Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Aleksander Vegsund
Balendra Sounthararajan
Joakim Røren Melum
Odin Velle Ulvestad

Common Query Engine and Viewer

Bachelor's thesis in Computer Science
Supervisor: Rituka Jaiswal
May 2023

NTNU
Norwegian University of
Science and Technology

Aleksander Vegsund
Balendra Sounthararajan
Joakim Røren Melum
Odin Velle Ulvestad

Common Query Engine and Viewer

Bachelor's thesis in Computer Science
Supervisor: Rituka Jaiswal
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Postnord AS wants us to explore solutions for extracting database information quick and easily for the non-SQL user. Using developer time to extract reports from a database takes developers away from developing and can become a costly bottle neck in daily operations.

In addition to developing a system for extracting database information, we look at different API's for communicating with the database and compare efficiency and ease of use.

This was accomplished by working with Scrum in an iterative approach as well as using other know development strategies.

The product developed in this project is a prototype for extracting database information with by using a web application with a simple user interface layout. In this way the non-SQL user can interact with the site and extract information without any prior knowledge of SQL.

Sammendrag

Postnord AS ønsker at vi undersøker løsninger for å hente ut informasjon fra database på en rask og enkel måte for brukere uten SQL-kunnskaper. Det å bruke utviklertid til å hente ut rapporter fra databaser tar utviklere bort fra utviklingen og kan være en kostbar flaskehals i bedriftens progresjonskurve.

I tillegg til å utvikle et system for å hente ut informasjon fra databaser, skal vi undersøke forskjellige API 'er for å kommunisere med back-end og sammenlikne brukervennlighet og effektivitet.

Dette ble oppnådd ved å benytte Scrum metodologien i som er en iterativ prosess samtidig som vi brukte andre kjente utviklingsstrategier.

Sluttproduktet som er utviklet i dette prosjektet er en prototype for uthenting av informasjon fra database, i form av en web-applikasjon med et enkelt brukergrensesnitt. På denne måten kan brukere uten som ønsker å hente ut informasjon enkelt gjøre dette uten SQL-kunnskaper.

Preface

This is the report for our bachelor thesis project, that has been done in close collaboration with Postnord AS.

We are grateful to the entire organization of Postnord AS - Åldalsnes for providing us with this software engineering problem to solve. Without their willingness to create the problem statement and offer it to us as students, this project would not have been possible. A special thanks to the IT Development department from Postnord AS - Åldansnes for their excellent communication and their time and effort in realizing this project. Their patience, positive attitude, feedback, and their commitment throughout the entire process were critical to our success.

We also want to express our gratitude to our supervisor, Rituka Jaiswal, for her exceptional guidance and support from the beginning to the end of the project. She helped us narrow down the problem domain, reserve group rooms, and ensure that we maintain a linear progression throughout the semester.

Lastly, we would like to thank the employees at the Department of ICT – NTNU for their invaluable assistance during the bachelor's thesis and throughout our bachelor's degree. They were always available to support us whenever we needed help.

Acronyms and abbreviations

HTML	Hypertext Markup Language.
CSS	Cascading Style Sheet.
JS	JavaScript.
IDE	Integrated Development Environment.
VS Code	Visual Studio Code.
CI	Continuous Integration.
CD	Continuous Deployment.
Vue	Vue.js.
JDBC	Java Database Connectivity.
JPA	Java Persistence API.
JSON	JavaScript Object Notation.
CRUD	Create, Read, Update, Delete.
IP	Internet Protocol.
VoIP	Voice over Internet Protocol.
CLI	Command Line Interface
NTNU	Norges Teknisk-Naturvitenskaplige Universitet (Norwegian University of Science and Technology).
CPU	Central Processing Unit.
OS	Operating System.
RAM	Random-Access Memory.
VM	Virtual Machine.
NDA	Non-Disclosure Agreement.
POM	Project Object Model.

Glossary

Virtual Machine simulated machine that runs on physical computer.

HTTP protocol for transferring information between devices on a network.

UX refers to a user’s experience on the web.

CI/CD practice to automate building, testing and deployment of applications

List of figures

Figure 1: The scrum process……………………………………………………………………………3

Figure 2: Components of DBMS……………………………………………………………………….7

Figure 3: Stages of database designing…………………………………………………………….8

Figure 4: Test pyramid……………………………………………………………………………………11

Figure 5: Simplified system architecture…………………………………………………………29

Figure 6: Project schedule w/milestones…………………………………………………………30

Figure 7: Snapshot of sprint board in Jira………………………………………………………31

Figure 8: Sprint review…………………………………………………………………………………32

Figure 9: Retrospective…………………………………………………………………………………32

Figure 10: Entities in the database model……………………………………………………33

Figure 11: Entities with their attributes…………………………………………………… ……35

Figure 12: Script to create customer entity…………………………………………… ……36

Figure 13: Code snippet with graphql and informix dependencies……………………36

Figure 14: Defining entity class and primary key using annotations………………37

Figure 15: Relationship between tables defined using JPA annotations……………37

Figure 16: Code snippet displaying method – graphql mapping …………………………38

Figure 17: Graphql schema……………………………………………………………………………38

Figure 18: Code snippet with method to fetch data from database……………………39

Figure 19: Unit test results ……………………………………………………………………………39

Figure 20: First sketch of Query Engine Viewer using Figma……………………………47

Figure 21: Use case from customer service and head of economics…………………42

Figure 22: Query engine viewer sketch using Figma………………………………………43

Figure 23: Code snippet showing global color variables…………………………………43

Figure 24: Basic structure of a component……………………………………………………44

Figure 25: Screen shot from IDE showing different components………………… ……45

Figure 26: Part of the customer page displaying search options……………… ……46

Figure 27: Table with results after filtering after last name…………………… ……46

Figure 28: Part of the customer page displaying parcel details per customer ………… ………46

Figure 29: The frontend pipeline that shows the passing stages……………………………………….……47

Figure 30: Frontend GitLab CI/CD configuration………………………………………………………………………38

Figure 31: The backend pipeline that shows the passing stages ……………………………………………49

Figure 32: Backend CI/CD configuration. Picture 1/3………………………………….……………………50

Figure 33: Backend CI/CD configuration. Picture 2/3…………………………………………………………..51

Figure 34: Backend CI/CD configuration. Picture 3/3…………………………………………………………..52

Figure 35: Gitlab project variables……………………………………………………………………………………53

Table of contents

Abstract ..... **v**

Sammendrag ..... **vi**

Preface ..... **vii**

Acronyms and abbreviations ..... **viii**

Glossary ..... **ix**

List of figures ..... **x**

1: Introduction **1**

1.1: Task description 1

1.2: Restrictions 1

1.3: Effect goals 2

1.4: Report Structure 2

2: Theory **3**

2.1: Agile Methodology 3

2.2: Scrum 3

2.2.1: Team 4

2.2.2: Scrum Master 4

2.2.3: Product Owner 4

2.2.4: Developer 4

2.2.5: Sprints 4

2.2.6: Planning 5

2.2.7: Reviews 5

2.2.8: Retrospectives 5

2.2.9: User stories 5

2.3: Database 5

2.3.1: Types of databases 6

2.3.2: Relational databases 6

2.3.3: Database Management System (DBMS) 6

2.3.4: Components of DBMS 7

2.4: Database design 7

2.4.1: Conceptual database model 7

2.4.2: Logical database model 8

2.4.3: Physical database model 8

2.4.4: Normalization 8

2.5: Test data ... 9
2.6: Query language ... 9
2.7: Programming Paradigm ... 9
2.7.1: Object-Oriented Paradigm ... 10
2.8: API – Application Programming Interface ... 10
2.9: Testing ... 11
2.9.1: Unit testing ... 11
2.10: Design patterns ... 11
2.10.1: Controller-Service-Repository ... 11
2.11: Code Reviews ... 12
2.12: Version control ... 12
2.12.1: Git ... 12
2.12.2: Git Flow ... 12
2.13: Design principles ... 13
2.13.1: Don Norman's 6 usability principles ... 13
2.13.2: Web design rules / Universal design ... 14
2.14: Microservices ... 15
3: Method ... 16
3.1: Preliminary project plan ... 16
3.1.1: Group contract ... 16
3.1.2: Schedule ... 16
3.1.3: 3-party agreement ... 16
3.2: Project organizing ... 16
3.2.1: Roles ... 17
3.2.2: Daily stand-up ... 17
3.2.3: Sprint planning ... 17
3.2.4: Sprint review ... 17
3.2.5: Sprint retrospective ... 18
3.2.6: Meeting notice ... 18
3.2.7: Meeting notes ... 18
3.3: Quality assurance ... 18
3.3.1: Code review ... 18
3.3.2: Testing ... 18
3.3.3: Continuous Integration and Continuous Deployment ... 19
3.4: Programming languages and frameworks ... 19
3.4.1: Java ... 19

3.4.2: Spring boot 20
3.4.3: JavaScript 20
3.4.4: Vue.js 20
3.4.5: GraphQL 21
3.5: Development tools 22
3.5.1: WebStorm 22
3.5.2: IntelliJ IDEA 23
3.5.3: GitKraken 23
3.5.4: Visual Studio Code 23
3.5.5: DBeaver 24
3.5.6: WinSCP 24
3.5.7: Putty 24
3.5.8: Figma 25
3.5.9: Wireshark 25
3.5.10: Browser Development Tools 25
3.6: Collaboration tools 25
3.6.1: Jira 26
3.6.2: Confluence 26
3.6.3: Discord 26
3.6.4: Microsoft Teams 26
3.6.5: GitLab 27
3.7: Hardware 27
3.7.1: Personal computer 27
3.7.2: Server 28
4: Results 29
4.1: System overview 29
4.2: Process 30
4.2.1: Schedule 30
4.2.2: Scrum 30
4.3: Database 32
4.3.1: Requirements 32
4.3.2: Designing phase. 33
4.4: Back-end development 36
4.4.1: Maven 36
4.4.4: Repository classes 38
4.4.5: Testing 39

4.5: Front-end development 40
4.5.1 Sketching 40
We decided to show the data in form of 42
4.5.3: Adopting universal design and principals. 43
4.5.4: Query engine viewer 44
4.5.5: Layout 45
4.5.6: Page content 45
4.5.7: Continuous Integration and Continuous Deployment (CI/CD) 47
4.5.7.1: CI/CD Frontend configuration and pipelines 47
4.5.7.2: CI/CD Backend configuration and pipelines 49
4.5.7.3: CI/CD Project variables 53
5: Discussion 54
5.1: Technical result 54
5.1.1: Database 54
5.1.2: Spring boot / GraphQL 54
5.2: Process 55
5.2.1: Collaboration 55
5.2.2: Jira and Scrum 55
5.2.3: Confluence 57
Chapter 6: Conclusion 58
6.1: Project conclusion 58
6.2: Recommendations 59
References 60
Appendices

1: Introduction

This chapter of the report introduces the assignment this bachelor thesis is founded upon, such as assignment description, constraints, and effect goals. The section also covers acronyms and abbreviations, and a structure of the report with a short description of what each chapter will cover.

1.1: Task description

Using developer time to extract reports from a database takes developers away from development and they quickly become a bottle neck for supporting day-to-day operations. This is expensive in the long term and the goal is therefore to explore solutions that takes developers out of the loop. The requester of information (should not need to know SQL) should be able to run the queries by themselves. The assignment consists of several parts:

- What is a good UX for the non-specialist end-user to retrieve information (check boxes, drop downs, JQL like Jira Querying language, …)
- Implementation of a prototype system
- Evaluation of the solution with end-users (Postnord business users)

1.2: Restrictions

Originally it was needed to use an NDA for this project, but the Postnord altered the task to avoid this. Instead of logging into their existing system that would require an NDA, we were instructed to make a simplified version of the system and use dummy-data.

When developing this project, the product owners wanted to use technologies that is like their current technology stack:

- IBM Informix as the database system
- Java 17 for back-end
- Vue.js for the front-end

1.3: Effect goals

Below is a list of requirements that determines the success of this projects:

- Implement database based on problem description document v2.
- Implement backend using Java with Spring boot.
- Implement a web-application using Vue.js.
- Create a prototype that can be tested.

1.4: Report Structure

A short description of what is covered in each chapter of the report.

Chapter 1 - Introduction: Introduces the reasoning behind this task, the problem domain, and constraints this report is based upon related to the effect goal for the reader.

Chapter 2 - Theory: A structured overview of the different theories that has been used as a foundation   for structuring the work and development in this project.

Chapter 3 - Method: Describes how the project was planned and the technical solutions and tools that were used in the project.

Chapter 4 - Results: A description of the results that was achieved in this project.

Chapter 5 - Discussion: A discussion of the technical results achieved, and the organizing of the project.

Chapter 6 - Conclusion: Conclusion of the project based upon the results and discussion.

2: Theory

After reading this section of the report, the reader should have knowledge of the theoretical material this project is based upon. The theory presented in this section which has been used is known practices and methods within the industry and has been proven useful when developing software projects by others in the past.

2.1: Agile Methodology

Agile methodology is based on an iterative approach to project management and software development. The idea behind this approach is that the development team can deliver small chunks of product to the customers in a rapid pace, instead of finishing the whole product before releasing it (Atlassian, 2023). In this way the development team is more adaptable to change in the product, get feedback quicker from the customers and can more easily roll out new functionality. The steps in an agile process for releasing product are plan, design, develop, test, and evaluate.

2.2: Scrum

Scrum is a way of structuring the working process within the agile methodology, and the word scrum originates from rugby. In rugby, scrum is where the team makes a compact formation to bring the ball forward together, while in this context Scrum is where the team comes together to deliver a product (Scrum.org, 2023). The most common Scrum terminologies are explained in sub-chapters of this chapter.

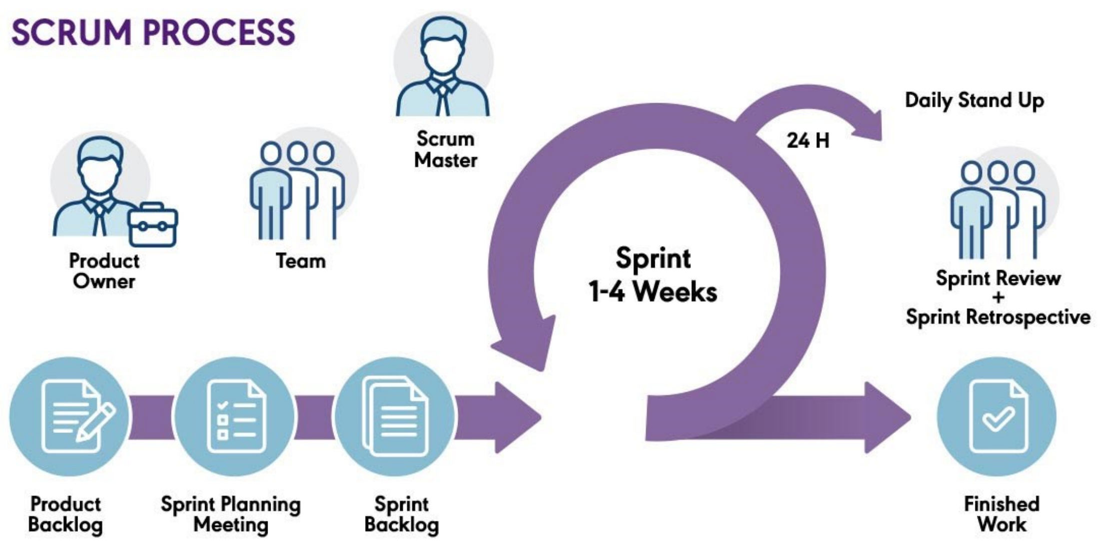


Figure 1: The Scrum process (www.pm-partners.com.au, 2021)

2.2.1: Team

The Scrum team consists of three parts, a product owner, a Scrum master, and the developers. The team should consist of people with the skillset to develop the product and solve the tasks at hand, and not rely on outer sources. The teams should ideally not exceed 10 people, since it is proven that smaller teams are more efficient. (Scrum.org, 2023)

2.2.2: Scrum Master

The Scrum master is the head of the Scrum team and may have many different tasks in the team. First and foremost, responsible for establishing the Scrum, setting up the team in the best way possible, and making sure that every team member is integrated in the team and knows the theory and practices to become efficient and independent. But the scrum master is also responsible for making sure that the team reach the product goals in time and works as a reminder for the developers making sure that they have done everything they should. (Scrum.org, 2023)

2.2.3: Product Owner

A product owner's responsibility is to make sure that the product goal is clear and understandable for the Scrum team. They set up the backlog, which is a prioritized list of what is to be in the product and further development of the product. Their most important role is to ensure that the product goals are met by the developers. The product owner also discusses feedback and new functionality for the product with stakeholders and customers, and has the final say in matters regarding changes or alterations in the product itself (Scrum.org, 2023).

2.2.4: Developer

A developer in the Scrum team is one of several persons that is responsible for developing the system. Since the Scrum team can consist of people with different background, they do not need to be a software developer, and may have skills in design, planning or other fields. But they are responsible for contributing to reach the product goals and complete each sprint set up by the team.

Software development is wide field with many topics, and if one team member has a skillset that is valuable for the team they may also be responsible for training and sharing knowledge with the other team members if this is valuable for the organization and team (Scrum.org, 2023).

2.2.5: Sprints

Sprints are one of the most important parts of the Scrum methodology and is a limited period where work is done regarding the product goal. Every sprint is planned from the backlog with the product goal as the objective, and the sprint goal is a subset of that.

Sprints are usually 1 – 4 weeks and, since little would be done in less time than one week, and 4 weeks might be to long of an iteration. The length of the sprint depends on the complexity of the problems to be solved in the given sprint, which are pulled from the backlog (Scrum.org, 2023).

2.2.6: Planning

Sprint planning is deciding the work that is to be done from the backlog together with the product owner and/or other people than may bring value to the planning. When planning the sprint, the team should try to answer three questions, and that's why, what, and how. As a thumb rule the sprint planning should not exceed more than two hours for a one-week sprint (Scrum.org, 2023).

2.2.7: Reviews

When a sprint is done the team presents the outcome of the sprint to the potential stakeholders and the product owners. Here the developers can show what has been done in the sprint, and what was not done in the cases they don't complete all the tasks from the sprint backlog. This also includes discussing problems that the sprint team met during the sprint, and showing how they solved the problems (Scrum.org, 2023).

2.2.8: Retrospectives

The final step in a sprint is the sprint retrospective, which focuses on the working process during the sprint. The retrospective focuses on improving efficiency amongst the team members, and routines might be added, altered, or removed based on the outcome of the given sprint. A sprint retrospective should not take more than one hour for a one-week sprint (Scrum.org, 2023).

2.2.9: User stories

User stories in web development, are a useful tool to describe how users interact with a system or product. They offer valuable insights on specific functionality a user might expect from a system as well as they show how different users might have varying expectations. A user story's primary goal could be to communicate technical requirements to stakeholders and product-owners, managing a projects scope or establish product requirements. (Wrike, 2023)

2.3: Database

Database is used to store a large amount of data. Data stored is often logically related and contains minimum amount of duplication. Database provides an efficient way to store, organize and manage data. In addition to the data, the database holds a description of the data. This is called metadata – *the data about data*. The important elements of a database are entity, attributes, and relationship. Entity represents a distinct object; attributes describe the entity and relationship represents the relationship between entities. These elements help databases to hold the data logically related. (Connolly & Begg, 2015)

2.3.1: Types of databases

There are many types of databases. The popular database types are hierarchical databases, relational databases, non-relational databases or NoSQL databases, object-oriented databases, distributed databases. The newest and largely growing types of database types are cloud databases and self-driving databases.

2.3.2: Relational databases

In relational databases, data is stored in tables which contain rows and columns. Rows represent tuple, a single row of unique data. Column represents attributes of the entity. Tables are logically connected using keys to access and manage data efficiently. Attributes of the table are used as keys. Primary key and foreign key are used to create logical relationship between tables. The primary key is an attribute which uniquely identifies each record in the table. Foreign key is an attribute which connects to the primary key of another table. (Connolly & Begg, 2015) The word relational does not mean relation between tables. The term relation roughly means table. Relational database represents data in tabular format (Batra, 2018).

These keys play an important role in the two principal rules of the relational databases, entity integrity and referential integrity. Entity integrity ensures that each tuple is unique using a primary key. Primary key cannot be null or empty to ensure the entity integrity. Referential integrity makes sure logical relationships created between tables are maintained to avoid inconsistency in data stored in tables. To ensure this foreign key must be identical or *null* to the primary of the connecting table. *Null* represents absence of the value, or the value is not applicable to the tuple.

2.3.3: Database Management System (DBMS)

Database Management System is used to define and manage the data stored in database. It acts as a bridge between database and the user´s application managing the data stored in database. DBMS consists of data and programs to access the data. Data Definition Language (DDL) is used to define datatype and relationship between data using constraints. Data stored in database is managed through Data Manipulation Language (DML). Users can Create, Read, Update, and Delete (CRUD) data using DML. (Connolly & Begg, 2015)

Data availability, data integrity, data security and data independence are the main objectives of DBMS. Data availability ensures that data is presented to wide range of users in a meaningful format, data integrity makes sure the correctness of data available in database, data security ensures that only authorized users get access to the data and data independence provides an easy way to manage the data, giving an abstract view of how data is stored. (Sumathi & Esakkirajan, 2007) MySQL, PostgreSQL, MongoDB, IBM Informix, and Microsoft Access are some of the popular DBMS systems.

2.3.4: Components of DBMS

Hardware, software, data, procedures, and people are the important components of DBMS. The system needed to run DBMS is hardware. It varies from organization to organization depending on their need. Software is DBMS itself, which is used to manage the data. Central and important component of DBMS is data. Procedure refers to information that is required to run the system. The last component people, refers to users involved in the system. They can be Data Administrator (DA), Database Administrator (DBA), database designers, application developers and end users. (Connolly & Begg, 2015)

End users are the group of people using the database to fetch the information they need. End users are classified into two groups according to their technical skills. Naïve users are users who do not have a technical understanding of DBMS and database. They fetch information using applications that are made to make it easy to fetch information without understanding DBMS. Sophisticated users have a good understanding of the DBMS and database. They are familiar with how the system works. (Connolly & Begg, 2015)

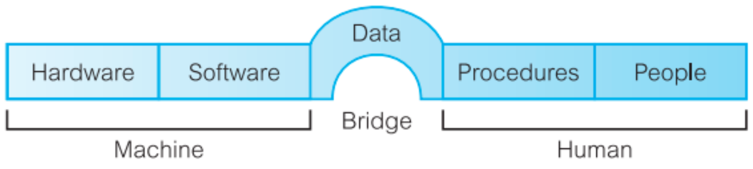


Figure 2: Components of DBMS (Connolly & Begg, 2015)

2.4: Database design

Database designing involves organizing the data according to the requirement. Conceptual database design, logical database design and physical database design are the three main stages of database design.

2.4.1: Conceptual database model

The first phase of designing a database is conceptual database design. In this phase, the focus is given to identifying the requirement. A data model is designed according to the requirements. A conceptual database model is created in this phase. This model is independent of implementation details and gives importance to the requirements. Entities and attributes are defined using the identified requirements. Relationship between entities is also defined in this phase. The conceptual model is used as source of the next phase of the database design, logical database design. (Connolly & Begg, 2015)

2.4.2: Logical database model

A logical database model is created in the second phase of designing a database. The model is tested and verified against the requirements during the process of designing. Data types and correctness of the model is tested using normalization in this phase. Normalization is discussed in detail in section 2.4.4. Logical model is used as source of the third phase of the design, physical database design. (Connolly & Begg, 2015)

2.4.3: Physical database model

In the third phase a physical database model is implemented. The main purpose of physical database design is to describe the process of implementing the logical database design and database-specific implementation of the data model. Storage structure and access methods for the data are identified to achieve optimum performance for the database system. Security protection is also ensured in this phase. Conceptual design and logical design are the most important phases of designing the database as they test and validate the requirements. Database design is done in iteration to provide the best design. (Connolly & Begg, 2015)

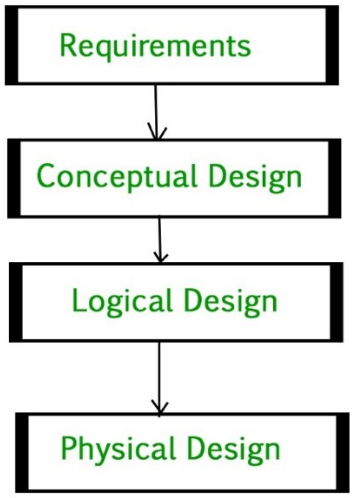


Figure 3: Stages of database designing (Geekforgeeks, 2023)

2.4.4: Normalization

Normalization is the process of organizing data in the database, to make the database more flexible by eliminating data redundancy and inconsistent dependency. Data redundancy takes more disk space and makes the data management difficult as changing data in one place leads to changing data in multiple places, which can lead to data mismatch. There are mainly three normal forms that should be followed to make sure that the database does not have data redundancy. (Learn Microsoft, 2023)

The three normal forms are,

- First Normal Form 1NF
- Second Normal Form 2NF
- Third Normal Form 3NF

In First Normal Form, a single cell must not hold more than one value, there should be a primary key and no duplicate rows or columns. In Second Normal Form, the table is in 1NF, and separate tables are used for values that are used in multiple records. Third Normal Form is achieved when the table is in 2NF, and all non-key attributes are directly dependent on the primary key and not on other non-key attributes. (Learn Microsoft, 2023)

2.5: Test data

In the world of software deployment, test data plays a crucial role in ensuring the software quality. With unity testing, developers can use the test data to verify potential defects early, improving code quality and allowing the developers to focus on programming instead of troubleshooting and fixing. Furthermore, using quality test data helps ensuring that software meets the necessary specifications.

2.6: Query language

Query language is a database programming language used to manage data in the database using DBMS. A query language should be able to create database and relation structure, manage data such as create, read, update, and delete, and execute queries. (Sumathi & Esakkirajan, 2007) Structured Query Language (SQL) is one of the most used query languages. Cypher, XQuery and SPARQL are some other examples of query language.

2.7: Programming Paradigm

The set of instructions given to a computer to solve a specific problem is a program. The set of notations used to write the program is a programming language. There are different categories of programming language such as first-, second-, third-, and fourth- generation. The higher the generation, closer it gets to the human language. Two important components of a program are data and algorithm. Data represents information and algorithm is steps involved in arriving at a solution to a problem. (Sharan & Davis, 2022)

The programming paradigm involves arriving at a solution by combining data and algorithms in different ways. Most used programming paradigms are imperative, procedural, declarative, functional, logic and object-oriented paradigm. (Sharan & Davis, 2022)

2.7.1: Object-Oriented Paradigm

An object plays an essential role in an object-oriented paradigm. Object encapsulates data and algorithm. In OOP objects interact with each other to solve a problem. Data defines the state of the object and algorithm defines the behavior of the object. Classes are the basic unit of OOP. They act as the blueprint of the objects. Objects are created from classes. An object is called also as instance of a class. Data or state is kept private in OOP. They are only available inside the object. Data is accessed using methods which are available outside of the class. Methods define the behavior of the object. Major principles of OOP are abstraction, encapsulation, inheritance, and polymorphism. (Sharan & Davis, 2022).

2.7.1.1: Cohesion

Cohesion refers to the degree of relationship between elements in a module, and separating responsibilities for classes/methods. Developers should aim for high cohesion, meaning that related functions are grouped together, and every function has a high degree of independence/ responsible for only one task (Pagade, 2022).

2.7.1.2: Coupling

Coupling is the degree of how tightly different software modules are connected. Developers should aim for loose coupling, meaning that changes in module will have little to no effect on other modules. This makes the system easier to maintain and expand (Pagade, 2022).

2.8: API – Application Programming Interface

Application Programming Interface (API) is a set of definitions and protocols that help different applications to communicate with each other. It acts as a middle layer that transfers data between different applications. Using APIs, systems can open their data and functionality to other systems. This makes the developing process efficient as other developers can make use of the existing data and functions via APIs. (IBM, 2023)

APIs can be implemented using different protocols. API protocols enable a standardized information exchange between systems by defining accepted data types, commands, and syntax. Different types of API protocols are Simple Object Access Protocol (SOAP), XML-Remote Procedure Call (XML-RPC), JavaScript Object Notation-RPC (JSON-RPC), Representational State Transfer (REST). Most of the APIs expose their data and functions over the internet. (IBM, 2023) The following are different types of web APIs Private, Public, Partner and Composite APIs. Public APIs can be accessed by everyone, Private APIs are mostly for internal use in companies, Partner APIs are used between companies of the same interest and Composite APIs combine two or more APIs to solve a complex problem. (amazon, 2023)

2.9: Testing

Software testing is reassurance that a given piece of code behaves the way it was intended to. There are many types of testing in software development, but the goal is to create more robust, bug free, less expensive, and efficient code (ibm.com, 2023).

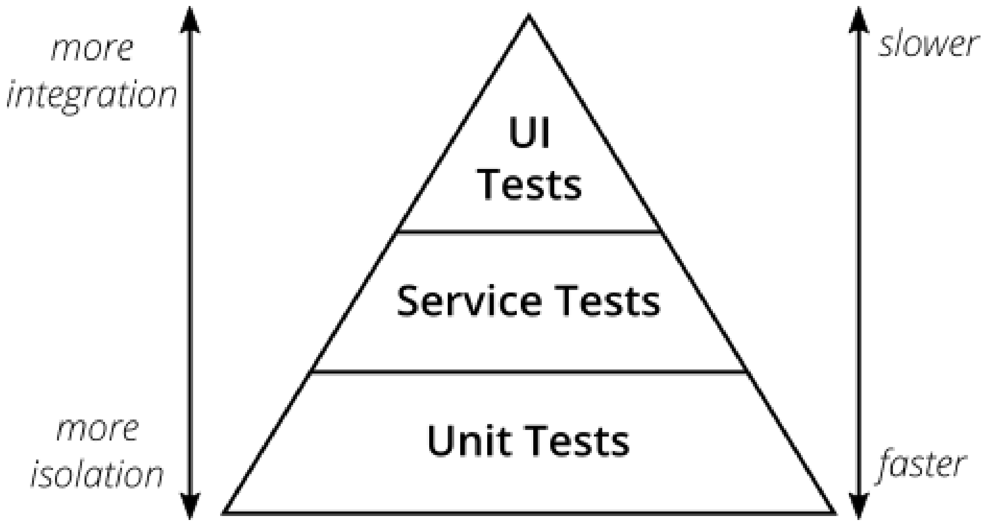


Figure 4 - Test pyramid (martinfowler.com) (Vocke, 2018)

2.9.1: Unit testing

Unit testing is a specific type of tests that is written for testing the functionality on a piece of code in isolation, typically a function or method (smartbear.com, 2023). As shown in figure 2, unit tests are faster, and makes the biggest part of the test pyramid. Preferably as many functions as possible should be tested in isolation.

2.10: Design patterns

In software engineering, design patterns are a solution to commonly known problems that occur in programming. A design pattern is not code, but a template or description on how to solve a problem that can be used in different situations (sourcemaking.com, 2023).

2.10.1: Controller-Service-Repository

A pattern used in some Spring boot applications is the Controller-Service-Repository pattern. This is mainly used for separation responsibilities, where the controller class is responsible for exposing functionality to end-points, service holds the business-logic, and repository communicates with the database (Collings, 2021).

2.11: Code Reviews

Code reviews as the name implies, is a chance to get a second opinion of the code before any CI/CD pipelines. The purpose of the reviews is to ensure better programming quality and structure as a second person goes through the code and potentially catching or identifying possible bugs before it's merged into production as an example. Having a second opinion on the code produced grants several advantages such as: preventing an unstable code base, teams might possibly learn new techniques and solutions, discovering bugs early, increase collaboration and improve code quality. (Gitkraken, 2023)

2.12: Version control

Version control has become an essential part of modern software development, which enables developers to track and manage changes to files over time, creating a historical record of the files in the project. This system offers several benefits to development teams, including the ability to collaborate on projects efficiently, merge different versions of files, and revert to an earlier version of the files when necessary. (Git-scm, 2023)Version control keeps "all the changes to files under revision control". (Chacon & Straub, 2014)

2.12.1: Git

Git is a popular open-sourced version control system that has become an important tool in modern software development. Its speed, efficiency, and flexibility make it a valuable tool for developers (Chacon & Straub, 2014). Git is a distributed system that allows developers to work offline, and merge changes when they are online again, thanks to each developer having their own local copy of the codebase. Git provides features like branching, merging, code review, and continuous integration, enabling development teams to collaborate more efficiently. Git also supports local and distributed collaboration, allowing more efficient collaboration and more effective code management. (Git-scm, 2023)

2.12.2: Git Flow

Git Flow is a branching model for Git, designed to simplify the software development process by providing a structure for organizing code changes with separate branches for development, testing, and release. This enables developers to collaborate more efficiently by defining rules for creating, merging, and managing branches (Atlassian, 2023). By creating separate branches for development, testing, and release, Git Flow helps developers to manage their codebase more effectively. It also includes features for version tagging and hotfixes. Git Flow also helps reduce conflicts and errors and enables development teams to deliver high-quality code (Gitkraken, 2023)

2.13: Design principles

2.13.1: Don Norman's 6 usability principles

Don Norman developed 6 principles for designing products, which is also applicable for IT systems. These principles revolve around visibility, feedback, constraints, mapping, consistency, and affordance.

The first principle is based on the concept of visibility. Given functions are visible to the user, it'll result in a higher chance of a user being able to find the functions and use them. How this relates to web design one can see by taking a navigation menu as an example. It's always better to show the menus/options in a visible form instead of hiding all the functions from the users in a hamburger menu hiding in a corner. If functions are visible, then the user will easier understand what options are available.

The second principle bases itself on feedback, this meaning for every action a user makes on a given system, the system should provide feedback to the user to confirm what action was taken. How this may apply to web design could range from the user clicking a button and bar displays the loading progress of the action, or an image plays a small animation if the user hovers their mouse over it. Regardless of the action the system should aim to inform the user with feedback to let them know their action was taken.

The third principle revolves around constraints, limiting the allowed actions a user can take on in each system or product. For web design, developers should constrain the user from being able to input invalid data and don't allow users to make actions which are too complicated. An example of this in practice could be to disable options before certain criteria is met such that the users doesn't break the program.

The fourth principle talks about mapping, in other words the relationship between the controls and their effect. Here the focus lays on how to properly place various elements in relation to their expected function. For web development this is an important principle for establishing a fluent feeling on web applications. Examples of this principle being used in practice for web applications could be displaying a menu to the user where the order of the menu follows the structure of section 2.4.4 the web page. (Find example to insert)

The fifth principle is about consistency, the system or product should be consistent within its own system as well other systems outside its own. This principle is important in the way that users will have an easier time using the system / product if it is like something they've interacted with before or recognize. For web development this could refer to a website's navigation menu staying on a consistent place all throughout the website. However, consistency also means that you cannot have two almost identical looking products which are different they should not look similar, otherwise the user will be confused.

The sixth and last principle, Affordance. If a user interface may be able to give its user a clue for it's possible use, then it will be easier to use. This applies to web development where developers should give their users a clue in the form of an example or preview on what a certain function does in the web application such that the user understands its intended use.

(Aela, 2023)

2.13.2: Web design rules / Universal design

Universal design refers to the term that a product should be useable by all people, without a need of adaptation or specialized solutions (Access computing, 2023). This also applies for web development, in the sense of a digital application being useable by all. For blind users, this can mean building a website using proper HTML semantics such that screen readers can properly tell the contents of a page. A user that's color blind should be able to read the contents of a page without problem. For these reasons there are a few web design rules to follow when building an application to help make sure it is useable by all.

2.13.2.1: Typography

Within typography it's generally smart to be aware of the font in use, as different types of fonts have varying levels of readability for users. The priority should be to choose a font that has easily distinctive letters and having airy and open characters as letter shapes won't easily blend or blur into each other. Another guideline to take note of is to keep between 45 to 75 characters per line as anything more than that could contribute to a readers-fatigue. As for font sizes, it depends on the platform is being read on, as smart phones would use font sizes in between 12-16pt and desktops would use around 16-20pt. (Toptal, 2022)

2.13.2.2: Color and themes

Color impacts websites more than one would initially expect and comes with a few guidelines on how to properly utilize them for web development. The first step for choosing a color theme for a website, could be to attempt to appeal to a color matching the sites personality. People tend to associate different colors with emotions, which will help impact the feeling of a website if implemented correctly. Take blue for instance, blue conveys a feeling/emotion often associated with trust and loyalty. Whereas a red color would represent a feeling of danger or compassion. (Chapman, 2023)

Another guideline could be to follow the 60/30/10 rule. The rule implies that the main color (generally a natural one) should be used on 60% of the page, whereas the secondary color should be used 30% of the page and the tertiary color for the remining 10%. By parting it up this way, it'll help to create a sense of balance on the website and reduce chaos (Gordon, 2021)

2.13.2.3: Images / icons

When choosing images for web design, it's generally a good rule to only use images that are relevant to the website being developed. These images should be used in such a way that improves the users experience or helps reduce the text overload on the website. Here icons can come into play which are able to reduce text overload, being to replace the text with a universal visual representation in its place.

Other rules when using images in web design include making sure the images are scalable, due to different platforms having different display resolutions (a phone has a different screen size than a PC monitor). Images should also be rescaled or cropped accordingly to fit the site, as well as they should not be too large. Since large images will affect the websites performance. (WebFX, 2023)

2.13.2.4: Whitespace

By utilizing whitespace properly, one can enhance a user's reading experience when browsing through websites. By adding a bit of spacing between compact blocks of text will make it more readable and comfortable for the users' eyes. Furthermore, whitespace may be used to help guide the user through a website as the empty spaces draws their eyes towards what is displayed. The use of whitespace also lets the user get some breathing room and reduce strain on their eyes while reading. (UX Engineer, 2023)

2.13.2.5: Responsive web design

In the earlier days most websites and applications were viewed on desktop computers with large screens, but today most of the traffic on internet is on mobile devices and tablets of different screen sizes. In 2023 60.06% of the global internet traffic came from mobile devices (oberlo.com, 2023).

Responsive web design is creating the application with the ability to automatically scale to fit the screen-size of the device it is viewed on. This is achieved by e.g., resizing elements, width, height, crop images, change font size and alter content to fit the screen size of the user's device. The reason behind this is to create a better user experience for every user on the site (w3schools.com, 2023).

2.14: Microservices

Microservices is an architectural style where the application is structured as a set of independent services that can communicate with each other via API's (microservices.io, 2023).

The point of microservices is to divide the application into smaller parts based on responsibilities and maintain loose coupling.

3: Method

After reading this section of the report the reader should know how the project were structured, as in planning, division of labor, procedures for quality assurance and code reviews. The chapter also covers all the tools and technologies used in development of this project. This chapter provides a detailed description of the research methodology that is related to the study.

3.1: Preliminary project plan

This sub-chapter covers the first steps in the planning process regarding this bachelor's thesis, which consists of group contracts, scheduling, and risk assessment. For more detailed information see **appendix A.**

3.1.1: Group contract

The group contract is a mandatory part of this project and is a contract that is created between the group members with input from NTNU. The reason behind the contract is that each of the students contributing to the project should agree on what is expected of them in terms of attendance, product results, risks connected to the project. This ensures that all team members know what is expected and don't skip work during the project.

3.1.2: Schedule

The schedule is a part of the preliminary project plan and is meant as a guideline for partial delivery during the whole development process. Our schedule consists of all important steps in the development process and key dates for delivery. The plan is meant as a reminder for the team and should help the team to keep a linear progression throughout the semester.

3.1.3: 3-party agreement

A 3-party agreement is mandatory to sign between the involved parties in the group when working on a bachelor's degree or master's degree in NTNU. The agreement exists to protect both the students, NTNU and the product owner, regarding the project and owner rights to the product. To avoid potential conflicts all parties involved in the project signed the document, meaning the head of department, supervisor, product owner and all group members.

3.2: Project organizing

After reading this section the reader should know how the project is organized. Much of the material in this section has already been introduced in chapter 2.

3.2.1: Roles

This project has been structured around the principles of Scrum, which was introduced in the theory section, which consists of the developers, scrum master and product owner.

- **Developer**: As a developer in the team, the responsibility and decision-making rights are equally divided amongst all team members. If one or more developers have certain skills that is crucial to the team, they share that information with the rest of the group for the benefit of the product.
- **Scrum master**: The supervisor also acts as a Scrum master, where she helps plan what is to be on the agenda for the upcoming week and guide us towards the goal. Besides doing this, the supervisor has helped us with discussion of technical solutions and narrowing down problems.
- **Product owner**: Postnord is the product owner of our application and have come up with ideas and thought about how the application hopefully will be like.

3.2.2: Daily stand-up

At the start of every day the team has a daily stand-up, which should not take more than five minutes. At the beginning after implementing Scrum the sprints took way longer, and we had to discuss how to make them more efficient. As mentioned in chapter 2, stand-up is a practice where every team member shares what they are doing for the day, if they have any issues that needs addressing before they continue with their work. This is also a nice opportunity to greet everyone for the day, and make sure that everyone is fit and ready to contribute to the project.

3.2.3: Sprint planning

Before starting to work on the actual project we had to plan what was necessary to complete the objectives given and did this by planning a Sprint. We had created a high-level list of all things that had to be done and picked objectives from that list.

Our team worked in one-week sprints which we agreed with our product owner and our supervisor.

3.2.4: Sprint review

At the end of every sprint the team reviews the product output for the latest week and discusses it with the product owners and our supervisor, separately. In these meetings we discuss difficulties and how problems were solved/not solved and get feedback on the output. After presenting for the product owners, the team members discuss the sprint internally before continuing.

3.2.5: Sprint retrospective

After every sprint is complete the team discuss if there are any improvements that can be done to the workflow process. This is not regarding the process, just on how things are done like our routines and how we work together. As an example, we realized here that we used too much time on the daily stand-up before starting to work, so the retrospectives were helpful.

3.2.6: Meeting notice

At the start of the semester meetings were called in by our team, once a week with the supervisor, and once a week with the product owners. But after a few weeks we decided to save time and sat up a fixed time to have the meetings with both parties. Meetings with product owners was a Teams meeting on Fridays at 13.00, and supervisor Wednesdays at 09.00.

3.2.7: Meeting notes

After every meeting with both the supervisor and the product owners, the team wrote a meeting note in Confluence to keep track of what had been discussed in all the different meetings at different times.

3.3: Quality assurance

This section covers the routines and principles we used for assuring quality in our work during this project.

3.3.1: Code review

A code review is normally done with the person who wrote the code snippet and the person that has the most expertise within the given language/framework the code is written in. In our team the code experience is the same, so we did them together as a group review. This was always done before merging the code into the given branch it was supposed to be merged to eliminate the possibility of badly written code going to production.

3.3.2: Testing

As we discussed in chapter 2.9, testing is a critical part of the project development process, and the team used several tools to ensure the quality of the code. For unit testing, the team used Junit, a popular testing framework for Java that allows developers to write and run tests easily. Junit is open source and widely used in the world, making it a natural choice for testing Java applications.

In addition to unit testing, the team also used Postman to test the RESTful API endpoints. Postman is a powerful tool that developers can use to test API endpoints and track response times, request size, and similar. It is user-friendly and supports a wide range of HTTP methods, making it a good choice for API testing. (Postman, 2023)

3.3.3: Continuous Integration and Continuous Deployment

Continuous Integration and Continuous Deployment hereafter referred to as CI/CD, have been utilized for both backend and frontend, which helps the development process.

When code is pushed to the repository, the CI/CD pipeline is triggered, which automatically builds, and deploys the changes to the server. This saves the team time and reduces the chances of human errors when deploying updates to the application.

For the frontend and backend, we have used Gitlab CI/CD as our CI/CD tool. Gitlab CI/CD is a powerful tool that allows for automated building, testing and deployment of code. It integrates with GitLab, which we used as our primary code repository and collaboration platform.

Using GitLab CI/CD allowed us to ensure that changes were properly tested and deployed. It also helped reducing the amount of manual work required for building, testing, and deployment, freeing up time for the team to focus on more important tasks. (Red Hat Inc., 2023) (GitLab, 2023)

3.4: Programming languages and frameworks

This section covers the programming languages and frameworks that is used in this project.

A framework is code for solving specific problems that has been written by other programmers, so that anyone solving a similar problem can take advantage of that. Frameworks are usually written for a specific language, and the benefit of using a framework is that it has been developed, used, and tested by other computer engineers (Codecademy, 2021).

3.4.1: Java

Java is an object-oriented programming language as well as a software platform. To make use of the programming language, it is required that a JDK (Java Development Kit) is downloaded which is supported on Mac OS, Windows, and Linux. When code is compiled, it’s transformed into a set of instructions (Java bytecode) for Java’s Virtual Machine (JVM) which is a part of the Java runtime environment (JRE). The transformed code runs on any system where JVM is supported and does not make any modifications on the system when it runs, allowing Java code to be run anywhere.

Java’s software platform consists of the JVM, Javas API and a complete development environment. JVM will parse and run through the compiled code. Java’s API is consisting of many sets on libraries including such as, basic objects, networking and security functions, web services and more. Taking javas object-oriented programming together with java’s software platform makes this a powerful tool software development. (IBM, 2023)

3.4.1.1: Maven

Maven is a build tool for Java-based projects (mainly) and was created to make developing Java-applications easier. Maven’s objectives are to simplify the build process, provide a uniform build system, increase quality of project information, and improve development practices (maven.apache.org, 2023). One of the fundamental concepts of Maven is the POM (an XML-file), which contains configuration details Maven use to build the project (maven.apache.org, 2023).

3.4.2: Spring boot

Spring boot is a java-based framework used to build and deploy web applications and microservices. It has its base from Spring framework. It is built upon the Spring framework. Building applications using Spring became time consuming and difficult as the spring framework became large. Spring boot made it easy to build and run a standalone application with minimum amount of time. The three core capabilities of Spring boot are autoconfiguration, an opinionated approach to configuration and the ability to create standalone applications. These core capabilities help to setup a spring application with minimal configuration and setup (IBM, 2023)

3.4.3: JavaScript

JavaScript is a programming language, as a client-side scripting language used to make dynamic and interactive webpages. In the initial stages the webpages displayed only static information. JavaScript made it possible to make the webpages dynamic. JavaScript supports many programming paradigms as discussed in chapter 2.7 . JavaScript can also be used a server-side coding language. Client-side JavaScript has many frameworks. Vue.JS, React and AngularJS are the popular among them. (Amazon, 2023)

3.4.4: Vue.js

During the development process of our project, we decided to use Vue.js as our primary front-end framework. The main reason for this decision was due to the demand from the product owner, who was looking for a modern and efficient front-end solution for our web application.

Vue.js, hereafter referred to as Vue, is a progressive JavaScript framework that has gained a lot of popularity over the years. It is a reactive framework that utilizes a virtual DOM to efficiently update the user interface without the need of re-rendering. Vue is open source, easy to use, learn and understand for building user interfaces. It is known for its simplicity, flexibility, performance and the easy to learn syntax. Its component-based architecture allows developers to create reusable components, making it easy to maintain, manage and makes it easy to scale the application as it grows.

Vue’s reactive system efficiently updates the user interfaces when data changes while being easy to maintain and manage. Vue provides tools to offer routing, state management and building single-page applications that dynamically update content while users interact with the application. It supports server-side rendering to enhance search engine optimization and performance.

Vue’s simplicity, easy to use, and efficiency make it a popular choice for building applications. It is flexible and can be integrated with other JavaScript frameworks and libraries to build applications that scale between small and simple to large and complex. Whilst Vue is fast and efficient, it might not be the best choice for every high-performance application. Other JavaScript frameworks might be more efficient.

In summary, Vue offers a versatile framework that provides efficiency, flexibility and simplicity through its reactive system, intuitive API, and rich ecosystem. Its component-based architecture and support for single-page applications make it easy to maintain and scale applications. Even though it is less popular than other JavaScript frameworks, Vue has excellent documentation and a supportive community, making it a great choice for developers looking to build high-quality applications fast and easily. (Vue.js, 2023) (Vue.js, 2023) (Sitepoint, 2023) (Sitepoint, 2023) (Sitepoint, 2023)

According to Vue’s documentation, the framework “builds on top of standard HTML, CSS and JavaScript with intuitive API and world-class documentation” and provides “truly reactive, compiler-optimized rendering system that rarely requires manual optimization” along with “a rich, incrementally adoptable ecosystem that scales between a library and a full-featured framework.” (Vue.js, 2023)

3.4.5: GraphQL

During the development process, the product owner expressed a desire for us to use GraphQL, a query language for APIs. GraphQL allows for efficient data retrieval, reducing the number of requests needed and improving performance. It also enables clients to specify exactly what data they need, reducing the amount of unnecessary data being transferred.

We decided to use GraphQL due to its benefits and suitability for the project’s needs. Its flexible and type-safe nature allowed us to easily modify our data schema as requirements changed, and its strong tooling made it easy to debug and optimize our queries.

The implementation of GraphQL went smoothly, and we were able to integrate it seamlessly with our backend server. The use of GraphQL enabled us to efficiently retrieve the necessary data and improved the overall performance of the application. Its flexibility and strong tooling made it a great choice for our project’s needs.

(altexsoft, 2023) (GraphQL, 2023) (Honeypot, 2023) (The GraphQL community, 2023)

3.4.6: REST API

Using a REST API is one common and effective approach for building web applications. In this project the Spring Boot backend serves as the server-side application, while Vue handles the client-side rendering and user interactions in the browser. The backend implements the REST API endpoints that exposes the application’s functionality and data and handles incoming HTTP requests form the Vue.js frontend, processing the requests, and returning appropriate responses. In this project, the frontend shall only have READ requests to the relevant endpoints. (The GraphQL community, 2023)

3.5: Development tools

Development tools are created to increase productivity and help developers across the world write better programs. In this section the reader will get an overview over the different technologies that has been used under the development of this project, such as Git-clients, IDEs, and more.

An IDE (Integrated Development Environment) is a tool that helps software developers write code faster, safer, and better. IDE’s have several benefits over a regular text program, such as building executables, debugging, autocompletion, syntax highlighting, and more. Some IDEs might be suited to support several programming languages, as others are meant for specific languages depending on the editor (Codecademy, 2023).

3.5.1: WebStorm

“WebStorm is an integrated development environment for JavaScript and related technologies. Like other JetBrains IDEs, it makes your development experience more enjoyable, automating routine work and helping you handle complex tasks with ease.” (JetBrains, 2023)

Developed by JetBrains, WebStorm has been an important tool for many developers, including those working on the front-end part of this project. It offers a wide range of advanced features that helps developers writing clean, efficient, and reliable code.

One of the most important features of WebStorm is the intelligent coding assistance, which provides developers with real-time feedback and suggestions as you code. This includes highlighting syntax errors, suggesting code completions, and provide code refactoring tools. There is also integrated debugging tools, that allows developers to debug code right from within the editor.

While WebStorm come with a price tag of € 198.75 for the first year, or € 19.88 per month both including VAT, JetBrains also offers a discount if you subscribe for two years or more. The Second year will cost € 158.75, and € 95 after three years, both including VAT, and is offering several benefits that can make it well worth the investment. For students at NTNU, the IDE is available for free, making it even more attractive for those who are starting with web development. (JetBrains, 2023)

3.5.2: IntelliJ IDEA

IntelliJ IDEA, hereafter referred to as IntelliJ, is a popular Integrated Development Environment (IDE) also developed by JetBrains that are primarily designed for Java and Kotlin development. IntelliJ provides a bunch of features and tools to help the development process, including code completion, debugging, and version control integration. IntelliJ also supports several languages, including Kotlin, Groovy, and Scala making it a good Choice for developers (JetBrains, 2023).

For the back-end development of this project, the group used IntelliJ as it provides integration with the Spring Boot framework. As a student of NTNU, you can get the ultimate edition of IntelliJ for free, which usually costs € 74.88 per month for individual use, making it even more attractive for those who are starting with web development. However, JetBrains also offers a community edition for free, which have fewer features included.

While IntelliJ IDEA Ultimate initially come with a price tag of € 748.75 for the first year, or € 74.88 per month, JetBrains also offers a discount if you subscribe for two years or more. For instance, the Second year will cost € 598.75, and € 448.75 after three years, all prices including VAT, and is offering several benefits that can make it well worth the investment. Investing in IntelliJ comes with several benefits that can make it worth the cost, including a user-friendly interface that helps speeding up the development process (JetBrains, 2023).

3.5.3: GitKraken

GitKraken is a tool for working with git, with a graphical user interface and is used by more than 10 million developers at 100,000 different organizations over the world (GitKraken, 2023).

One of the main reasons for using Git via GUI over CLI is to increase productivity. It is much easier to keep track over all the commits and merged branches since this become graphical in GitKraken. GitKraken stores all the commits for all the different team members, for each branch in the repository. This makes it easy to look up anything that has been done in the project if the team have good routines for committing. And if a user really needs the CLI to do a specific task, there is an option to open this directly in GitKraken.

3.5.4: Visual Studio Code

Visual Studio Code, commonly known as VS Code and hereafter referred to as VS Code, is a lightweight and high-performance code editor that has a widespread popularity among developers. It supports various programming languages, making it a wise tool for front-end and back-end development, as well as with other types of projects. One of the most significant advantages of VS Code is its platform-independent nature, as it can be used on macOS, Windows, and Linux based systems, and it is free to download and use, making it an attractive option for developers looking for a cost-effective solution (Visualstudio, 2023).

For this project, VS Code and WebStorm were the preferred choice of code editor for the front-end development phase, as it provides an extensive range of features and plugins that helps with the development. With VS Code, developers were able to take advantage of the provided support for various tools and frameworks, including Vue.js GraphQL, and similar. Due to its user-friendly interface it is easier for developers to work on complex projects efficiently.

3.5.5: DBeaver

DBeaver is an open-source database tool, which supports various databases; including the Informix database used for this project (DBeaver, 2023). The database tool provides an easy-to-use GUI where you can manipulate data as well as creating/exporting data to various file formats such as CSV. (DBeaver, 2023)

3.5.6: WinSCP

We used WinSCP primarily for transferring files between our local computer and servers. This included transferring large data sets, code files, and other relevant data that was required for the proper functioning of our system. WinSCP's secure and efficient file transfer capabilities ensured that our data was transferred safely and quickly, without any loss or corruption. Its user-friendly interface and automation feature also made it easy to manage our files and streamline our workflow. Overall, WinSCP played a critical role in the smooth operation of our system by providing a reliable and secure means of transferring files.

(WinSCP.net, 2023) (WinSCP.net, 2023)

3.5.7: Putty

PuTTY is a widely used terminal emulator that allows for remote server management through various network protocols such as SSH and Telnet. Its user-friendly interface, support for multiple operating systems, and configuration options make it an essential tool for managing servers remotely. PuTTY also provides secure encryption during communication, ensuring the protection of sensitive data from interception or compromise. Its flexible display options and ability to save multiple sessions make it a versatile and valuable tool for managing servers remotely.

We primarily used PuTTY for remote server management, which involved connecting to our servers remotely through SSH and Telnet protocols. This allowed us to manage and monitor our servers' performance, install and configure software, and troubleshoot any issues without physically being on site. PuTTY's user-friendly interface, secure encryption, and support for multiple operating systems made it an ideal tool for our remote server management needs. Overall, PuTTY played a crucial role in the efficient and secure management of our servers, ensuring that our systems were always accessible and performing optimally. (Putty, 2023) (Putty, 2023)

3.5.8: Figma

Figma is an online browser-based design tool where multiple people can collaborate on the same file, allowing developer teams to work together and design prototypes for everything from desktop applications to phone apps. Figma also has a large community where users can make widgets which are public for everyone to use, making this design tool a strong choice for creating strong wireframes. (Figma, 2023)

For this project, Figma was used in the earlier stages of development to establish a general idea of how the layout of the site should look. Later, these sketches were improved upon but were not referenced completely as new ideas rose during development.

3.5.9: Wireshark

Wireshark is a network protocol analyzer that we used to diagnose traffic between our computer and servers. It helped us identify and resolve issues, such as slow response times caused by incorrectly configured routers, by analyzing the packets of data being sent and received. Wireshark's ability to provide detailed information about network protocols and filter captured packets made it a valuable tool for ensuring the smooth and efficient operation of our system. (Wireshark, 2023) (Wireshark, 2023)

3.5.10: Browser Development Tools

The browser development tools are a set of powerful resources that can be an essential resource for web developers and can make it easier for web developers to design, debug, and optimize web applications effectively. These tools provide solutions to create, analyze and improve web pages, making the development more efficient and effective.

In the browser development tool, you can find inspect element tool, console tool, sources tools, network tools, performance tools, memory tools, application tools,

The tools include the Inspect Element tool, which allows developers to inspect and modify the code of a web page. There is also a Console tool that provides a command-line interface for executing JavaScript code and makes it possible to debug errors. (Mozilla.org, 2023)

3.6: Collaboration tools

The collaboration tools used during the development process are discussed in this section of the report, covering the file sharing, Voice over Internet Protocol (VoIP), and video conferencing tools that were utilized. These tools played an important role in creating effective communication and collaboration among team members.

The file sharing tools made it easy and secure to share files and documents, which helped the team work together more efficiently. This improved collaboration among team members and made it easier to complete tasks. The VoIP tools allowed team members to talk to each other in real-time, no matter where they were located. This helped the team to communicate more effectively and reduced delays in completing tasks. The video conferencing tools allowed team members to hold virtual meetings, which made it easier to discuss project progress and make decisions.

The use of these collaboration tools helped to improve teamwork efficiency, which led to increased productivity and better project outcomes. In today’s modern workplace, collaboration tools are an essential component of teamwork, and we were grateful to have had access to them during the development process.

3.6.1: Jira

Jira is a tool created by Atlassian for organizing work when using an iterative approach such as Scrum. By using Jira all team members can get an overview over previous and current sprints, tasks, and log their work in a simple way. Since this is a web-application it’s also possible for team members that works remote to keep an overview of the sprint.

3.6.2: Confluence

Confluence was created by Atlassian and released in 2004 as a collaborative documentation tool for software projects. By using Confluence team members can create, organize, and share documents in a simple way.

Confluence also integrates with Jira, but where Jira focuses on the sprints, Confluence is meant as a wiki-tool for the project. Typical information that is stored here is sprint review notes, retrospective notes, documentation for the system, and design principles amongst other things.

By storing all such information in one place, it’s easy for the team members to look up the documentation for the system and logs if needed during development of the project.

3.6.3: Discord

Discord is VoIP collaboration tool where people can send text, talk via microphone, and share video with others in simple steps. The service can either be installed as a desktop application or run in the browser by logging in on discord’s home page (Discord, 2023).

This is the main communication tool that has been used during this project when working away from campus, because it is easy to set up, log into and use. Discord lets users stream their screens live with a few simple steps, which is a useful tool when working remote and not in group rooms at campus. In this way team members can easily ask the others for help and showing it visual instead of explaining which is much more time consuming.

3.6.4: Microsoft Teams

Microsoft teams is a collaboration application meant for organizations. The app has the possibility for videoconference, filesharing, grouping and messaging all in one place (Microsoft, 2023). Teams has been the application that we have used to have video meetings with the product owner, and for file sharing regarding the bachelor thesis report and preliminary project report.

3.6.5: GitLab

As mentioned in chapter 2, section 2.9.1 – version control such as Git allow developers to track changes made to the project files that is under version control.

When working on a project in collaboration with others, it’s essential to share project files and make them easily accessible for all team members. GitLab is a provider of remote repositories and is widely used amongst developers all over the world. By using GitLab, team members can access, update, and share changes to project files with little effort in an efficient way (git-scm, 2023).

Another benefit by using remote repositories for project work is that they serve as backup for the project files, in case of issues on the local computer(s). This feature of remote repositories ensures that crucial project data is not lost due to unfortunate events with the local machine.

3.7: Hardware

In this section, we will be discussing the hardware that was utilized to successfully complete the project. This information is essential in understanding how the project was executed and what components were involved in its implementation. We will provide details on the specific hardware used and their roles in the project.

It is worth noting that hardware refers to the physical components of a computer system or device, which includes the input and output devices, storage devices, processing units, and other related peripherals. In the case of this project, the hardware used was critical to the success of the project, and therefore, it’s important to outline their roles.

We will discuss the specific hardware components used in this project and how they were utilized to achieve the project objectives. This may include details such as the type of hardware, specifications, and other relevant information.

3.7.1: Personal computer

As for the Personal Computer (PC) aspect of our work, each team member has utilized their own laptop, which typically runs on either a newer version of macOS or Windows. Some members have opted to use an external monitor to provide a more comprehensive overview of their work. This setup has proven to be quite effective, allowing us to work on our tasks in a flexible and efficient manner. However, it’s worth noting that we have also taken the necessary security measures to ensure that our personal devices do not pose a risk to the project’s confidentiality or integrity.

3.7.2: Server

The ICT department at NTNU has granted us access to an OpenStack server, which is essentially a virtual machine (VM). The server is equipped with 2 virtual CPU cores, 6 GB of RAM, and a 40GB hard drive, and it runs on the Ubuntu 22.04 operating system. We are currently utilizing this server to host our backend, frontend, and database components, all of which are operating within Docker containers that are specially configured to run on this server.

4: Results

This is the main section of the report, and this section will cover all the results and findings throughout the whole development process. The section is divided into subsections where each section covers the results related to the given topic.

4.1: System overview

The general result of this project is that the team made a web-application for extracting information from a database, using the different tools and frameworks precented in chapter 2 and 3.

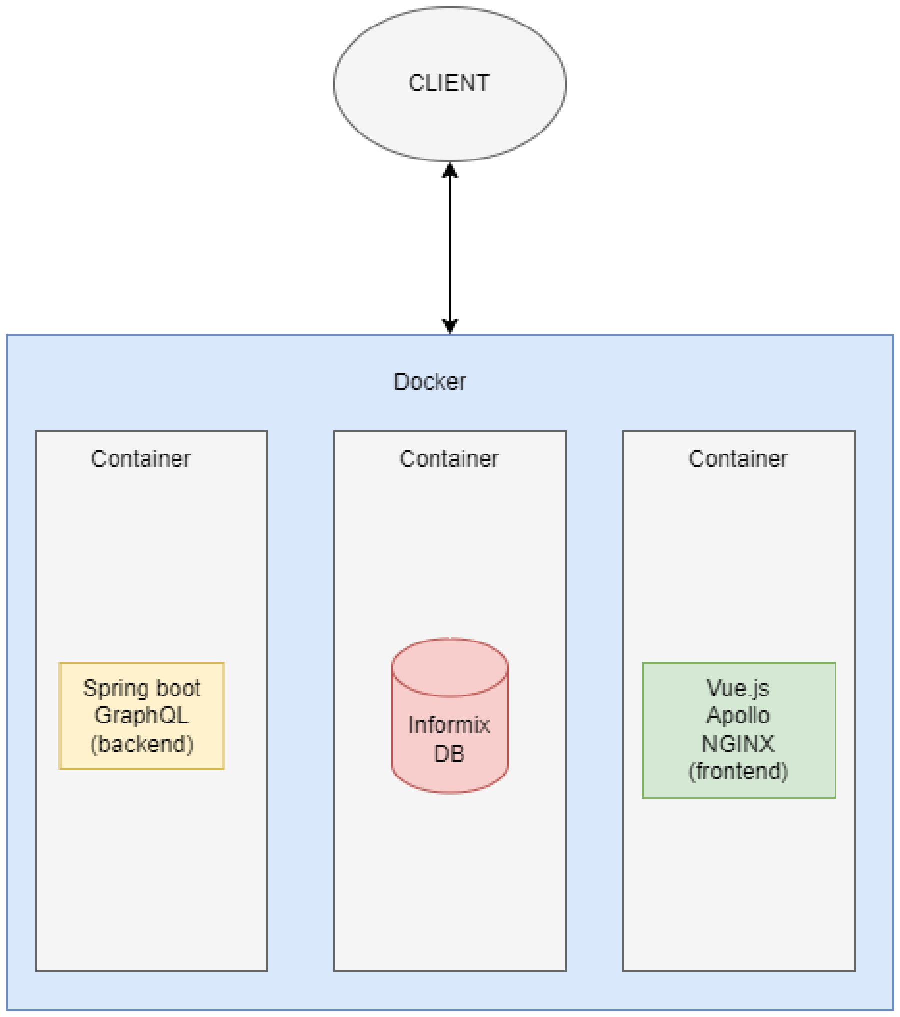


Figure 5: Simplified system architecture

Our application is deployed on an Ubuntu Server and follows a microservice architecture, utilizing containerization. The architecture consists of three components: the database server, frontend server, and backend server. Each of these components are encapsulated within its own containers, allowing them to be managed independently and ensuring scalability and flexibility for our application. By adopting the microservice approach, we have decoupled different functionalities into services, enabling easier development, deployment, and maintenance. Containerization enhances the efficiency of our application, allowing it to be run consistently across different environments. This modular and scalable architecture promotes agility, scalability, and fault tolerance, providing a robust foundation for our applications operation.

4.2: Process

This section covers the results of the working process throughout the project, meaning the working methodology, planning and structure of the project.

4.2.1: Schedule

In the start of the project, we created a schedule (as mentioned in section 3.1.2) with milestones and different activities, and due dates for those activities. This plan was posted at the front page on Confluence to be easily accessed when working with the project and remind us of the progression while working.

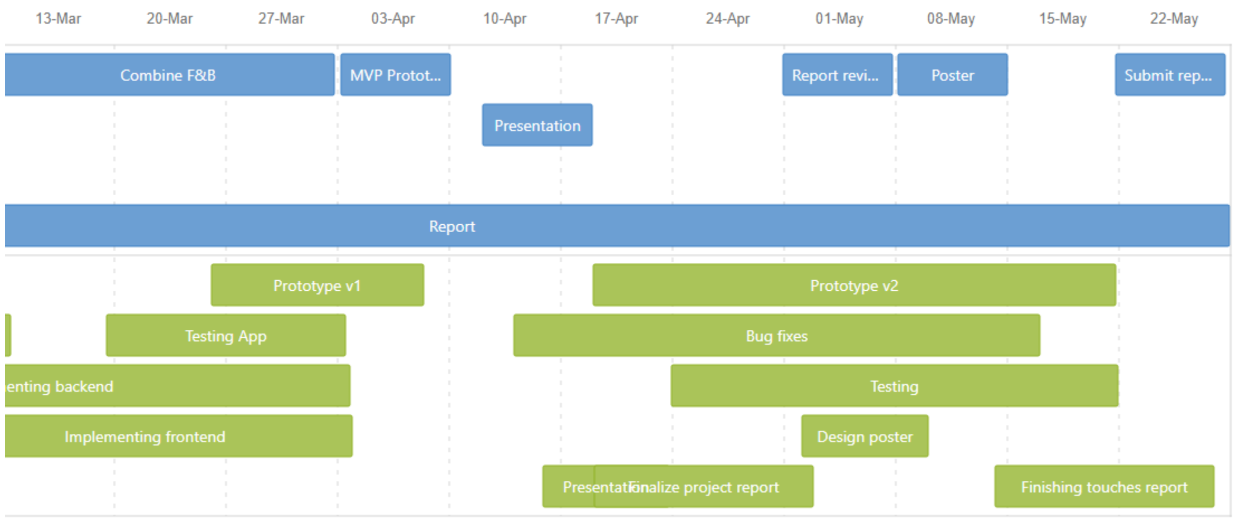


Figure 6: Project schedule w/milestones

As shown in figure 3, the blue represents milestones and important dates during the project, and the green represents things that we should work on at the given time.

As it's known, software projects are known for exceeding the limit on either time and/or money, so it is important to have a good and clear progression plan.

4.2.2: Scrum

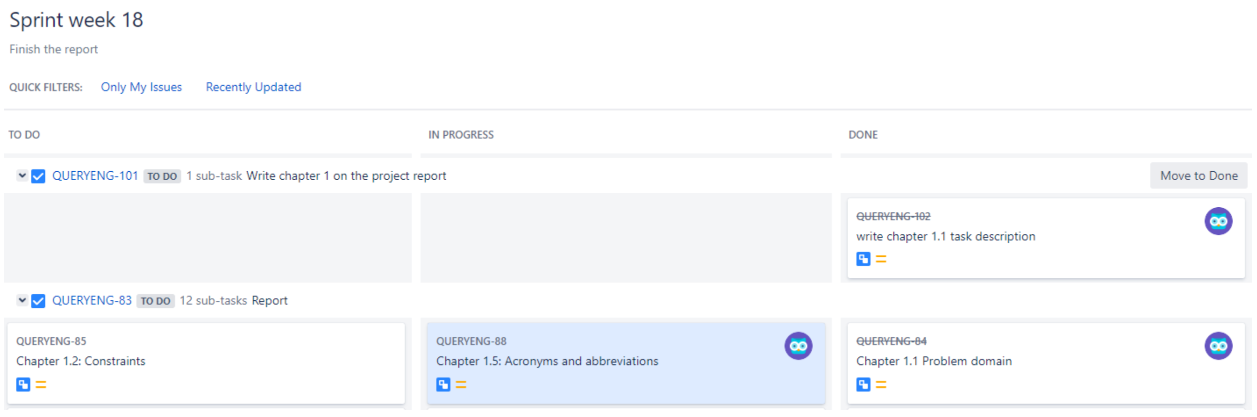
Covers the results from implementation of Scrum into the project.

4.2.1.1: Backlog

As mentioned in section 2.2.5, Sprint objectives are chosen from the backlog while prioritizing the most important tasks first. While working with the Scrum methodology this was the way our sprints was planned. The tasks put in the backlog were based on the schedule for the given week, in addition to issues that might have been discussed during the ongoing sprint.

4.2.1.2: Sprint

Scrum sprints (2.2.5) were used through the whole process except from when we were working on the preliminary project plan.


Sprint week 18

Finish the report

QUICK FILTERS: Only My Issues Recently Updated

TO DO	IN PROGRESS	DONE
QUERYENG-101 TO DO 1 sub-task Write chapter 1 on the project report		Move to Done
		~~QUERYENG-102~~ write chapter 1.1 task description
QUERYENG-83 TO DO 12 sub-tasks Report		
QUERYENG-85 Chapter 1.2: Constraints	QUERYENG-88 Chapter 1.5: Acronyms and abbreviations	~~QUERYENG-84~~ Chapter 1.1 Problem domain

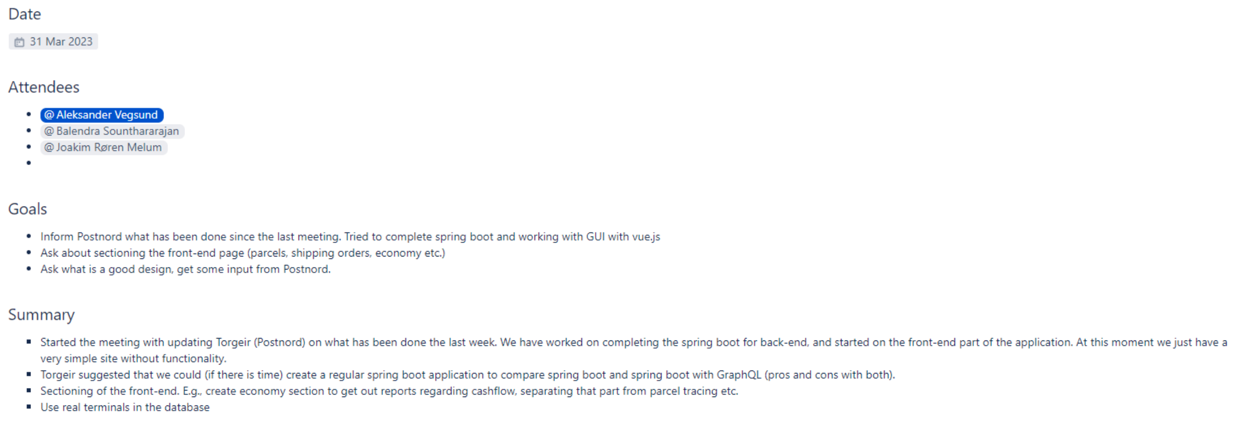
Figure 7: Snapshot of sprint board in Jira

The sprint board is divided into three columns; to do, in progress and done. Team members picked issues from the column named 'to do' and assigned them to themselves. The issue is then moved to the 'in progress' column, where it will remain until the task is done. When it is done, it is moved to the column named accordingly.

4.2.1.3: Reviews

Sprint reviews (section 2.2.7) were used weekly for updating the product owners, showing what had been done in the latest sprint and what difficulties were faced during the sprint. This was also a great way to get some input from the product owners in cases where we were a bit stuck.

For every sprint review we did, we wrote down some notes so we could have a reference to what was agreed upon in the meeting.


Date

31 Mar 2023

Attendees

- @Aleksander Vegsund
- @Balendra Sounthararajan
- @Joakim Røren Melum
-

Goals

- Inform Postnord what has been done since the last meeting. Tried to complete spring boot and working with GUI with vue.js
- Ask about sectioning the front-end page (parcels, shipping orders, economy etc.)
- Ask what is a good design, get some input from Postnord.

Summary

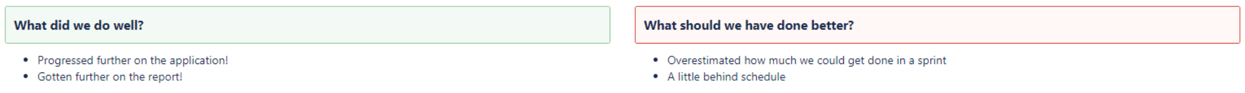
- Started the meeting with updating Torgeir (Postnord) on what has been done the last week. We have worked on completing the spring boot for back-end, and started on the front-end part of the application. At this moment we just have a very simple site without functionality.
- Torgeir suggested that we could (if there is time) create a regular spring boot application to compare spring boot and spring boot with GraphQL (pros and cons with both).
- Sectioning of the front-end. E.g., create economy section to get out reports regarding cashflow, separating that part from parcel tracing etc.
- Use real terminals in the database

Figure 8: Sprint review

As shown in **figure 8**, this is the sprint review notes from 31-March-2023. The notes are divided into three sections, attendees, goals, and summary for the given sprint. Attendees includes names of attending team members, goals was to be the output of the meeting, and summary is a short reflection on what was discussed in the meeting and how it went.

4.2.1.4: Retrospectives

After every sprint the team had a sprint retrospective (section 2.2.8) to evaluate the process and performance over the last sprint. By doing this from the start the team could see progress in efficiency, both in the meetings and in working with the project.


What did we do well?

- Progressed further on the application!
- Gotten further on the report!

What should we have done better?

- Overestimated how much we could get done in a sprint
- A little behind schedule

Figure 9: Retrospective

Above is a snapshot from part of a retrospective. The green header is what went well the past sprint, and the red is things that need to be improved.

4.3: Database

Modelling the database was the first step in the project. Post Nord explained how a simplified version of Post Nord database should be and what should be included in the database.

4.3.1: Requirements

In the simplified version of a parcel delivery system, all customers sending and receiving parcels must be registered. Customers should have the option of sending multiple parcels as one order. Parcels should have a barcode which can be used to track the status of the parcel and terminal the parcel is scanned at. The database model should consider that the customer can add additional services when the parcel is under transport. The cost of sending a parcel is calculated using its weight. A parcel can be transported through several terminals before being delivered to the destination.

Parcel will be scanned at terminals, where a fixed amount will be added as the revenue to the terminal. The same amount will be added as the cost of delivering a parcel in the system. The revenue of the parcel is calculated by subtracting the total amount of cost generated while scanned at the terminals from the initial cost paid by the customer when sending the parcel.

When a parcel is registered in the system, an invoice will be created with an amount to pay and a due date. Invoices can be paid by sender, receiver or third parties. Third parties refer to the persons / organizations other than sender or receiver who pays the invoice. When a customer adds an additional service to a parcel, an additional invoice will be created.

4.3.2: Designing phase.

4.3.2.1: Conceptual phase of the design

In the first phase - conceptual designing, we identified the important keywords in the requirements to represent them as entities of the database model. After multiple iterations we ended up with the following entities.

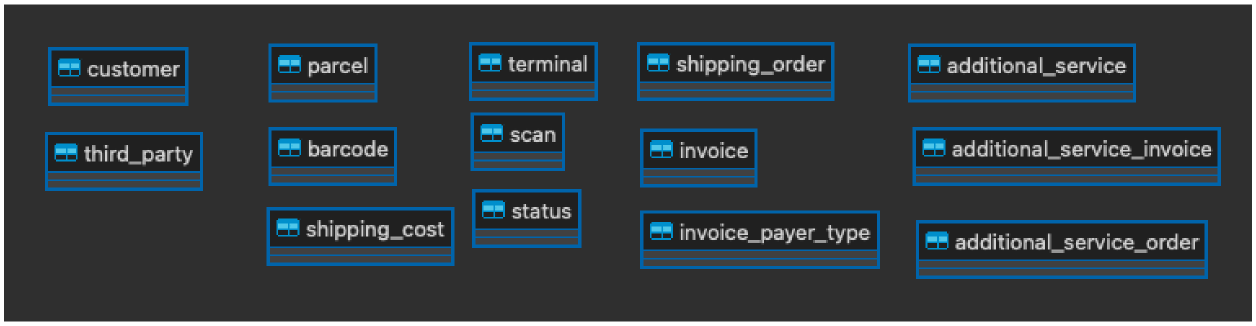


Figure 10: Entities in the database model

Customer – A customer is a registered person or an organization that sends and receives parcels using Post Nord.

Third party – A third party refers to an organization or person who pays an invoice on behalf of the customer.

Parcel – A parcel is an item that is being sent or received through the postal service.

Barcode –A barcode is a unique identifier assigned to a parcel. Used to track parcels.

Shipping cost – A shipping cost is the amount charged to send a parcel. It is based on the weight of a parcel.

Terminal – A terminal is a location where parcels are received and sorted. It is where parcels are processed and then forwarded to another terminal or destination for further transportation.

Scan – A scan refers to the process of scanning the barcode of a parcel at the terminal. This scan updates the system with the current location and status of the parcel.

Status – Represents the status of the parcel.

Shipping order – A shipping order contains information about the sender, receiver, and the parcel. It is generated when a customer sends a parcel.

Additional service – Additional service are optional services that can be added to a parcel shipment, after the parcel is sent.

Additional service order – An additional service order is triggered when a customer requests an additional service for their parcel. It contains information about the specific additional service requested and is associated with the original shipping order.

Invoice – An invoice is a document that provides payment information for a parcel shipment. It includes details such as the shipping cost, additional service charges, and the total amount due.

Invoice payer type – The invoice payer type refers to the person / organization paying the invoice.

Additional service invoice – an additional service invoice is a separate invoice specifically for the additional services added to parcel shipment. It provides detailed information about the additional service charges and payment instructions.

After identifying the entities, we discussed the possible attributes of the entities. Our primary focus was to establish relationships between these entities and their associated attributes. At this stage our aim was to define the entities, their relationship, and their fundamental characteristics, without going into the specifics of implementation discussed in chapter 2.4.1.

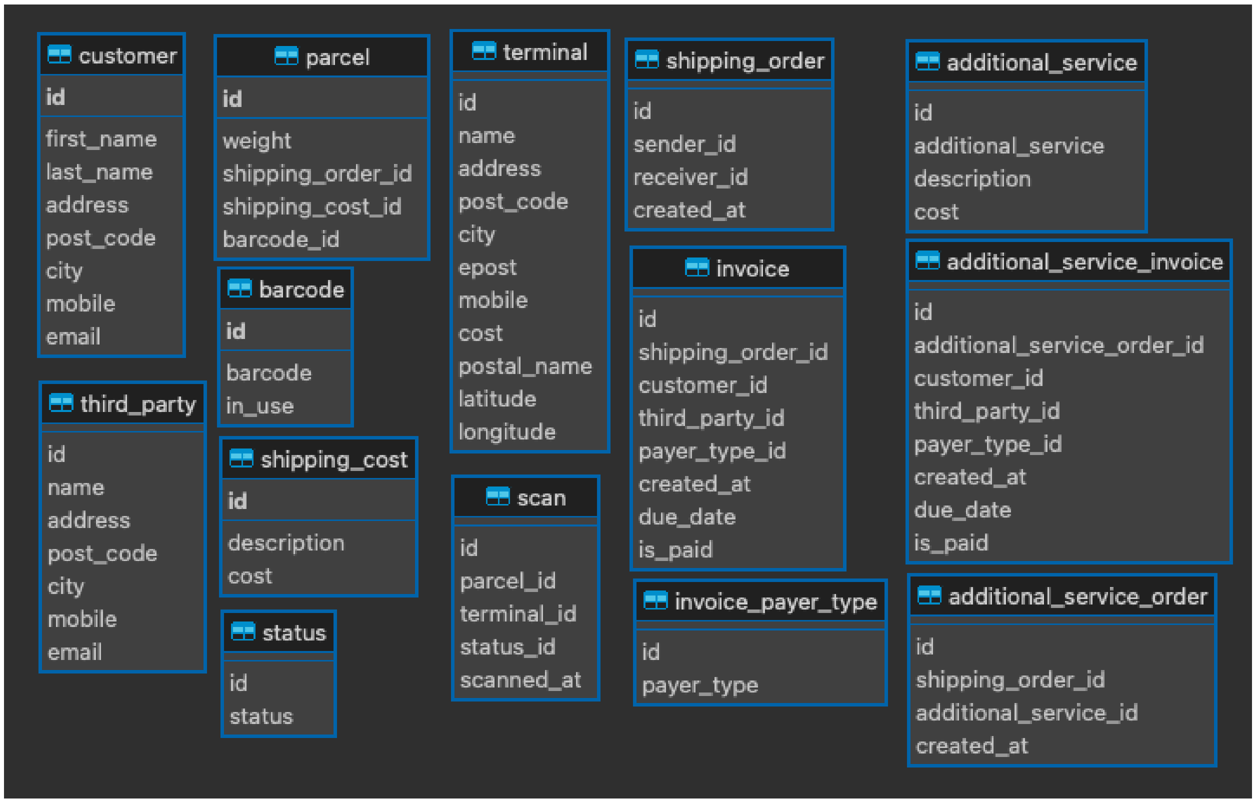


Figure 11: Entities with their attributes.

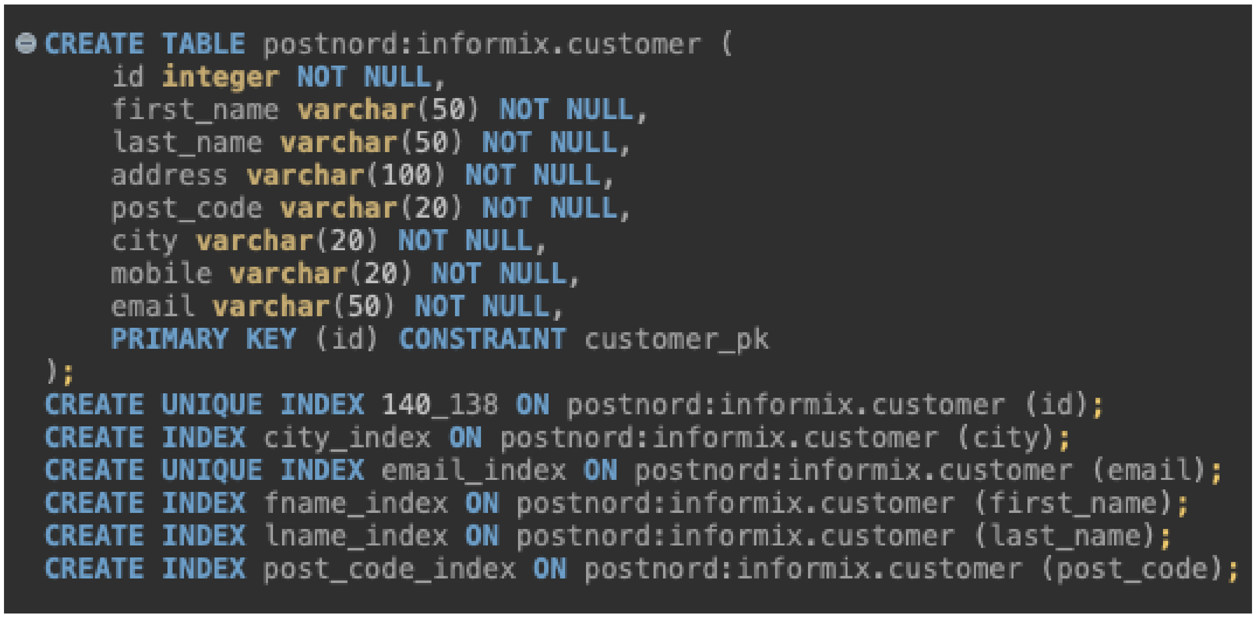
4.3.2.2 Logical phase of the design

In the second phase, we proceeded to determine the suitable data types for the attributes of the entities. Primary keys were established in each entity by identifying the unique attribute. In addition, we used normalization to minimize redundancy and dependencies, as discussed in chapter 2.4.2.

Initially, the parcel entity included both customer details and status. After many iterations of normalization, we moved customer details attributes and status attributes out of the parcel entity as it caused redundancy by repeating customer names and status in many rows. We then joined these tables using foreign keys based on their relationships. This is one example of how we made use of normalization.

4.3.2.3 Physical phase of the design

In the third phase, the logical model created in the second phase is adopted to IBM Informix Server. We implemented all the entities into tables in the database. Created relationships between tables. Identified which datatype in Informix is same as the datatypes we arrived at the second phase.



```sql
CREATE TABLE postnord:informix.customer (
    id integer NOT NULL,
    first_name varchar(50) NOT NULL,
    last_name varchar(50) NOT NULL,
    address varchar(100) NOT NULL,
    post_code varchar(20) NOT NULL,
    city varchar(20) NOT NULL,
    mobile varchar(20) NOT NULL,
    email varchar(50) NOT NULL,
    PRIMARY KEY (id) CONSTRAINT customer_pk
);
CREATE UNIQUE INDEX 140_138 ON postnord:informix.customer (id);
CREATE INDEX city_index ON postnord:informix.customer (city);
CREATE UNIQUE INDEX email_index ON postnord:informix.customer (email);
CREATE INDEX fname_index ON postnord:informix.customer (first_name);
CREATE INDEX lname_index ON postnord:informix.customer (last_name);
CREATE INDEX post_code_index ON postnord:informix.customer (post_code);
```

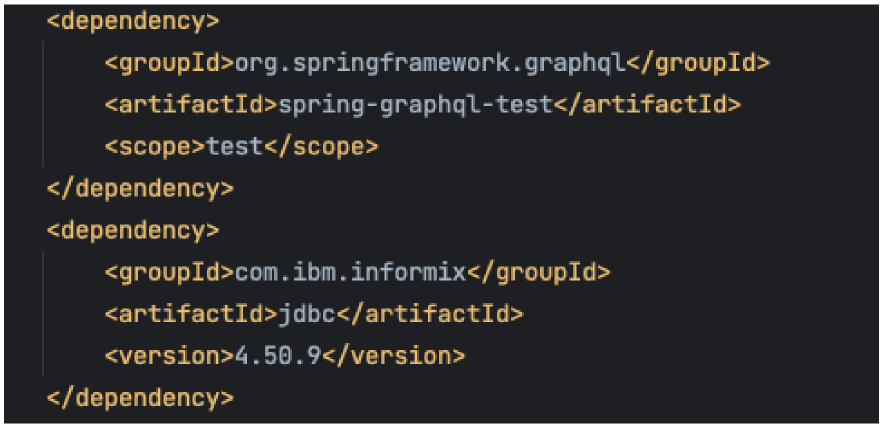
Figure 12: Script to create customer entity.

4.4: Back-end development

This section covers the results achieved from the backend development. The backend part of this project consists of Java with Spring boot, and GraphQL queries to fetch data from the database.

4.4.1: Maven

For the backend of this project, we used Maven as our build tool, as discussed in Chapter 3.4.1.1. We utilized pom file to obtain all the necessary dependencies for the project, including GraphQL, Faker, Informix and Junit. These dependencies were added to the pom file to ensure that our project had all the resources it needed to function properly.

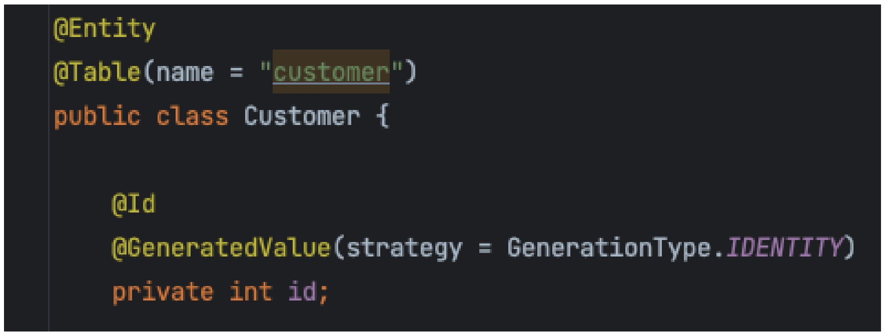


```xml
<dependency>
    <groupId>org.springframework.graphql</groupId>
    <artifactId>spring-graphql-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>com.ibm.informix</groupId>
    <artifactId>jdbc</artifactId>
    <version>4.50.9</version>
</dependency>
```

Figure 13: Code snippet with graphql and informix dependencies

4.4.2: Entity classes

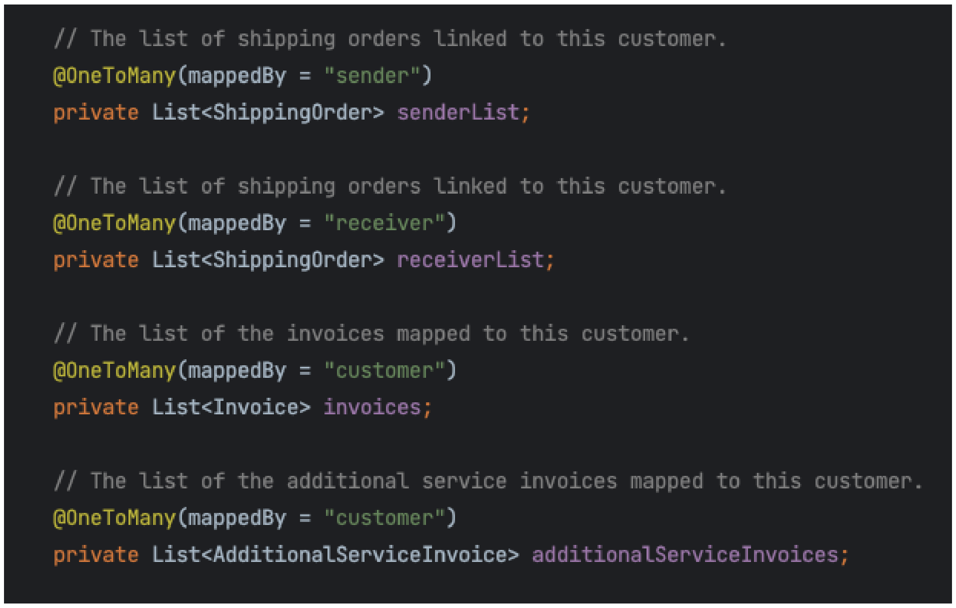
Our initial step involved converting all tables in the database into their corresponding entity classes using the Java Persistence API (JPA). To accomplish this, we annotated each class with @Entity to map it to the appropriate table. As specified by JPA, we also annotated the primary key field of each class with @Id.



```
@Entity
@Table(name = "customer")
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
```

Figure 14: Defining entity class and primary key using annotations.

The relationship between tables is defined in entity classes using @OneToMany, @ManyToOne, @OneToOne annotation depending on the relationship between the tables.

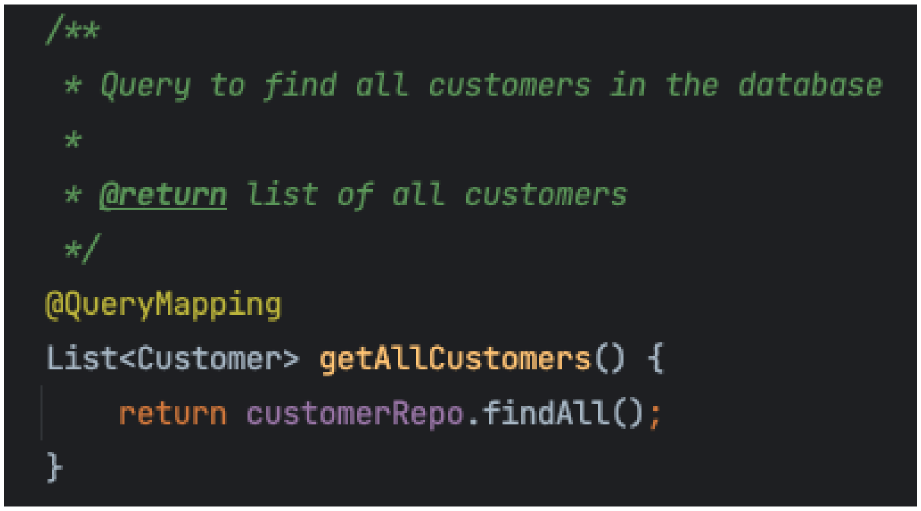


```
// The list of shipping orders linked to this customer.
@OneToMany(mappedBy = "sender")
private List<ShippingOrder> senderList;

// The list of shipping orders linked to this customer.
@OneToMany(mappedBy = "receiver")
private List<ShippingOrder> receiverList;

// The list of the invoices mapped to this customer.
@OneToMany(mappedBy = "customer")
private List<Invoice> invoices;

// The list of the additional service invoices mapped to this customer.
@OneToMany(mappedBy = "customer")
private List<AdditionalServiceInvoice> additionalServiceInvoices;
```

Figure 15: Relationship between tables defined using JPA annotations.

4.4.3: Controller classes

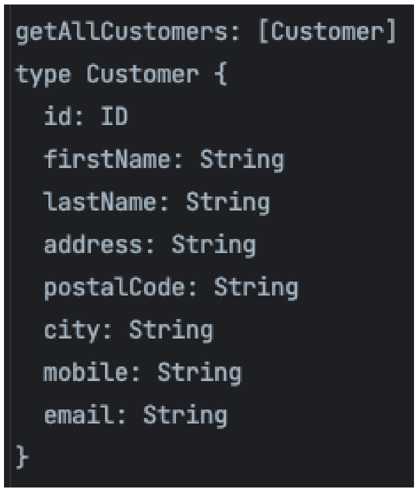
We associated every entity class with corresponding controller classes. These classes contain methods for retrieving data from the database using the repository classes described in 4.3.4. These methods the annotated with @QueryMapping to associate them with GraphQL queries, allowing data to be fetched from the frontend. This annotation allows to specify which controller method should handle a particular GraphQL query.



```
/**
 * Query to find all customers in the database
 *
 * @return list of all customers
 */
@QueryMapping
List<Customer> getAllCustomers() {
    return customerRepo.findAll();
}
```

Figure 16: Code snippet displaying method – graphql mapping.

The code snippet below shows how the controller method is referenced in *graphqls.schema* file. This serves as an endpoint that can be accessed from the front end to retrieve data.

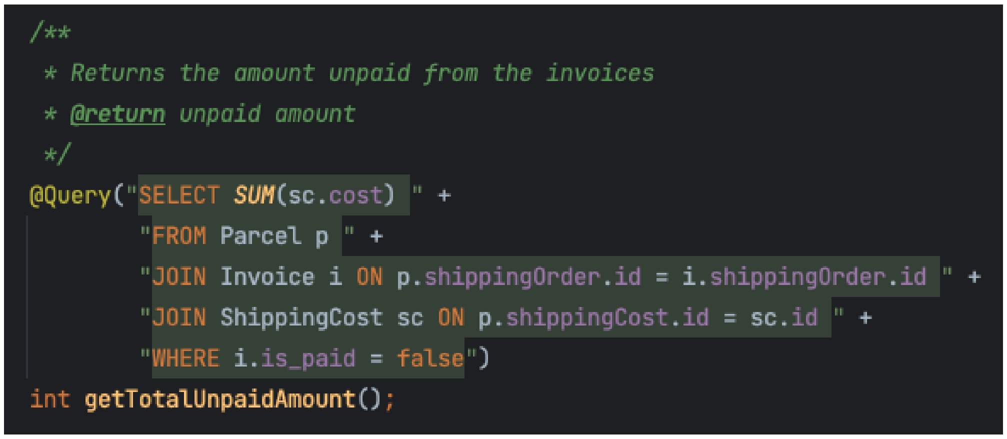


```
getAllCustomers: [Customer]
type Customer {
  id: ID
  firstName: String
  lastName: String
  address: String
  postalCode: String
  city: String
  mobile: String
  email: String
}
```

Figure 17: qrahqls schema

4.4.4: Repository classes

Repository classes are used to manage data fetching. They provide an abstraction between data fetching and rest of the application. As we are working only with reading data from the database, we have only methods to read data from the database. Repository classes provide predefined methods, which can be used to fetch data. In addition, we have used complex SQL queries in native form for more complex data access.



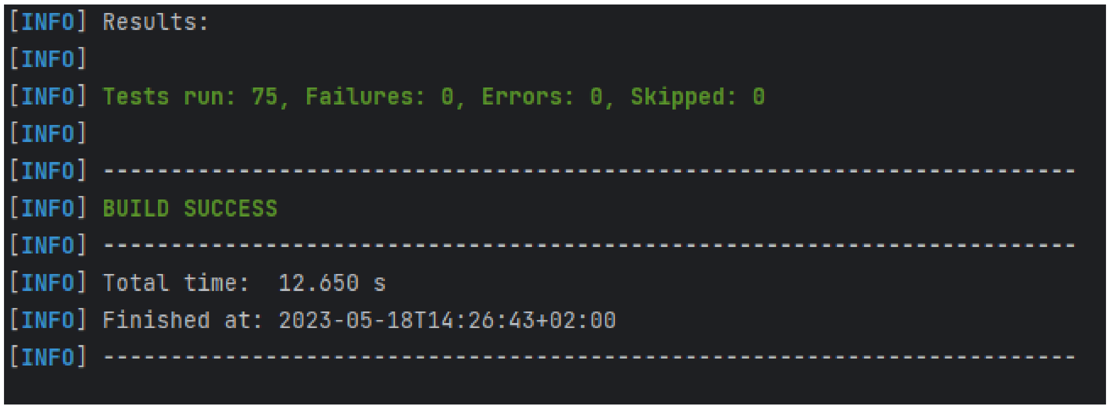
```
/**
 * Returns the amount unpaid from the invoices
 * @return unpaid amount
 */
@Query("SELECT SUM(sc.cost) " +
        "FROM Parcel p " +
        "JOIN Invoice i ON p.shippingOrder.id = i.shippingOrder.id " +
        "JOIN ShippingCost sc ON p.shippingCost.id = sc.id " +
        "WHERE i.is_paid = false")
int getTotalUnpaidAmount();
```

Figure 18: Code snippet with method to fetch data from database.

The code snippet above gets the sum of all invoices that are unpaid in the database. This is done by joining three tables Parcel, Invoice and Shipping Cost.

4.4.5: Testing

Software testing (section 2.9) is an important step in software development for making sure that every piece of code that is written behaves exactly how we want it to behave. As mentioned, there are several ways of testing an application, but unit testing (2.9.1) is one of the most common ways to test an application.



```
[INFO] Results:
[INFO]
[INFO] Tests run: 75, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.650 s
[INFO] Finished at: 2023-05-18T14:26:43+02:00
[INFO] ------------------------------------------------------------------------
```

Figure 19: Unit test results

As you can see from **figure 19**, 75/75 tests for the application run successfully with no failures. Here is an example of a unit test for a given function:

```
@BeforeEach
Void setUp(){
    Barcode = new Barcode (13, “1012398712”, true);
    Parcel = new Parcel(1, 30);
}
```

Above is a code snippet that runs before each test in this test class and is responsible for creating a new object of a barcode and a parcel. Both objects are created with the given attribute values that is written inside the parenthesis, which works as the conditions to test the objects upon.

```
@Test
@DisplayName (“test getId() method”)
Void testGetId() {

    assertEquals (13, barcode. getId());

}
```

Above is an actual unit test that tests that the `getId()` method works on the barcode object, and that the method returns the expected value. In the `assertEquals` method we can see that the first number in the parenthesis is the expected value of the barcode object’s id, which is the same id as the barcode is created with in the setup method. The second parameter is a method that returns the actual value. For the test to pass in this case, the number expected must match the actual value of the number returned. The test passed as can be seen in **figure 19**.

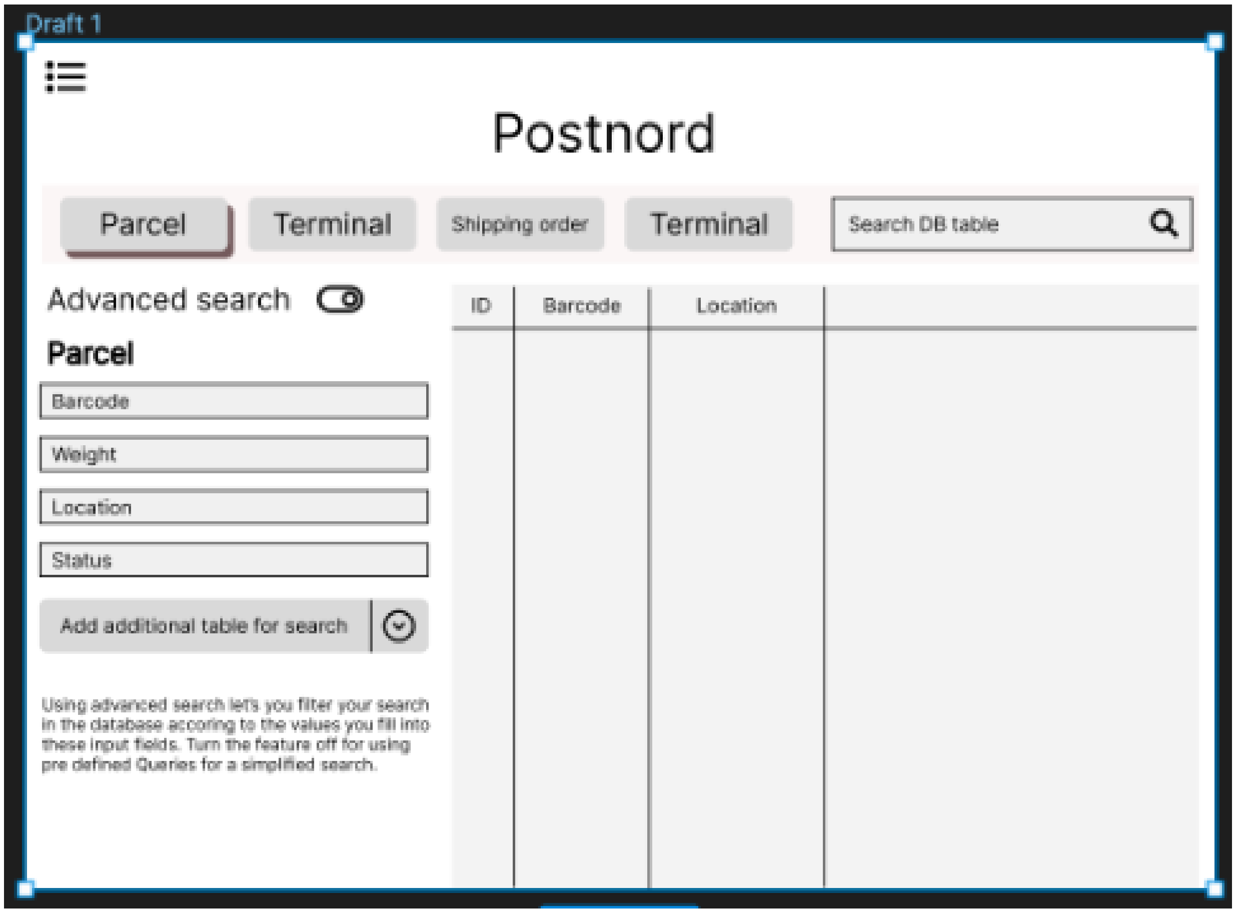
4.5: Front-end development

As discussed earlier in the database section 4.3, Post Nord have a large production database with nearly 1000 tables. Post Nord faced a bottleneck when retrieving data from their large production database. They wished that the data needed should be fetched directly by the individuals themselves instead of forwarding to the database administrators or developers. To address this bottleneck issue, we developed a web application called Query engine viewer, which will make it easy for the employees working in Post Nord to retrieve the desired information directly without forwarding request to others in the organization. They can in addition fetch the information they need in the format they need.

4.5.1 Sketching

4.5.1.1 Sketch of Initial idea

We started sketching a web application using Figma tool. The initial idea was to allow the users to search through all the tables. Then they can select the necessary fields wanted with filters they wanted.



Using advanced search let's you filter your search in the database accoring to the values you fill into these input fields. Turn the feature off for using pre defined Queries for a simplified search.

Figure 20: First sketch of Query Engine Viewer using Figma.

After completing the first draft of our design we developed several use cases to test its functionality. One such use case involved a financial manager trying to view the total amount of unpaid invoices withing a specific period. To execute this, the manager would need to know the name of the table where the information on invoices is stored. This would be feasible if the size of the database was small. But Post Nord has a large database, where it is nearly impossible to know the structure of the database to access information like unpaid invoices. Additionally, the information about invoices will be spread across multiple tables, which should be joined to get the desired result. This is one of the use cases where this approach of allowing the users to access data directly from tables failed.

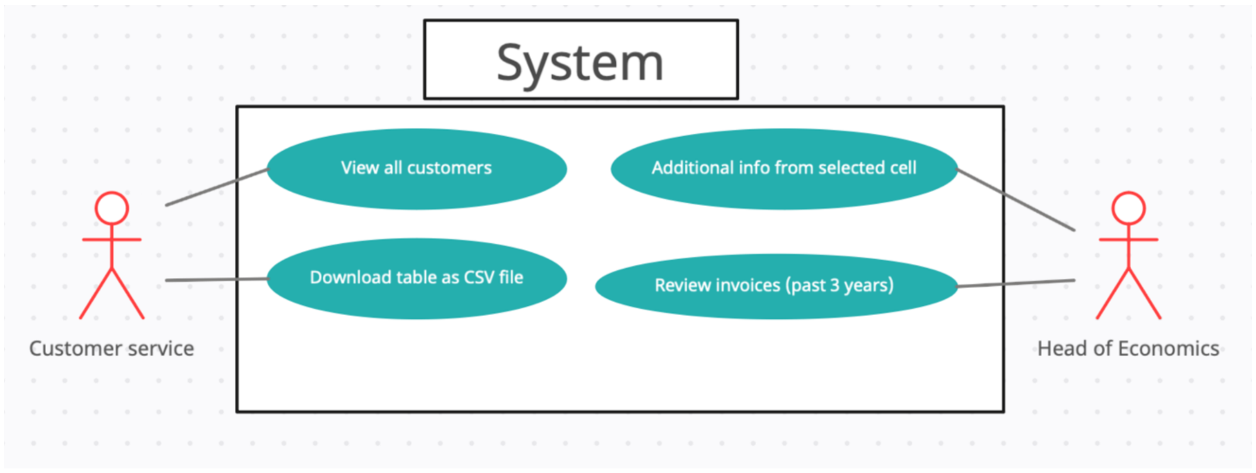

System
View all customers
Additional info from selected cell
Download table as CSV file
Review invoices (past 3 years)
Customer service
Head of Economics

Figure 21: Use case from customer service and head of economics.

4.5.1.2 Sketch of final idea

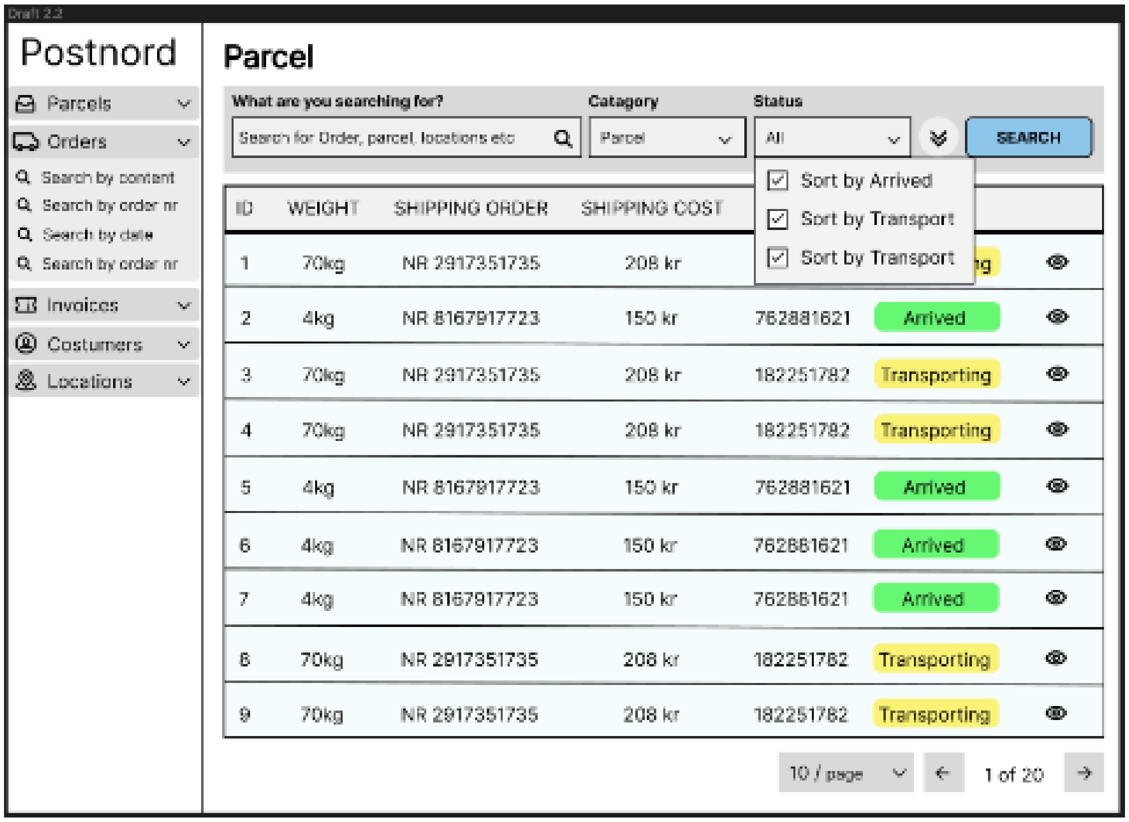
After many iterations, we ended up dividing the database data into main categories. We ended up with dividing the database data in three main categories such as

- Customers
- Invoices
- Parcels

We decided to show the data in form of

- Key numbers
- Tables
- Charts

Highlighted key numbers will give the users information about the best and worst cases within a category. Tables will be displaying more detailed information about every use case, while charts offer a visual representation of the data, allowing for abstract understanding without the need to examine the numbers in detail.

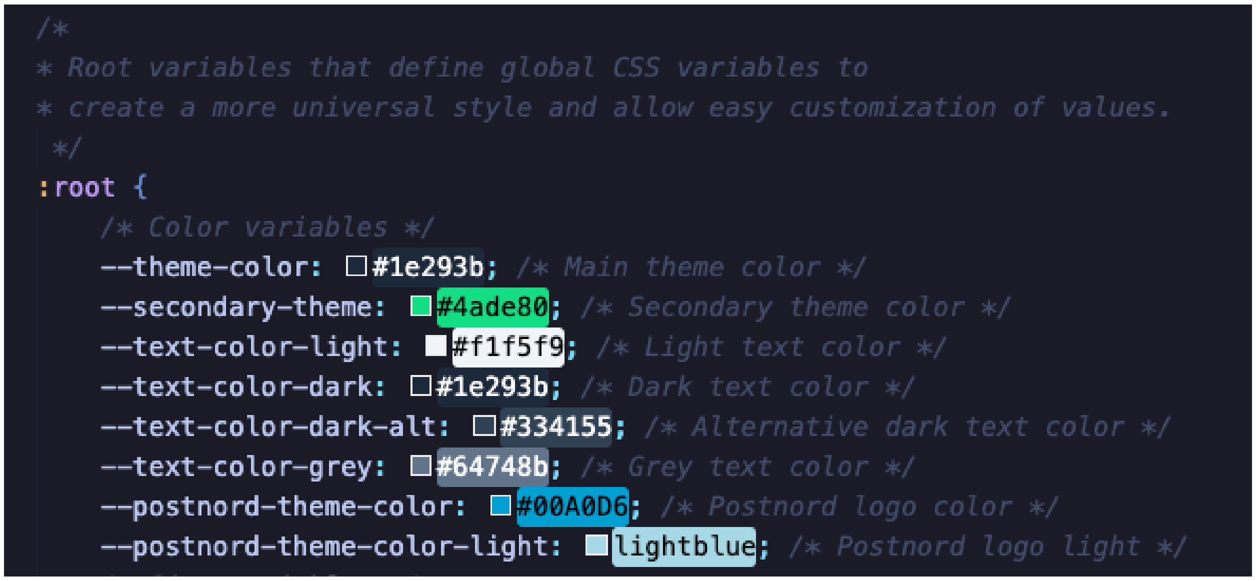


ID	WEIGHT	SHIPPING ORDER	SHIPPING COST			
1	70kg	NR 2917351735	208 kr		ng	👁
2	4kg	NR 8167917723	150 kr	762881621	Arrived	👁
3	70kg	NR 2917351735	208 kr	182251782	Transporting	👁
4	70kg	NR 2917351735	208 kr	182251782	Transporting	👁
5	4kg	NR 8167917723	150 kr	762881621	Arrived	👁
6	4kg	NR 8167917723	150 kr	762881621	Arrived	👁
7	4kg	NR 8167917723	150 kr	762881621	Arrived	👁
8	70kg	NR 2917351735	208 kr	182251782	Transporting	👁
9	70kg	NR 2917351735	208 kr	182251782	Transporting	👁

Figure 22: Query engine viewer sketch using Figma.

4.5.3: Adopting universal design and principals.

When we had our final design of the web application, we used design principles discussed in x and universal design discussed in x to make decisions about font style, color, and layout of the web application.



```css
/*
* Root variables that define global CSS variables to
* create a more universal style and allow easy customization of values.
 */
:root {
    /* Color variables */
    --theme-color: #1e293b; /* Main theme color */
    --secondary-theme: #4ade80; /* Secondary theme color */
    --text-color-light: #f1f5f9; /* Light text color */
    --text-color-dark: #1e293b; /* Dark text color */
    --text-color-dark-alt: #334155; /* Alternative dark text color */
    --text-color-grey: #64748b; /* Grey text color */
    --postnord-theme-color: #00A0D6; /* Postnord logo color */
    --postnord-theme-color-light: lightblue; /* Postnord logo light */
```

Figure 23: *Code snippet showing global color variables.*

We have taken several measures to ensure that our web page is user-friendly and accessible to everyone. Our font colors and background colors have been carefully selected to provide enough contrast for users with reduced vision. The layout is intuitive and familiar which makes the navigation easy to the users. The web page is also interactive, providing users with information after each action where it is possible. Additionally, we have included a wiki start page with a brief introduction on how to use the page.

4.5.4: Query engine viewer

After completing sketching, defining global variables and layout, we started coding the web page. For developing we made use of Vue framework, which is discussed in more detail in chapter 3.4.4. We have made use of two strong features of Vue framework, single page application and component-based development.

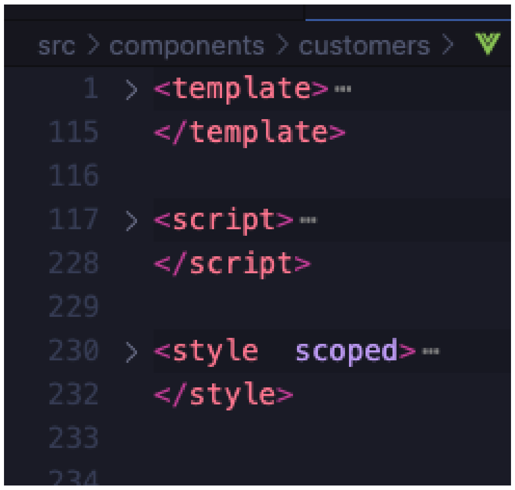
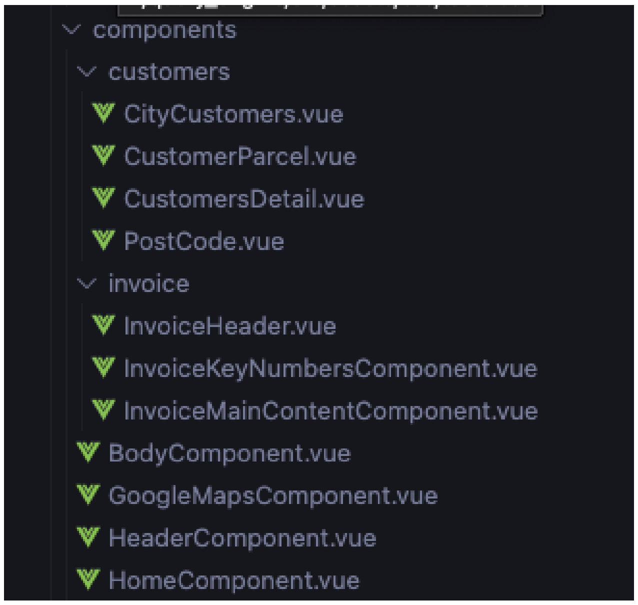

```
src > components > customers >
<template>…
</template>

<script>…
</script>

<style  scoped>…
</style>
```

Figure 24: Basic structure of a component.

We aimed to make every feature as components where possible, allowing us to reuse components multiple times. This made it easy to add and remove components from the page without making significant changes. As a result, each component became independent of each other. This approach made the code. This led to low coupling and high cohesion as discussed in chapter 2.7.1.1 and chapter 2.7.1.2.

For instance, we have a component named *CityCustomers.vue*, which is used to display a table and a chart of the top 10 cities based on the number of customers. This component contains only code necessary to fetch and display table data. The same data is also used to generate the chart. The component is focused solely on displaying information about customers and the cities.



```
components
  customers
    CityCustomers.vue
    CustomerParcel.vue
    CustomersDetail.vue
    PostCode.vue
  invoice
    InvoiceHeader.vue
    InvoiceKeyNumbersComponent.vue
    InvoiceMainContentComponent.vue
  BodyComponent.vue
  GoogleMapsComponent.vue
  HeaderComponent.vue
  HomeComponent.vue
```

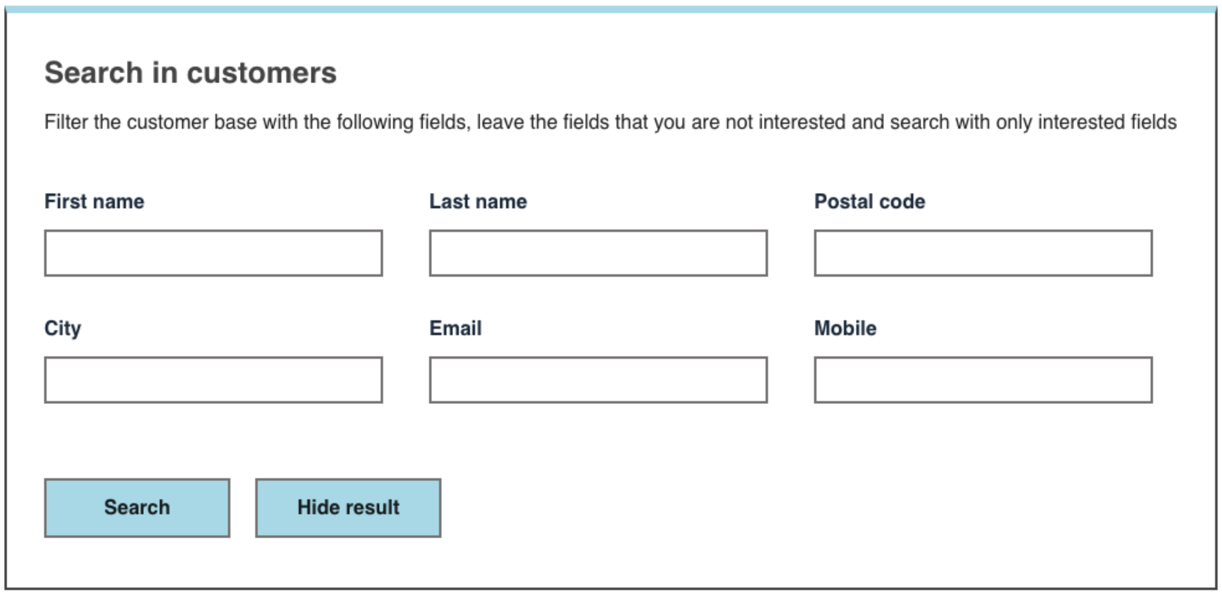
Figure 25: Screen shot from IDE showing different components.

4.5.5: Layout

We have developed our web page as a single page application. One of the key features of a single page application is their performance, which is an important criterion for our page. Our goal was to provide our users with information they needed without any significant delay. We implemented a sidebar which can be minimized and maximized to show different categories. When the user navigates to a category, the relevant information is displayed in the body area. This approach allows for fast and efficient navigation.

4.5.6: Page content

For instance, customer page provides necessary information about the registered customers. Users can search through the entire database to find specific customers using the filters as shown in figure 26. Page also display tables and charts. Tables displayed can be downloaded by the user for to work further with the key numbers. Additionally, we have highlighted key numbers such as postcode with highest and lowest number of customers. This information is intended to give the users visiting the customer page an idea about the trend in the organization´s customer base. Similarly, we have an invoice and parcel page with comparable functionalities. These pages are designed to demonstrate possible ways of displaying the result to the user.

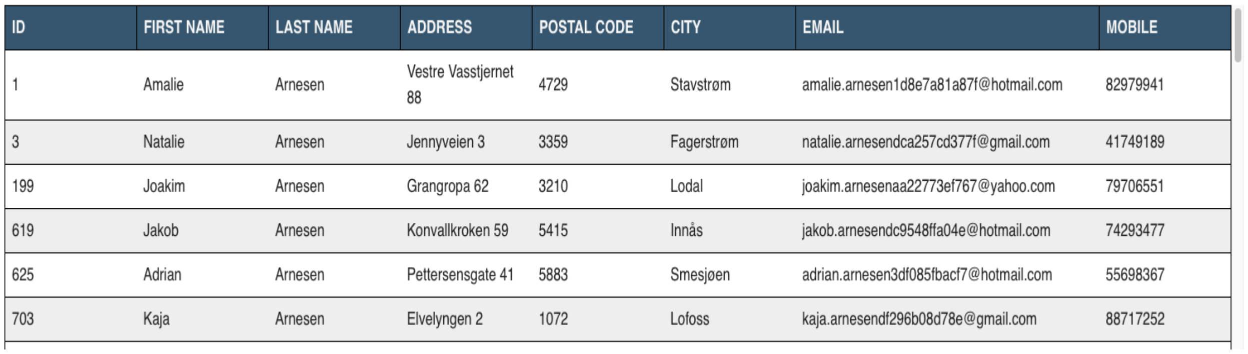


Search in customers

Filter the customer base with the following fields, leave the fields that you are not interested and search with only interested fields

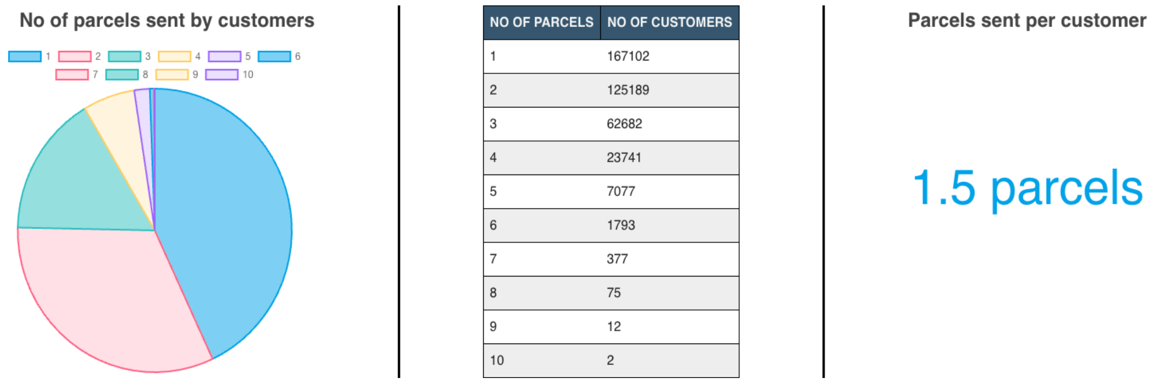
First name	Last name	Postal code
City	Email	Mobile

Figure 26: Part of the customer page displaying search options.



ID	FIRST NAME	LAST NAME	ADDRESS	POSTAL CODE	CITY	EMAIL	MOBILE
1	Amalie	Arnesen	Vestre Vasstjernet 88	4729	Stavstrøm	amalie.arnesen1d8e7a81a87f@hotmail.com	82979941
3	Natalie	Arnesen	Jennyveien 3	3359	Fagerstrøm	natalie.arnesendca257cd377f@gmail.com	41749189
199	Joakim	Arnesen	Grangropa 62	3210	Lodal	joakim.arnesenaa22773ef767@yahoo.com	79706551
619	Jakob	Arnesen	Konvallkroken 59	5415	Innås	jakob.arnesendc9548ffa04e@hotmail.com	74293477
625	Adrian	Arnesen	Pettersensgate 41	5883	Smesjøen	adrian.arnesen3df085fbacf7@hotmail.com	55698367
703	Kaja	Arnesen	Elvelyngen 2	1072	Lofoss	kaja.arnesendf296b08d78e@gmail.com	88717252

Figure 27: Table with results after filtering after last name "Arnesen".



NO OF PARCELS	NO OF CUSTOMERS
1	167102
2	125189
3	62682
4	23741
5	7077
6	1793
7	377
8	75
9	12
10	2

Parcels sent per customer

1.5 parcels

Figure 28: Part of the customer page displaying parcel details per customer.

4.5.7: Continuous Integration and Continuous Deployment (CI/CD)

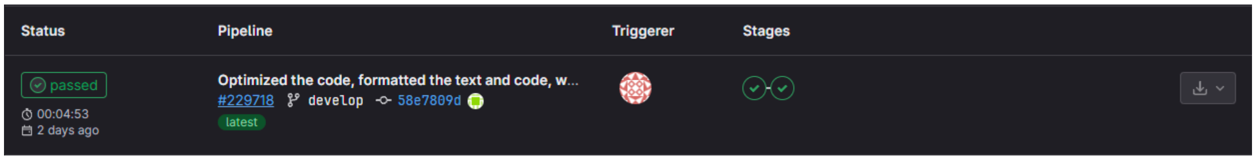
Continuous Integration and continuous Deployment (CI/CD) have played a vital role in increasing the efficiency of our software development process and delivering numerous benefits. Implementing CI/CD has resulted in significant time saving across various areas, such as automated build and deployment, accelerating testing cycles, and more.

One of the key advantages by using CI/CD in our project is the automation of essential tasks like building, testing, and pushing changes to the server. This eliminates the need for developers in our team to manually run tests or build the application each time they want to deploy changes. Instead, CI/CD takes care of sopping the current container, starting a new one, creating Docker image, and pushing it to Dockerhub for easier access. By removing the need of manual operations, CI/CD frees up considerable time that would otherwise be spent on repetitive, and error-prone tasks.

Faster testing cycles are another significant benefit from using CI/CD in our project. With automated processes in place, the developers in our time can focus on crucial tasks, leading to more efficient code changes. Feedback are delivered almost instant as code changes are automatically validated, reducing the chances of human errors. As a result, updates can be released to production faster and more reliably. This efficiency allows the team to allocate more time to implementing new features, optimizing existing functionality, and experimenting with new ideas, leading to continuous improvement.

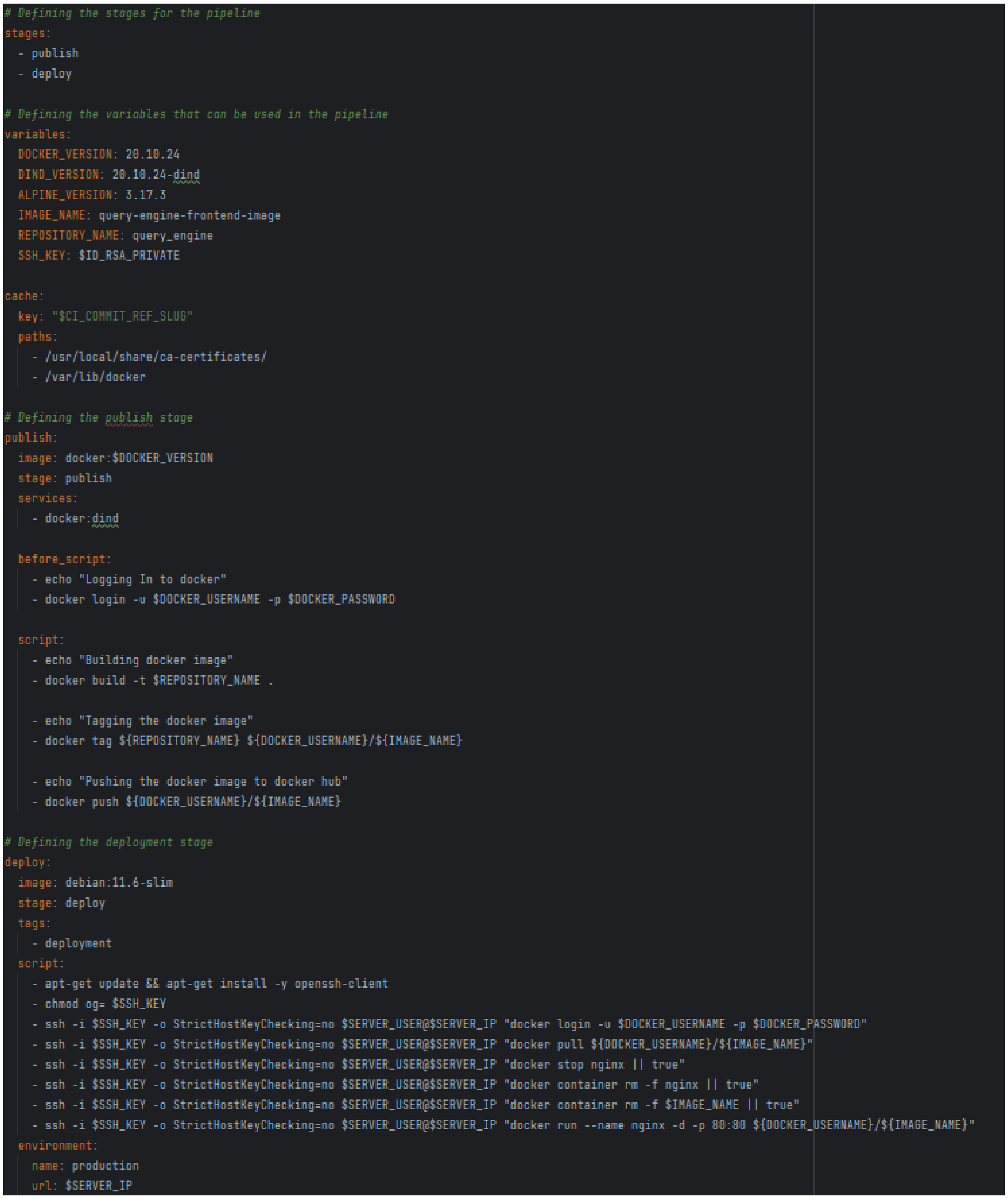
CI/CD also ensures early bug detection, increasing efficient collaboration within our team. By automating processes and workflows, it saves valuable time while maintaining a high software quality. Both our frontend and backend development have been integrated with CI/CD, providing an easy-to-understand overview of pipelines and their current status. If any issues appear during a pipeline, such as stages failing, the pipeline status will be marked as failed, alerting us about a potential problem.

4.5.7.1: CI/CD Frontend configuration and pipelines



Status	Pipeline	Triggerer	Stages
passed 00:04:53 2 days ago	Optimized the code, formatted the text and code, w... #229718 develop 58e7809d latest		✓ ✓

Figure 19: The frontend pipeline that shows the passing stages.



```yaml
# Defining the stages for the pipeline
stages:
  - publish
  - deploy

# Defining the variables that can be used in the pipeline
variables:
  DOCKER_VERSION: 20.10.24
  DIND_VERSION: 20.10.24-dind
  ALPINE_VERSION: 3.17.3
  IMAGE_NAME: query-engine-frontend-image
  REPOSITORY_NAME: query_engine
  SSH_KEY: $ID_RSA_PRIVATE

cache:
  key: "$CI_COMMIT_REF_SLUG"
  paths:
    - /usr/local/share/ca-certificates/
    - /var/lib/docker

# Defining the publish stage
publish:
  image: docker:$DOCKER_VERSION
  stage: publish
  services:
    - docker:dind

  before_script:
    - echo "Logging In to docker"
    - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD

  script:
    - echo "Building docker image"
    - docker build -t $REPOSITORY_NAME .

    - echo "Tagging the docker image"
    - docker tag ${REPOSITORY_NAME} ${DOCKER_USERNAME}/${IMAGE_NAME}

    - echo "Pushing the docker image to docker hub"
    - docker push ${DOCKER_USERNAME}/${IMAGE_NAME}

# Defining the deployment stage
deploy:
  image: debian:11.6-slim
  stage: deploy
  tags:
    - deployment
  script:
    - apt-get update && apt-get install -y openssh-client
    - chmod og= $SSH_KEY
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker pull ${DOCKER_USERNAME}/${IMAGE_NAME}"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker stop nginx || true"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker container rm -f nginx || true"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker container rm -f $IMAGE_NAME || true"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker run --name nginx -d -p 80:80 ${DOCKER_USERNAME}/${IMAGE_NAME}"
  environment:
    name: production
    url: $SERVER_IP
```

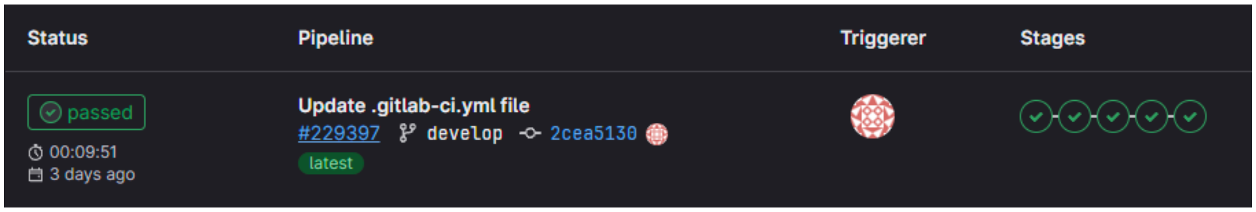
Figure 20: Frontend GitLab CI/CD configuration

In the diagram shown above (Figure 3), we have defined two stages in our pipelines: the publish stage and the deployment stage.

In the publish stage, we have specified that before running the script, it needs to authenticate with Docker. After that, the Docker image is built, tagged, and pushed to Docker Hub.

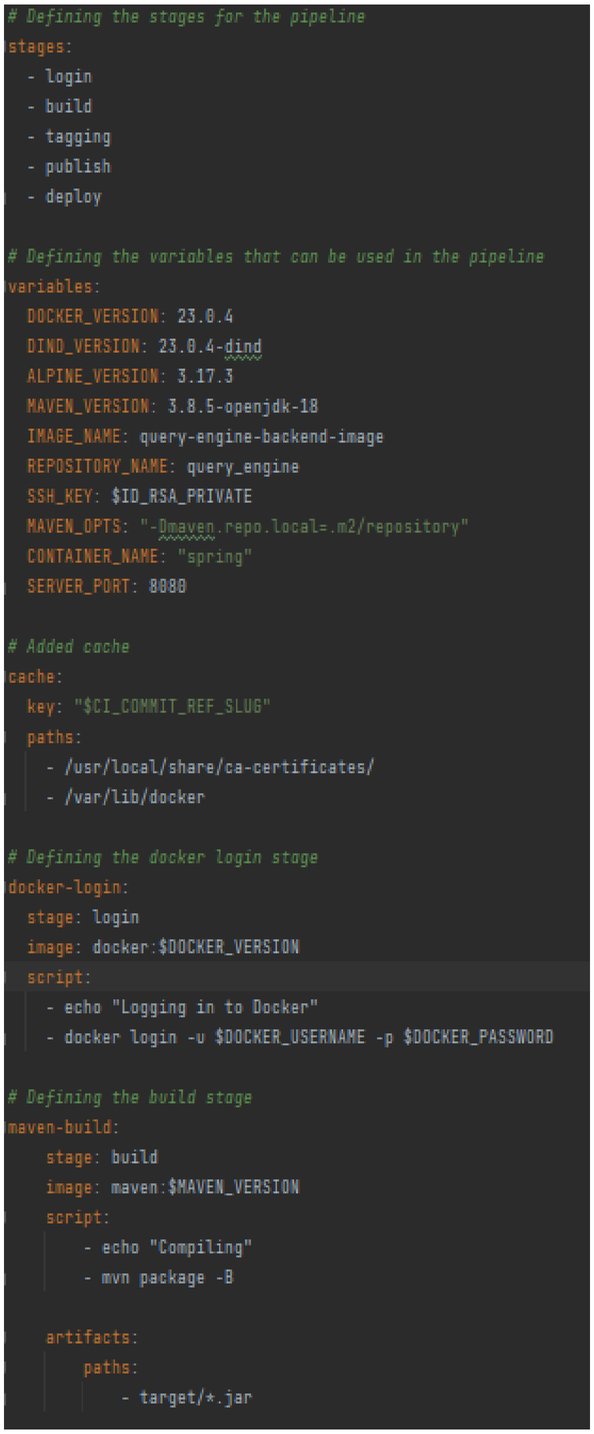
Moving on to the deployment stage, we first make sure that everything is up to date. Then, we install the OpenSSH client and authenticate with Docker. Next, we pull the Docker image, stop the running container, remove it, and run the new container using the specified port and image.

4.5.7.2: CI/CD Backend configuration and pipelines



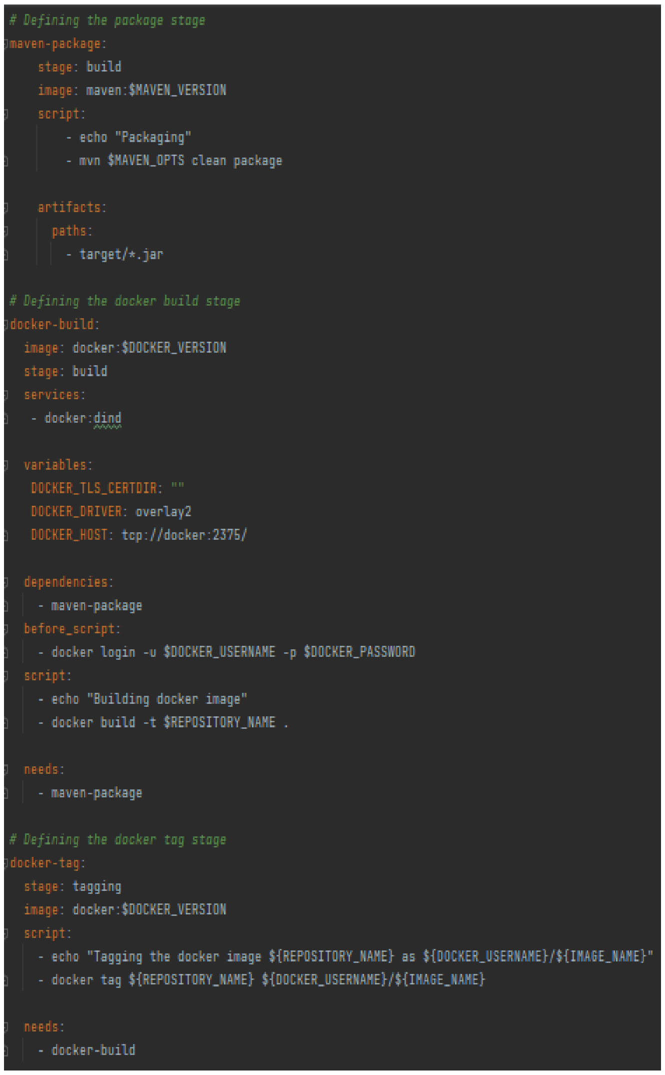
Status	Pipeline	Triggerer	Stages
passed 00:09:51 3 days ago	Update .gitlab-ci.yml file #229397 develop 2cea5130 latest		✓ ✓ ✓ ✓ ✓

Figure 31: The backend pipeline that shows the passing stages.



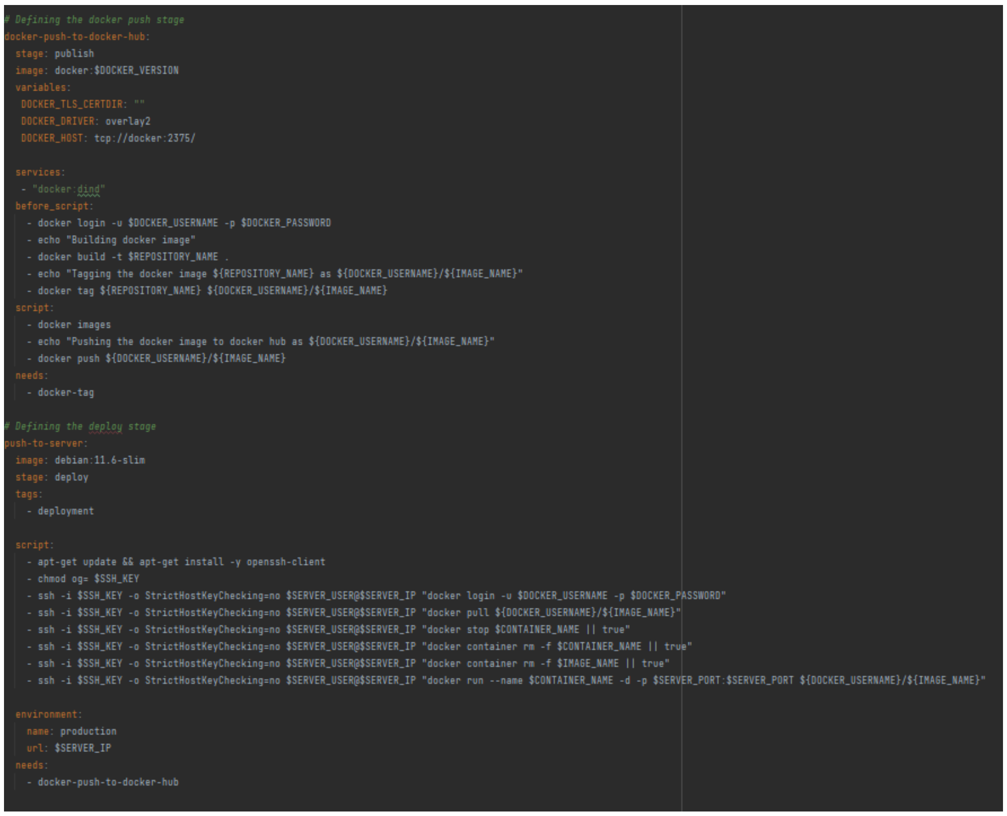
```yaml
# Defining the stages for the pipeline
stages:
  - login
  - build
  - tagging
  - publish
  - deploy

# Defining the variables that can be used in the pipeline
variables:
  DOCKER_VERSION: 23.0.4
  DIND_VERSION: 23.0.4-dind
  ALPINE_VERSION: 3.17.3
  MAVEN_VERSION: 3.8.5-openjdk-18
  IMAGE_NAME: query-engine-backend-image
  REPOSITORY_NAME: query_engine
  SSH_KEY: $ID_RSA_PRIVATE
  MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"
  CONTAINER_NAME: "spring"
  SERVER_PORT: 8080

# Added cache
cache:
  key: "$CI_COMMIT_REF_SLUG"
  paths:
    - /usr/local/share/ca-certificates/
    - /var/lib/docker

# Defining the docker login stage
docker-login:
  stage: login
  image: docker:$DOCKER_VERSION
  script:
    - echo "Logging in to Docker"
    - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD

# Defining the build stage
maven-build:
    stage: build
    image: maven:$MAVEN_VERSION
    script:
        - echo "Compiling"
        - mvn package -B

    artifacts:
        paths:
            - target/*.jar
```

Figure 32: Backend CI/CD configuration. Picture 1/3



```yaml
# Defining the package stage
maven-package:
    stage: build
    image: maven:$MAVEN_VERSION
    script:
        - echo "Packaging"
        - mvn $MAVEN_OPTS clean package

    artifacts:
      paths:
        - target/*.jar

# Defining the docker build stage
docker-build:
  image: docker:$DOCKER_VERSION
  stage: build
  services:
   - docker:dind

  variables:
   DOCKER_TLS_CERTDIR: ""
   DOCKER_DRIVER: overlay2
   DOCKER_HOST: tcp://docker:2375/

  dependencies:
    - maven-package
  before_script:
    - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
  script:
    - echo "Building docker image"
    - docker build -t $REPOSITORY_NAME .

  needs:
    - maven-package

# Defining the docker tag stage
docker-tag:
  stage: tagging
  image: docker:$DOCKER_VERSION
  script:
    - echo "Tagging the docker image ${REPOSITORY_NAME} as ${DOCKER_USERNAME}/${IMAGE_NAME}"
    - docker tag ${REPOSITORY_NAME} ${DOCKER_USERNAME}/${IMAGE_NAME}

  needs:
    - docker-build
```

Figure 33: Backend CI/CD configuration. Picture 2/3



```yaml
# Defining the docker push stage
docker-push-to-docker-hub:
  stage: publish
  image: docker:$DOCKER_VERSION
  variables:
   DOCKER_TLS_CERTDIR: ""
   DOCKER_DRIVER: overlay2
   DOCKER_HOST: tcp://docker:2375/

  services:
   - "docker:dind"
  before_script:
    - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
    - echo "Building docker image"
    - docker build -t $REPOSITORY_NAME .
    - echo "Tagging the docker image ${REPOSITORY_NAME} as ${DOCKER_USERNAME}/${IMAGE_NAME}"
    - docker tag ${REPOSITORY_NAME} ${DOCKER_USERNAME}/${IMAGE_NAME}
  script:
    - docker images
    - echo "Pushing the docker image to docker hub as ${DOCKER_USERNAME}/${IMAGE_NAME}"
    - docker push ${DOCKER_USERNAME}/${IMAGE_NAME}
  needs:
    - docker-tag

# Defining the deploy stage
push-to-server:
  image: debian:11.6-slim
  stage: deploy
  tags:
    - deployment

  script:
    - apt-get update && apt-get install -y openssh-client
    - chmod og= $SSH_KEY
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker pull ${DOCKER_USERNAME}/${IMAGE_NAME}"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker stop $CONTAINER_NAME || true"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker container rm -f $CONTAINER_NAME || true"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker container rm -f $IMAGE_NAME || true"
    - ssh -i $SSH_KEY -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker run --name $CONTAINER_NAME -d -p $SERVER_PORT:$SERVER_PORT ${DOCKER_USERNAME}/${IMAGE_NAME}"

  environment:
    name: production
    url: $SERVER_IP
  needs:
    - docker-push-to-docker-hub
```

Figure 34: Backend CI/CD configuration. Picture 3/3

The backend pipeline configuration is more advanced and consists of multiple stages. We have divided the pipeline configuration into smaller parts for better organization.

In the login stage, our focus is on ensuring successful authentication with Docker. Moving on to the building stage, we verify that it can properly package the project files into a single .jar file and build the Docker image.

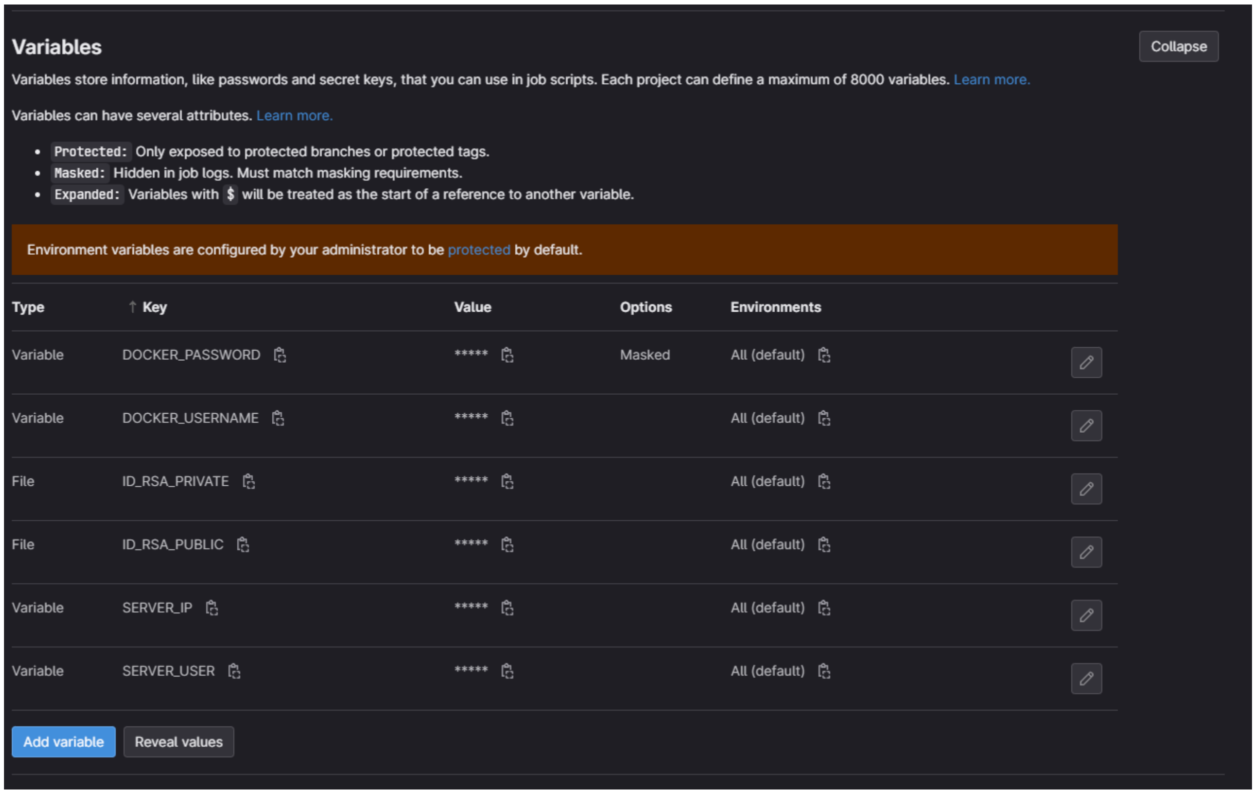
The next stage is tagging, where we assign a tag to the Docker image. In the publish stage, we set up the Docker-in-Docker (DinD) service and establish authentication with Docker. Additionally, we build and tag the image before proceeding to push it to Docker Hub using a script.

Moving on to the deployment stage, our first step is to ensure that all components are up to date. We then proceed to install the OpenSSH client, authenticate with Docker, and pull the desired Docker image. After stopping and removing the existing container, we run the new container with the specified port and image.

It’s worth noting that the publish stage incorporates error handling for the login, building, and tagging processes, as they could potentially lead to issues. This inclusion in the publish stage ensures that any errors are addressed before the script execution.

4.5.7.3: CI/CD Project variables

In order to streamline our project pipelines and enhance the security of CI/CD-related logs, we have implemented variables. These variables allow us to store masked values for sensitive information such as passwords and API keys, preventing them from being visible in logs. To access these variables, simply type a dollar sign followed by the variable name. This approach significantly reduces the need for unique configurations for each project pipeline, minimizes code duplication, and promotes a more universal configuration.



Variables

Variables store information, like passwords and secret keys, that you can use in job scripts. Each project can define a maximum of 8000 variables. Learn more.

Variables can have several attributes. Learn more.

- `Protected:` Only exposed to protected branches or protected tags.
- `Masked:` Hidden in job logs. Must match masking requirements.
- `Expanded:` Variables with `$` will be treated as the start of a reference to another variable.

Environment variables are configured by your administrator to be protected by default.

Type	↑ Key	Value	Options	Environments
Variable	DOCKER_PASSWORD	*****	Masked	All (default)
Variable	DOCKER_USERNAME	*****		All (default)
File	ID_RSA_PRIVATE	*****		All (default)
File	ID_RSA_PUBLIC	*****		All (default)
Variable	SERVER_IP	*****		All (default)
Variable	SERVER_USER	*****		All (default)

Figure 35: 3Gitlab project variables

5: Discussion

This section of the report covers assessment of the methods and results from this project. After reading this chapter the reader should know about limitations, changes and/or deviations that the team encountered during the development of this project.

5.1: Technical result

This section is a discussion of the technical results achieved in the bachelor thesis.

5.1.1: Database

In our project, we aimed to design a database for simplified version of Post Nord parcel delivery system. Throughout the process, we made multiple iterations to identify the important entities and their relationships. Our goal was to achieve Third Normal Form as discussed in chapter 2.4.4 , initially we faced some confusion about how data would be created and edited in the database. But after discussing the issue with the Post Nord, we decided to focus on creating a database that reflected the state of the system at a single moment, without considering how the data would be inserted or modified. Focusing solely on organizing the data made the designing process easier. However, as discussed in chapter 4.3.1, our final design represents a simplified version of the Post Nord database. It does not cover all scenarios of the real system.

5.1.2: Spring boot / GraphQL

Spring boot was easy to setup with Maven build system by using pom file to get all the necessary dependencies. We used JPA to convert all the tables to objects/entities in spring boot. This part of the development went without any challenges. Time that was used to fetch data from the database was the challenging part in the backend. To overcome this, we created tables with the aggregate functions. This made it easy to fetch the data directly from the result tables instead of running queries with aggregate functions when fetch request is sent.

GraphQL made it easy to fetch only required information to display in different part of the pages. It prevented from over fetching and under fetching. As a strong type supported language, it was challenging to fetch aggregate function results, which are not defined as entities in spring boot. We had to create entities in spring boot to fetch these results, as GraphQL supports only defined types as results.

5.1.3: Vue.js

Our web application demonstrates how data from the database can be displayed to the user in various formats. Component based feature allowed us to create many components individually that could be used in different pages.

We had rendering issues when we started fetching data from the database. As we have a large amount of data in our database, aggregate functions took long time to return the results. This led to components being not displayed when they were waiting for the results. This is solved by using reactive feature of the Vue. This made the components to re-evaluate and update whenever there was a change in state of the data.

5.2: Process

This sub-chapter covers discussion about the results from collaboration between team members, collaboration tools and working process.

5.2.1: Collaboration

The communication between the group has been impeccable throughout the whole development process. All team members have always been punctual, and in cases where one or more team members needed to be absent for various reasons the remaining team members were always informed in good time. This applies to both physical and digital meetings.

If cases where there was a discussion, all team members had an equal say on the matter. All members contributed with their views, and the solution that was made was always unanimous and in the best interests of the project.

Time spent on the project varies amongst members, for different reasons. Amongst other things we experienced some days with sickness, and two of the team members have children. With a lot of public holidays during April and May this affected how much some of the team members could contribute in certain periods.

The general result of the teamwork can be considered as a success. There were no situations where it was necessary to mediate between group members, so the collaboration worked well.

5.2.2: Jira and Scrum

Jira(3.6.1) is a tool that the team was introduced to in the start of the semester, and none of the members had any prior experience using it. This is also the case regarding working with the Scrum methodology(2.2).

At the beginning of the project the experience was varying since the team was new to working in this way. One of the main issues was that the sprint planning was sub-optimal, where the team did not consider all the tasks that needed to be done in each sprint. We did not use the full strength of the program or the methodology, so we encountered a period of poor planning.

5.2.2.1: Sprint planning

The backlog was not complementary enough from the start, which resulted in some challenges when planning for the sprint(3.2.3). The team had made some of the elements needed to fulfill the project, but not well enough to execute it properly. This was more of an issue in the beginning and got better gradually.

5.2.2.2: Time assessment

Time planning were one of the biggest challenges the team faced during each sprint. Often when planning the next sprint(3.2.3) the team forgot to set an estimated time for the task to be finished, which resulted in tasks being carried over from one sprint to another.

5.2.2.3: Time logging

The timesheets in Jira proved helpful, where each team member could track their hours easily by picking a date and log it on a given task. The backside of our project is that since the issues were not properly divided into clear subtasks at all times, many hours were logged on the same issue repeatedly.

5.2.2.4: Daily standup

In the beginning of the project the team used way too much time on the daily standup(3.2.2), where the longest meeting ran for about an hour. After discussing it internally and with the supervisor, we made some adjustments, so the meetings got shortened to about 5 minutes.

5.2.3: Confluence
Confluence (3.6.2) initially took some time to learn to utilize properly as the group all were new to the program, though as time went on the team became more efficient. The program quickly became strong for us as we learned how to properly adept its templates, allowing us to efficiently plan meetings, record meeting summaries, reflect on previous sprints with retrospectives and collaboratively share concepts, ideas, and sketches with each other during the development process. (3.2)

5.3.3.1: Calling in meetings
At the start of the project, we set up weekly meetings (3.2.6) on teams with the product owner (2.2.3) of our project, hence we don’t have any documentation for calling in meetings along with the attachments of the report. Though we have stayed well in touch with the project owner the entire journey through development and made sure they’ve been kept up to date with our progression on a weekly basis. After every meeting we took a recap on what was mentioned during it and wrote down a summary of the meeting (3.2.7)

5.3.3.2: Sprint retrospectives
On each week after we finished a sprint we would gather around and reflect on what had gone well, what we could’ve done better and whether we managed to reach the goals we had set for the sprint (3.2.5). These reflections were often rushed and brushed over during the project as we often did not reach the goal we had set up at the end of the week. If we had spent more time trying to figure out why we missed our goals, we possibly could’ve made our workflow more efficient. Though furthermore we should’ve realized something was off as we kept missing goals, or the goals were too grand for the given timespan.

Chapter 6: Conclusion

This chapter covers the conclusion based on the work and the results throughout the project, as well as recommendations for others who want to create a similar project.

6.1: Project conclusion

Based on the technical results of this project, the group is somewhat satisfied with the product. The group managed to create a web-application using the Vue.js framework for frontend, Spring boot and GraphQL for the backend and a relational database, and set it live on a server. The team also created a CI/CD pipeline for automatic building, testing, and deploying the application, which is part of the desired solution.

The main goal was to create an application for the non-SQL user to extract internal information from the database, to prevent taking unnecessary time away from the developers. This criterion was met, but the scope of functionality is much smaller than the team intended when starting the task. While we are satisfied with having solved the problem, we would preferably have solved it in a better way.

Retrospectively, if the team had started the project all over again with the knowledge gained from this experience, we would have done it a bit differently. This is regarding both the working process and the system. The team should have tested the database logic earlier on and considered several options for the structure. Late in the project we realized that some of the tables were not connected in the best suitable way to solve the problem, and it was difficult doing right queries, as well as the queries had poor execution time. This resulted in a sub-optimal performance of the web-application when extracting information for certain tables, and other parts worked as intended.

Another aspect of the project that took a lot longer than expected was the frontend development. We used a lot of time on sketching up some solutions, but ended up with a result that was completely different in some parts of the site. This meaning that we spent a lot of time on planning something that never became useful for the project. In addition, none of the team members had any prior experience with GraphQL, Informix, Vue.js, and similar JavaScript frameworks, so we all needed to use a lot of time on different tutorials under the whole development process.

The product owners also wanted us to do a comparison between Spring boot and GraphQL, for the academical perspective. This is a topic they were interested in seeing a result, but with time this became less likely to realize due to minimal time left for implementing this.

From a learning perspective the team concludes that the project was interesting and an enjoyable process. Learning new technologies is an important skill in the field of software development and comparing the start and the end of the project, we felt that our knowledge and skills in the tools and frameworks improved.

6.2: Recommendations

The team would like to recommend anyone who attempt to recreate this project or make a similar project to do a bit more research on the subject. Look for more projects that is in the same genre to see if there is anything that could be useful to your own project and do proper planning.

If implementing the Scrum methodology, make sure to create user-stories from the start, and implement a solid backlog before starting the work. If possible, divide every task into concrete subtasks, and use a method for time estimation so that there is some form of measurements on the problems solved.

If the person(s) attempting to recreate the project has any prior experience with similar technologies, we would recommend using similar languages to solve the problem, due to getting used to new frameworks, programming languages and tools could become time consuming, and can become a decisive factor if there is a deadline.

References

Access computing, 2023. *The Alliance for Access to Computing Careers.* [Online]
Available at: https://www.washington.edu/accesscomputing/what-difference-between-accessible-usable-and-universal-design
[Accessed 17th May 2023].

Aela, 2023. *Aela.* [Online]
Available at: https://aelaschool.com/en/interactiondesign/interaction-design-principles/
[Accessed 2nd May 2023].

altexsoft, 2023. *altexsoft.com.* [Online]
Available at: https://www.altexsoft.com/blog/engineering/graphql-core-features-architecture-pros-and-cons/

altexsoft, 2023. *altexsoft.com.* [Online]
Available at: https://www.altexsoft.com/blog/engineering/graphql-core-features-architecture-pros-and-cons/

amazon, 2023. *aws.* [Online]
Available at: https://aws.amazon.com/what-is/api/
[Accessed 04 Mai 2023].

Amazon, 2023. *aws.amazon.* [Online]
Available at: https://aws.amazon.com/what-is/javascript/
[Accessed 4 Mai 2023].

Atlassian, 2023. [Online]
Available at: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Atlassian, 2023. *Atlassian.* [Online]
Available at: https://www.atlassian.com/agile
[Accessed 19 April 2023].

Batra, R., 2018. *SQL Primer - An Accelerated Introduction to SQL Basics.* 1. edition ed. s.l.:Apress.

Chacon, S. & Straub, B., 2014. *Git-scm.* [Online]
Available at: https://git-scm.com/book/en/v2

Chapman, C., 2023. *toptal.* [Online]
Available at: https://www.toptal.com/designers/ux/color-in-ux
[Accessed 2nd May 2023].

Codecademy, 2021. *Codecademy.com.* [Online]
Available at: https://www.codecademy.com/resources/blog/what-is-a-framework/

Codecademy, 2023. *codecademy.org.* [Online]
Available at: https://www.codecademy.com/article/what-is-an-ide
[Accessed 01 Mai 2023].

Collings, T., 2021. *tom-collings.medium.com.* [Online]
Available at: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5
[Accessed 18 May 2023].

Connolly, T. M. & Begg, C. E., 2015. *Database Systems.* 6. edition ed. s.l.:Pearson Education Limited.

DBeaver, 2023. *DBeaver.* [Online]
Available at: https://dbeaver.io/about/
[Accessed 4th May 2023].

DBeaver, 2023. *DBeaver.* [Online]
Available at: https://dbeaver.com/docs/wiki/Data-transfer/
[Accessed 4th May 2023].

Discord, 2023. *discord.com.* [Online]
Available at: https://discord.com/safety/360044149331-what-is-discord
[Accessed 2 May 2023].

Figma, 2023. *Figma.* [Online]
Available at: https://www.figma.com
[Accessed 4th May 2023].

Geekforgeeks, 2023. *Geekforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/
[Accessed 15 05 2023].

Gitkraken, 2023. [Online]
Available at: https://www.gitkraken.com/learn/git/git-flow

GitKraken, 2023. *gitkraken.com.* [Online]
Available at: https://www.gitkraken.com/about
[Accessed 3 May 2023].

GitLab, 2023. *GitLab.* [Online]
Available at: about.gitlab.com/features/continuous-integration/

Git-scm, 2023. *Git-scm.* [Online]
Available at: https://git-scm.com/video/what-is-version-control

git-scm, 2023. *git-scm.com.* [Online]
Available at: https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes
[Accessed 1 May 2023].

Git-scm, 2023. *What is Git?.* [Online]
Available at: https://git-scm.com/video/what-is-git

Gordon, K., 2021. *Nielsen Norman Group.* [Online]
Available at: https://www.nngroup.com/articles/color-enhance-design/
[Accessed 2nd May 2023].

GraphQL, 2023. *graphql.org.* [Online]
Available at: https://graphql.org/learn/

Honeypot, 2023. *Youttube.* [Online]
Available at: https://www.youtube.com/watch?v=783ccP__No8

ibm.com, 2023. *ibm.com.* [Online]
Available at: https://www.ibm.com/topics/software-testing
[Accessed 18 May 2023].

IBM, 2023. *IBM.* [Online]
Available at: https://www.ibm.com/topics/java

IBM, 2023. *ibm.com.* [Online]
Available at: https://www.ibm.com/topics/java-spring-boot
[Accessed 3 Mai 2023].

IBM, 2023. *ibm.com.* [Online]
Available at: https://www.ibm.com/topics/api
[Accessed 3 Mai 2023].

JetBrains, 2023. *jetbrains.* [Online]
Available at: https://www.jetbrains.com/webstorm/

JetBrains, 2023. *jetbrains.com.* [Online]
Available at: https://www.jetbrains.com/idea/features/
[Accessed 01 May 2023].

Learn Microsoft, 2023. *learn.microsoft.com.* [Online]
Available at: https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description
[Accessed 21 May 2023].

maven.apache.org, 2023. *maven.apache.org.* [Online]
Available at: https://maven.apache.org/what-is-maven.html
[Accessed 18 May 2023].

maven.apache.org, 2023. *maven.apache.org.* [Online]
Available at: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html
[Accessed 19 May 2023].

microservices.io, 2023. *microservices.io.* [Online]
Available at: https://microservices.io/
[Accessed 18 May 2023].

Microsoft, 2023. *Microsoft.com.* [Online]
Available at: https://support.microsoft.com/en-us/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29
[Accessed 3 May 2023].

Mozilla.org, 2023. *Mozilla.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Tools_and_setup/What_are_browser_developer_tools

oberlo.com, 2023. *oberlo.com.* [Online]
Available at: oberlo.com/statistics/mobile-internet-traffic
[Accessed 17 May 2023].

Pagade, G., 2022. *baeldung.com.* [Online]
Available at: https://www.baeldung.com/cs/cohesion-vs-coupling
[Accessed 18 May 2023].

Postman, 2023. *postman.* [Online]
Available at: https://www.postman.com

Putty, 2023. *Putty.* [Online]
Available at: https://www.putty.org

Putty, 2023. *tartarus.* [Online]
Available at: https://tartarus.org/~simon/putty-snapshots/htmldoc/Chapter1.html#intro

Red Hat Inc., 2023. *Red Hat.* [Online]
Available at: https://www.redhat.com/en/topics/devops/what-is-ci-cd

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-team/what-is-a-scrum-master
[Accessed 19 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-team/what-is-a-product-owner
[Accessed 19 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-team/what-is-a-developer
[Accessed 19 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-team
[Accessed 19 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-events/what-is-a-sprint
[Accessed 19 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-

events/what-is-sprint-planning
[Accessed 22 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-events/what-is-a-sprint-review
[Accessed 22 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-events/what-is-a-sprint-retrospective
[Accessed 22 April 2023].

Scrum.org, 2023. *Scrum.org.* [Online]
Available at: https://www.scrum.org/learning-series/what-is-scrum/what-is-scrum
[Accessed 22 April 2023].

Sharan, K. & Davis, A. L., 2022. *Beginning Java 17 Fundamentals: Object-Oriented Programming in Java 17.* 3. edition ed. s.l.:Apress.

Sitepoint, 2023. *Sitepoint.* [Online]
Available at: https://www.sitepoint.com/vue-vs-react/

Sitepoint, 2023. *Sitepoint.* [Online]
Available at: https://www.sitepoint.com/

Sitepoint, 2023. *Sitepoint.* [Online]
Available at: https://www.sitepoint.com/community/t/how-to-tell-if-vue-js-is-the-right-framework-for-your-next-project/318572

smartbear.com, 2023. *smartbear.com.* [Online]
Available at: https://smartbear.com/learn/automated-testing/what-is-unit-testing/
[Accessed 18 May 2023].

sourcemaking.com, 2023. *sourcemaking.com.* [Online]
Available at: https://sourcemaking.com/design_patterns
[Accessed 23 April 2023].

Sumathi, S. & Esakkirajan, S., 2007. *Fundamentals of Relational Database Management Systems.* 1. edition ed. s.l.:Springer.

The GraphQL community, 2023. *https://www.howtographql.com.* [Online]
Available at: https://www.howtographql.com/basics/1-graphql-is-the-better-rest/

Toptal, 2022. *Toptal.* [Online]
Available at: https://www.toptal.com/designers/typography/web-typography-infographic
[Accessed 2nd May 2023].

UX Engineer, 2023. *UX Engineer, White space, Princeples of design.* [Online]
Available at: https://uxengineer.com/principles-of-design/white-space/

Visualstudio, 2023. *code.visualstudio.com.* [Online]
Available at: https://code.visualstudio.com/docs/editor/whyvscode
[Accessed 3 May 2023].

Vocke, H., 2018. *martinfowler.com.* [Online]
Available at: https://martinfowler.com/articles/practical-test-pyramid.html
[Accessed 18 May 2023].

Vue.js, 2023. *Vue.js Documentation.* [Online]
Available at: https://vuejs.org

Vue.js, 2023. *Vue.js Github Repository.* [Online]
Available at: https://ithub.com/vuejs/vue

w3schools.com, 2023. *w3schools.com.* [Online]
Available at: https://www.w3schools.com/html/html_responsive.asp
[Accessed 17 May 2023].

WebFX, 2023. *WebFX.* [Online]
Available at: https://www.webfx.com/web-design/learn/tips-for-using-images-in-website-design/
[Accessed 5th May 2023].

WinSCP.net, 2023. *WinSCP.net.* [Online]
Available at: https://winscp.net/eng/index.php

WinSCP.net, 2023. *WinSCP.net.* [Online]
Available at: https://winscp.net/eng/docs/introduction

Wireshark, 2023. *Wireshark.* [Online]
Available at: https://www.wireshark.org

Wireshark, 2023. *Wireshark.* [Online]
Available at: https://www.wireshark.org/faq.html

Wrike, 2023. *Wrike.* [Online]
Available at: https://www.wrike.com/blog/what-is-a-use-case/
[Accessed 18th May 2023].

Appendices

A Preliminary project plann

A

Common Query Engine and Viewer
Preliminary project plan

Version <1.0>

Revision history

Date	Version	Description	Author
26.01.2023	0.1	First draft	Alle

Table of contents

Table of contents **2**
1 .... **Goals and outlines** **3**
1.1 Orientation 3
1.2 Problem description / project description and performance goals 3
1.3 Performance measure 4
2. Organizing **4**
3. Execution **5**
3.1. Main activities 5
3.2. Milestones 5
4. Follow-up and quality assurance **7**
4.1 quality assurance 7
4.2 reporting 8
5. Risk assessment **9**
6. Appendix .... Feil! Bokmerke er ikke definert.
6.1 Schedule .... **Feil! Bokmerke er ikke definert.**
6.2 Agreement documents .... **Feil! Bokmerke er ikke definert.**
6.2.1 Employment contract for bachelor group.......... **Feil! Bokmerke er ikke definert.**
6.2.2 3-party agreement.......... **Feil! Bokmerke er ikke definert.**

1. Goals and outlines

1.1 Orientation

After attending a meeting about the bachelor thesis in October 2022, all the bachelor projects were published on Blackboard a couple of weeks after that. The four of us working together on the project sat down together and looked at all the different projects. First, we looked through all the available projects before narrowing it down to the ones we would like to work on. We found many of them interesting but decided to go for a project where it seemed like we could have some prior knowledge to solve the task.

The project we got "Common Query Engine and Viewer", which we would write for Postnord was our first choice, and we got the project. The reason for picking this project was that the problem description seemed interesting. To create a system so that non-experts in SQL could do queries to free up developer time to do other things. And the way we interpreted the task it seemed like this would give us the chance to work more with databases, front-end and back-end development, which we liked from previous courses.

1.2 Problem description / project description and performance goals

<u>Problem description</u>

Using developer time to extract reports from a database takes developers away from development and they quickly become a bottle neck for supporting day-to-day operations. This is expensive in the long term and the goal is therefore to explore solutions that takes developers out of the loop. The requester of information (should not need to know SQL) should be able to run the queries by themselves. The assignment consists of several parts:

- What is a good UX for the non-specialist end-user to retrieve information (check boxes, dropdowns, JQL like Jira Querying language, …)
- Implementation of a prototype system with GUI
- Evaluation of the solution with end-users (PostNord business users)

If there is enough time, compare the efficiency between Spring Boot and GraphQL when querying from the database.

Performance goals

Having a user interface that responds to query operations so that people with no prior SQL knowledge can extract reports from the database in a simple manner.

1.3 Performance measure

The long-term goal for this assignment is to give PostNord a Query search engine prototype, which they can use and take inspiration from in order to improve their current database system. They would also like us to explore different options regarding efficiency between Java REST API and GraphQL, as they wish to know which has better performance in case they do end up expanding their current database solution. When this assignment is completed, PostNord will be left with a decent insight into the technology used to develop the prototype as well as gain knowledge of the recommended technologies to further improve their database system within the company.

For each student knowledge and experience in a real case is the goal itself, since there are no financial benefits of doing the project. We get to work with some new technologies and tools as well as some tools and technologies we have previous experience with.

2. Organizing

NTNU, PostNord and Student group

3. Execution

3.1. Main activities

For this project, main activities include sketching a data model and implementing a database which includes the requirements of the PostNord system. Developing a backend system using Spring boot to query the database and creating REST Apis. As required, by PostNord a GUI will be developed using Vue JS, where the end user can use checkbox, drop down, text boxes to extract information from the database. PostNord wishes to spare the time used by the developers, querying the database by using this Query engine viewer application.

Scrum methodology will be used in the development of application. One-week sprints will be used to quality check the work done the week before and discuss the work to be done. There will daily standup meetings to check the progress of the scheduled plan. PostNord will be given information about the progress in the project in two-week sprint meetings. The supervisor and the student group will have one-week sprint meetings to discuss the progress and problems regarding the project. Jira and confluence will be used as platforms to document the scrum methodology.

The group will start with an agreement, where members are agreed to where and when the meetings will be held and what the group aims for as the result at the end of the project. Sketches and models of the application will be available to all the members such that members are aware of how the final product should be. Group members will be using Jira and Confluence to document and plan the process of the project.

3.2. Milestones

Milestones		
Month	**Date**	**Task**
January	13.01.2023	Initial meeting with product owner (Postnord)
	28.01.2023	Deadline preliminary project report w/appendix
February	10.02.2023	MVP DB-model
March	03.03.2023	Finishing implementing back-end
	17.03.2023	Combining front-end and back-end
April	07.04.2023	MVP application
		Project presentation in English
May	05.05.2023	Sending in project report to supervisor for last feedback iteration
	12.05.2023	Design poster for project presentation
	22.05.2023	Deadline bachelor thesis / final presentation

4. Follow-up and quality assurance

4.1 quality assurance

Group members are scheduled to meet every Thursday and Friday. This will ensure that there is no misunderstanding about the work to be done and what is expected from the members. The work done the week before will be discussed and checked to ensure that it keeps up with the group's expectations. If the quality of the work is not up to expectations, then it will be redone by the group. The members of the group will be given enough time and help to redo the work.

As it is mentioned in group agreement, it is expected that the members ask for help when the need arises as soon as possible to deliver the product within the deadline. In addition, the group will be using Scrum methodology, which includes daily standup meetings, sprint planning, sprint review and sprint retrospective. These scrum techniques will be used to evaluate the quality of the work done by the members.

4.2 reporting

During this project, we will have to report to our supervisor and the product owner regularly.

Supervisor

Name: Rituka Jaiswal

Email: rituka.jaiswal@ntnu.no

Sprint frequency: 1 week – meeting every Friday

Product owner

Contact person: Torgeir Grothe Lien

Email: torgeir.lien@postnord.com

Sprint frequency: 2 weeks – meeting in the end of the week every other week

5. Risk assessment

Probability / consequence		**Insignificant**	**Moderate**	**Serious**	**Critical**
		1	**2**	**3**	**4**
Likely	**4**		Misunderstanding with the employer. Learning a new programming language		Workload from other subjects
Possible	**3**	Coming late to the meetings	Knowledge and competence Conflicts and disagreements Group member not attending scheduled meeting Sick members or children		
Less likely	**2**			Product scope is not well defined	Not delivering a working product
Rarely	**1**		Canceled boats or ferries	The scope of the project may change Misunderstandings Communication issues	Work not done by deadline.

Members agreed on the procedure, which holds information about meeting notice, notification of absence document management, submission of group work and interaction, which includes information about attendance, engagement and disagreement in the group. This information is documented in the group contract where all the members are agreed upon and signed. This group contract will be used to solve most of the problems that come under risk assessment.

To avoid confusions about the scope of the product, the group will make multiple sketches in iteration till the scope is well defined. This will be done in sprint meetings with the employer PostNord. The group will be working on Thursdays and Fridays till week 12 due to the workload from the other subject. From week 13, the whole week will be used to work on the project. Every member in the group is responsible for quality and delivering work before the deadline.

Members must learn Vue JS framework to code in the project. Prior programming knowledge and knowledge about how to learn frameworks should help the members to overcome this task. In addition, has the group agreed to learn Vue from the beginning of the project when time permits.

NTNU
Norwegian University of
Science and Technology