Design document for IAS

BluDevil digital marketplace



Members: Alexandru, Emilian E.M.

Tutors: Geurts, Jaap J. ; Guayrin, Brice B.P.B.J.P.

Version: 1.3

# Table of Contents:

[Table of Contents: 2](#_Toc72317701)

[1. RESTful API design 1](#_Toc72317702)

[1.1 Product API section 1](#_Toc72317703)

[1.2 Image API section 2](#_Toc72317704)

[1.3 User API section 1](#_Toc72317705)

[2. Diagrams 2](#_Toc72317706)

[2.1 C4 diagrams 2](#_Toc72317707)

[2.1.1 Context diagram 2](#_Toc72317708)

[2.1.2 Container diagram 3](#_Toc72317709)

[2.1.3 Components diagrams 4](#_Toc72317710)

[2.1.3.1 Product components section 4](#_Toc72317711)

[2.1.3.2 Image components section 5](#_Toc72317712)

[2.1.3.3 User components section 6](#_Toc72317713)

[2.2 CI setup diagram 7](#_Toc72317714)

[2.3 Entity relation diagram 8](#_Toc72317715)

[The product section: 9](#_Toc72317716)

[The user section: 9](#_Toc72317717)

[The order section: 9](#_Toc72317718)

[2.4 UML diagrams 10](#_Toc72317719)

[2.4.1 User section 10](#_Toc72317720)

[2.4.1.1 Components 10](#_Toc72317721)

[2.4.1.2 Models 11](#_Toc72317722)

[2.4.2 Product section 12](#_Toc72317723)

[2.4.2.1 Product components 12](#_Toc72317724)

[2.4.2.2 Product models 13](#_Toc72317726)

[2.4.3 Image and common section 14](#_Toc72317728)

[2.4.4 Exception section 15](#_Toc72317729)

[3. Design decisions 16](#_Toc72317730)

[3.1 Back-end: 16](#_Toc72317731)

[3.1.1 Spring boot: 16](#_Toc72317732)

[3.1.2 Date persistence 17](#_Toc72317733)

[3.2 Front-end: 18](#_Toc72317734)

[3.2.1 Frameworks criteria 18](#_Toc72317735)

[3.2.2 Frameworks 18](#_Toc72317736)

[3.2.2.1 Vue 18](#_Toc72317737)

[3.2.2.2 React 18](#_Toc72317738)

[3.2.2.3 Angular 19](#_Toc72317739)

[3.2.3 Framework conclusion: 19](#_Toc72317740)

[3.2.4 Bootstrap 19](#_Toc72317741)

[3.2.5 Font Awesome 20](#_Toc72317742)

[4. Wireframes 21](#_Toc72317743)

[4.1 Navigation & Product catalog 21](#_Toc72317744)

[4.2 Product details 22](#_Toc72317745)

[5. Quality assurance metrics tool results 23](#_Toc72317746)

[Last test results: 23](#_Toc72317747)

[6. Design document versioning 24](#_Toc72317748)

[7. References: 25](#_Toc72317749)

# 

# 1. RESTful API design

## 1.1 Product API section

|  |  |  |
| --- | --- | --- |
| **Operation** | **URL** | **Description** |
| GET | api/products/{id} | Get the product by id. |
| POST | api/products | Create a new product. |
| PUT | api/products/{id} | Update a product. |
| DELETE | api/products/{id} | Delete a product by its id. |
| GET | api/products?page={pageNr}&size={size} | Get all products with pagination. |
| GET | api/products?page={pageNr}&size={size}&price={price} | Get all products with pagination by the given price. |
| GET | api/products/platform/{platformName}?page={pageNr}&size={size} | Get all products with pagination by given platform name. |
| GET | api/products/platform/{name}?page={pageNr}&size={size}&price={price} | Get all products with pagination by given price and platform name. |
| GET | api/products/search/{productName}?page={pageNr}&size={size} | Get all products with pagination by the given name. |
| GET | api/products/statistics | Get the general statistics of the products. Returns, the total number of all products, the total number of the products by platform. |
| GET | api/products/statistics/{id} | Get the product turnover of the product with the given id. |
| GET | api/products/statistics/top/{yyyy-MM-dd} | Get the top 5 most sold products during the week of the given date. |
| GET | api/productPlatforms?retrieval={mode} | Get all product platforms. |
| GET | api/productPlatforms?page={pageNr}&size={size} | Get all platforms by pagination. |
| GET | api/productPlatforms/{id} | Get a product platform by id. |
| POST | api/productPlaforms | Create a new product platform. |
| PUT | api/productPlarforms/{id} | Update a product platform. |
| GET | api/cdKeys/product/{id}?page={pageNr}&size={size}&used={isUsed} | Get all available or unavailable cdKeys with pagination by a product id. |
| GET | api/cdKeys/{id} | Get a cdKey by its id. |
| POST | api/cdKeys | Create a new CdKey for a product. |
| PUT | api/cdKeys/{id} | Update a CdKey for a product. |
| DELETE | api/cdKeys/{id} | Delete a CdKey by its id. |
| GET | api/genres?page={pageNr}&size={size} | Get all genres by pagination. |
| GET | api/genres?retrieval={mode} | Get all genres. |
| GET | api/genres/{id} | Get a genre by its id. |
| PUT | api/genres/{id} | Update a genre by its id. |
| POST | api/genres | Create a new genre. |

## 1.2 Image API section

|  |  |  |
| --- | --- | --- |
| **Operation** | **URL** | **Description** |
| POST | api/images/upload | Upload a JPEG, PNG, GIF image. |
| GET | api/images/getImage/{imageName: .+} | Get an image by name. |
| POST | api/images/upload/product/{id} | Uploads the image file to the server for a product with the given id |

## 1.3 User API section

|  |  |  |
| --- | --- | --- |
| **Operation** | **URL** | **Description** |
| POST | api/auth/signin | Login a user in the system |
| POST | api/auth/singup | Registers a member to the application |
| POST | api/auth/refreshToken | Gives back the new access token if the given refresh token is available |
| GET | api/auth/userInfo/{refreshToken} | Returns the user details based on the refresh token |
| GET | api/users/{id} | Returns a user with the given id |

# Graphical user interface, application, website Description automatically generated Diagrams

## 2.1 C4 diagrams

### 2.1.1 Context diagram

### Diagram Description automatically generated2.1.2 Container diagram

### Diagram Description automatically generated2.1.3 Components diagrams

[To products](#_2.1.3.1_Product_components) [To images](#_2.1.3.2_Image_components) [To users](#_2.1.3.3_User_components)

#### 2.1.3.1 Product components section

[To container](#_2.1.2__Container) [To system context](#_2.1.1_Context_diagram) [To UML](#_Product_section)

#### 2.1.3.2 Image components section

[To container](#_2.1.2__Container) [To system context](#_2.1.1_Context_diagram) [To UML](#_Image_components_&)

Diagram

Description automatically generated

#### 2.1.3.3 User components section

Diagram

Description automatically generated[To container](#_2.1.2__Container) [To system context](#_2.1.1_Context_diagram) [To UML](#_2.4.1_User_section)

## 2.2 CI setup diagram

Diagram

Description automatically generated

## Diagram, Word Description automatically generated2.3 Entity relation diagram

The major parts of the ERD diagram can be summarized into three sections:

### The product section:

The main table for storing the product information is the “Products” table. It contains the common fields that each type of product shares and the primary key as “product\_id”, used in other tables for reference. One of the common fields, the “platform\_id”, stores as a foreign key the primary key of the “ProductPlatform” table, representing the many to one relationship as many products have exactly one platform.

The cdkey table, used to store the cdkey id and sequence, has the field “product\_id” as a reference for the many to one relationship, many products have one cdkey.

Currently, there are only two types of products video games and software products, each type has its table and contains the “product\_id” as their primary key which is also a foreign key.

The video game table has a many to many to relationship to the genres table, many products have many or one genre. The relationship is represented in the “Product\_Genres” table which has as fields the “product\_id” and the “genre\_id” as foreign keys.

The limitations for the products are that once a product has been persisted its type cannot be changed and to persist a product its platform and genres need to firstly exist in their tables. For persisting a cdkey its product must first exist in the “Products” table.

### The user section:

The main table of the user section is the “Users” table which contains the user information like the first name, last name, email, and password as fields. The users' table has many to many relationship to the Roles table used to store each role a user can have. The relationship is represented in the “Users\_Roles” table that contains the primary key of the users the table and the primary key of the roles table as foreign keys and fields.

The limitations of the users are that to persist a user a role must first exist and that a user role cannot be removed.

### The order section:

The main table of the order section is the “Orders” table which stores the “order\_id” as the primary key, the creation date, and the total price as fields. The order table has a many to many relationship with the “OrderItems” table, many orders have many orders items, and a many to zero or many relationship with the “Users” table as an order has one user represented by the field “user\_id” in the “Orders” table. The order items relationship is represented in the “Orders\_OrderItem” table.

The “OrderItems” table contains the ordered product quantity and the “product\_id”. As relationships, it has a many to one relationship as many order items have exactly one product and a many to many relationship with the CdKey table as many order items have many cdkeys represented in the “OrderItems\_Cdkeys”.

The limitation of the order section is that to persist an order the user and the product from its order items must exist.

## 2.4 UML diagrams

The following diagrams are for the back-end architecture.

The controllers are responsible for the communication with the client thru the means of endpoints and the request types. Each resource has its own controller and each controller has at least one reference to a service interface

The service layer is responsible for retrieving, creating, and updating the models, performing application-specific logic and manipulations

The repository layer will call the database and perform the requested operations. Each entity has a repository. Spring Data JPA is used for managing the entities so each abstraction extends from one of the JPA repositories type (JpaRepository, CrudRepository, PagingAndSortingRepository).

### 2.4.1 User section

[To C4component](#_2.1.3.3_User_components)

#### Graphical user interface, text Description automatically generated2.4.1.1 Components

#### A screenshot of a computer Description automatically generated with low confidence2.4.1.2 Models

### 2.4.2 Product section

#### 2.4.2.1 Product components

[To C4component](#_2.1.3.1_Product_components)

#### Graphical user interface Description automatically generated with medium confidence

#### 2.4.2.2 Product models

### A screenshot of a computer Description automatically generated with medium confidence

### 2.4.3 Image and common section

[To C4component](#_2.1.3.2_Image_components)

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated

### 2.4.4 Exception sectiGraphical user interface, text Description automatically generatedon

# 3. Design decisions

## Spring Boot: The Most Notable Features You Should Know ...3.1 Back-end:

### 3.1.1 Spring boot:

One of the frameworks that I choose for this project is Spring boot, a back-end framework based on Java.

**Some of spring boots disadvantages:**

* From a learning perspective, Spring boot doesn’t cover most of the details of Spring, so if you never worked with Spring before details like proxies, dependency injection and AOP you’ll have a harder time during troubleshooting or modifications
* Due to its automation you can very easily miss many concepts of springs ecosystems like Spring Security, Spring Integration etc..

**I prefer it because:**

* It is flexible as it allows configuring beans in multiple ways like XML, Annotations and JavaConfig. I mostly tends towards annotations as they offer the fastest and easiest solutions but as a second option or by necessity I would rely on JavaConfig.
* It makes developing spring-powered applications better by making Spring or Spring MVC easier to use with its auto configuration
* It provides starters, like the spring boot starter web with all the needed dependencies prepackaged for developing an application to expose RESTful services
* It is open-source and quite popular with developers offering good documentation
* It simplifies integration with JPA/Hibernate ORM
* It very compatible with most technologies as it supports NoSQL databases, Oracle, PostgreSQL, MongoDB and several other processes
* It utilizes dependency injection as Spring at its core it’s a dependency injection container

### 3.1.2 Date persistence

For data persistence, I will be using Spring Data JPA as it will speed up the development time of the application, keep everything simple and remove a lot of boilerplate code. Spring Data JPA is an add-on for JPA, an abstraction over the Data Access Layer that uses JPA(Java Persistence API) and ORM implementation like Hibernate. Because it is based on a JPA specification it uses all its defined features, it enables the entity objects and their metadata mappings and it also enables the entity manager, responsible for persisting and retrieving the entities from the database, while keeping mostly everything under the hood.

Spring Data JPA also removes the need to write native SQL statements by providing a set of Interfaces(Repositories) that define query methods for dealing with data and for which Spring

automatically provides an implementation. The name of the methods declared in the repository will be converted to low-level SQL queries

## 3.2 Front-end:

### 3.2.1 Frameworks criteria

The most relevant criteria’s are for me:

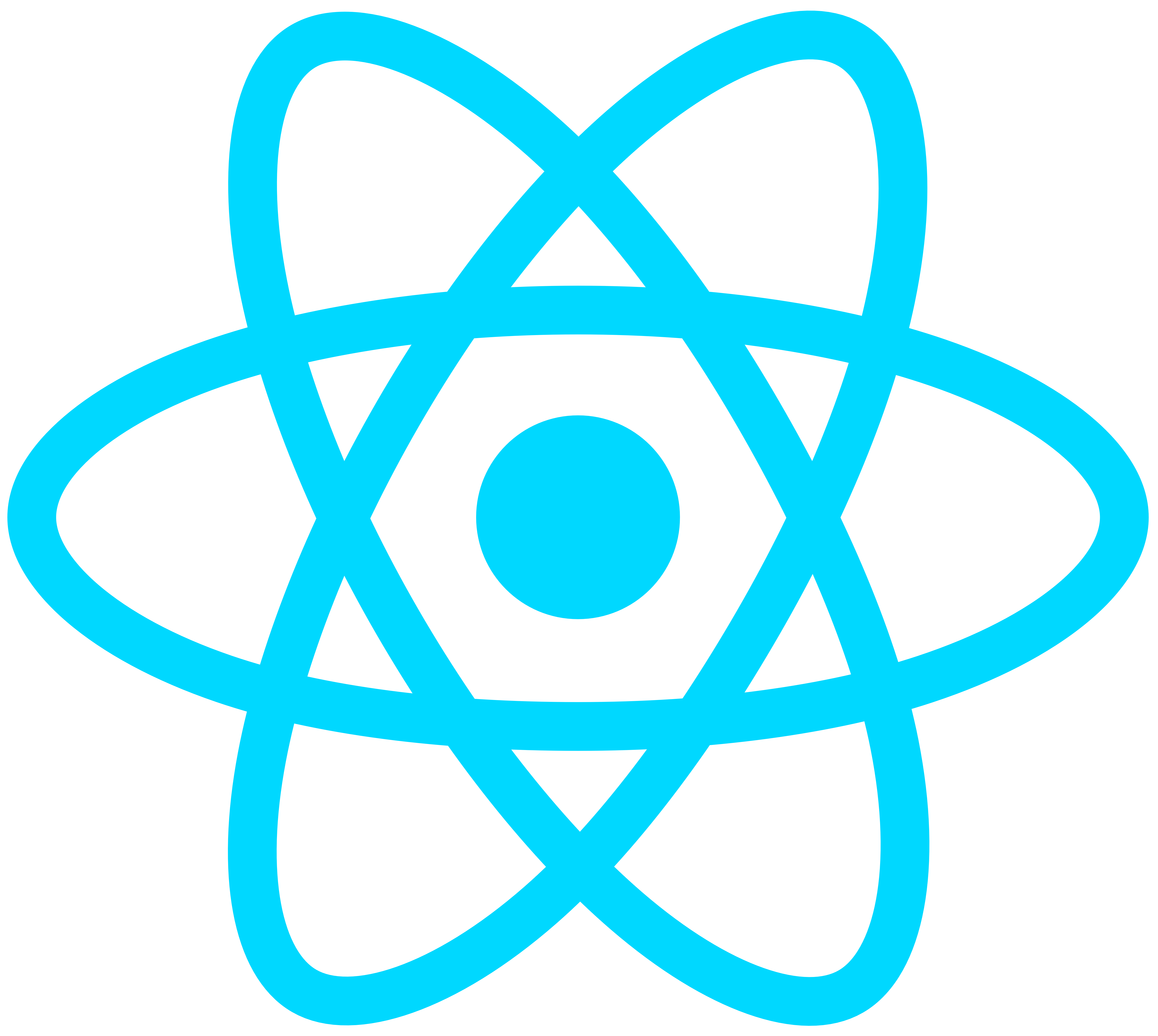
|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Angular | React | Vue |
| Support community | 9 | 9 | 7 |
| Opinionated | 9 | 6 | 6 |
| Documentation | 9 | 8 | 7 |
| Ease of learning | 6 | 7 | 7 |
| Ease of use | 6 | 7 | 7 |

### 3.2.2 Frameworks

Although there are a lot of frameworks when it comes to the front-end I try to limit myself to one of the three recently popular frameworks: Angular, React and Vue.

#### VUE Logo PNG Transparent & SVG Vector - Freebie Supply3.2.2.1 Vue

|  |  |
| --- | --- |
| Pros | Cons |
| Empowered use of HTML  Circumstantial documentation  Adaptable and flexible  Good integration, can be used for both building single-page applications and more difficult web apps  Tiny size, can weight around 20KB  UI and behavior are part of components  One-way and two-way data-binding | Young framework  Lack of resources  Risk of over flexibility |



#### 3.2.2.2 React

|  |  |
| --- | --- |
| Pros | Cons |
| Use of JSX (an HTML-like syntax) for templating  Detailed documentation  Very fast due to the Virtual DOM implementation and various rendering optimizations  Great support for content-focused applications  Implements Functional Programming  Offers support also for typescript  Good size, around 43k  UI and behavior are part of components | JSX mixes templating with logic which can become confusing as some points  Data-binding is one-way only  React is unopinionated so most of the developing choices are up to the developer  React is moving away from class-based components |

#### Angular (web framework) - Wikipedia3.2.2.3 Angular

|  |  |
| --- | --- |
| Pros | Cons |
| Empowered use of HTML  Mature framework  Exceptional support for typescript  Allows intellisense and autocomplete inside of component external HTML template files  It comes with the Angular CLI  Detailed documentation  Two-way data-binding  Uses dependency injection  Complete setup on startup | Enforces the MVVM pattern  Steep learning curve due to the variety of different structures (Injectables, Components, Pipes, Modules etc.)  Relatively slower performance compared to the other frameworks  Is quite bloated, the zipped file size is around 143k  Is not as flexible and universal |

### Angular (web framework) - Wikipedia3.2.3 Framework conclusion:

In conclusion, Angular is my framework of choice because it exceeds the others regarding detailed documentation, community support, and is highly opinionated. As a framework, it provides an easy and complete startup by making use of the Angular CLI. Being the most “opinionated” Angular provides all the tools needed to make it easy to build my web application, including but not limited to state management, routing, and dependency management. The enforced MVVM pattern is advantageous as it provides a separation of concern. I want to have a better understanding and correct use of the pattern as this will be my first project tackling it.

Due to its variety of structures, Angular scores are lower in ease of use and ease of learning criteria. These structures will make the project more challenging but, after a deeper dive in understanding them, they will surely prove useful in the long run.

### Bootstrap 4 is released - SD Times3.2.4 Bootstrap

An increasingly popular and feature rich front-end design framework. Bootstrap focuses on the design aspect of the application UI enchanting the already existing elements that come with HTML5 but also providing lots of custom additions.

**Some disadvantages:**

* The styles are verbose and can lead to lots of output in HTML
* It increases loading time for the pages

**The main reasons for choosing bootstrap are:**

* It’s extremely easy to set up either by CDN or npm
* The speed of development is drastically increased, as bootstrap provides ready-made components which I can easily customize and build on top using CSS or jQuery scripts
* I am familiar with the framework as I have used it a bit in the past
* It uses jQuery for handling scripts which I am familiar with
* It has very good community support and documentation

### Font Awesome Icons | Drupal.org3.2.5 Font Awesome

Font Awesome is another design framework that I decided to use because of the big options of icons that it provides:

**Some advantages:**

* A very light design framework
* It’s extremely easy to set up and use either by CDN or npm
* It provides a lot of icons to choose from
* It is open source

**Some disadvantages:**

* Not all icons and variations of the icons are available in the free version

# 4. Wireframes

## 4.1 Navigation & Product catalog

## Product details

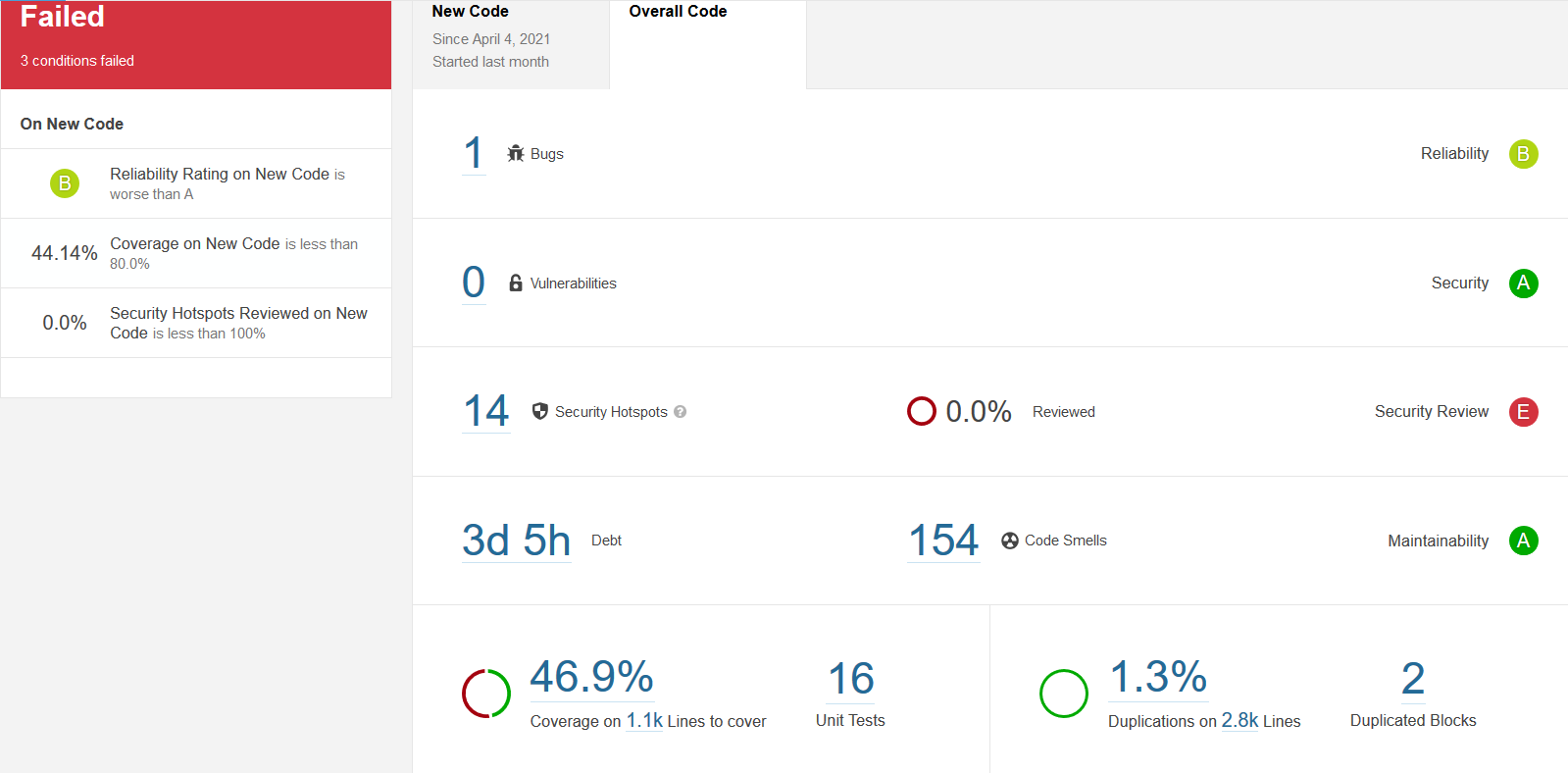
Table

Description automatically generated

# 5. Quality assurance metrics tool results

For the inspection of the code quality I’ll be using SonarQube which provides a detailed report of the bugs, code smells, vulnerabilities and code duplications inside the application. The tests are usually run on each commit and are addressed in the next one. Sometimes tests might be runed locally during development to ensure a bug free code. For measuring the code coverage I have added JaCoCo which generates a xml report after if each test task that is later read by SonarQube. Based on the results run so far, the applications remaining main issues are related to the security, the high number of code-smells, the low code coverage and the little amount of comments. So far I have tackled the bugs, lowered the high number of code-smells and the number of duplications. For rising my code coverage I have implemented 13 additional test methods.

## Last test results:



# 6. Design document versioning

|  |  |
| --- | --- |
| **Version/Date** | **Description** |
| V1.0 / 04-03-2021 | * Created ERD based on temporary data description from Project Plan V1.0 (currently contains products, users and orders data) |
| V1.1 / 10-03-2021 | * Updated the ERD to contain a legend for cardinality component (relationships) V1.1 * Created UML diagram V1.1 * Created wireframes for product listing and product details * Wrote design decisions |
| V1.2 /16-04-2021 | * Modified diagrams to match new implementations * Splitted diagrams into sections * Added a descriptive text for all the diagrams * Added data persistence section * Added sonarqube analysis section * Updated the Front-end frameworks section from the design decision |
| V1.3/18-05-2021 | * Reorganised sections * Added hyperlinks for easier page nav * Updated the SonarQube test results * Updated UML diagrams content * Create c4 diagrams sections |

## 7. References:

A. (2020, July 13). *Why Choose Spring Boot for your next Business Application*. Inexture. https://www.inexture.com/why-choose-spring-boot-for-your-next-business-application/

Janssen, T. (2020a, May 10). *What is Spring Data JPA? And why should you use it?* Thorben Janssen. https://thorben-janssen.com/what-is-spring-data-jpa-and-why-should-you-use-it/

Janssen, T. (2020b, May 10). *What’s the difference between JPA, Hibernate and EclipseLink*. Thorben Janssen. https://thorben-janssen.com/difference-jpa-hibernate-eclipselink/

Kumar, A. (2021, March 1). *React vs. Angular vs. Vue.js: A Complete Comparison Guide*. Dzone.Com. https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu

Kuts, R. K. (2021, January 24). *Angular vs React vs Vue.js: Comparison of Frameworks - TechMagic*. TechMagic. https://blog.techmagic.co/reactjs-vs-angular-vs-vuejs-what-to-choose-in-2020/

Mihalchenko, A. (2020, December 10). *Pros and Cons of Using Spring Boot*. SCAND. https://scand.com/company/blog/pros-and-cons-of-using-spring-boot/

Neuhaus, J. (2018, September 21). *Angular vs. React vs. Vue: A 2017 comparison - pixelpassion*. Medium. https://medium.com/pixelpassion/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176

Velázquez, L. (2021, January 14). *How to Choose the Right Backend Framework (after years of experience)*. DEV Community. https://dev.to/levivm/how-to-select-a-backend-framework-after-years-of-experience-4fej