# Technical Documentation

A red and black logo

Description automatically generated

# Introduction

With the following technical documentation, we hope to give a guide on how our web application works and to better define the development and implementation details of this project. This is done so that future software engineers who need to extend our product with additional functionality or migrate it to another IT technology, can do so more easily by reading this document.

In this project we made a website that is designed to streamline the laptop upgrade service offered by Mediamarkt. The main functionality of this website is a CRUD (create, read, update, delete) system for users, images, and laptops. In this document you can expect the intent and workings behind this website.

Contents

[Technical Documentation 1](#_Toc157975615)

[Introduction 2](#_Toc157975616)

[Contents 3](#_Toc157975617)

[Product vision 4](#_Toc157975618)

[Introduction 4](#_Toc157975619)

[Product Description 4](#_Toc157975620)

[Target Users 4](#_Toc157975621)

[Future growth 4](#_Toc157975622)

[User stories 5](#_Toc157975623)

[Deploy website. 5](#_Toc157975624)

[As Admin I want to be able to see and order the image statistics 5](#_Toc157975625)

[As Admin I want to be able to order the user page 5](#_Toc157975626)

[Class Diagrams 6](#_Toc157975627)

[Models UML 6](#_Toc157975628)

[UML JPA 6](#_Toc157975629)

[Challenges 7](#_Toc157975630)

[Password hashing 7](#_Toc157975631)

[Deploy 7](#_Toc157975632)

[Making relations between the models 7](#_Toc157975633)

[Deployment Diagram 8](#_Toc157975634)

[Analytical reflection 9](#_Toc157975635)

# Product vision

## Introduction

For the product vision for this website, we intend a user-friendly tool for workers to create, check and update the status of images associated with laptops. This process aims to enhance the overall efficiency of the laptop service.

## Product Description

This application includes a CRUD system managing users, images and laptops. Workers and administrators have clear roles and access levels within the application ensuring secure and controlled access.

## Target Users

The targeted users for this application are in-store workers and administrators. Workers will be able to create and pick up images. Administrators are also able to edit images, create laptops and edit/create users. The main interest of the in-store workers is that they can effectively search and pick up images. We accomplish that by streamlining the process. Administrators also want to be able to manage laptops and other users.

## Future growth

For the future growth of this application there are a lot of improvements to be made integrating this application with storing the images so users can upload images directly to this application. In this way this application would not only be a way to check image statuses but also provide an all-in-one place to store the images.

# User stories

## Deploy website.

The deployment of our website was one of the most important ones so that users could use this. We accomplished this by using Docker and deploying those images.

Accept Criteria:

* Website is online.
* Frontend completed.
* Backend completed.
* URL routing works.
* database is online.

Definition of Done:

* Code is commented where needed.
* Code complies with HBO-ICT conventions.
* Code is tested.
* No unused code.

## As Admin I want to be able to see and order the image statistics

This was important so that the application is more user friendly. We accomplished this by creating endpoints

Accept Criteria:

* Stats are visible.
* Backend filter completed.
* Frontend filters completed.

Definition of Done:

* Code is commented where needed.
* Code complies with HBO-ICT conventions.
* Code is tested.
* No unused code.

## As Admin I want to be able to order the user page

This was important so that the application is more user friendly.

Accept Criteria:

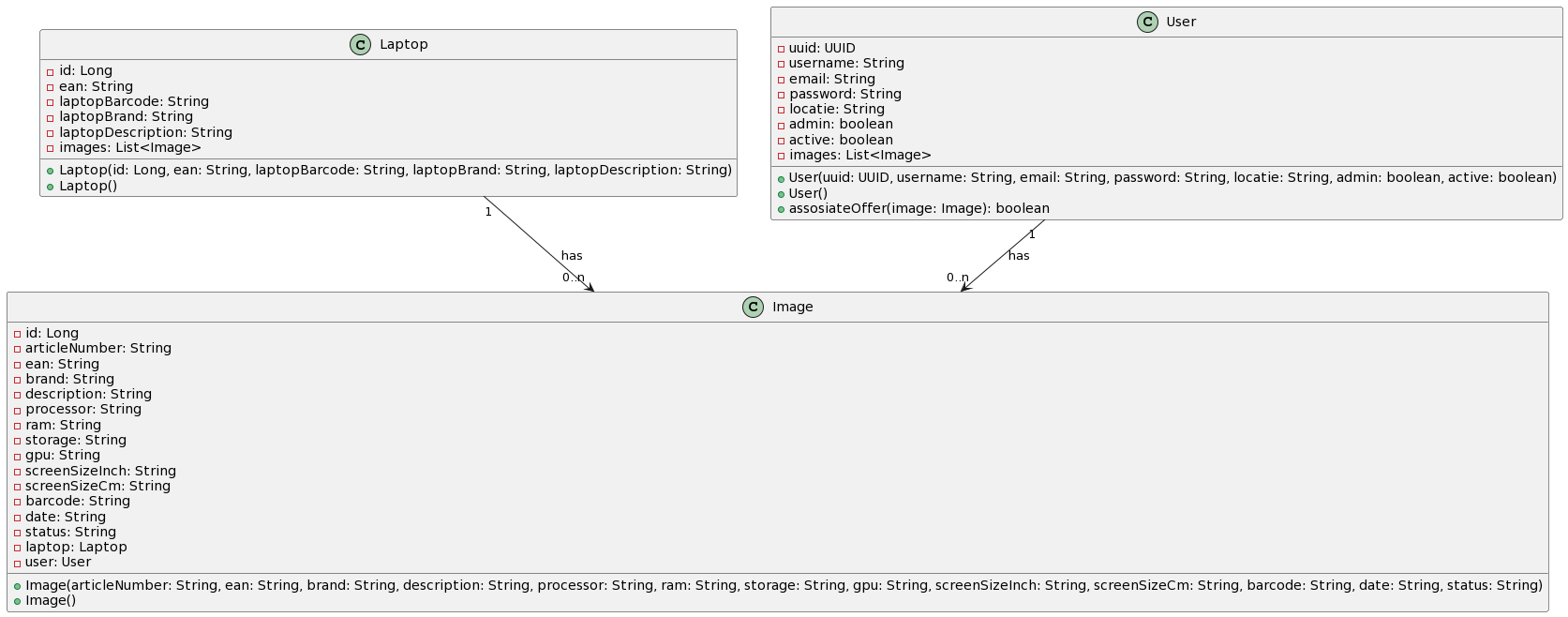
* Backend filter completed.
* Frontend filters completed.

Definition of Done:

* Code is commented where needed.
* Code complies with HBO-ICT conventions.
* Code is tested.
* No unused code

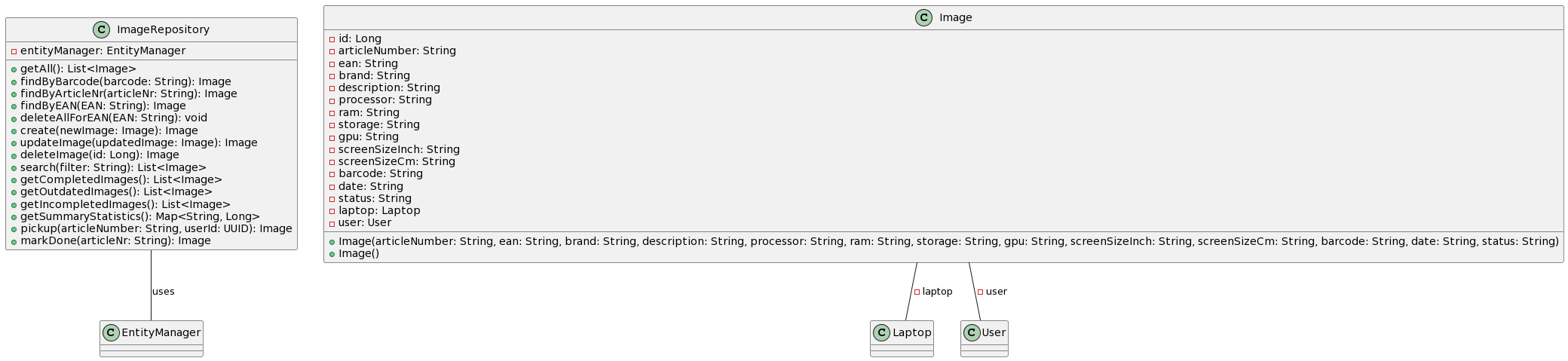
Class Diagrams

## Models UML

We use three models for our product, each with attributes and functions. These models include User, Image, and Laptop. These models are an integral part of our application, which are also used for the MYSQL database. All our functions and methods depend on these models.

In our models an image is associated with a laptop and a user. The relationships between these models are both one to many relations between entities, providing a navigable diagram.

## UML JPA

This diagram shows the imageRepository, the ImageRepository is responsible for implementing the methods to handle the methods to manipulate an Image instance. We should have used an external interface here to make sure all our repositories shared a basic functionality, but we forgot to implement that and when we figured out it was to late. The methods are: getAll, findByBarcode, findByArticleNr, findByEAN, deleteAllForEAN, create, updateImage, deleteImage, search, getCompletedImages, getOutdatedImages, getIncompletedImages, getSummaryStatistics, getCountByStatus, getCompleted, getCountOfOutdatedImages, pickup, markDone. These functions are mostly different filter functions and basic CRUD (Create, Read, Update, Delete) operations.

# Challenges

## Password hashing

One of our challenges was password hashing. The problem was that we didn't have a good idea on how to, and that we made everything before the hashing which caused us to have to find the correct dependency. We accomplished hashing passwords by using BCrypt. We hash the user password before storing it, making it impossible to reverse the process and obtain the original password. An alternative approach could involve considering the use of Argon2, a memory-hard hashing algorithm. This alternative provides resistance against brute-force attacks and offers flexibility for future security enhancements. Choosing BCrypt, however, is advantageous for its widespread industry adoption and proven track record in security, ensuring a robust and reliable solution.

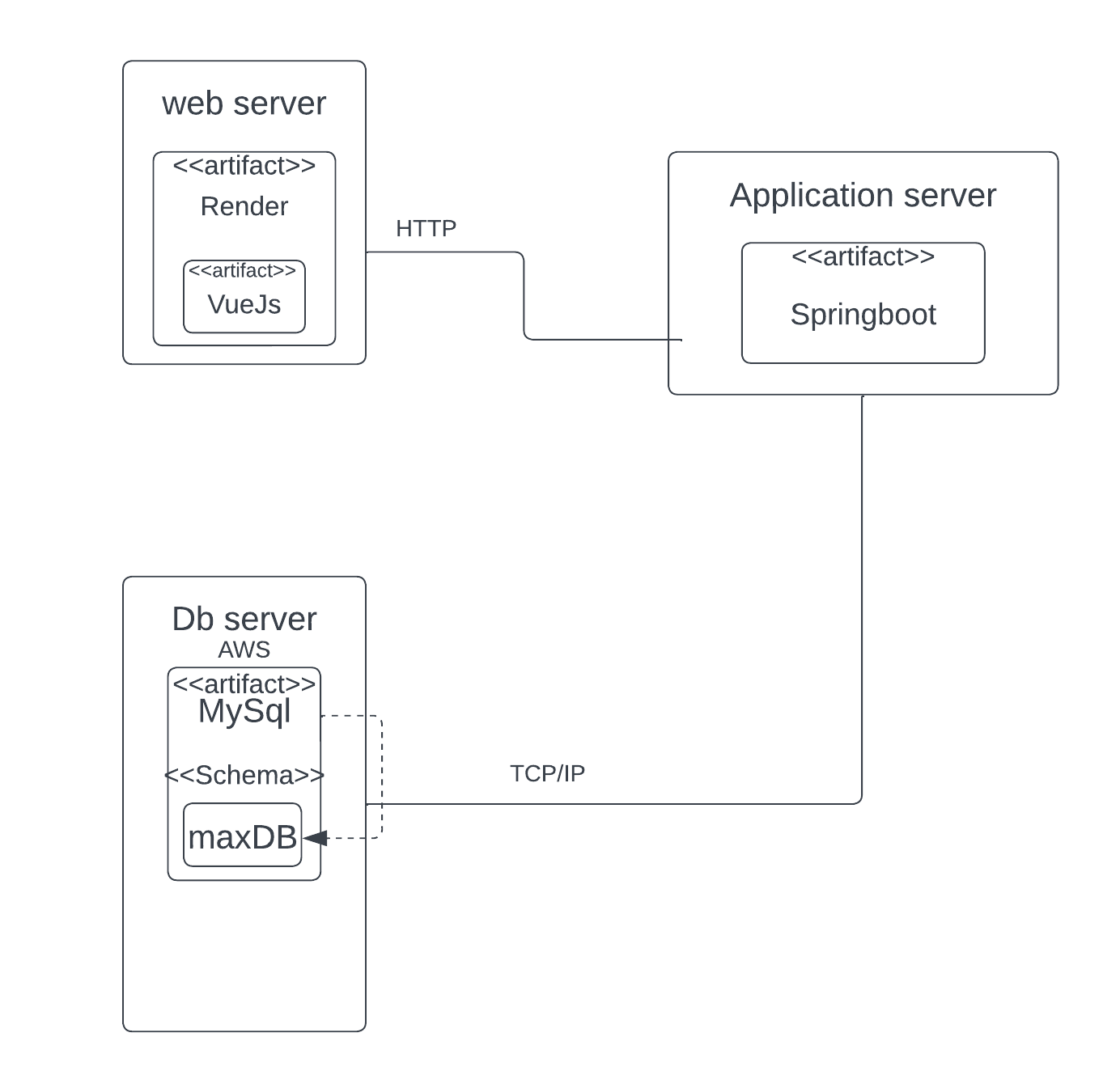
## Deploy

Another of our challenges was deploying the website. With our limited knowledge of Docker and deploying websites, this is a thing we struggled with. We deployed our website by learning Docker and using the provided tutorials by school. While we haven't fully automated deploying our website in the CI/CD, we prioritize regular deployment to improve security and keep our site updated. Alternatively, exploring other tools like Kubernetes could streamline the deployment process, offering scalability and better resource utilization. Opting for Docker aligns with industry standards and simplifies the deployment process, allowing for quick and efficient updates.

## Making relations between the models

We faced challenges creating relationships between models in our back-end application. To overcome this, we searched online for how relations in Spring Boot work and looked at our web-framework projects to see the difference. We fixed this by working on the foreign key constraints and the cascading operations. Another option could be considering the use of an Object-Relational Mapping (ORM) tool like Hibernate, simplifying the management of relationships and enhancing database interaction efficiency. Our choice to address foreign key constraints directly ensures a more hands-on and tailored solution, maintaining control over the application's specific needs.

# Deployment Diagram



In this diagram we have a webserver that hosts the website via render with a vueJs framework that can make request to the backend application server running Springboot we chose this because we learned render in school, and it is free. With VueJs we made the front end of the application because it is easy to learn and sustainable. For our back end we used Springboot for its effectiveness and so we can access the Database server running AWS to run MySQL scripts. For the database we chose MySQL because it is used a lot and is easy to maintain.

# Analytical reflection

Looking back at the four sprints, the website did well in using Docker effectively, creating a strong CRUD system, and setting up good role-based access control for workers and administrators. The use of JWT for secure authentication also fits well with the project's main goals. The implementation of several filters to filter out different images also helped us reach the goals. However, there are clear areas where things can be better. The user interface needs to be improved to make it more user-friendly, more consistent and responsive, providing an overall better experience. It is also important to take a closer look at how the project is deployed to make it more efficient and minimize downtime in future releases.

Challenges and Ways to Improve

Even though the project met its requirements, there was some difficulty making the requirements such as implementing the unit tests. It took us a long time to make jest work so maybe next time we can ask the teachers about it or read documentation on the website. We had some specific bugs that were hard to solve relating to databases, request filtering and the barcode scanner that costed us a lot of time to fix.

there's potential to add more features that could enhance the overall performance of the website. One idea is to implement location-based access restrictions, limiting access to specific Mediamarkt shops. This not only improves security but also tailors the user experience to the intended audience's needs. A big thing that we did wrong in the project is our design. If we work more on design the next time, we can ensure that the first impressions of our product are better. We can do that by using more colors or using more space because now our product can feel a bit empty. These improvements show a commitment to always getting better and staying adaptable.

Conclusion

Looking back highlights the project's successes and strong foundations. The areas for improvement are opportunities to learn for future development, showing a dedication to doing things well. This reflection sets the stage for ongoing improvements, making sure the website keeps evolving to be dynamic, secure, and user focused.