Design document for IAS

BluDevil digital marketplace

Members: Alexandru, Emilian E.M.
Tutors: Geurts, Jaap J. ; Guayrin, Brice B.P.B.J.P.
Version: 1.5

1 Abbreviations and Definitions

Terms	Description
SPA	Single Page Application
AOP	Aspect Oriented Programming
XML	Extensible Markup Language
JPA	Java Persistence API
ORM	Object-Relational Mapping
HTML	Hypertext Markup Language
DOM	Document Object Model
MVVM	Model-view-viewmodel
MVC	Model-view-controller
CLI	Command-line interface
HTTP	Hypertext Transfer Protocol
NPM	Node Package Manager
RCE	Remote Code Execution
GDPR	General Data Protection Regulation
JSON	JavaScript Object Natation
JWT	JSON Web Token
SMTP	Simple Mail Transfer Protocol
STOMP	Simple Text Orientated Protocol
FTP	File Transfer Protocol
TLS	Transport Layer Security
SOAP	Simple Object Access Protocol
DoS	Denial of Service

2 Table of Contents

3 Introduction

This document is a written report of the BluDevil SPA design that facilitates the applications overall architecture, overall analysis and decisions.

4 Project description

The project is a digital-marketplace named BluDevil, a SPA that focuses on displaying, managing digital products. Currently the application supports product types such as video games and software products.

The application contains the following systems:

- Product system
- User system
- Chat system

The product system allows administrators to do operations such as creating, deactivating and updating on the products and all the additional product resources(ex. platforms and genres). Viewing, filtering and searching the products is possible by any type of user.

The user system allows the users to register, login and edit their personal information, by traveling to their profile page.

The chat system allows members and admins to have one on one conversations. Additionally, an administrator can delete his chat with a selected member.

5 Design decisions

5.1 Back-end

5.1.1 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

5.1.2 Data 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 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.

5.2 Front-end

5.2.1 Researched 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.

5.2.1.1 Vue

Table 1-Vue-Framework

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

5.2.1.2 React

Table 2-React-Framework

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

5.2.1.3 Angular

Table 3-Angular-Framework

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

5.2.2 Framework criteria

The most relevant criteria's are for me:

Table 4-Framework-Criteria

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

5.2.3 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.

5.2.4 Bootstrap design framework

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

5.2.5 Font awesome design framework

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

6 Diagrams

6.1 C4 diagrams

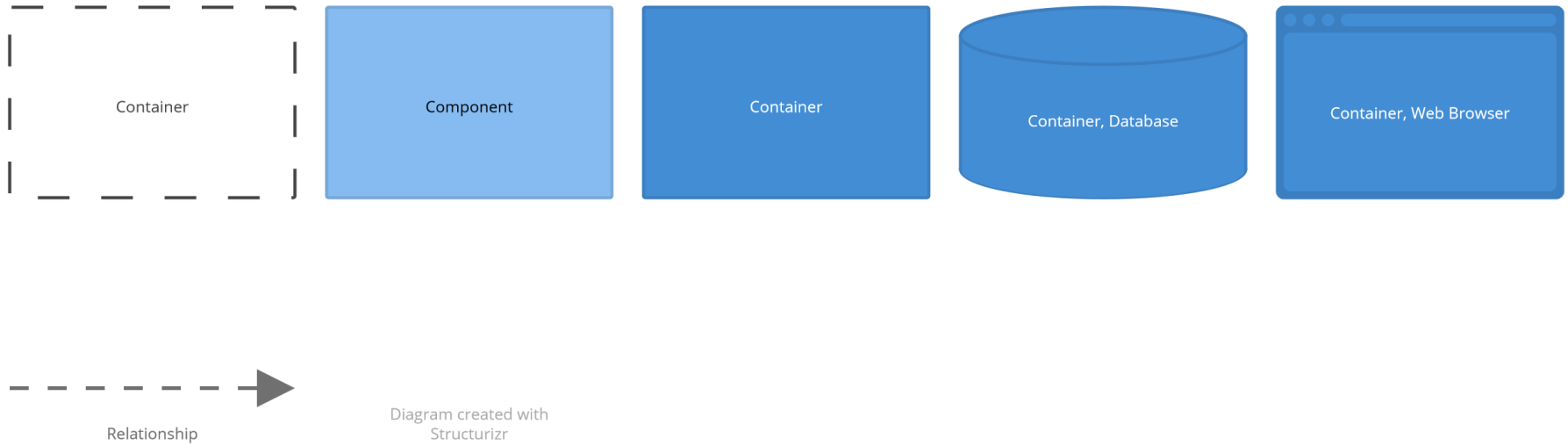


Figure 1-C4-Keys

The shown diagram(Figure 1-C4-Keys) represent the diagram keys and it servs as a legend for the C4 diagrams section.

6.1.1 Context diagram

The following diagram(Figure 2-C4-Context) is the system context diagram and it represent each actors interaction with the BluDevil digital-marketplace.

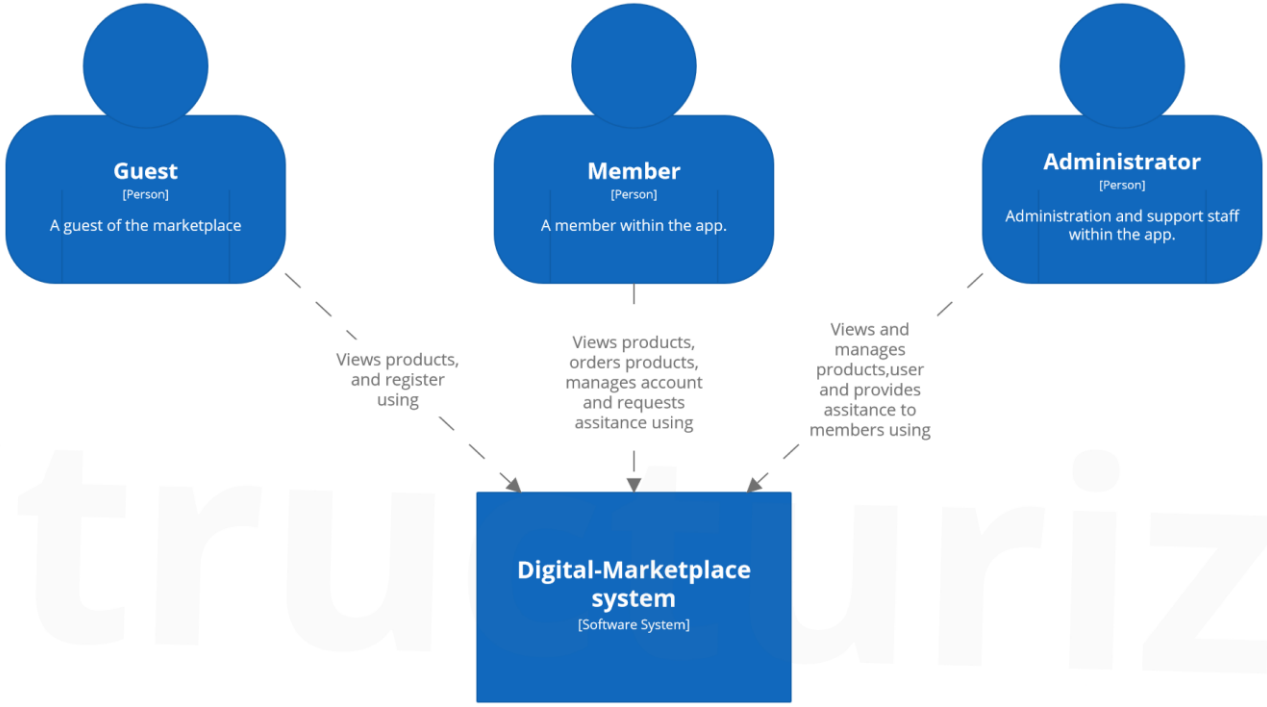


Figure 2-C4-Context

6.1.2 Container diagram

The container diagram(Figure 3-C4-Container) show the high-level shape of the BluDevil software architecture and how each responsibility are distributed across it. Each actors communicates with the GUI of front-end application. The front-end application sends HTTP request to the API application endpoints in order to perform the operations like getting, creating, updating and deleting data.

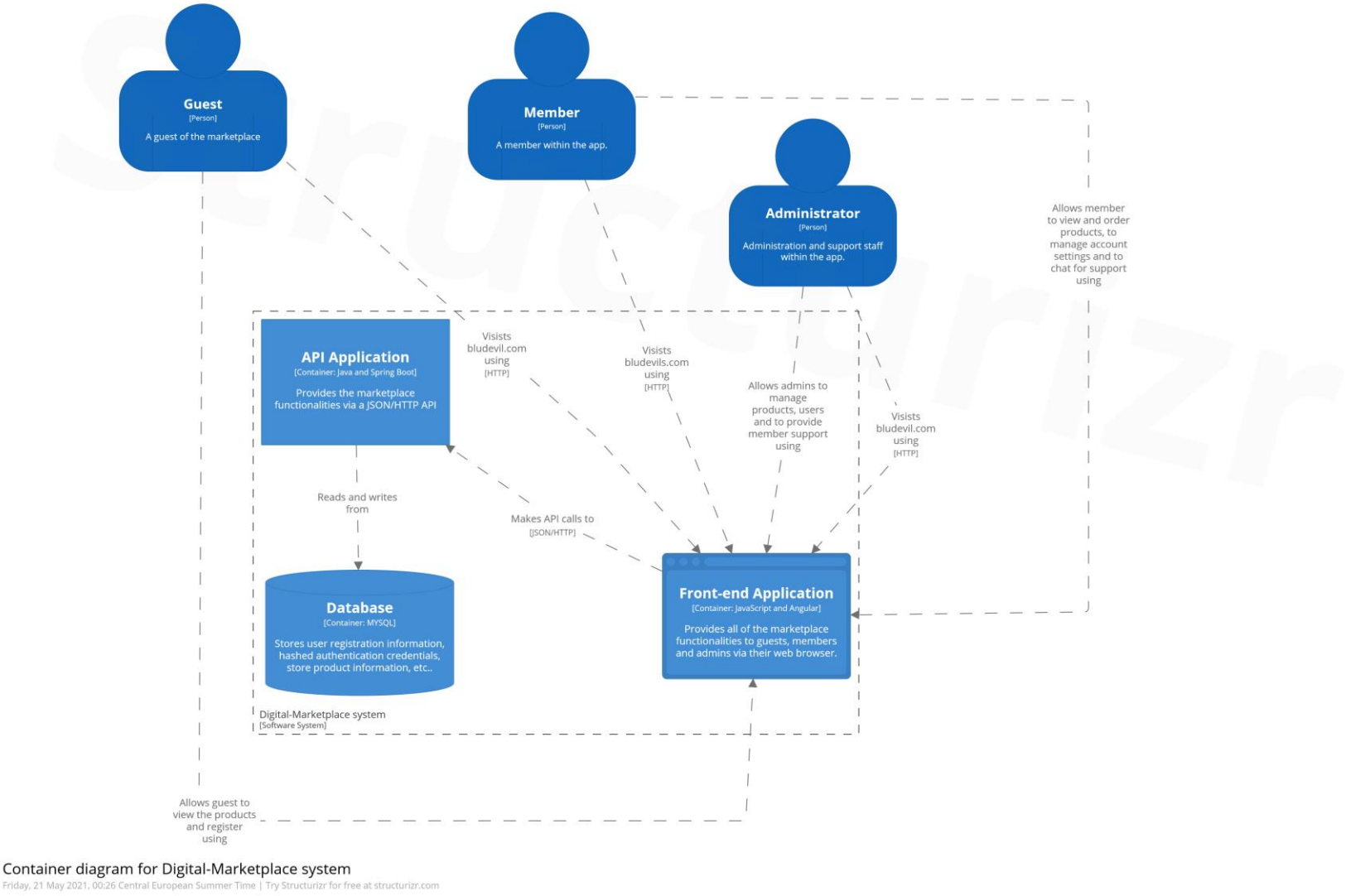
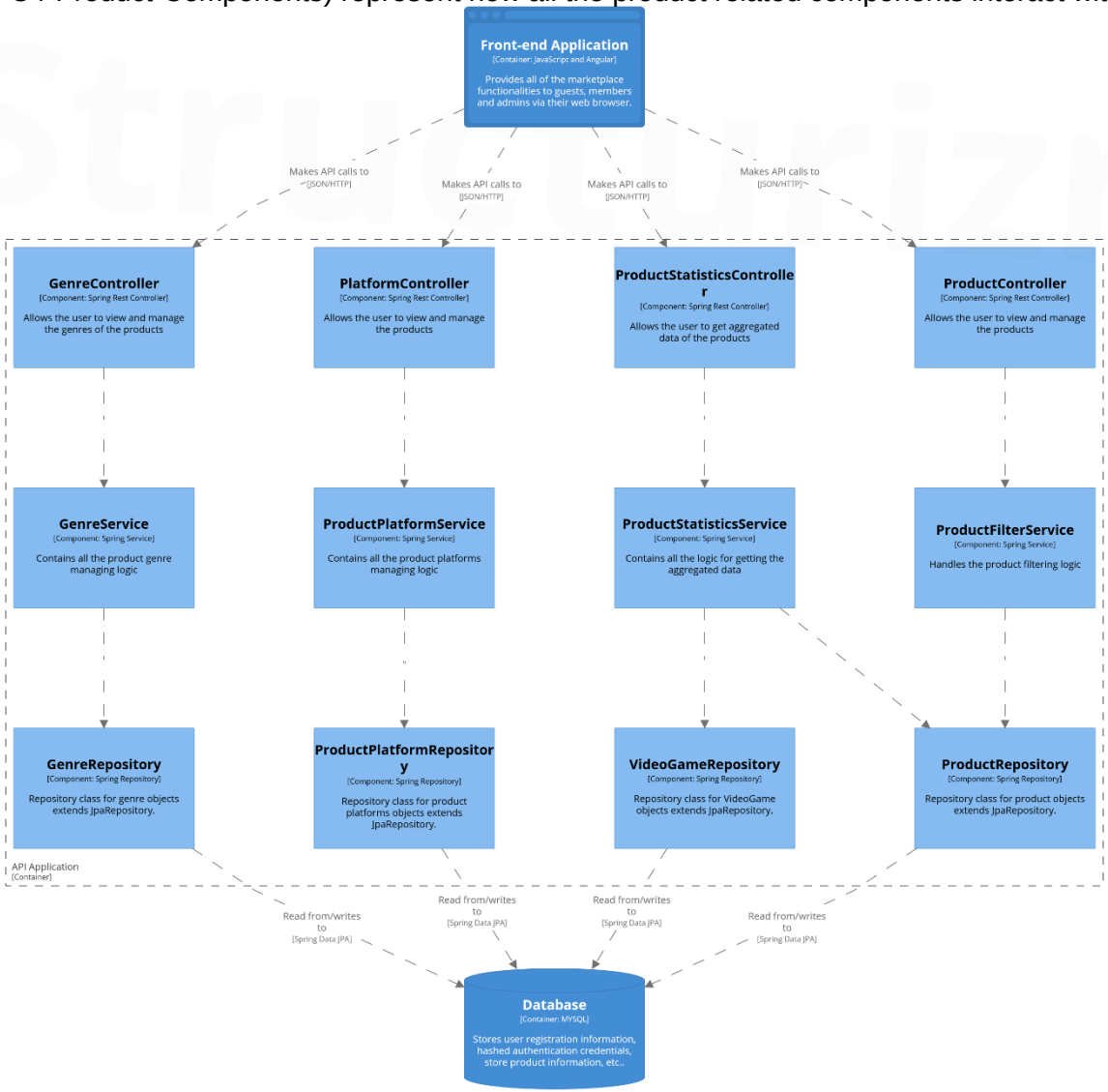


Figure 3-C4-Container

6.1.3 Components diagrams

6.1.3.1 Product C4 components

The following diagram(Figure 4 C4-Product-Components) represent how all the product related components interact with each other.



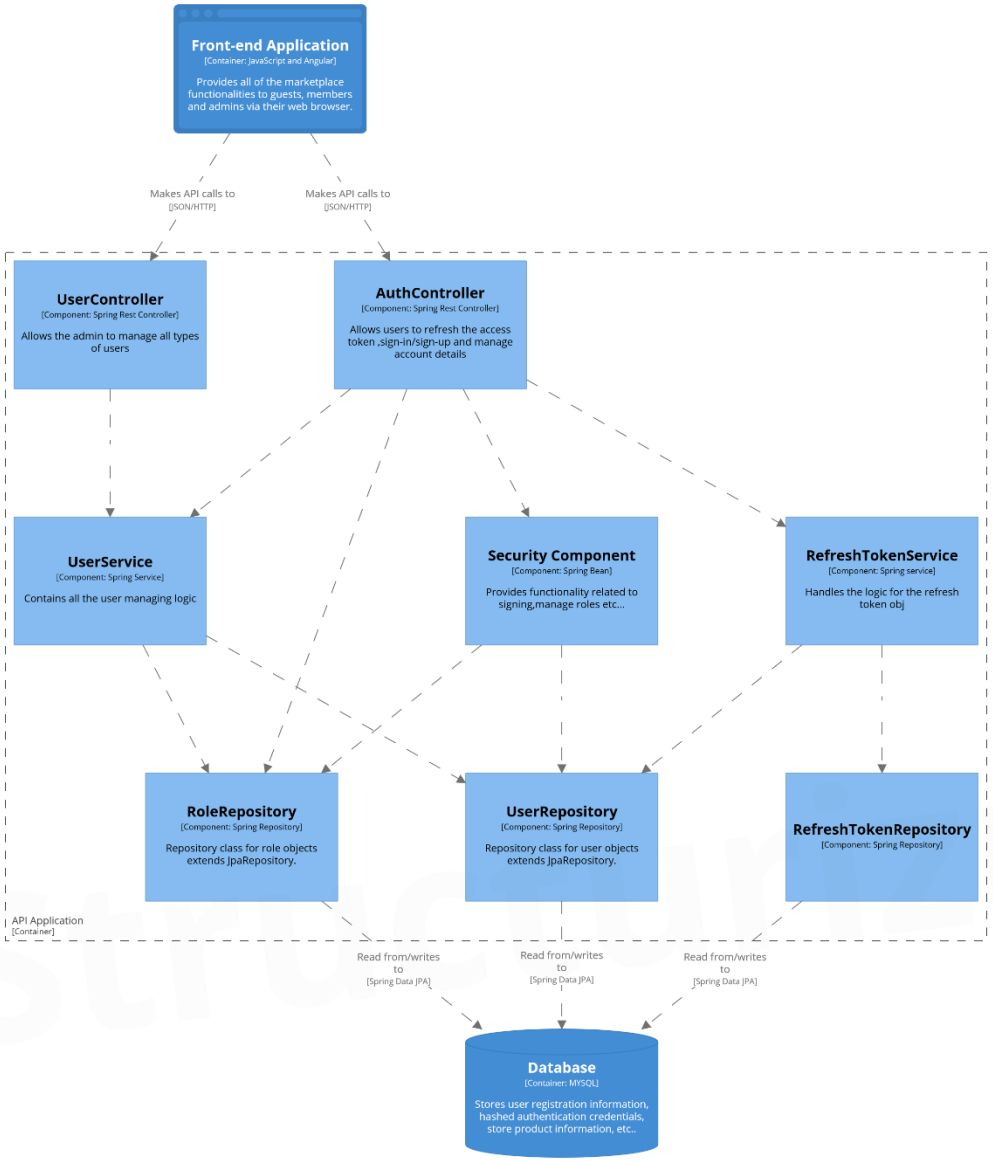
Component diagram for the product section of the WebApplication-v1.0

Figure 4 C4-Product-Components

Try Structurizr for free at structurizr.com | Saturday, 19 June 2021, 17:47 Central European Summer Time

6.1.3.2 User C4 components

The following diagram(Figure 5 C4-User-Components) represent how all the user related components interact with each other.



Component diagram for the user section of the WebApplication-v1.1

Try Structurizr for free at structurizr.com | Saturday, 19 June 2021, 17:36 Central European Summer Time

Figure 5 C4-User-Components

6.1.3.3 Image C4 components

Figure 6 C4-Image-Components shows how all the image related components interact with each other.

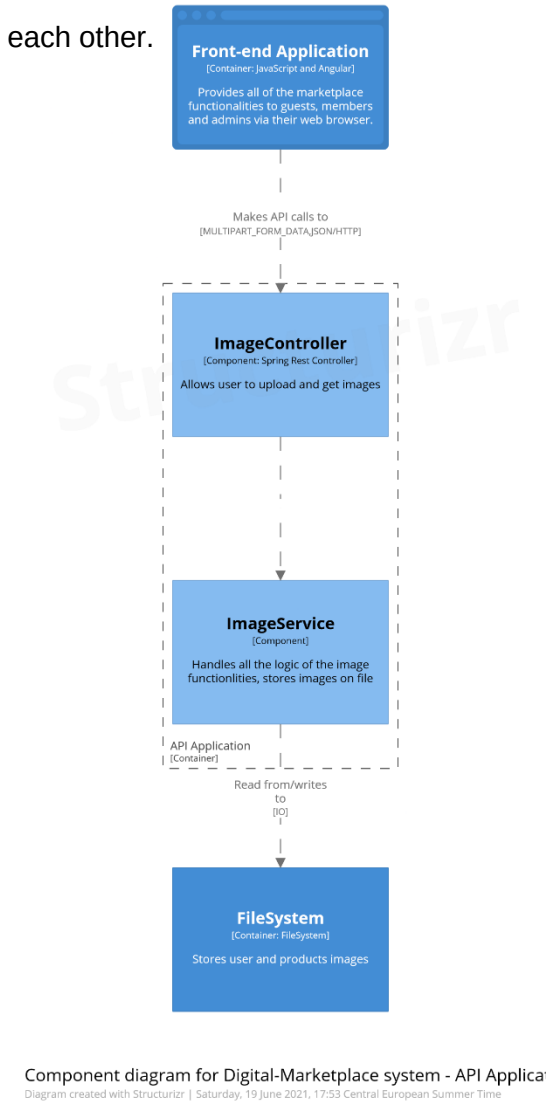


Figure 6 C4-Image-Components

6.1.3.4 Chat C4 components

Figure 7 C4-Chat-Components shows how all the chat components interact with each other.

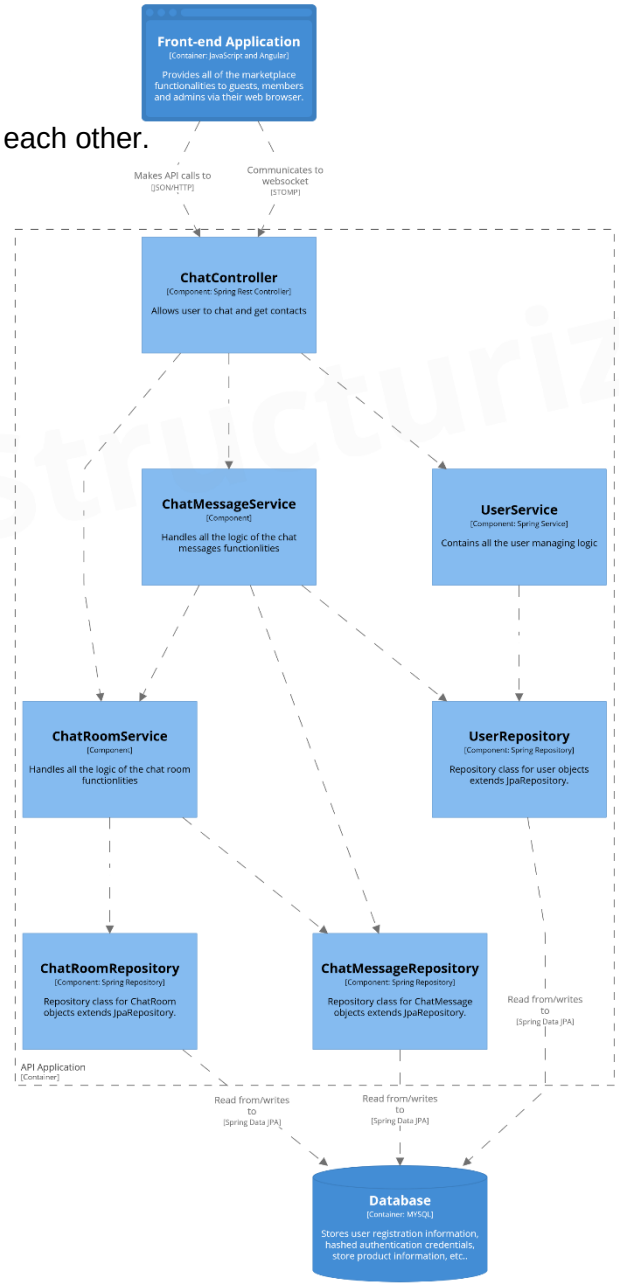


Figure 7 C4-Chat-Components

6.2 CI/CD flow diagram

The following diagram (Figure 8 CI/CD flow)shows the CI/CD pipeline flow.

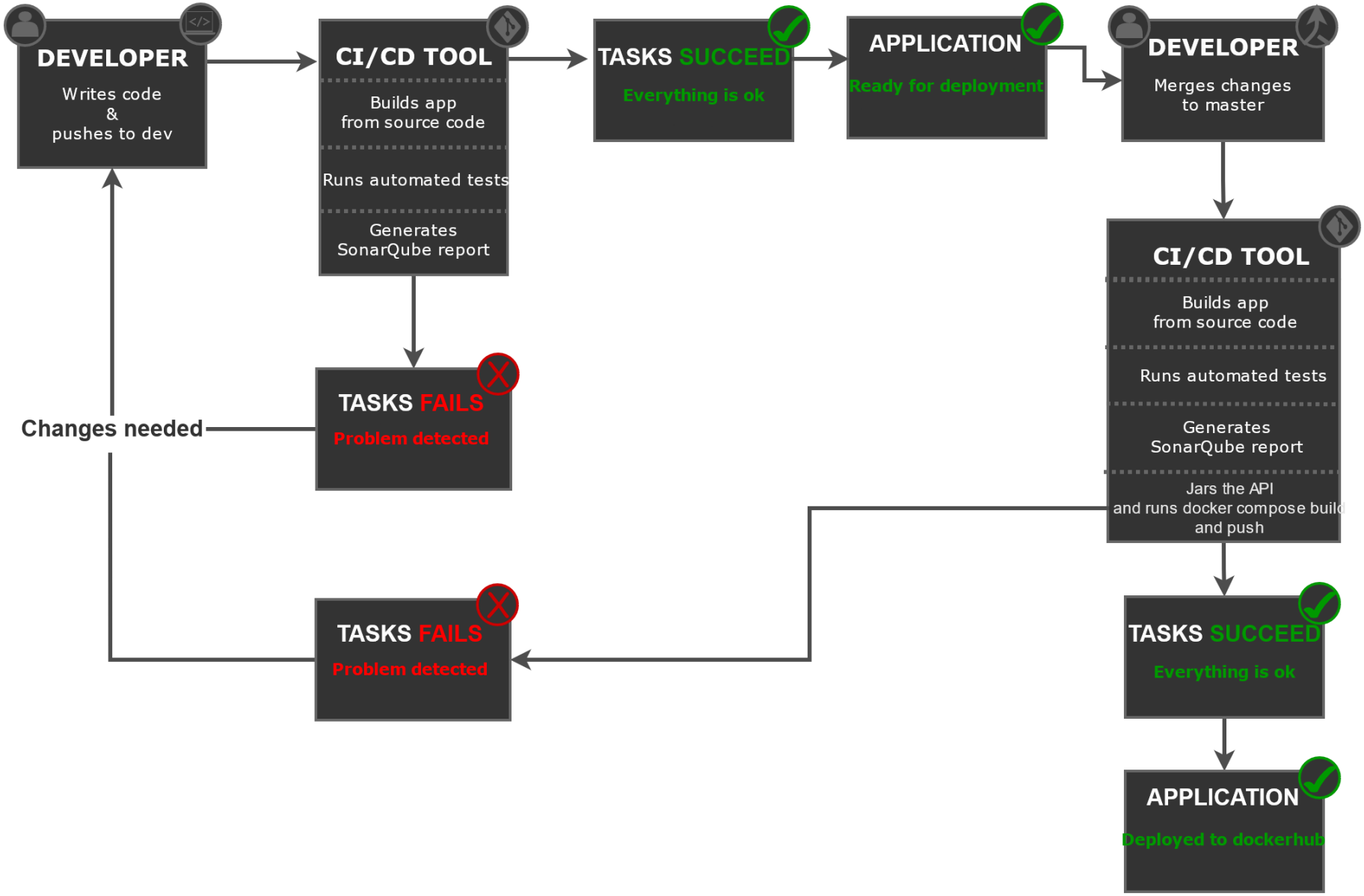


Figure 8 CI/CD flow

6.3 Entity relation diagram(ERD)

The major parts of the ERD diagram can be summarized in three sections, the product section (Figure 9 ERD-Product), the user section(Figure 10 ERD-User) and the chat section (Figure 11 ERD-Chat).

6.3.1 ERD Product

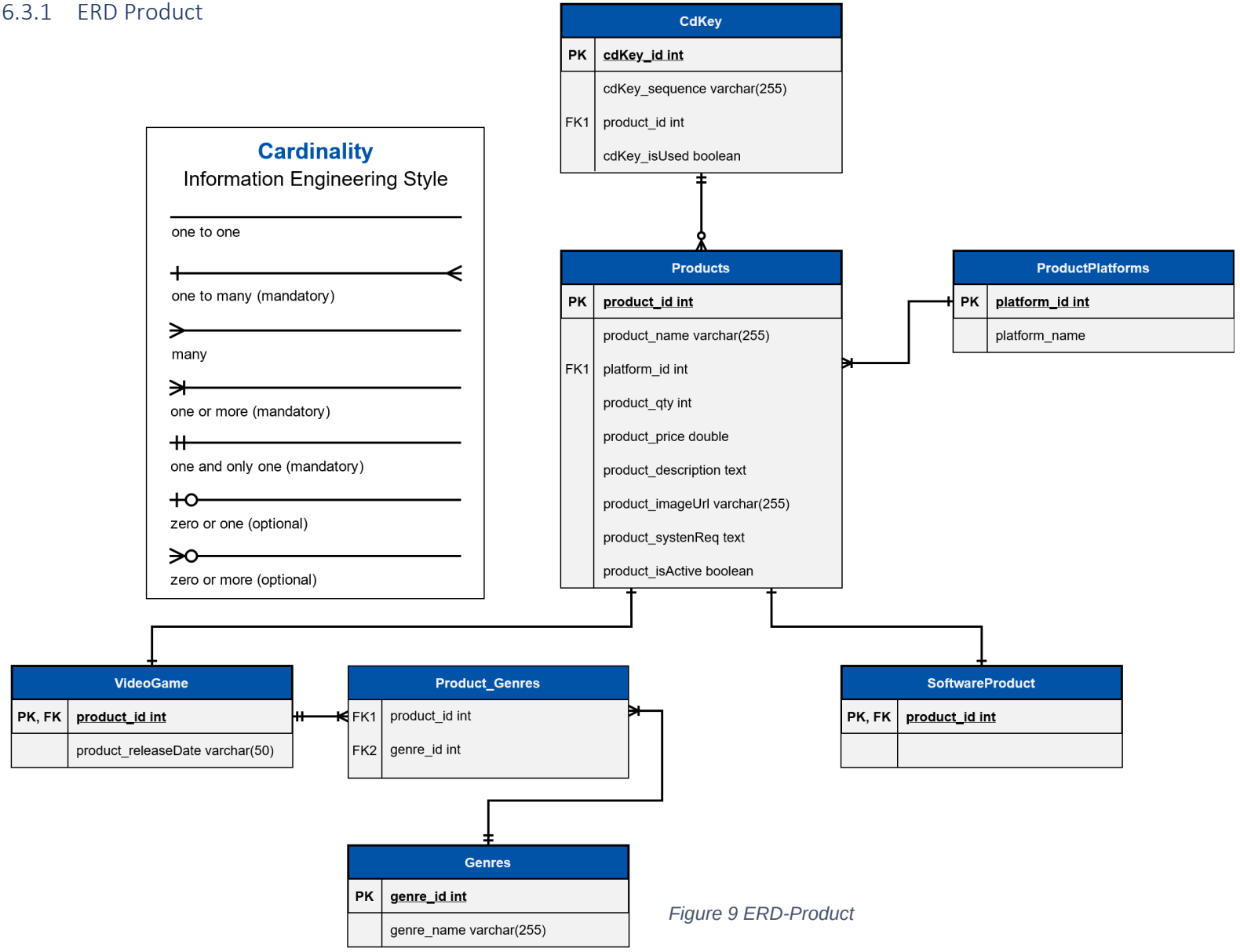


Figure 9 ERD-Product

Looking at **Error! Reference source not found.**, 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 “product_id” and “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.

6.3.2 ERD User

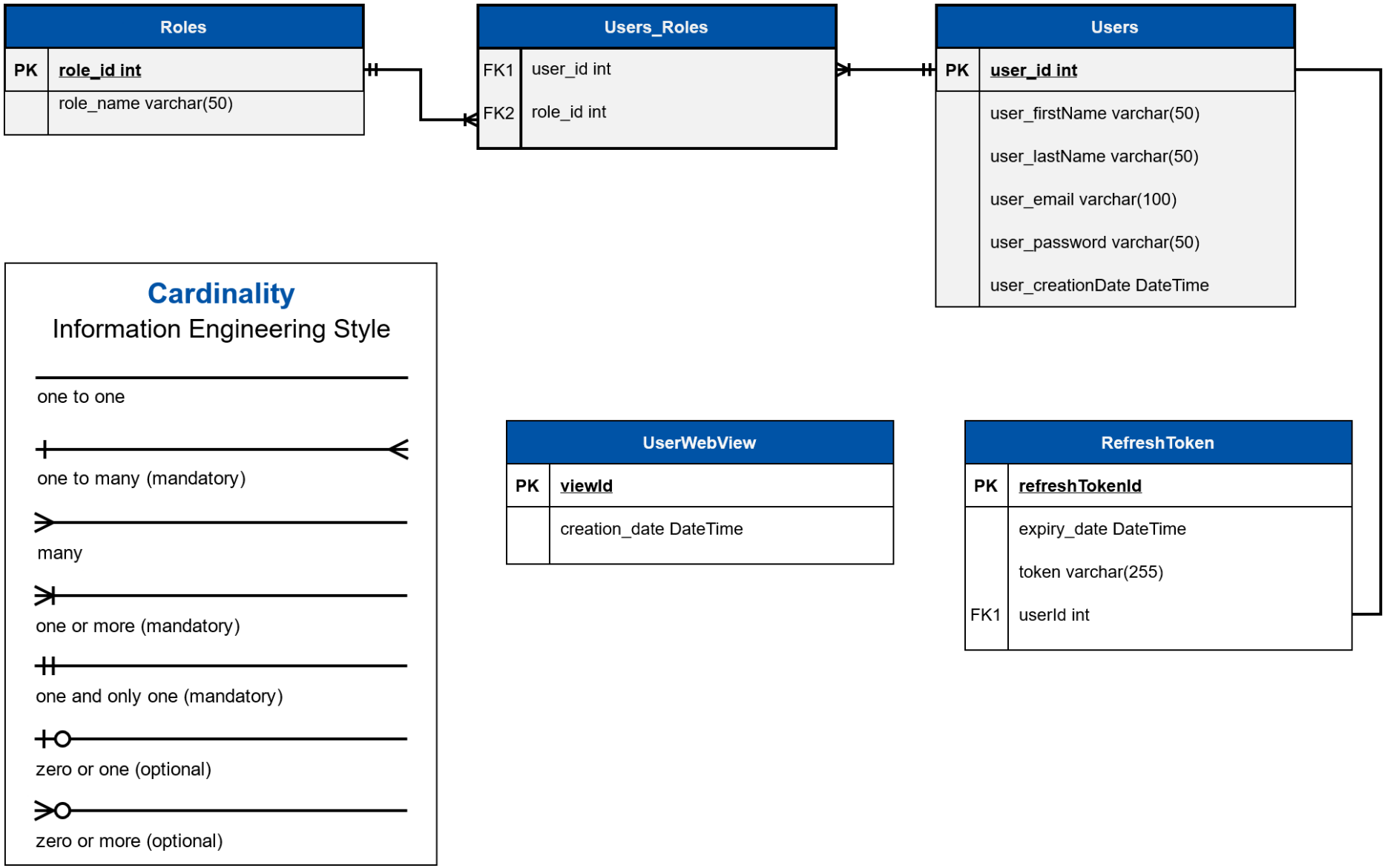


Figure 10 ERD-User

Looking at **Error! Reference source not found.**, 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.

6.3.3 ERD chat

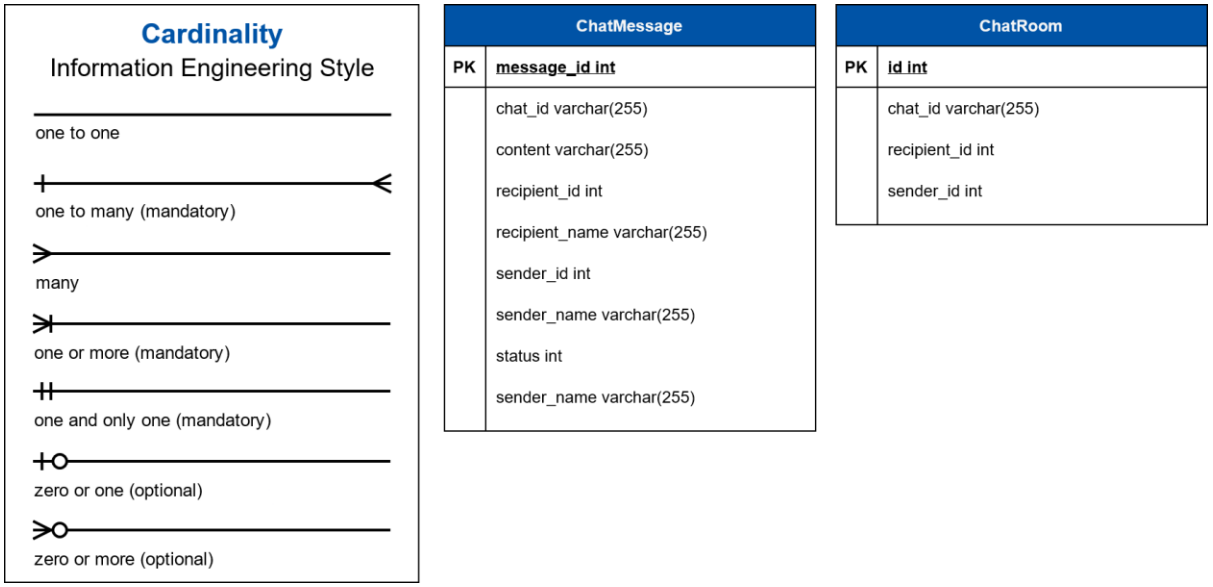


Figure 11 ERD-Chat

Looking at **Error! Reference source not found.**, the main tables of the chat section are the “ChatMessage” table and the “ChatRoom” table. The tables do not have any relations between them but they share a common value which is the “chat_id”. Before persisting any of the two entities the “chat_id” is formatted by concatenating the “sender_id” with the “recipient_id”. The “status” value from the “ChatMessage” is based on the order of the Enum “MessageStatus”(Figure 19 UML-Chat-Models) values.

6.4 UML diagrams

All the UML diagrams represent 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 types (JpaRepository, CrudRepository, PagingAndSortingRepository).
The diagrams are separated by four sections the user section(Figure 15 UML-User-Components & Figure 16 UML-User-Models), the product section(Figure 12 UML-Products-Components & Figure 13 UML-Product-Models), the image section(Figure 17 UML-Image-Components) and the chat section(Figure 18 UML-Chat-Components & Figure 19 UML-Chat-Models).

6.4.1 UML product

6.4.1.1 UML product components

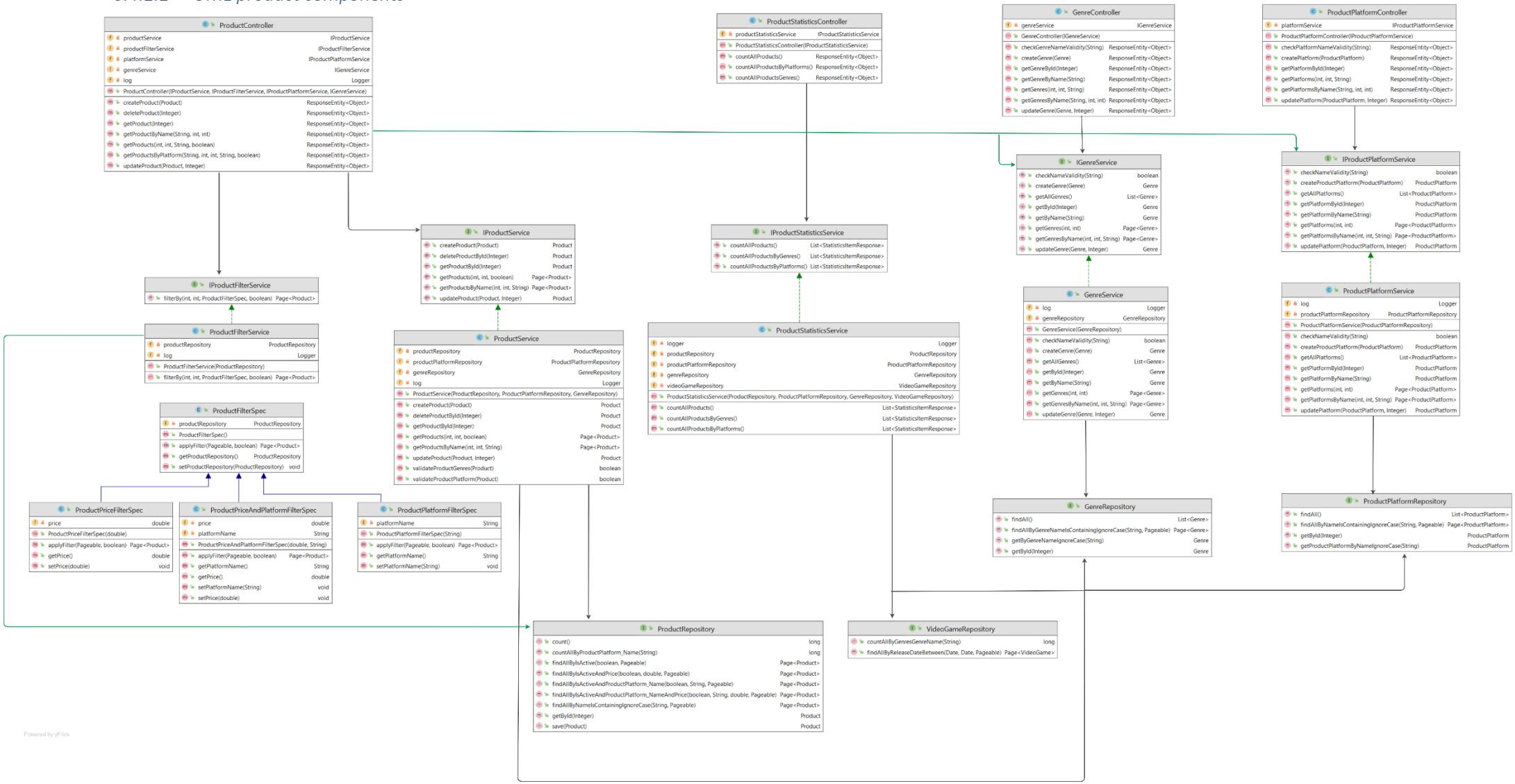


Figure 12 UML-Products-Components

6.4.1.2 UML product models

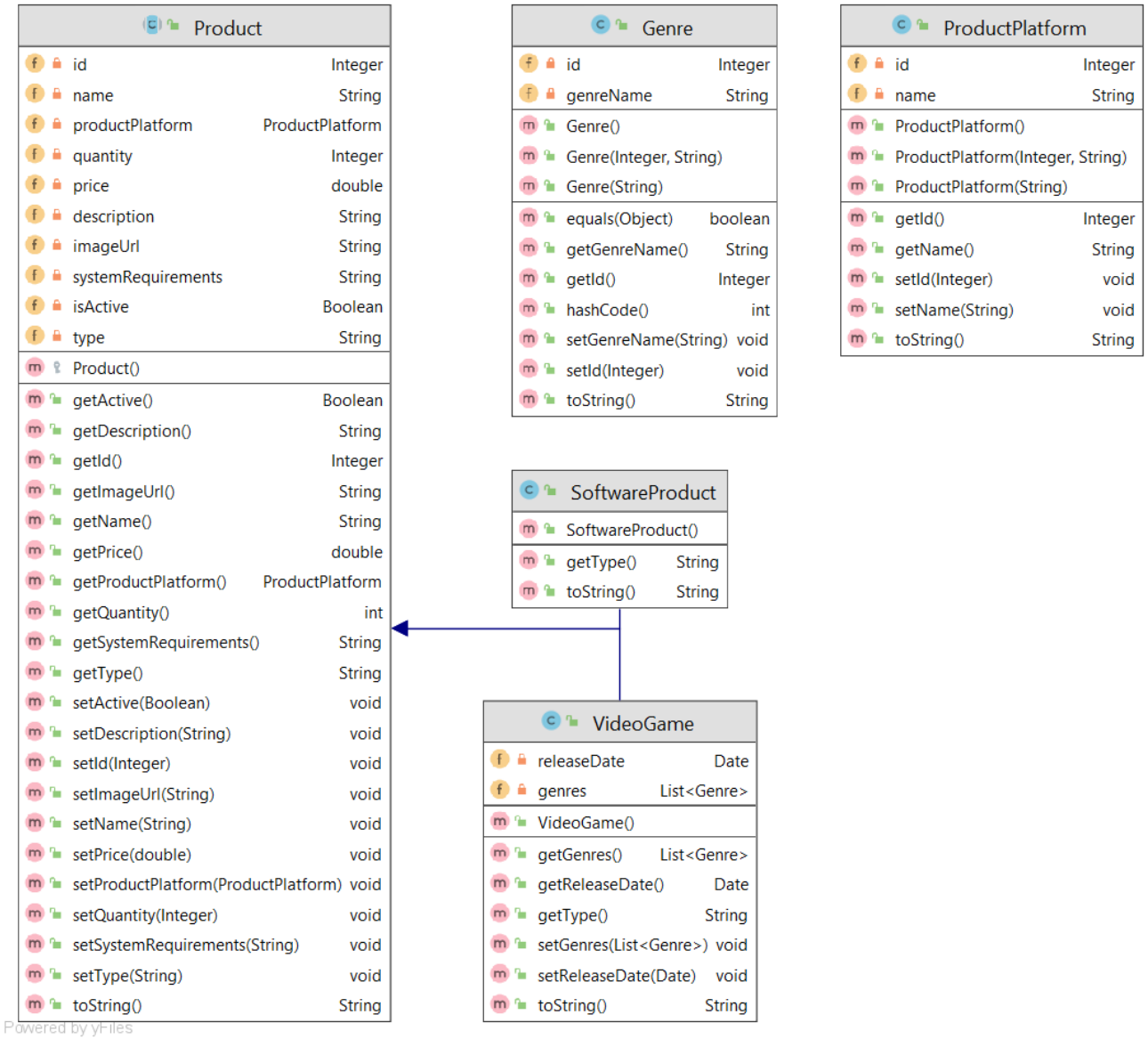


Figure 13 UML-Product-Models

6.4.2 UML user

6.4.2.1 UML user components

Figure 15 UML-User-Components

6.4.2.2 UML user models

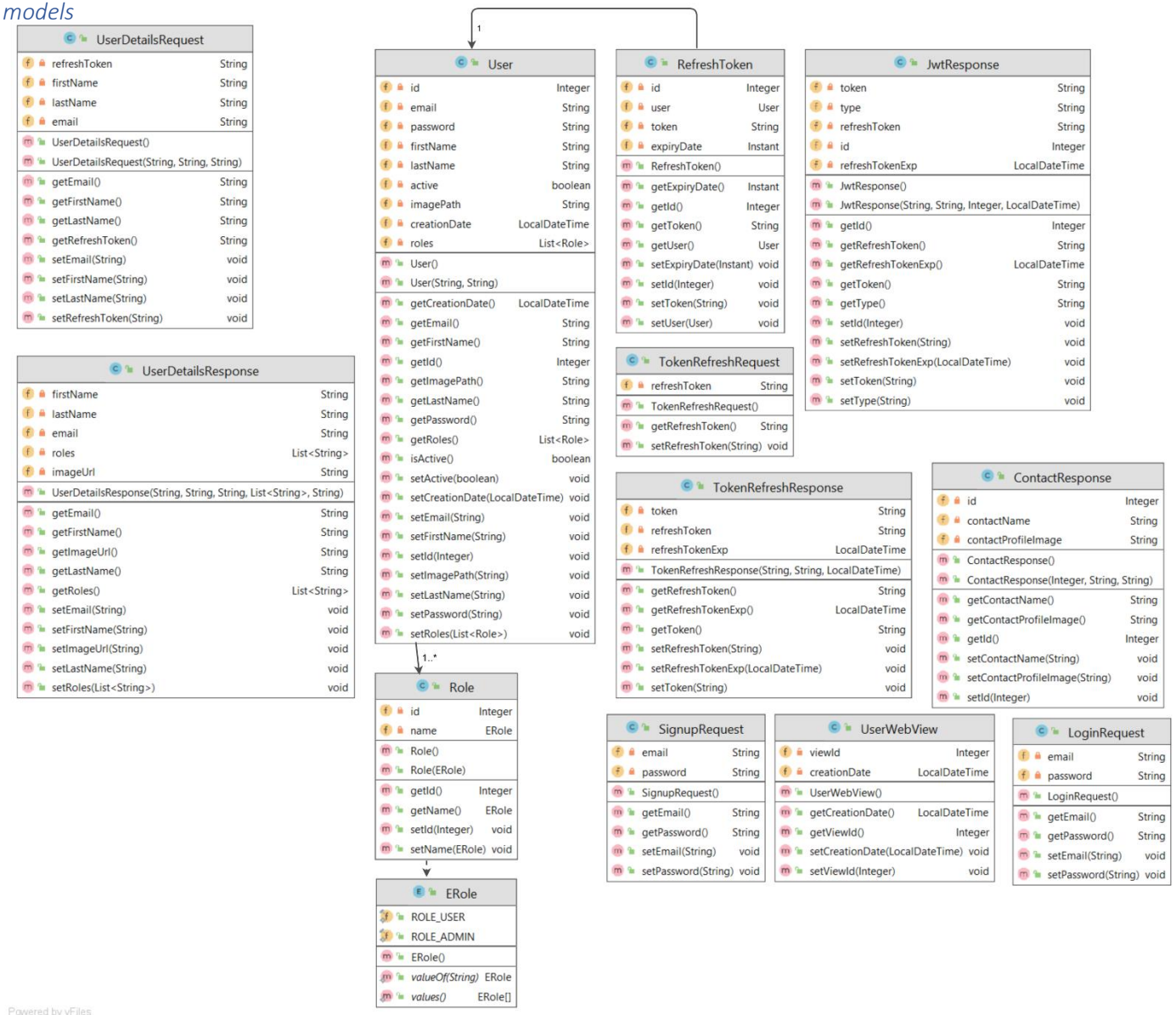


Figure 16 UML-User-Models

6.4.3 UML image

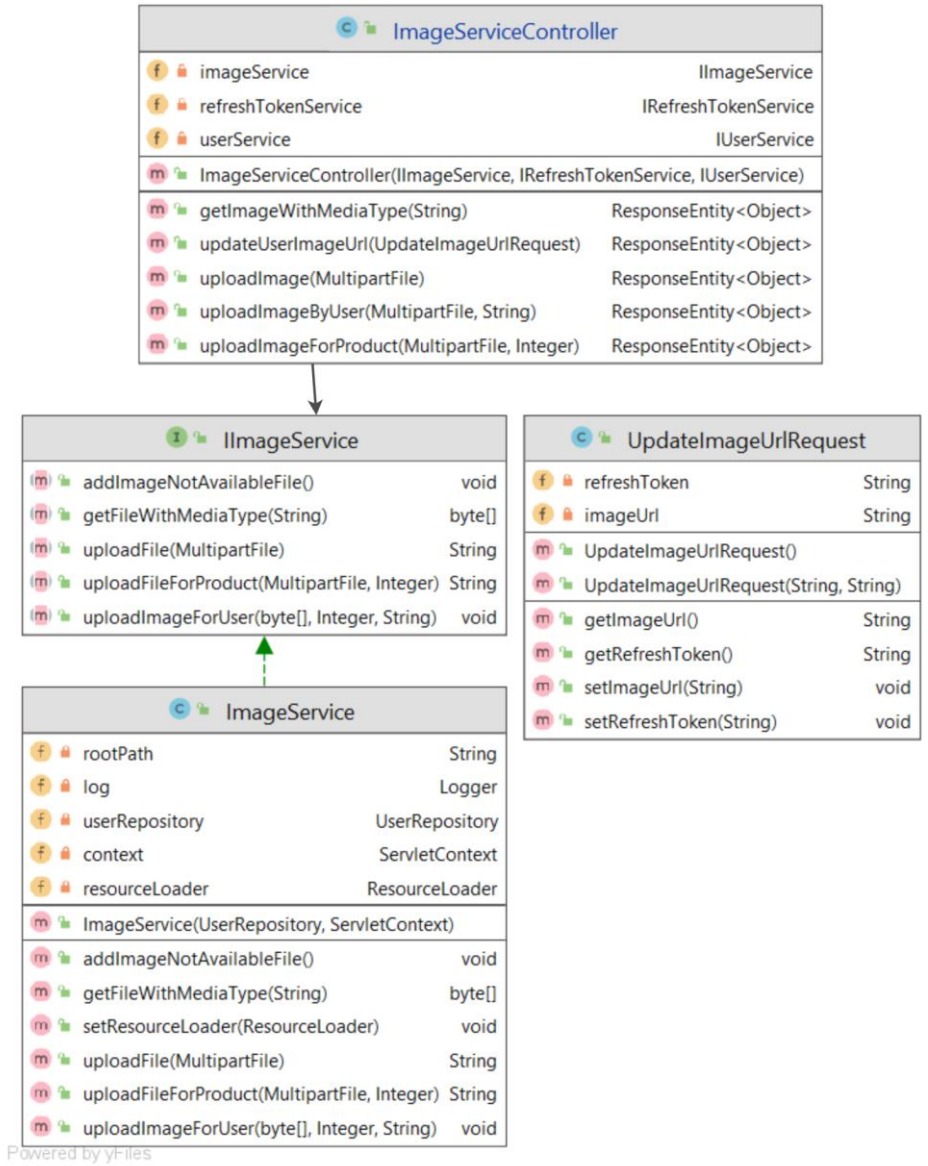


Figure 17 UML-Image-Components

6.4.4 UML chat components

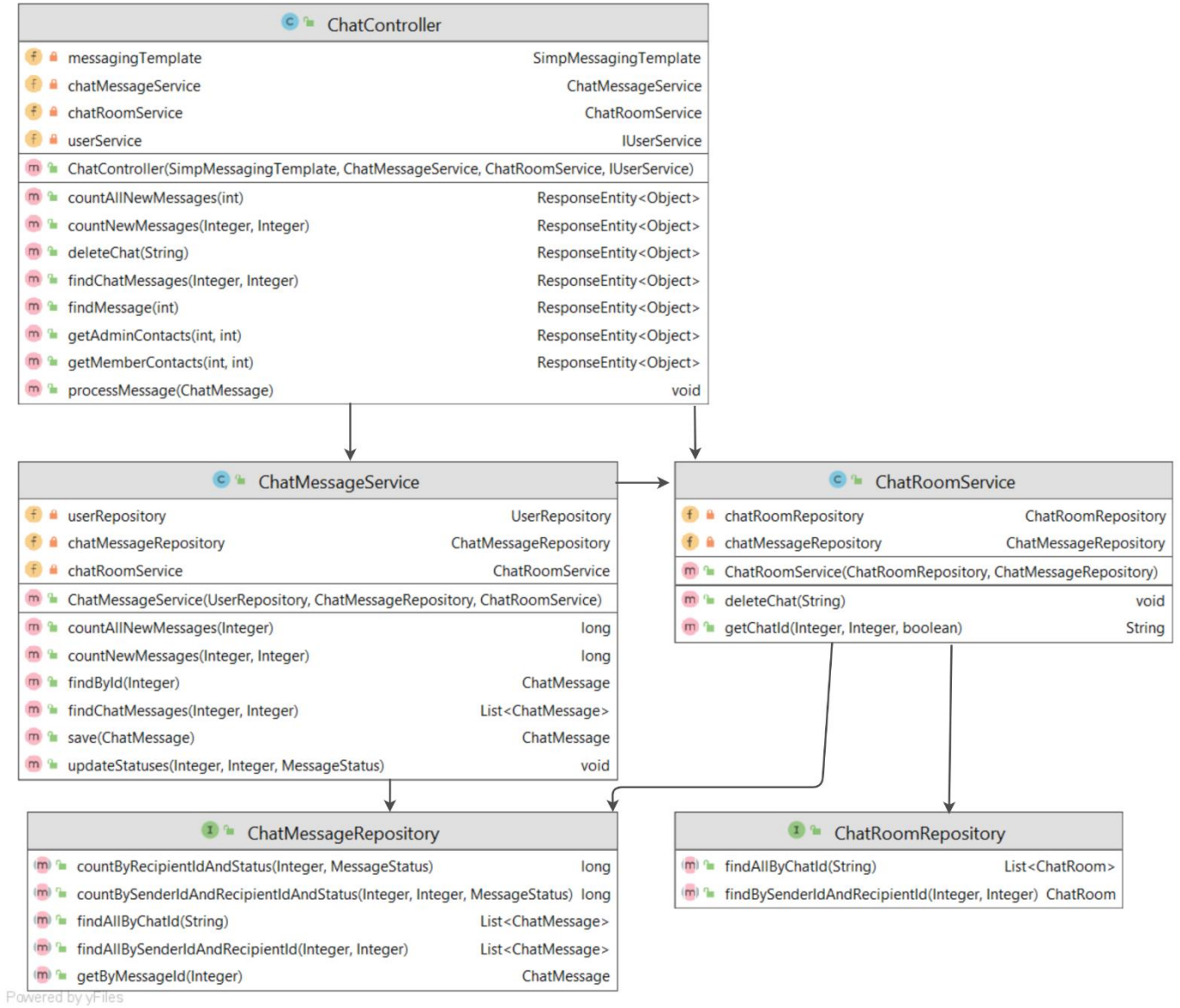


Figure 18 UML-Chat-Components

6.4.5 UML chat models

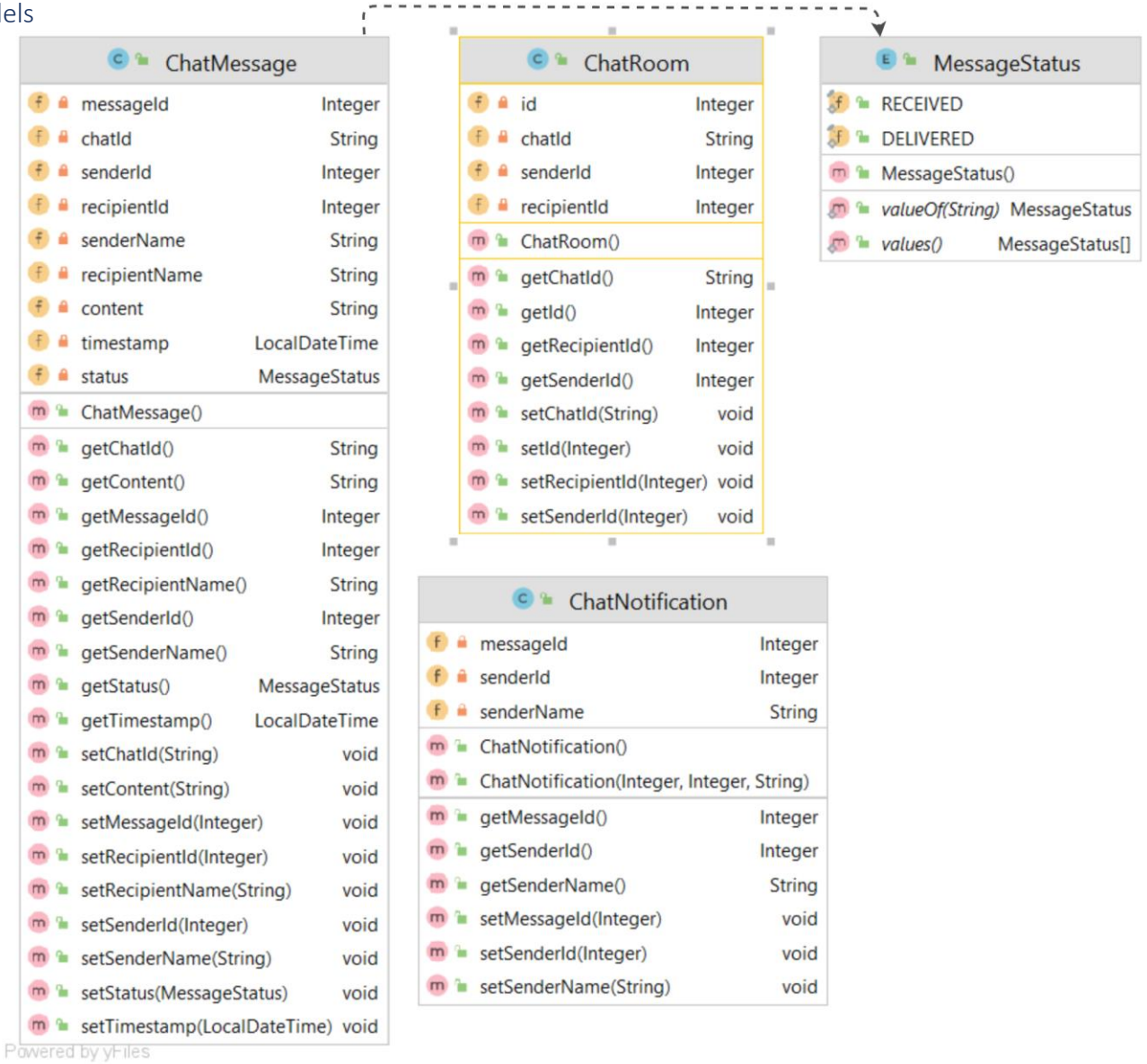


Figure 19 UML-Chat-Models

6.5 Sequence diagram

The following diagrams show the sign-up flow of a guest starting from the angular client application.

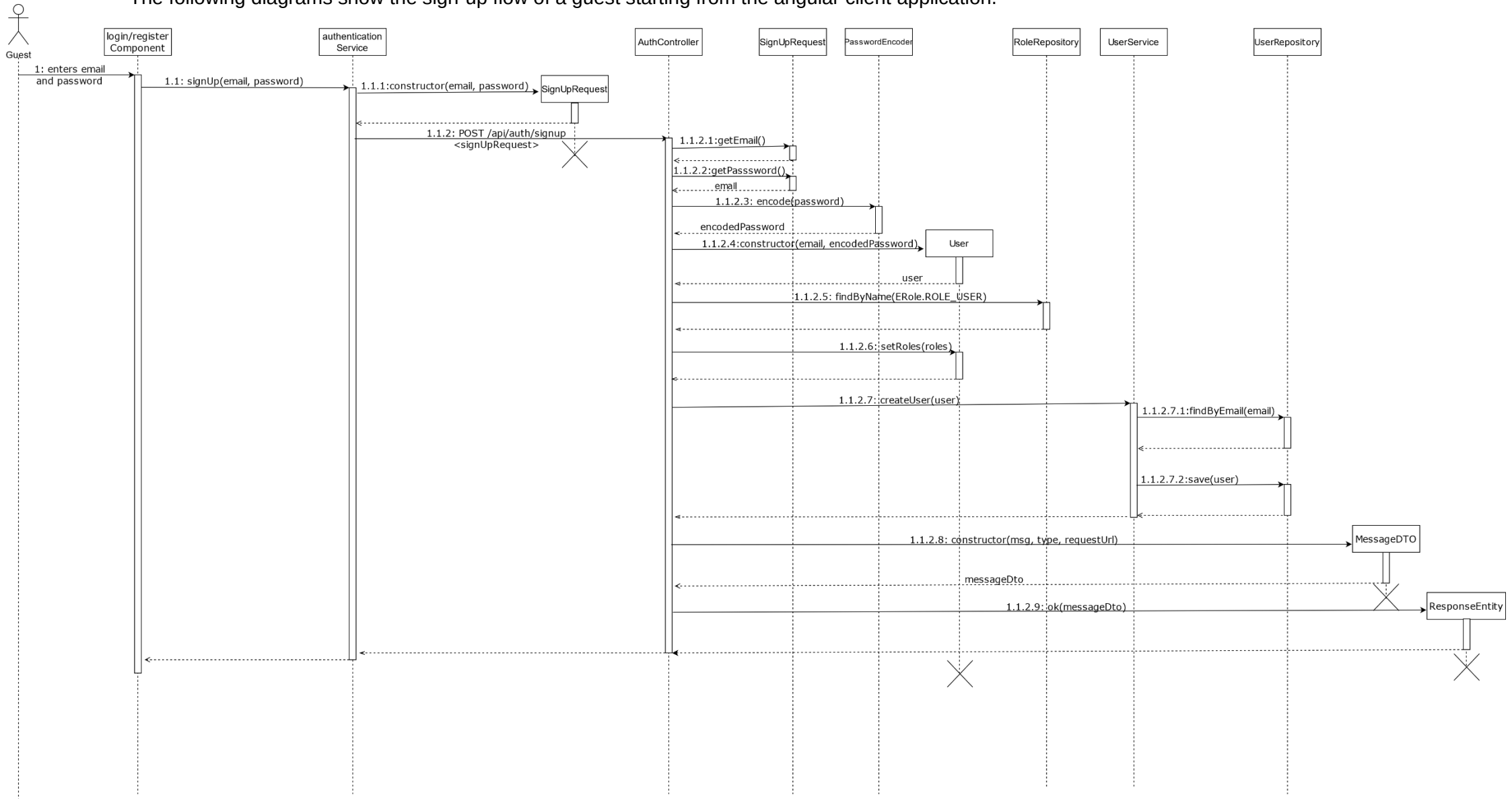


Figure 20 Sequence-User-SignUp

7 RESTful API design

This section shows the endpoints from the API that a client can call in order to perform operations on entities.

7.1 Product API design

Table 5-Product-API-Design

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/countPlatforms	Get the total number of platforms.
GET	api/products/statistics/countGenres	Get the total number of genres.
GET	api/products/statistics/countAll	Get the total number of products

GET	api/productPlatforms?retrieval={mode}	Get all product platforms. Retrieval mode options “All“ & “Page”.
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/genres?page={pageNr}&size={size}	Get all genres by pagination.
GET	api/genres?retrieval={mode}	Get all genres. Retrieval mode options “All“ & “Page”.
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.

7.2 User API design

Table 6-User-API-Design

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.
GET	api/users/statistics/countAll	Gets the total number of users.
GET	api/users/statistics/userRoleRatio	Gets the total number of users separated per role.

GET	api/users/dailyRegistered	Gets the total number of daily registered users.
PUT	api/users/updateInfo	Updates the personal information of a logged in member.
GET	api/views/countAll	Returns the total number of website views.
GET	api/views/countAllDaily	Returns the total number of website view during the time of the current day.

7.3 Image API design

Table 7-Image-API-Design

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.
POST	api/images/upload/user/{refreshToken}	Uploads the user profile image.
POST	api/images/update/user	Updates the imageUrl of the user.

7.4 Chat API design

Table 8-Chat-API-Design

Operation	URL	Description
GET	api/messages/{senderId}/{recipientId}/count	Count and read all the messages from the given sender id and recipient id
GET	api/messages/{recipientId}/count	Count all new messages for recipient
GET	api/messages/{senderId}/{recipientId}	Get and read all messages
GET	api/messages/{id}	Get a message by its id

GET	api/images/update/user	Updates the imageUrl of the user.
GET	api/contacts/members	Get all admin contacts
GET	api/contacts/admins	Get all member contacts
DELETE	api/chat/{chatId}	Delete a chat by its id

8 Wireframes

The following diagrams show the first concepts of the BluDevil digital-marketplace.

8.1 Navigation & Product catalog wireframe

This wireframes represent the product listing viewed by a guest and the navigation menu. The wireframe on the left is showing the desktop version of the page and the right wireframe shows the mobile version. The navigation does not required a mobile version as it is build to function on both version.

As shown in the wireframes on the left side we have the menu button that when pressed it opens the navigation. Still on the left side on the product listing page we can view all the possible filters for the products as well as a search bar to search a product by name. On mobile the products are arranged in a column position so to make the user experience better a back-to-top button, that contains a upward pointing arrow, can be found in the bottom right corner. Once selecting a product card the user is redirected to the product details page of the selected product, see Figure 22 Wireframe-Product-Detail.

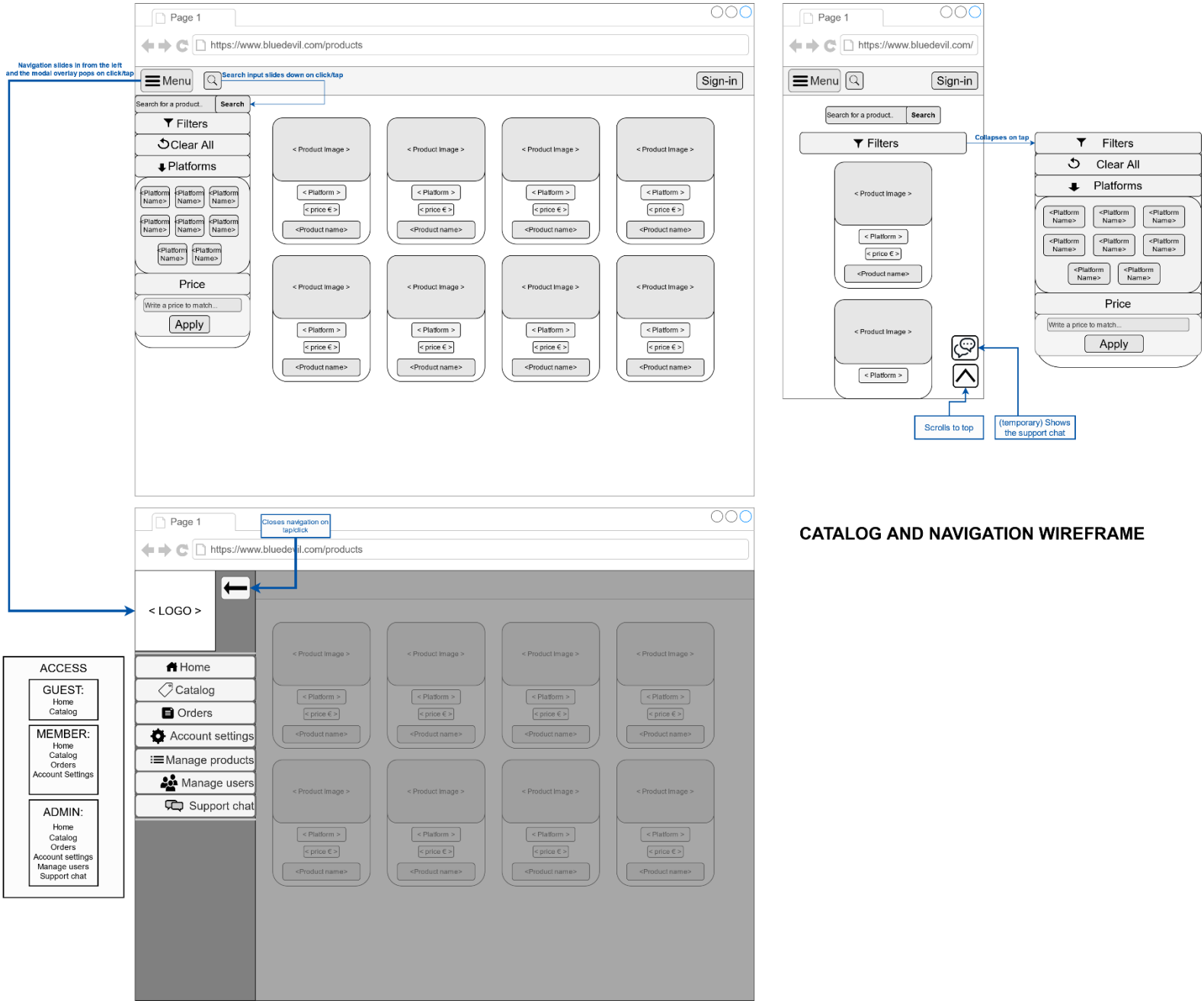


Figure 21 Wireframe-Catalog&Navigation

8.2 Product details wireframe

The wireframe on the left is showing the mobile version and the one on the right the desktop version of the details wireframe. The product state is displayed above the image at it changes styles based on the product availability.
The left point arrow button next to the product name navigates the user back to the product listing page, see Figure 21 Wireframe-Catalog&Navigation.

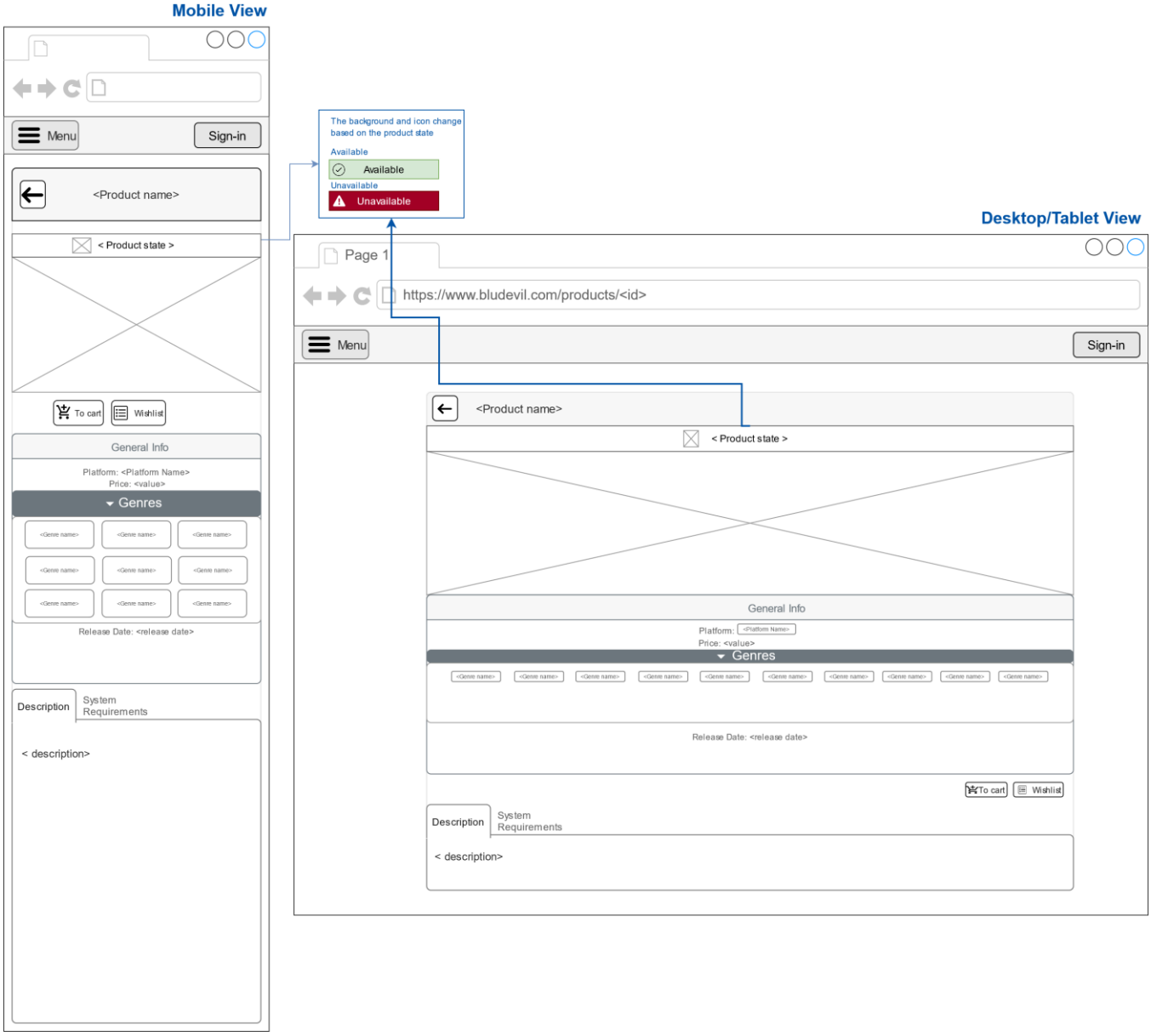


Figure 22 Wireframe-Product-Details

9 UX survey results

9.1 General questions

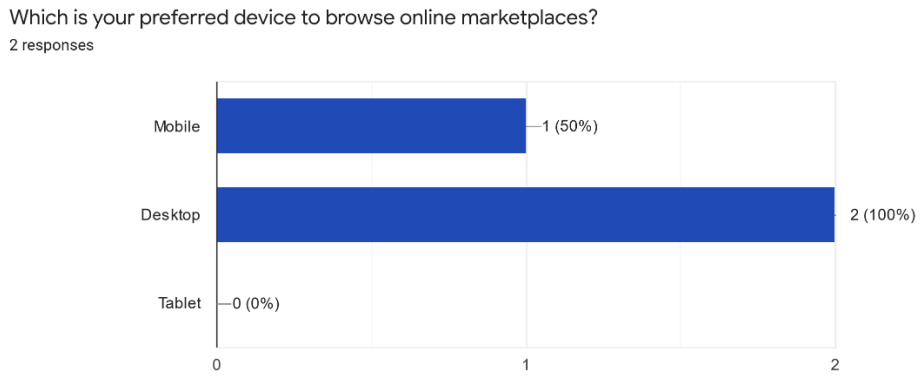


Figure 23 Preferred-device-chart

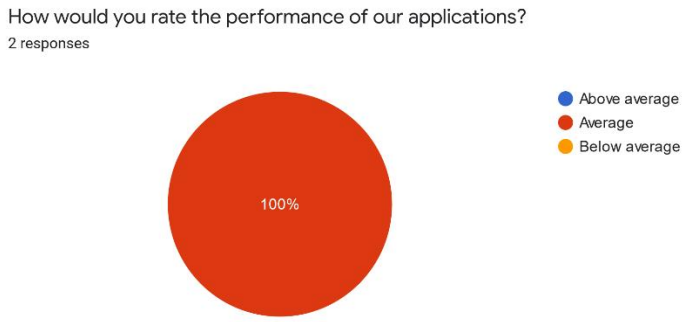


Figure 24 Performance-Chart

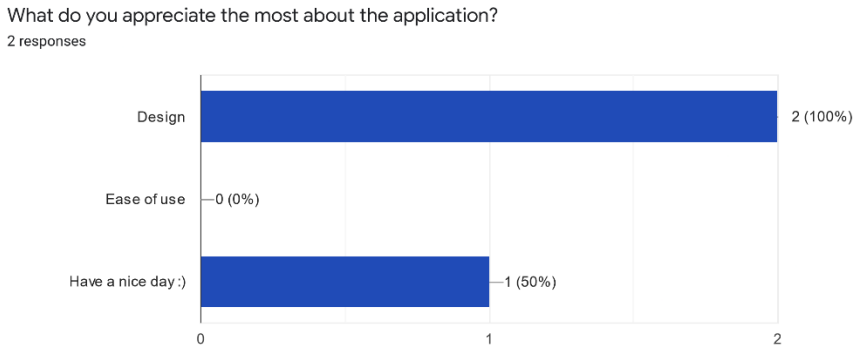


Figure 25 Most-Appreciated-Chart

Based on the given results, the focus for the applications looks should be put first in the desktop view and secondly the mobile view. An improvement could be made to the performance as the application scored average on all survey results.

9.2 Ease of use

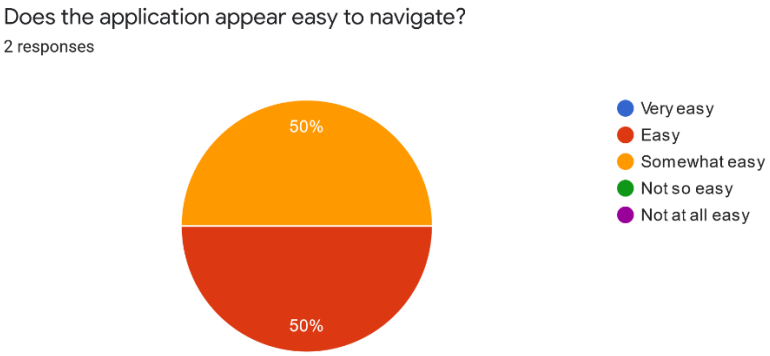


Figure 26 Navigation-Difficulty-Chart

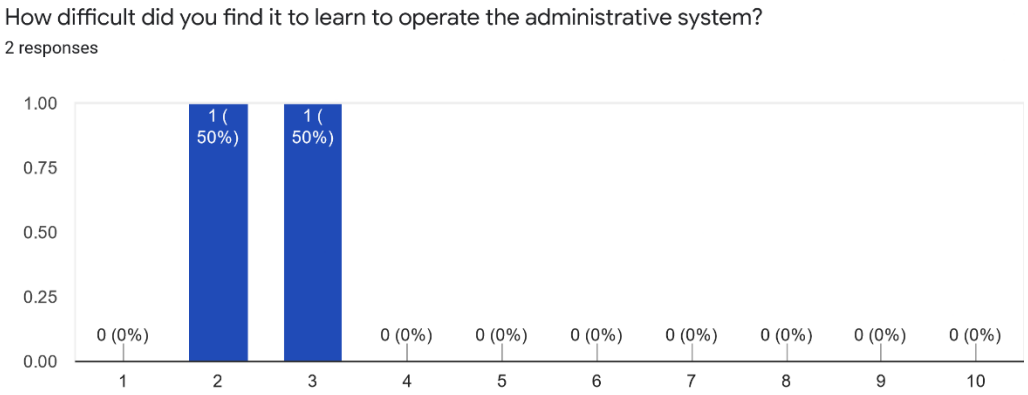


Figure 27 Admin-System-Operate-Chart

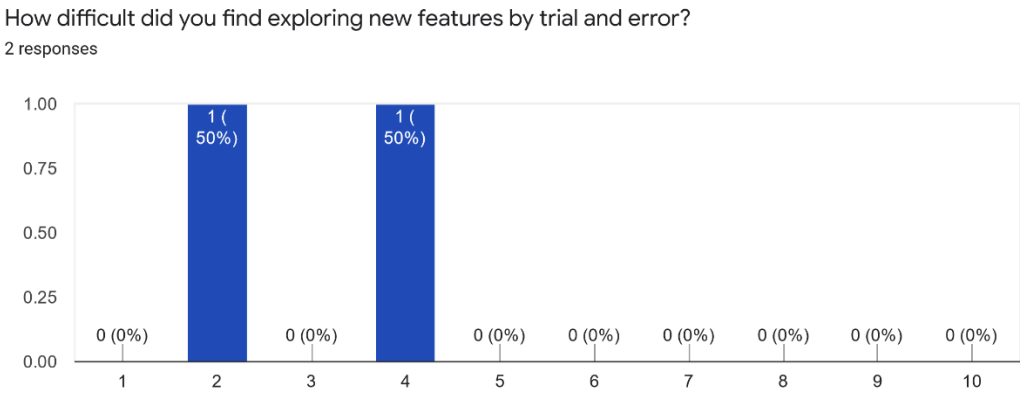


Figure 28 Trail&Error-Chart

The applications ease of use scored nicely, it is intuitive enough but improvements could still be made as the design is the most appreciated, as shown in Figure 25 Most-Appreciated-Chart.

9.3 Product Details

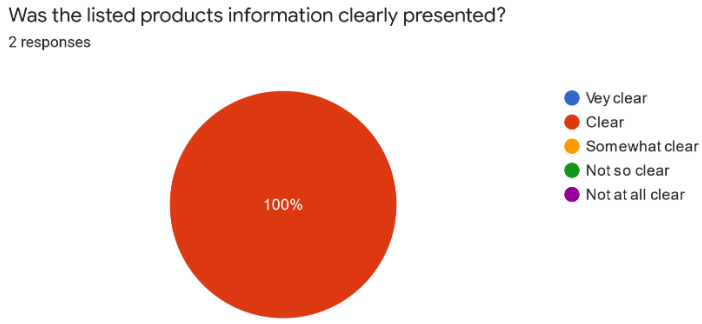


Figure 29 Product-Details-Chart

What other product information should we provide inside our application?
Answers:

- More images and maybe videos

The product information is clear to the user. Improvements can be made by showing the user additional information thru the means of images and videos.

9.4 Member functionalities

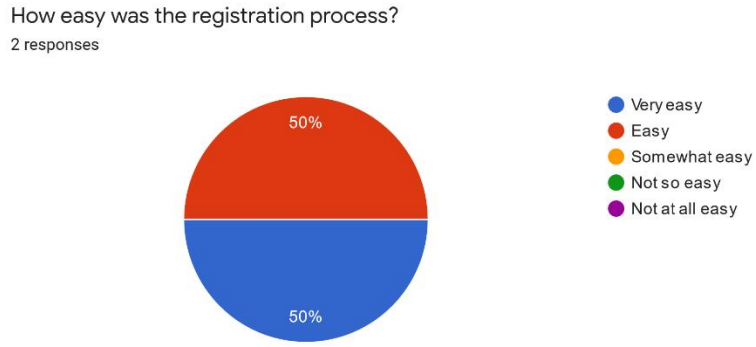


Figure 30 Registration-Chart

Figure 31 Update-User-Details-Chart

What was your overall impression of the profile page?
Answers:

- Minimalistic but sufficient.
- Overall good looking but a bit empty

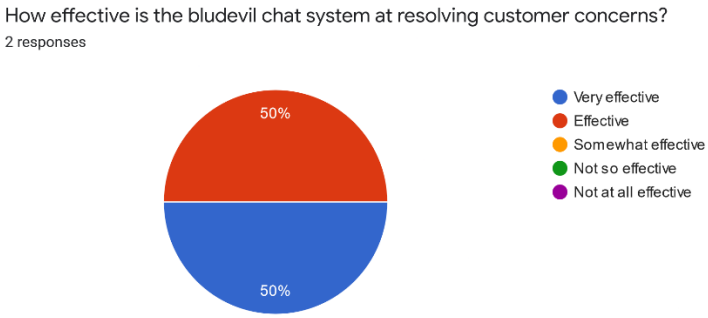


Figure 32 Chat-System-Chart

What other features would you like to see as a member?
Answers:

- More robust product and price filtering options, genre button to work as platform button.
- A password reset, a checkout

As a registered user, is there anything that doesn't work the way you expected it to?
Answers:

- Editing my profile, at first glance, I thought that I could edit/add my info inthe text fields. Tried and failed and then noticed the edit button.
- Some errors did not clear/stayed on screen. A bit confusing, I had an error red box with no info.
- Some of the error did not show any guiding text and updating the information was confusing at start

From the received feedback all the participants were able to successfully utilize all the member functionalities. The user support is highly effective. The forms validation needs to be revised as they do not work as intended in the production environment. The profile page is functional but a bit confusing for the user when it comes to editing his information. Additionally, there is a lack of features for the members so in the next updates it is highly advised to increase the number and complete unfinalized features. Some user suggested additions are password reset, checkout page, and more filtering options.

9.5 Admin functionalities

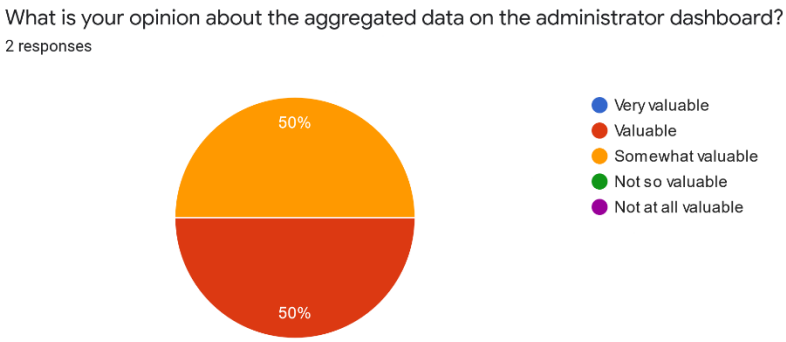


Figure 33 Dashboard-Chart

As an administrator, is there anything that doesn't work the way you expected it to?
Answers:

- Nope.
- I expected the clear all button to also clear the errors

Do you have any thoughts on how to improve the administrative systems?2 responses

- Fluid design?
- More information on the dashboard, maybe completely separate or minimize the genre and platforms boxes, the admin page for products seems to clustered and smth for managing other users

Overall the administrative system for the products is clearly delivered to the user. Providing more aggregated data and reworking the design and errors functionalities of the products manage page will increase the user experience. Additionally, the system would benefit from more administrative features.

10 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 hotspots, the high number of code-smells, the low code coverage and the little amount of comments.

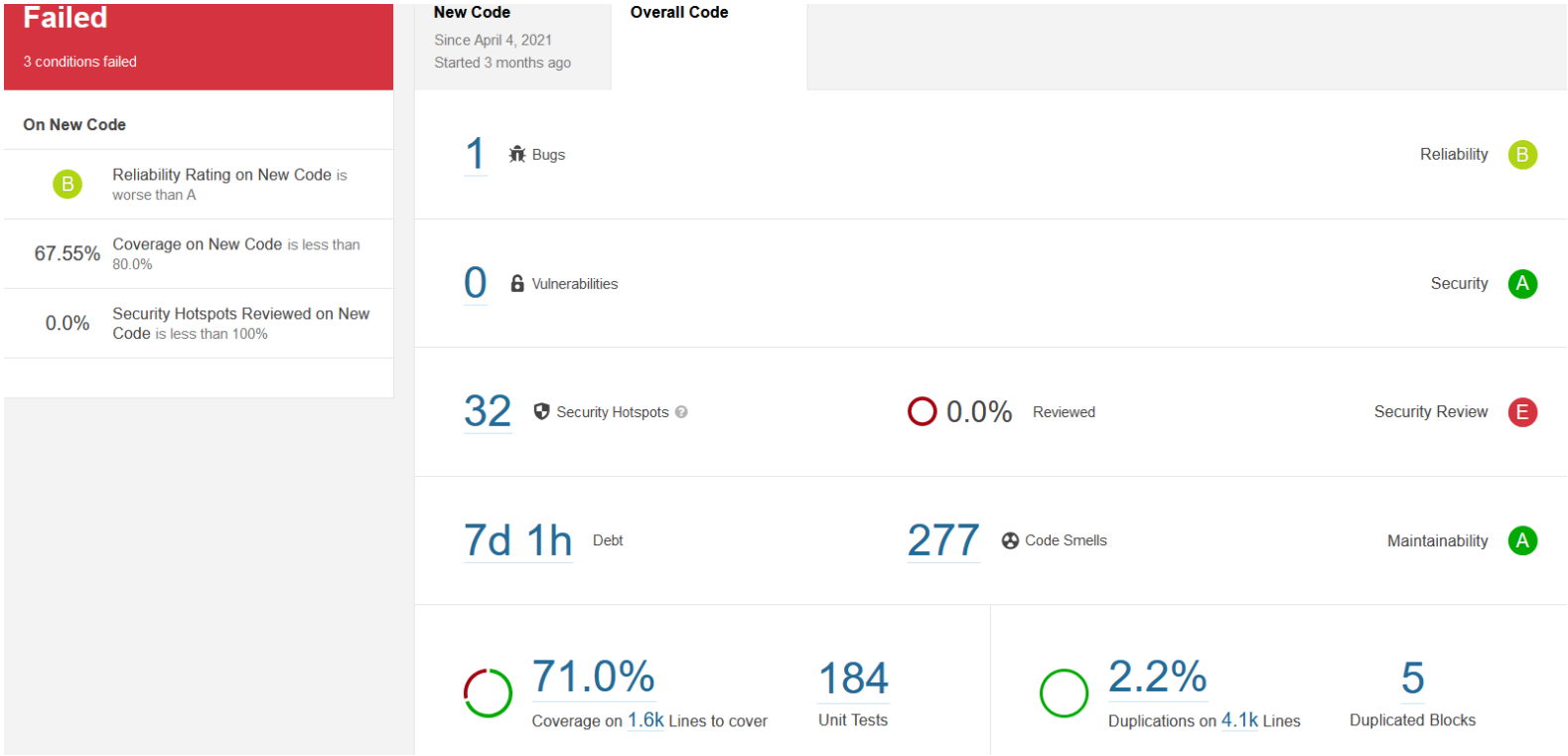


Figure 34 SonarQube-Results

11 OWASP report (Top 10)

This section contains the top most encountered risks from OWASP. In each subsection there is a short description of the risk and how it applies to the BluDevil system.

11.1 Injection

Code injection also known as RCE is when an unknown party supplies malicious code to a compromised application. If the compromised data does not get sanitized before being processed by the system or an ORM, it might contain malicious commands or queries that once executed can expose more data, modify or completely remove stored data.

Vulnerabilities are often found in SQL, NoSQL queries, OS commands, XML parsers, HTTP requests, expression languages, and ORM queries. **This risk can result in data loss, corruption, disclosure to unauthorized parties, or a complete takeover!** Said risk can be minimised with combined actions such as sanitizing the received input for special characters, using ORMs or using parameterized queries, using “LIMIT” on queries to prevent mass exposure.

The following risk applies to the BluDevil system as users can send malicious data from the front-end application to the RESTful API.

The front-end validation is very light, allowing the user to pass special characters and not limiting the user input length.

The back-end API does not sanitize the received data but it relies on Spring data JPA which makes use of Hibernate, an ORM thus proving parameterized queries.

To improve defense against injection, both the front-end application and the back-end API needs to sanitize the user input by checking for special characters and provide fixed boundaries for the length of the input.

11.2 Broken authentication

Broken authentication refers to several vulnerabilities that can be exploited to impersonate legitimate users. Generally, this refers to weaknesses in credential management and session management.

Some common attacks, often automated, can consist of credential stuffing, where the attacker possesses a list of valid credentials and brute force tactics.

Using weak or infective credential recovery like “knowledge-based answers”, plain text, or weekly hashed passwords increases the vulnerability of applications.

A poorly configured session management can consist of the exposure of the Session IDs in the URL, not rotating the session ID, not properly invalidating the Session ID during logout, or a period of inactivity.

The following risk applies to the BluDevil system as it permits automated attacks like brute force by not implementing multi-factor authentication, not providing a limit, increasingly delay during sign-in, and not logging errors during failed authentications. The system exposes the refresh token, used in a one-hour rotation to refresh the JWT access token, in the URL and it does not properly invalidate the refresh token during logout or a period of inactivity. In addition, weak password checks such as checking the created or updated password against a list of the top 10000 worst passwords or a list of known exposed passwords during breaches are not implemented in the front-application or back-end application.

11.3 Sensitive data exposure

Sensitive data exposure occurs when the entity holding sensitive data fails to secure and inadvertently exposes said data. This also happens when an attacker executes man-in-the-middle attacks to steal clear text data off the server while it is in transit or from the user’s client itself. Vulnerabilities appear when passwords, personal information, trade secrets, etc.. do not have the required extra protection, particularly if that data falls under privacy laws or regulations like GDPR. Other vulnerabilities appear when the data is transmitted in clear text using protocols such as HTTP, SMTP, and FTP, poorly configured or missing security directives or headers.

BluDevil makes use of Bcrypt for a strong, adaptive, and salted hashing of the password. BluDevil also does not unnecessarily store sensitive user information.

The risk is still applicable to the system as all transit data is not secured with protocols such as TLS, cipher prioritization, and secure parameters.

11.4 XML external entities

Attackers can exploit XML processors if they can upload XML or include malicious content in an XML document, exploiting vulnerable code, dependencies, or integrations.

The system as is, uses less complex data formats like JSON, it is not using SOAP or a version before 1.2 but it does allow any file upload so it is inherently at risk.

11.5 Broken access control

Access control weaknesses are common due to the lack of automated detection, and lack of functional testing. It enforces policies that a user cannot act outside of their given permissions. Failing to provide a defense against this risk may lead to the user performing an out-of-bounds function, accessing unauthorized data, and modifying or deleting the data in question. Manual testing is the best way to detect missing or ineffective access control, including HTTP methods, controllers, direct object references, etc…

The front-end application of the system uses angular routes to limit the user's access control based on roles. The back-end application secures each endpoint by filtering each request, extracting the JWT token, and providing role-based access.

The risk applies to the system due to the JWT tokens not being invalidated in the server after logout, the web socket missing a security configuration for access control, and access to static resources not configured.

11.6 Security misconfiguration

Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Misconfiguration or flaws at any level can give attackers unauthorized access to some system data or functionality that may result in a complete system compromise.

The BluDevil system is vulnerable as it does not set secure values for all the libraries and the database, it removes the default admin account only on startup leaving the risk of the default account information unchanged, the angular framework and spring boot version is not up to date and some of the returned error responses keep the default format showing the stack traces and other overly informative messages to the user.

11.7 Cross-site scription(XSS)

The impact of XSS is moderate for DOM XSS, and severe for stored XSS, the remote code is executed on the victim's browser as such giving the possibility to steal credentials, sessions, or deliver malware to the victim.

The BluDevil front-end application is protected from XSS as Angular treats all values as untrusted by default. When a value is inserted into the DOM from a template binding or interpolation, Angular sanitizes and escapes untrusted values. The risk applies to the system as the back-end application does not clear the user input, it does not contain an XSS filter that would strip the malicious code from the headers, parameters, and bodies of each request.

11.8 Insecure deserialization

Insecure Deserialization is a vulnerability that happens when untrusted data is used to abuse the logic of an application, inflict DoS attacks, or execute arbitrary code upon it being deserialized.

The BluDevil RESTful is vulnerable to insecure derealization as the input is unsanitized and the functions responsible for converting data into an object assume that the data is trusted. An attacker may format the serial data in such a way that the result of deserialization is malicious.

11.9 Using components with known vulnerabilities

Once a vulnerable component is exploited, an attack can provide a serious data loss or server takeover. Applications that use components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

The following risk is applicable as both the front-end and back-end applications contain several other vulnerabilities. The API utilizes out-of-date and unused dependencies. Additionally, the spring boot framework and angular application are not up to date.

11.10 Insufficient logging and monitoring

Insufficient logging and monitoring allows attackers to further attack systems, maintain persistence, and tamper, extract, or destroy data.

The risk applies to the system as the back-end API does not provide sufficient logging for the stored and processed data. Logs are not generated during all logins, access control failures and the server-side input validation is mostly not present. Additionally, it does not provide a user context to identify suspicious or malicious accounts.

12 Document versioning

This section uses the versioning table to keep track of what was updated on the design document and when.

Table 9-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
V1.4/20-06-2021	• Added introduction • Added project description • Rearranged document • Updated all diagrams • Updated the SonarQube test results • Increased sections text • Added security report • Added ux feedback section • Added appendix

13 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/

A1:2017-Injection | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html

A2:2017-Broken Authentication | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication

A3:2017-Sensitive Data Exposure | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

A4:2017-XML External Entities (XXE) | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE)

A5:2017-Broken Access Control | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control

A6:2017-Security Misconfiguration | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration

A7:2017-Cross-Site Scripting (XSS) | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)

A8:2017-Insecure Deserialization | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization

A9:2017-Using Components with Known Vulnerabilities | *OWASP*. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities

A10:2017-Insufficient Logging & Monitoring | OWASP. (n.d.). OWASP. Retrieved June 20, 2021, from https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring

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

14 Appendix

UX survey link: https://forms.gle/5tZKqKeRMvFACjVE8

14.1 UML diagrams

14.1.1 Error advice UML diagrams

The ExceptionHandlerControllerAdvice takes action once one of the custom exceptions is thrown inside the RESTful API. The class intercepts the response and return a custom error response with a pre-defined status and content.

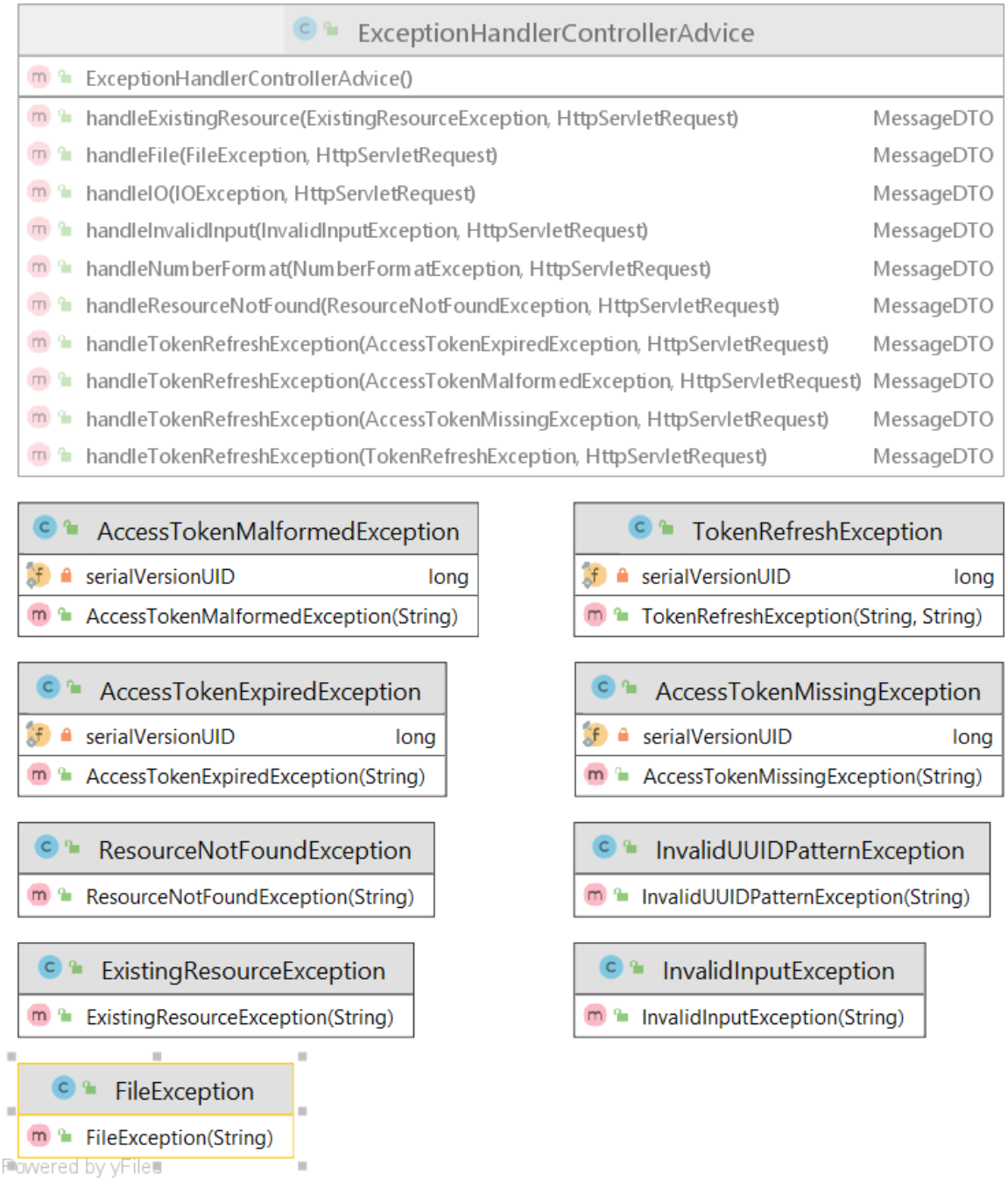


Figure 35 UML-Error-Advice

14.1.2 Request and response UML diagrams

The following classes are extra and nearly all them are generalized responses.
The AdminAccountGenerator is responsible to create an default admin account on API start-up using the credentials specified inside the application.properties files.

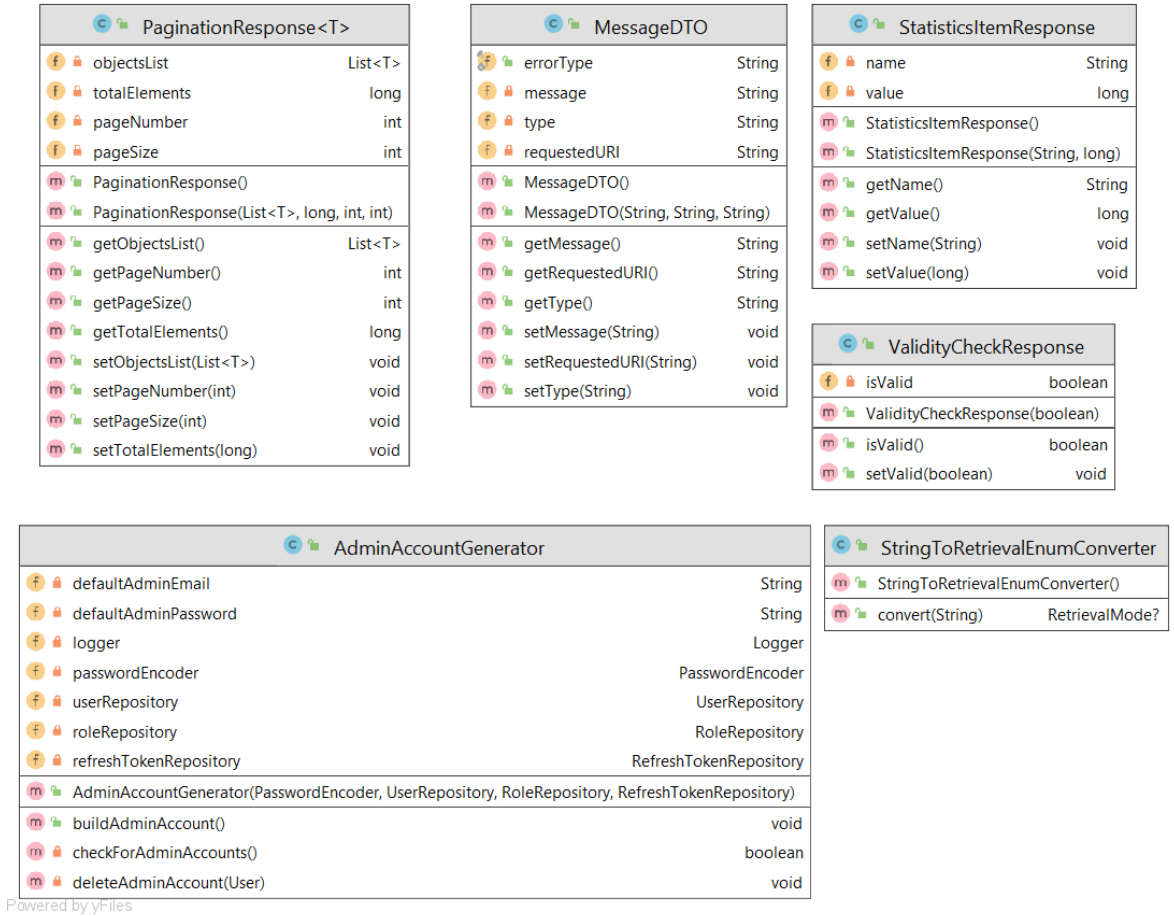


Figure 36 UML-Requests&Responses