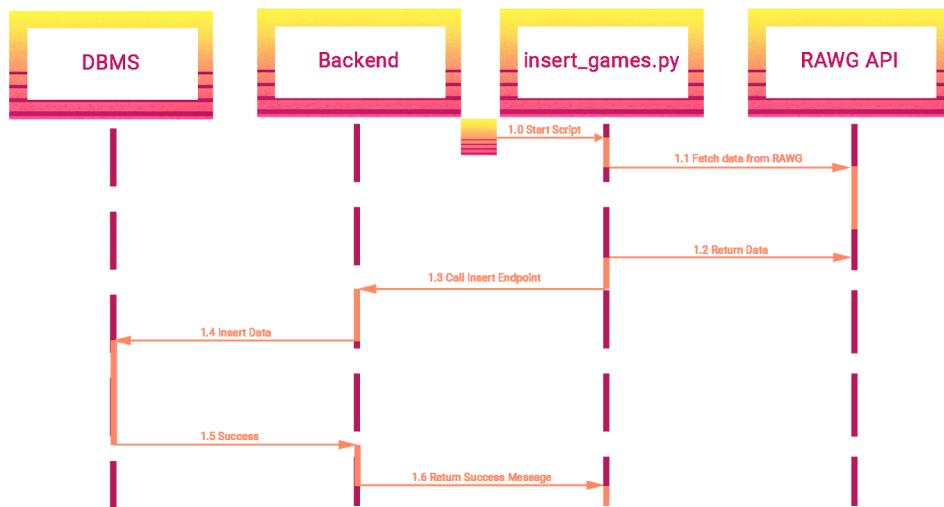


Img 4.2 Sequence Diagram showcasing the normal interaction when loading a new page/screen

It should also be noted that we were faced with the problem of having to populate our database with several games (preferably thousands) information. This information should include developers, genres, publishers, descriptions, name, launch date and cover picture. We basically had two options. The first was to randomly generate strings and values for all of these fields. This would give us a lot of data, yes, but it wouldn't be realistic and even for the purposes of a better project we decided would be ill-advised. We then found the **RAWG API**. This extensive API allowed us to fetch information from a rather large database of video games in order to populate our database, and as such a script was created to invoke both an endpoint on the RAWG API to get developers, publishers, genres and games, as well as one on our own API to add that information to our DB.

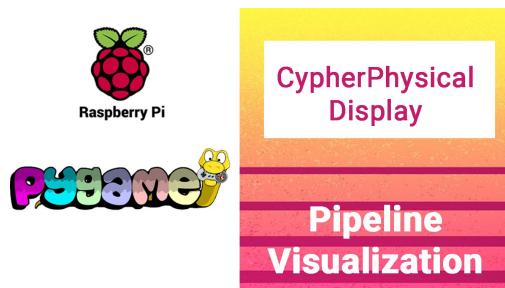


Img 4.3 RAWG API - "World's Largest Video Game Database"



Img 4.5 Sequence Diagram showcasing the population of our DB

One other thing we wanted to include was the usage of a Cyberphysical Display that would connect to our GitHub Actions CICD pipeline and inform us of the stage of each stage without us having to actually go on GitHub. We had several ideas for this, the first being the usage of a Raspberry 3 and an LED strip that would light on and off depending on the execution of each stage. This however, was not possible since we wouldn't be able to get the necessary hardware in time. Two of our team members already had a Raspberry 3 though, and one of them came with the idea of utilizing a PyGame program to graphically display the stage of each pipeline, kind of like a stoplight. There would be several circles in a line, each representative of a stage, and they would light up grey, yellow, red and green for not completed, in progress, failed and succeeded, respectively. This program would then be ran continuously on a Raspberry computer. This was indeed implemented and works as expected, giving the user of the Raspberry information of whenever a pipeline is ran on the master branch, who was the author of the commit and the state of all stages.



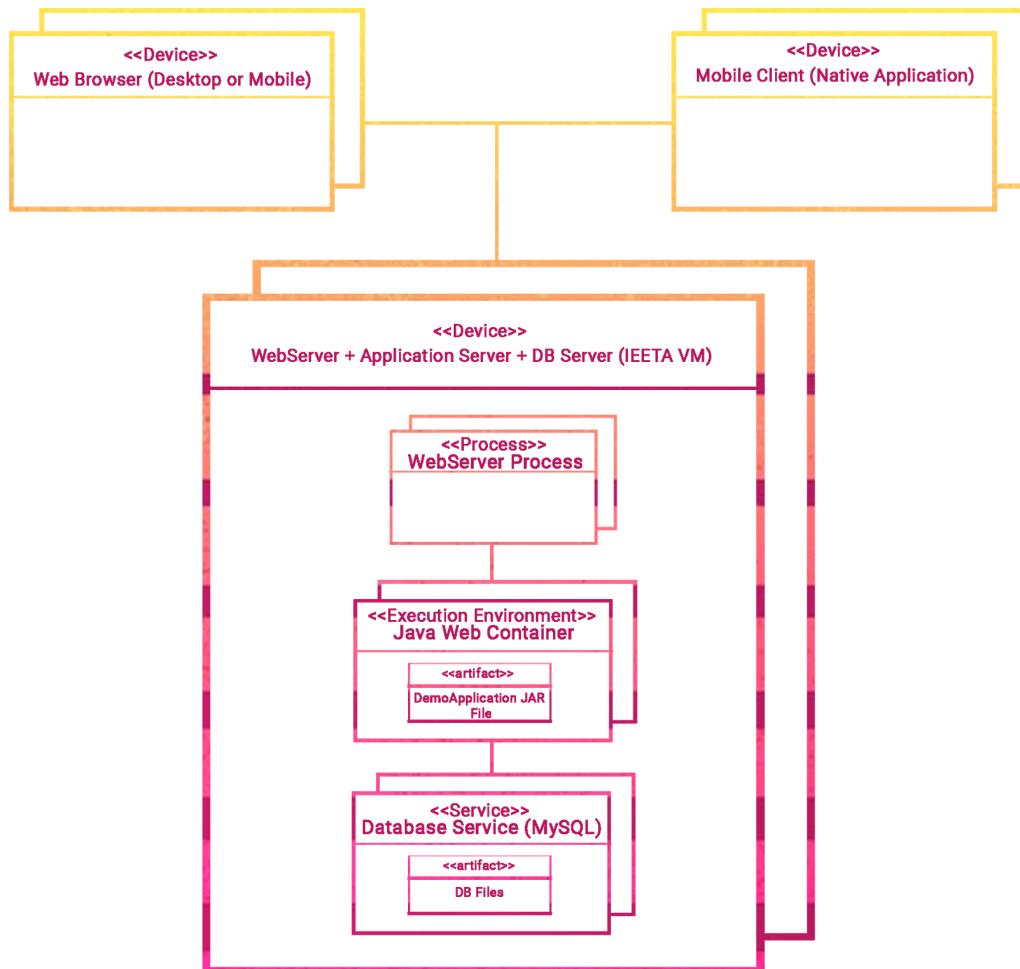
Img 4.6 Our CyberPhysical component

### 4.3 Deployment architecture

As for the deployment of our end product, and as this is a course project, we were graciously provided a Virtual Machine from the IEETA institution to serve as a server. On this server we placed all of our backend services, i.e, our SpringBoot Application and our MySQL server. Our web-app was also placed within this machine whilst our mobile app is available for download as an APK. All of these backend services have been placed inside Docker Containers upon deployment.

It should be noted that, due to the nature of the machine, our platform is accessible only from the Eduroam network or Aveiro's University VPN.

Below is a diagram that illustrates what has been said so far



Img 4.7 Our Deployment Diagram

## 5 API for developers

Speaking about the API, we opted to implement a REST API utilizing SpringBoot as our framework. At first we were actually misnaming our endpoints. We were starting them with the root **/grid** and then defining the process the user is accessing. For example, when trying to add a new user review we called the **/grid/add-user-review** endpoint. This was in fact not following the REST nomenclature, and after having been instructed by our teacher, we were informed that we were actually using SOAP rather than REST (i.e, accessing processes rather than resources).

With this knowledge in hand, we fully changed all of our endpoints to go in accordance with REST, changing their names so that they pointed to a resource rather than process. That endpoint, for example, became **/grid/reviews/user** being called with a **POST**.

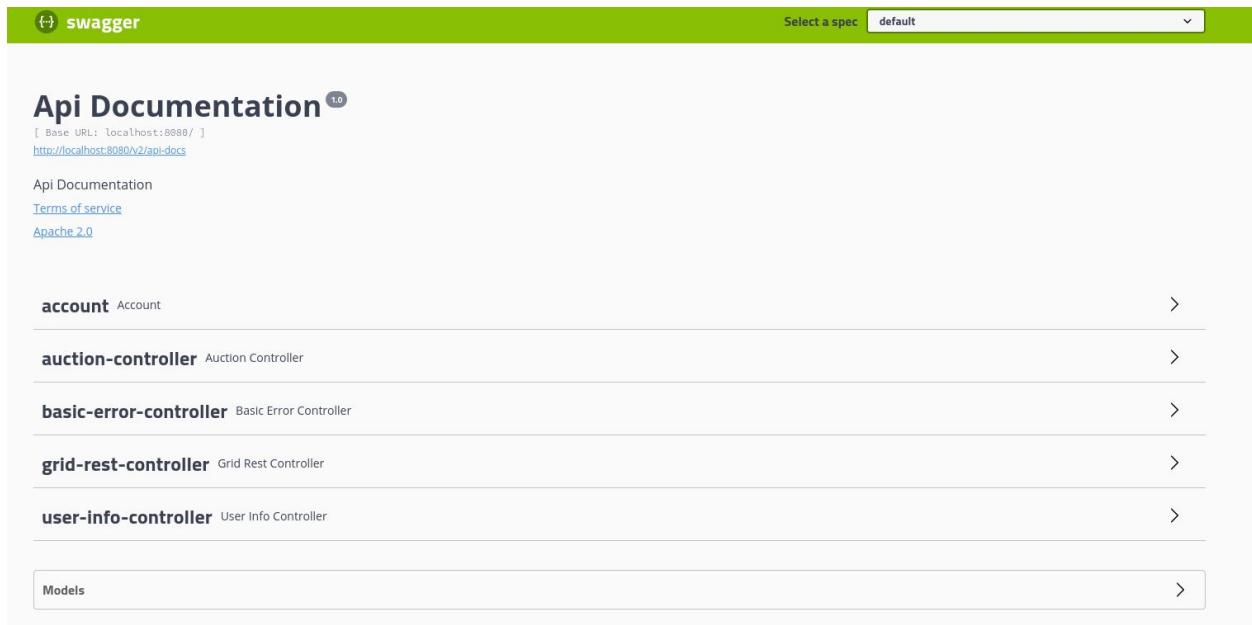
Feature-wise our API allows for the control of basically our entire service. There are several **GET** endpoints to gather information on games, publishers, developers, users, etc; **POSTs** to add new games, users or sales, and such to our database, **PUTs** to update user info and auction winners and even **DELETEs** aimed at canceling sell listings and removing user accounts.

These requests are separated in four main controllers: an account controller to deal with sign ups, login and deleting accounts; a user information controller that has other user-centered operations, such as updating information, adding funds and get the user information; an auction controller that focuses around the feature of being able to start an auction for a game and have different users bid on it; and finally, the main controller, Grid Rest Controller, that has all the main operations: search games, check game information, and buy and sell games.

All of our endpoints are well documented, both in terms of their path, but also what their request parameters/body is, whether they require authorization, what their response is and what types of errors they might throw. These are all specified in our Swagger available at

<http://192.168.160.56:8080/swagger-ui.html>

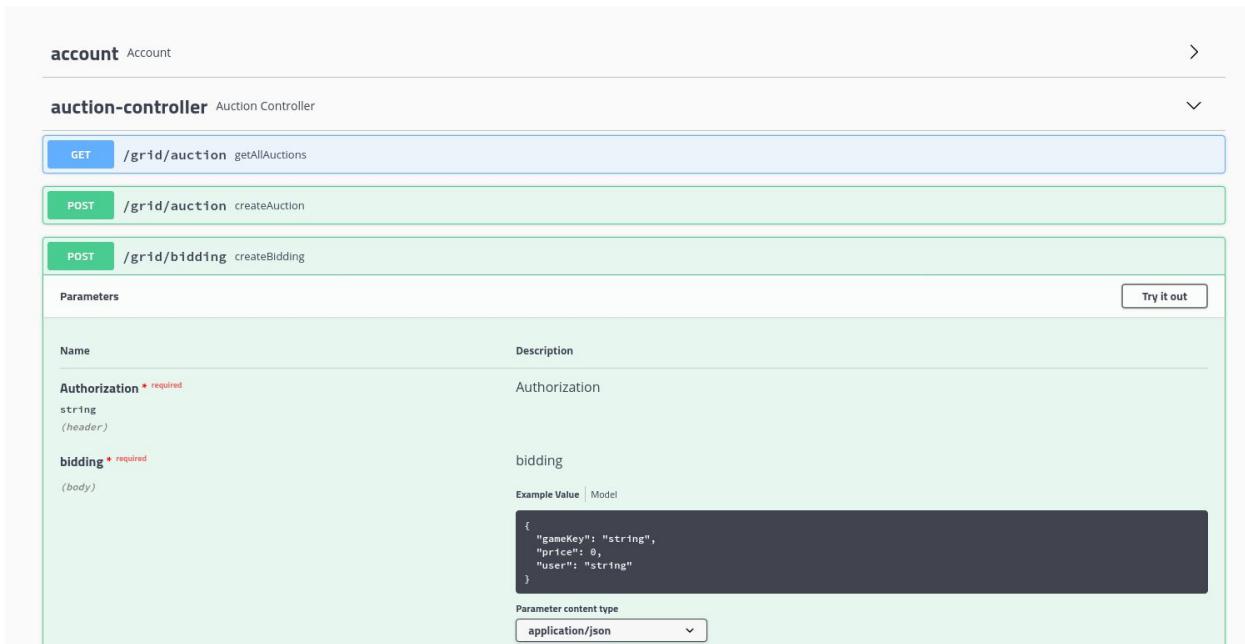
It should also be explained that due to our usage of the SpringSecurity module most endpoints require the sending of a Basic Auth Authentication Header. This can be sent as null for the endpoints that don't require a user to be signed in to see (such as the endpoint that returns all games, as a user doesn't need an account to simply peruse our storefront). But for endpoints that in fact require a user account (such as selling a game), valid user credentials are needed, otherwise the endpoint will return a HTTP 401 error.



The screenshot shows the Swagger UI interface for the API documentation. At the top, there is a green header bar with the 'swagger' logo and a dropdown menu labeled 'Select a spec' set to 'default'. Below the header, the title 'Api Documentation 1.0' is displayed, along with the base URL '[ Base URL: localhost:8080 / ]' and the API specification URL 'http://localhost:8080/v2/api/docs'. Under the title, there are links for 'Api Documentation', 'Terms of service', and 'Apache 2.0'. The main content area lists several controllers as expandable sections:

- account** Account
- auction-controller** Auction Controller
- basic-error-controller** Basic Error Controller
- grid-rest-controller** Grid Rest Controller
- user-info-controller** User Info Controller
- Models**

Img 5.1 Our Controllers



Img 5.2 An example of an endpoint

## 6 References and resources

For the elaboration of this entire project and in order to achieve the presented final product, the following resources were vital:

- <https://www.baeldung.com/rest-with-spring-series>
- <https://reactnavigation.org/>
- <https://material-ui.com/pt/>
- <https://www.creative-tim.com/product/material-kit-react>
- <https://www.creative-tim.com/product/material-kit-react-native>
- <https://reactnative.dev/>
- <https://jestjs.io/>
- <https://enzymejs.github.io/enzyme/>
- <https://api.rawg.io/docs/>
- <https://spring.io/projects/spring-security>
- <https://www.restapitutorial.com/>
- <https://guides.github.com/introduction/flow/>