Developing a statistics interface for the Klippa DocHorizon application

Bachelor's Project Computing Science

July 2024

Author: Tomás Leote Falcão
Student Number: S4246683
First supervisor: Andrea Capiluppi
External supervisor: Klaas Jelmer Boskma

Abstract

This thesis presents the development of a statistical interface for Klippa's Doc-horizon platform, a document management solution that automates and simplifies business processes. The objective of the interface is to provide users with a tool for visualising their platform usage data, offering insights into their interactions with Klippa's product.

The primary objective of this project is to enhance the decision-making abilities of Klippa users and employees, and to improve the overall user experience by offering detailed visualisations of usage patterns and metrics. This was achieved by implementing new queries and schemas on the back-end to collect and aggregate usage data with GraphQL, and a front-end interface using Angular to support dynamic filtering and grouping of data. The interface enables users to tailor their data views according to a range of criteria, including date ranges, projects, service types, and different metric codes.

The thesis demonstrates the practical application of combining data collection, transformation, and visualization techniques to create a user-friendly interface. Additionally, it outlines the challenges encountered during development and proposes future enhancements.

Overall, this project aims to bridge the gap between complex data sets and user-friendly visualizations within Klippa's platform, contributing to the enhancement of the Doc-horizon platform and providing valuable insights for its different stakeholders.

Contents

1 Glossary

Angular: A platform and framework for building single-page client applications using HTML and TypeScript, developed by Google.

API (Application Programming Interface): A set of protocols and tools for building software and applications that allow different software entities to communicate with each other.

ClickHouse: An open-source columnar database management system designed for online analytical processing (OLAP), known for its high performance.

CSV (Comma-Separated Values): A simple file format used to store tabular data, such as a spreadsheet or database.

Golang (Go Programming Language): A statically typed, compiled programming language designed at Google, known for its simplicity and efficiency in building scalable and reliable software.

GraphQL: A query language for APIs and a runtime for executing those queries by allowing clients to request exactly the data they need.

HITL (Human-in-the-loop): An approach that integrates human judgment into the processing or analysis workflow to ensure high accuracy and compliance.

JSON (JavaScript Object Notation): A lightweight data-interchange format that is easy for humans to read and write and for machines to parse and generate.

KPI (Key Performance Indicator): A measurable value that demonstrates how effectively an organization or individual is achieving key business objectives.

MVP (Minimum Viable Product): A product with the minimum features needed to satisfy early adopters and provide feedback for future development.

OCR (Optical Character Recognition): The technology used to convert different types of documents, such as scanned paper documents, PDFs, or images captured by a digital camera, into editable and searchable data.

SDK (Software Development Kit): A collection of software tools and libraries designed to help developers create applications for specific platforms.

UI/UX (User Interface/User Experience): Design principles focused on optimizing the interaction between users and the product, ensuring usability, accessibility, and pleasure in the interaction.

MoSCoW (Must-have, Should-have, Could-have, Won't-have): A prioritization technique used in project management to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement.

Back-end: The part of a computer system or application dealing with data storage and business logic, typically running on a server.

Front-end: The part of a computer system or application with which the user interacts directly, typically running in a web browser.

Module: A self-contained unit of code that encapsulates specific functionality in a software application.

Component: A reusable piece of the user interface in a front-end framework such as Angular.

Service: A piece of software that provides reusable business logic and data management functionality in an application.

Highcharts: A charting library written in JavaScript that is used to create interactive charts for web applications.

Dashboard: An interface that provides a consolidated view of multiple data visualizations and metrics for monitoring and analysis purposes.

Visualization: The representation of data in graphical format, such as charts or graphs, to facilitate understanding and analysis.

Data transformation: The process of converting data from one format or structure to another.

Data seeding: The process of populating a database with initial data for testing or setup purposes.

Scalability: The capability of a system to handle increased load or demand by adding resources.

Maintainability: The ease with which a software system can be modified to fix defects, improve performance, or adapt to a changed environment.

Usage metrics: Quantitative measures that provide insights into how a system or application is being used.

2 Introduction

2.1 Background

In the rapidly evolving landscape of document management and digital transformation, there is a growing demand from businesses for efficient and user-friendly platforms that can facilitate the streamlining of their processes. Klippa, a company specialized in document management solutions, has developed the Doc-Horizon platform with the objective of addressing these needs. The Doc-Horizon platform offers a simplified and automated approach to document processing, encompassing capture, verification, and management of documents [11].

Despite the widespread availability of document management solutions, there remains a significant gap in providing users with intuitive and user-friendly interfaces to visualise their usage data. This gap presents an opportunity to develop a statistical interface for the Doc-Horizon platform, enabling users to gain detailed insights into their usage patterns and metrics.

The primary objective of this thesis is to evaluate the characteristics of an effective charting library for this use case using an industry-as-laboratory approach and to develop a tool that enables Klippa users to visualise and analyse their usage data effectively within the Doc-Horizon platform.

2.2 Aim of the Paper

The objective of this paper is to provide a detailed documentation of the development process undertaken for the design and implementation of a statistical interface for the analysis of platform usage data on the Doc-Horizon platform. This project seeks to enhance the decision-making capabilities of different stakeholders, by offering detailed insights into their interactions with the platform's various services and metrics.

Specifically, this paper aims to:

1. Describe the technical approach and methodologies used in developing the interface.
2. Detail the selection process of identifying a suitable charting library for this project.
3. Discuss the final product and the learning curve

By achieving these objectives, the paper intends to provide a comprehensive overview of the project's contribution to the Doc-Horizon platform and its potential impact on user engagement and satisfaction.

2.3 Paper Structure

This paper is organized into several sections to provide a comprehensive overview of the development and implementation of the statistical interface for Klippa's Doc-Horizon platform.

The section entitled "State of the Art" presents a review of existing technologies and methods for visualising usage data. It examines the ways in which these technologies are employed within the industry and identifies shortcomings that the current project seeks to rectify, thus providing a basis for comprehending the project's contributions.

In the System Overview, the paper provides an introduction to the Doc-Horizon platform, including an overview of its various services and metrics. This serves as a foundation for the subsequent chapters of this thesis, which are based upon the aforementioned platform. This section elucidates the manner in which the statistical interface is integrated with the platform, thereby affording a comprehensive insight into the constituent components and functionalities thereof.

The Requirements section provides a detailed account of the project stakeholders, epics, and user stories, which is essential for a comprehensive understanding of the project requirements and potential improvements.

The Design chapter provides an in-depth examination of the system architecture and design principles employed in the development of the interface. It covers the high-level overview, user interface, conceptual view, module view, and scalability and maintenance considerations, ensuring an efficient, scalable, and maintainable solution.

The Implementation chapter details the methodologies and phases involved in the development process. It covers the onboarding and technology stack, selection of an effective library, MVP development, and subsequent improvements based on stakeholder feedback. It also discusses the integration and testing processes to ensure the system meets the requirements and performs efficiently.

The Results, Discussion Testing section presents the outcomes of the project, including the stakeholder's feedback. The final interface and its capabilities are presented as examples, demonstrating how the project met its objectives.

In conclusion, the thesis presents a summary of its principal findings and contributions. It offers an evaluation of the project as a whole, its influence on the Doc-Horizon platform, and its broader implications for the fields of data visualisation and user interface design.

3 State of the Art

To develop a robust statistics interface for the DocHorizon platform, it is crucial to examine how similar interfaces are designed and implemented by other companies. By analyzing these existing solutions, I can identify best practices and areas for improvement to ensure Klippa's interface provides superior functionality and user experience. This section draws upon the research conducted in my thesis proposal [4], particularly the review of existing platform usage statistics interfaces.

3.1 Existing Interfaces

In discussions with Klippa, it became clear that the goal for their statistical interface is to build upon existing solutions rather than "reinventing the wheel". By investigating how other platforms offer statistical services, I can develop an approach tailored to Klippa's needs. Therefore, it is essential to explore and understand the interfaces currently used in the industry.

3.1.1 Azure Monitor

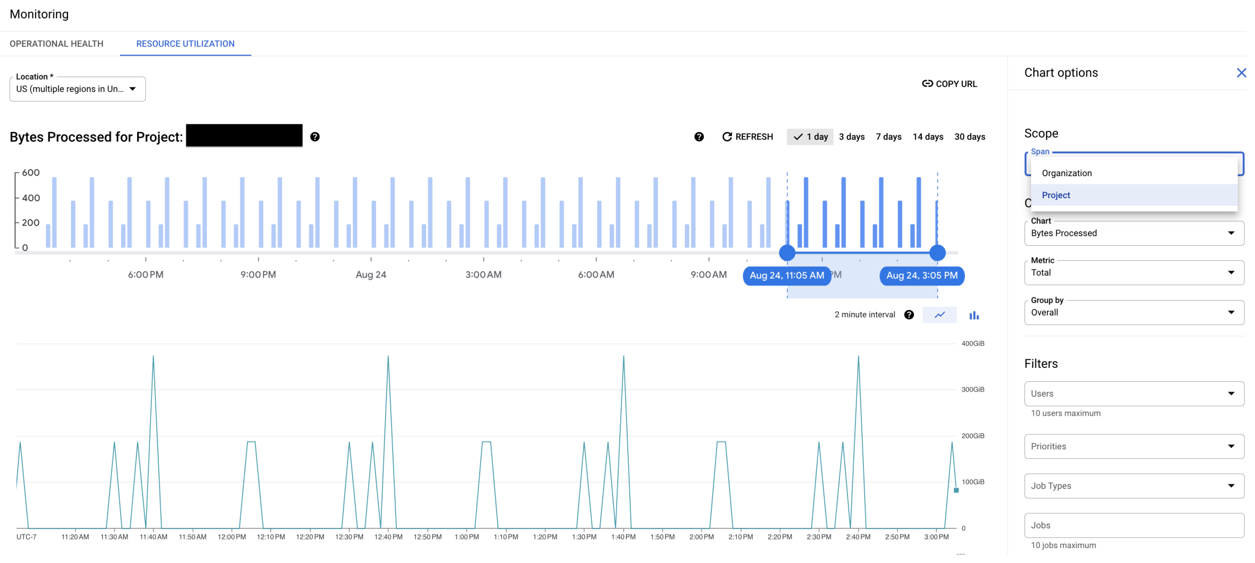
One notable example is the Azure Monitor interface, which is renowned for its comprehensive dashboard features. Azure Monitor integrates various data types into customizable panels, providing real-time insights and performance metrics [1]. Its interactive visual elements allow users to delve deeper into specific metrics or logs for detailed analysis. For the DocHorizon platform, adopting similar dashboard features could enhance its statistical interface, enabling users to effectively monitor document processing activities and analyze data trends within a unified interface.

3.1.2 Google Cloud Platform Monitoring Tool

Google Cloud Platform (GCP) represents a sophisticated and comprehensive suite of tools designed for the monitoring, analysis, and visualisation of cloud resources and service usage. GCP's monitoring tool, which offers users the ability to create customised dashboards, provides real-time insights into the usage of the platform through a variety of charts and graphs [2]. A comprehensive examination of the platform's components and features is presented below.

- **Filters and Data Selection:** GCP offers extensive filtering options, including predefined time intervals, custom time ranges, and various filters to refine displayed data, enabling users to gain precise insights into their data metrics.
- **Live Data Toggle:** This feature allows users to switch between live and historical data, facilitating real-time monitoring and improving load times by fetching the most current data only when necessary.

- **Chart Options:** Users can select from different chart types, line or bar chart, and choose metrics like total bytes processed, latency, and request counts. These options allow for detailed data grouping and comprehensive analysis.
- **Cost Management:** GCP includes tools for monitoring resource usage and costs. It provides visibility into expenditure and includes features for setting policies and alerts to optimize and manage costs effectively.



Location * US (multiple regions in Un...

Bytes Processed for Project: [redacted]

Chart: Bytes Processed

Metric: Total

Group by: Overall

Figure 1: GCP Usage statistics

In developing the Doc-Horizon statistics page, several functionalities were inspired by the approaches used in the GCP Monitoring tool, ensuring an intuitive and robust monitoring solution. Key similarities include:

- **Filters Bar:** I adopted a similar approach by implementing a filters bar that includes relevant filters for the type of data displayed on the page.
- **Date Range Buttons:** Predefined date range buttons were added, allowing users to quickly select commonly used ranges such as yesterday, past 7 days, past 30 days, etc.
- **Chart Configuration Options:** A section for chart customization was included, enabling users to select different options, such as the 'Group by' feature, to tailor the visualization to their needs.
- **Projects Table:** Similar to GCP's jobs table, I implemented a projects table where users can get a tabulated overview of their projects, offering insights comparable to the GCP jobs table.
- **Chart Types:** Both line and bar charts were implemented, allowing users to choose their preferred chart types for data visualization.

However, the Doc-Horizon statistics page addresses some of the limitations observed in the GCP Monitoring tool by introducing additional features:

- **Extensive Chart Customization:** While GCP offers some level of chart customization, my interface provides more focused customization options, such as adjusting the chart interval, to visualize data with more or less granularity to better suit the specific needs of the platform users.
- **Custom Date Ranges:** Unlike GCP, which restricts date selection to fixed intervals (1, 3, 7, or 30 days), my interface allows users to select any custom date range. This flexibility enables the visualization of larger and older datasets, providing more comprehensive usage analysis.
- **General Statistics Section:** I observed that users did not have access to a quick overview of specific statistics. Following discussions with the product designer, it was determined that this was a missing feature in GCP that could significantly enhance the user experience within Doc-Horizon. The incorporation of this feature was intended to provide users with immediate access to key metrics, thereby enhancing overall usability and efficiency.

In conclusion, while GCP Monitoring offers a comprehensive and versatile solution for cloud monitoring statistics and Azure Monitor provides an extensive and customisable dashboard approach, I utilised these platforms as inspirations to tailor a solution that specifically addresses the unique needs of the project stakeholders. By focusing on detailed insights into platform usage, I ensured that the adopted user-centric design remains both robust and easy to use, thereby facilitating effective data-driven decision-making and enhancing the overall user experience.

4 System Overview

4.1 Introduction to Doc-Horizon

Doc-Horizon has been designed with the objective of streamlining and automating various aspects of document processing for businesses. As an advanced solution in the field of digital transformation, Doc-Horizon employs state-of-the-art technologies to provide a comprehensive range of functionalities, with the objective of enhancing efficiency, accuracy, and user experience in document management.

4.1.1 OCR Data Extraction

Klippa's Doc-Horizon platform employs advanced Optical Character Recognition (OCR) technology to enhance document processing capabilities. The OCR technology is designed to extract text from a variety of document types, including invoices, receipts, contracts, and identification documents [15]. This process converts scanned documents, images, and PDFs into machine-readable text, significantly reducing the need for manual data entry.

4.1.2 Identity Verification

Klippa's Doc-Horizon platform also offers robust identity verification services. By leveraging machine learning algorithms and advanced image analysis, the platform can authenticate the legitimacy of identification documents such as passports, ID cards, and driver's licenses [14]. This feature is particularly beneficial for businesses requiring secure and reliable identity verification processes.

4.1.3 Human in the loop

In addition to document capture and verification, Doc-Horizon integrates human-in-the-loop (HITL) processes, allowing for manual review and intervention when necessary [13]. This hybrid approach ensures high accuracy and compliance, catering to the nuanced needs of various industries, including finance, healthcare, and logistics.

4.1.4 Flow builder

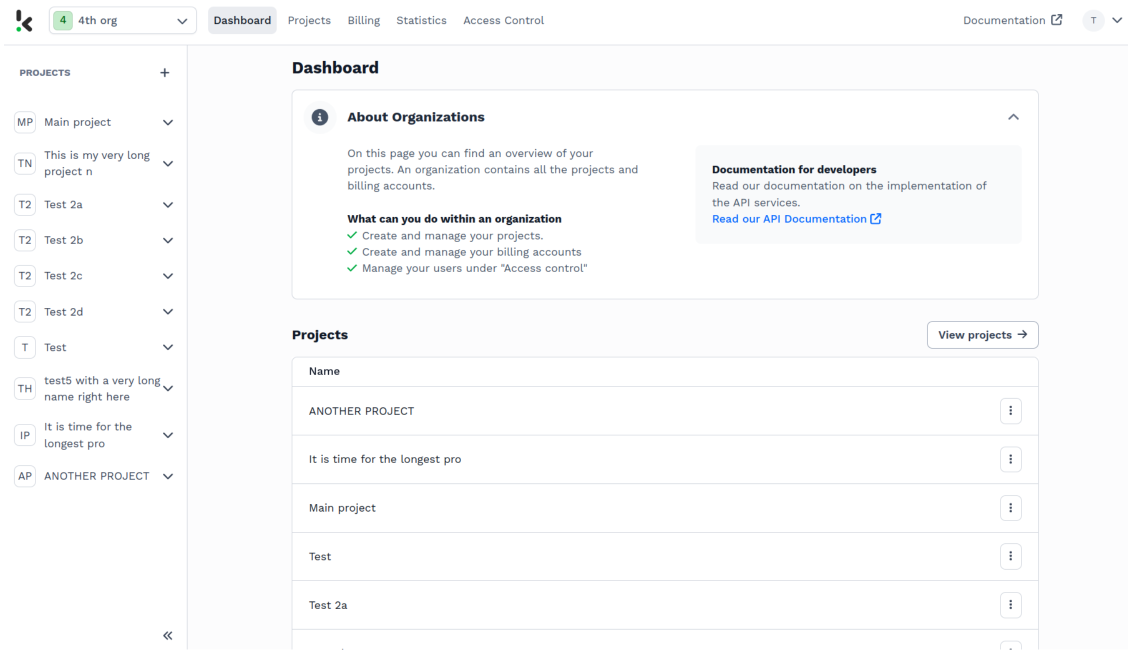
The Flow Builder in Klippa's Doc-Horizon platform enables users to automate their document workflows by creating custom processing flows. This tool allows the configuration of various steps such as document classification, data extraction, validation and integration with other systems [12]. By providing a visual interface for designing workflows, the Flow Builder simplifies complex automation tasks, increasing efficiency and reducing manual intervention.

Overall, Doc-Horizon provides a powerful, versatile solution for modern businesses seeking to optimize their document management processes.

4.2 Platform Overview

In order to interact with the different services, the Klippa Doc-Horizon platform boasts a user-friendly front-end interface, which enables users to manage a variety of projects and organisations, as well as customise the services provided for each individual project. The main sections are as follows:

1. The Dashboard, provides an overview of the user projects for the selected organisation.
2. Projects page, users can manage their projects, create new ones, and access all the functionalities within a project.
3. Billing page where users can manage their billing details, view invoices and select payment plans.
4. Access Control page manages user permissions, ensuring data protection and role-specific access.
5. The Statistics page is where the new statistical interface is implemented, which we will delve into throughout this paper



Dashboard

About Organizations

On this page you can find an overview of your projects. An organization contains all the projects and billing accounts.

What can you do within an organization
✓ Create and manage your projects.
✓ Create and manage your billing accounts
✓ Manage your users under "Access control"

Documentation for developers
Read our documentation on the implementation of the API services.
Read our API Documentation

Projects

Name
ANOTHER PROJECT
It is time for the longest pro
Main project
Test
Test 2a

Figure 2: Doc-Horizon platform

This structured interface enhances the functionality of the Doc-Horizon platform, making it a powerful tool for optimizing document management and workflow automation.

4.2.1 Organisation & Projects

The Doc-Horizon platform is structured around the concepts of organisations and projects. Each organisation can have multiple projects, and various services are applied at the project level. Users can belong to different organisations and manage multiple projects within each organisation. Each project utilises services uniquely, reflecting the specific needs and workflows of the project.

This framework of organisations and projects is crucial for understanding the implementation of the statistics page discussed later in this paper, as it allows for detailed and customized data visualization based on specific organisational and project parameters.

4.2.2 Usage Tracking Data

On the statistics page, users can view detailed data about their usage tracking. To fully understand this data, it's important to clarify what it measures.

The term "usage tracking data" is used to describe the process of capturing the amount of engagement with specific services and associated metrics. The data can be filtered by metric codes and services in order to provide context regarding the source of the usage. To illustrate, if a project makes an API call to the identity verification service to verify an ID card, this action is counted as one instance of usage for the Identity Verification service, categorised under the 'per_document' metric code.

The statistics encompass all services offered by Klippa, including:

1. Company Information
2. Document Capturing Services
3. Document Toolkit
4. E-Invoicing
5. Flow Builder
6. Human In The Loop
7. Identity Verification
8. Storage

Users can filter their data using various metric codes, which include:

1. Flow Builder Component Call
2. Number of HITL reviews
3. Total documents processed
4. Extra page processed

5. Page processed
6. Request processed
7. Number of users

4.3 Technology Stack

Having gained an understanding of Klippa's platform, it is now appropriate to provide a brief overview of the technology stack employed by Klippa. The aforementioned technologies include Go for the back-end, Angular and GraphQL for the front-end, and ClickHouse for the database. While other technologies are also included in the entire Klippa technology stack, they are not relevant for this project and thus will not be discussed here. Instead, we will provide an overview of each technology and how they work together.

4.3.1 Back-end: Golang

Go, also known as Golang, is a statically typed, compiled programming language developed by Google. In the Doc-Horizon platform, Go is used to handle back-end services, managing data processing and business logic. Go's strong concurrency support and efficient performance make it ideal for building scalable back-end systems. It enables the collection, aggregation and transformation of usage data, which is then made available to the front-end via GraphQL APIs.

4.3.2 Front-end: Angular and GraphQl

Angular is a front-end framework developed by Google for the developmnet of dynamic web applications. In the Doc-Horizon platform, Angular is used for the implementation of all user interface features. The component-based architecture of Angular enables the development of modular and maintainable code, which is of paramount importance for the creation of interactive elements within the statistical interface. Angular provides the requisite tools and libraries to handle complex data binding, dependency injection, and routing, thereby ensuring a smooth and responsive user experience.

GraphQL is a query language for application programming interfaces (APIs) developed by Facebook. It is designed to retrieve precisely the data that clients require. In the Doc-Horizon platform, GraphQL serves as an intermediary between the front-end and back-end. This enables Angular components to dynamically query the back-end services implemented in Go, retrieving the necessary data for visualization and display. This integration enables the efficient and flexible retrieval of data, tailored to the specific needs of the statistical interface.

4.3.3 Database: Clickhouse

ClickHouse is an open-source columnar database management system designed for online analytical processing (OLAP). It is highly efficient at handling large volumes of data and complex queries, making it ideal for the extensive data processing required by the Doc-Horizon platform. ClickHouse's ability to perform real-time data analysis with high performance and low latency ensures that the statistical interface can provide up-to-date and accurate visualizations of usage data

When I started my internship, I had no prior programming experience with the technologies that were employed in this project. Consequently, the development process proved to be an invaluable learning opportunity, enabling me to gain proficiency in these new technologies. My learning was twofold: it entailed both mastering the theoretical fundamentals requisite for my thesis and applying these principles to develop a functional, real-world product.

Furthermore, I was able to utilise an existing back-end framework, as a schema for the usage tracking data had already been established. The pre-existing structure thus provided a solid foundation for the construction of the necessary queries

5 Requirements

This chapter outlines the essential requirements for the development of the statistical interface for Klippa's DocHorizon platform. It includes the identification of key stakeholders, the definition of high-level goals (epics), and the detailing of user stories that will guide the subsequent development process.

The user stories for the Product Designer stakeholder were initially developed to create the minimum viable product (MVP). Subsequently, user stories for Sales and Customer Success were created based on feedback gathered following the deployment of the MVP.

5.1 Stakeholders

The term 'stakeholder' is used to describe individuals or groups with an interest in the outcome of the project. According to a practical guide on stakeholder analysis [19], it is crucial to understand the needs and expectations of stakeholders to ensure the delivery of a high-quality interface that meets requirements and achieves goals

5.1.1 Klippa

Klippa is the primary organization responsible for the development and integration of the statistical interface. Within Klippa, there are several key stakeholders:

1. **Product Designer:** Provides initial design guidelines and expectations for the statistical interface, ensuring that the design is user-friendly, meets both aesthetic and functional requirements, and is coherent with the rest of the Klippa platform.
2. **Sales Team:** Uses the statistical interface to monitor the utilisation of the Klippa platform by clients who have entered into contractual agreements with the company, as well as those who have not yet done so. This enables the team to ascertain whether a new contract can be offered or to guarantee the viability of the existing contract.
3. **Customer Success Team:** Uses the interface to monitor clients' usage of the platform to ensure they are being successful and to identify any problems that may occur. This helps in providing better support and ensuring client satisfaction.
4. **Doc-Horizon Development Team:** Provide technical feedback on the implementation, performance, and scalability of the interface. It is the responsibility of the team to ensure that the new interface integrates smoothly with existing systems and that technical standards are met.

5.1.2 Commercial Users

Commercial users are Klippa's clients who utilize the DocHorizon platform for various purposes such as document processing, data extraction, and workflow automation. These clients work with large data sets and require tools to effectively manage and understand their data. It is imperative that the statistical interface provides representative data to enhance the business processes of the users.

5.1.3 Bachelor Student

The bachelor student working as an intern developer, my role, is responsible for the implementation of the statistical interface. This role involves learning the required technology stack, developing the minimum viable product (MVP), and conducting iterative improvements based on feedback from other stakeholders. It is the student's responsibility to ensure that the project aligns with the overall goals of Klippa and delivers a functional and user-friendly interface.

5.2 Epics

The main epic for this project is an **Usage Tracking Statistics Interface**. This epic comprises several distinct features, each further divided into individual user stories with specific requirements that must be met for successful completion. Each feature and its corresponding user stories have been assigned a unique identifier to facilitate the connection between them.

The **Usage Tracking Statistics Interface** is comprised of four main components, which together form the interface. The filters section, a general statistics section, the chart container, and the projects table section. These components are integrated and operate in unison to facilitate an effective and user-friendly system for visualising and analysing platform usage data.

5.2.1 Feature: Filters section

This feature enables users to tailor their data views by applying a range of filters, including project, service type, metric, and date range. It also enables the user to modify the displayed data in real time.

User Stories: **US-F-M2**, **US-F-M8**, **US-F-S2**, **US-F-C1**, **US-F-W2**, **US-F-W4**

5.2.2 Feature: General statistics

This feature presents key metrics in a readily accessible card format, offering users a concise overview of pertinent data such as the total number of processed documents or the total number of API calls. The statistics cards are dynamically updated in accordance with the applied filters.

User Stories: **US-F-M3**, **US-F-M7**, **US-F-S7**

5.2.3 Feature: Chart container

This feature provides dynamic visualisations of the data. It supports multiple chart types, such as line and bar charts, which are updated in real-time based on user-selected filters.

The chart container allows users to customise their data view with specific chart options, thereby facilitating the identification of trends and patterns.

User Stories: **US-F-M4**, **US-F-M6**, **US-F-S3**, **US-F-S4**, **US-F-S5**, **US-F-W1**, **US-F-W3**

5.2.4 Feature: Projects table

This feature provides a comprehensive overview of all projects within an organisation, accompanied by pertinent statistical data for each project. By presenting project-specific data in a structured format, the projects table allows users to efficiently compare and evaluate the use of different projects.

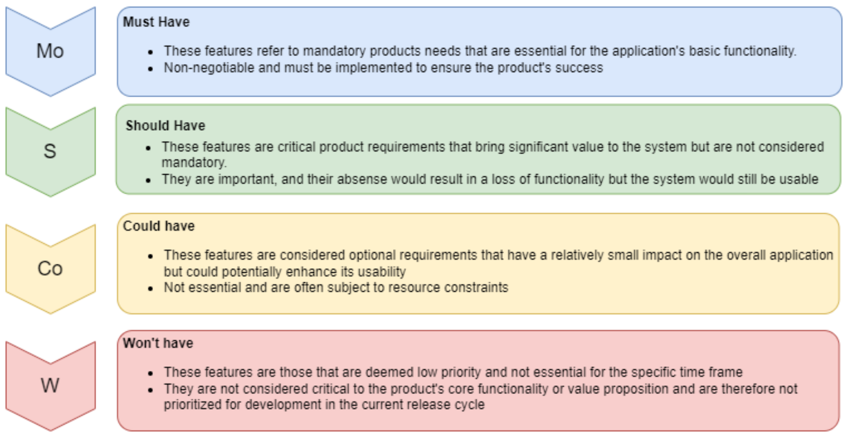
User Stories: **US-F-M5**, **US-F-S6**

5.3 User Stories

We can use the user stories in this project to provide a high-level overview of the features to be implemented, as envisioned by the stakeholders identified earlier in this document. Each user story includes the stakeholder, the specific feature they wish to implement, and an explanation of how this feature contributes to the overall system.

Additionally, each user story outlines testable requirements and acceptance criteria that must be met for the story to be considered complete. These criteria define the specific conditions necessary to demonstrate successful implementation of the feature functionality, ensuring alignment with stakeholder expectations.

To prioritize the implementation of user stories, the MoSCoW (Must-have, Should-have, Could-have, Won't-have) technique is employed. This method classifies stories into four distinct categories, helping to manage scope and focus on delivering essential features first [7]. Each user story is assigned a unique identifier for easy reference and tracking throughout the development process.



Must Have
- These features refer to mandatory products needs that are essential for the application's basic functionality.
- Non-negotiable and must be implemented to ensure the product's success

Should Have
- These features are critical product requirements that bring significant value to the system but are not considered mandatory.
- They are important, and their absense would result in a loss of functionality but the system would still be usable

Could have
- These features are considered optional requirements that have a relatively small impact on the overall application but could potentially enhance its usability
- Not essential and are often subject to resource constraints

Won't have
- These features are those that are deemed low priority and not essential for the specific time frame
- They are not considered critical to the product's core functionality or value proposition and are therefore not prioritized for development in the current release cycle

Figure 3: MoSCoW Prioritization Diagram

To identify user stories, each one is assigned a unique ID, formatted as US-[NF/F]-[M/S/C/W][number], where:

- **US** represents User Story
- **F/NF** indicates whether it is Functional or Non-functional
- **M/S/C/W** stands for Must-have, Should-have, Could-have, or Won't-have
- **number** denotes the specific number within the MoSCoW category

Additionally, requirements are uniquely identified using the format US-[M/S/C/W][numberUS]-RQ-[NF/F][numberRQ], where:

- **US** represents User Story
- **M/S/C/W** indicates Must-have, Should-have, Could-have, or Won't-have
- **numberUS** is the user story number within the MoSCoW category
- **F/NF** denotes whether it is Functional or Non-functional
- **numberRQ** represents the specific requirement number within the category

5.3.1 Must Have

Must-have are those product requirements that are indispensable for the application to perform its fundamental functions. It is imperative that these features are included in the product MVP, as their absence would jeopardise its success. In this project, must-have user stories were part of the requirements of the MVP.

US-F-M1: As a product designer, I need the statistics interface to have its own specific page so that users can easily access and focus on their usage data.

Requirements:

- **US-M1-RQ-F1:** Create a dedicated statistics page accessible from the main navigation.

Acceptance criteria:

- A dedicated statistics page accessible from the main navigation.

US-F-M2: As a product designer, I need a filters section on the statistics page where users can select to filter by project, service, metric, and date range to customize their data view.

Requirements:

- **US-M2-RQ-F1:** Implement filters for project, service, metric, and date range on the statistics page.
- **US-M2-RQ-F2:** Ensure the filters update the displayed data dynamically.

Acceptance criteria:

- Filters for project, service, metric, and date range available on the statistics page.
- Filters should update the displayed data dynamically.

US-F-M3: As a product designer, I need a general statistics section on the statistics page that displays key metrics in card format, such as Total Processed Documents or Total API Calls.

Requirements:

- **US-M3-RQ-F1:** Develop a section with statistics cards displaying key metrics.
- **US-M3-RQ-F2:** Ensure each card shows specific numbers relevant to the users.
- **US-M3-RQ-F3:** Make sure cards are updated based on the applied filters.

Acceptance criteria:

- A section with statistics cards displaying key metrics.
- Each card shows specific numbers relevant to the users.
- Cards update based on the applied filters.

US-F-M4: As a product designer, I need a chart section on the statistics page where usage data is displayed based on the selected filters to provide a visual representation of the data.

Requirements:

- **US-M4-RQ-F1:** Implement a chart section that visualizes usage data.
- **US-M4-RQ-F2:** Ensure charts update dynamically based on selected filters.

Acceptance criteria:

- A chart section that visualizes usage data.
- Charts update dynamically based on selected filters.

US-F-M5: As a product designer, I need a projects table section where an overview of all projects of the organization is displayed, including specific statistics and a button to select a project.

Requirements:

- **US-M5-RQ-F1:** Develop a table listing all projects with relevant statistics.
- **US-M5-RQ-F2:** Include a button to select the project in each row.

Acceptance criteria:

- A table listing all projects with relevant statistics.
- Each row includes a button to select the project.

US-F-M6: As a product designer, I need the chart container to support at least line and bar chart types

Requirements:

- **US-M6-RQ-F1:** Ensure the chart container supports line and bar charts.
- **US-M6-RQ-F2:** Ensure customization options apply changes dynamically.

Acceptance criteria:

- Chart container supports line and bar charts.
- Customization options apply changes dynamically.

US-F-M7: As a product designer, I need the general statistics cards to show a percentage change based on a previous period to provide context.

Requirements:

- **US-M7-RQ-F1:** Include a percentage change from a previous period on each card.
- **US-M7-RQ-F2:** Ensure data and percentage change update based on filters.

Acceptance criteria:

- Each card shows a percentage change from a previous period.
- Data and percentage change update based on filters.

US-F-M8: As a product designer, I need users to be able to combine different filters in the filters section to get specific data tailored to their needs.

Requirements:

- **US-M8-RQ-F1:** Allow users to apply multiple filters simultaneously.
- **US-M8-RQ-F2:** Ensure data displayed reflects the combined filters.

Acceptance criteria:

- Users can apply multiple filters simultaneously.
- Data displayed reflects the combined filters.

5.3.2 Should Have

Should-have represent critical product requirements that bring significant value to the system, yet are not considered mandatory. These features are of significant importance, and their absence would result in a loss of functionality. However, the system would still be usable. In this project, the should-have user stories were developed based on feedback received from the MVP by the sales, customer success, and developer stakeholders.

US-NF-S1: As a Customer Success Manager, I need the interface to be intuitive and user-friendly, with clear naming conventions and filtering options to efficiently access the data I need.

Requirements:

- **US-S1-RQ-NF1:** Have clear naming of filters and statistics
- **US-S1-RQ-NF2:** Include tooltips if necessary in order to provide clarification regarding specific nomenclature or iconography.

Acceptance criteria:

- Clear and descriptive names for all filters and statistics are implemented.
- Tooltips are available for any filters or statistics that may require additional clarification.

US-F-S2: As a Customer Success Manager, I need to be able to select more than one metric or service simultaneously, to facilitate comparisons between different usage patterns between metrics or services.

Requirements:

- **US-S2-RQ-F1:** Implement selection of multiple metrics or services at the same time.

Acceptance criteria:

- Interface allows selection of multiple metrics, services, and projects.

US-F-S3: As a Sales Manager, I need to export platform usage data into different formats such as PNG, CSV, or JPEG to share with various stakeholders or applications.

Requirements:

- **US-S3-RQ-F1:** Implement export functionality for displayed data on the chart container.
- **US-S3-RQ-F2:** Ensure CSV, PNG or JPEG formats include all relevant metrics and data points.

Acceptance criteria:

- Export functionality available for all displayed data on the chart container.
- CSV, PNG or JPEG format includes all relevant metrics and data points.

US-F-S4: As a Sales Manager, I need an option to select the chart interval I want to see to get more granular insights into client usage patterns.

Requirements:

- **US-S4-RQ-F1:** Ensure the possibility of selecting the time interval of the graph from an hourly to a yearly basis
- **US-S4-RQ-F2:** Implement visualizations that support hourly data.

Acceptance criteria:

- Option to aggregate data by interval available.
- Visualizations support and display hourly data.

US-F-S5: As a Sales Manager, I need the option to view more than one project on the chart to compare projects and to enhance sales presentations and pitches.

Requirements:

- **US-S5-RQ-F1:** Implement the option to select and view multiple projects on a single chart with a group by project option.

Acceptance criteria:

- Option to select and view multiple projects on a single chart.

US-F-S6: As a Sales Manager, I need to see which billing accounts are linked to projects to better manage client accounts and understand usage patterns.

Requirements:

- **US-S6-RQ-F1:** Display billing account information linked to each project on projects table.

Acceptance criteria:

- Billing account information is displayed and linked to each project on the projects table.

US-F-S7: As a Sales Manager, I want to see specific meaningful general statistics cards depending on the selected services.

Requirements:

- **US-S7-RQ-F1:** Implement logic to display different general statistics cards based on the selected services.
- **US-S7-RQ-F2:** Ensure the interface dynamically updates the general statistics cards when different services are selected.
- **US-S7-RQ-F3:** Define and display relevant statistics for each service type.

Acceptance criteria:

- General statistics cards update based on the selected services.
- Relevant statistics are displayed for each service type.
- The interface dynamically updates to reflect changes in service selection.

US-F-S8: As a Sales Manager, I need predefined range buttons on the calendar to quickly select usual ranges such as past 30 days or past week to streamline the process of viewing common data ranges.

Requirements:

- **US-S8-RQ-F1:** Implement predefined range buttons for common intervals such as past 7 days, past 30 days, past 90 days, and past year.
- **US-S8-RQ-F2:** Ensure the calendar interface updates dynamically based on the selected range.

Acceptance criteria:

- Predefined range buttons are available and functional on the calendar.
- Data updates correctly according to the selected predefined range.

US-F-S9: As a Developer, I need more efficient and general queries to better handle large datasets to improve performance and reduce load times.

Requirements:

- **US-S9-RQ-F1:** Optimize database queries to handle large datasets more efficiently by using more generalized queries.
- **US-S9-RQ-F2:** Implement batching and pagination for data retrieval to avoid overloading the system.

Acceptance criteria:

- Generalized queries are implemented and improve performance with large datasets.
- Batching is functional and reduces load times significantly.

US-F-S10: As a Developer, I need filter presets such that users can select between predefined presets, and this should be easily extendable to allow custom filter presets in the future.

Requirements:

- **US-S10-RQ-F1:** Implement predefined filter presets for common selections such as all projects, all services, and all metrics.
- **US-S10-RQ-F2:** Design the system to allow easy addition of custom filter presets by users in the future.
- **US-S10-RQ-NF3:** Ensure that the interface for selecting filter presets is intuitive and user-friendly.

Acceptance criteria:

- Predefined filter presets are available and functional.
- The system is designed to easily accommodate custom filter presets in the future.
- Users can easily select and apply filter presets from the interface.

5.3.3 Could Have

Could have are defined as an optional requirement that has a relatively minor impact on the overall functionality of an application, yet has the potential to enhance its usability. In this project, the could-have user stories are based on feedback that was not implemented due to time limitations, but could be considered for future development by Klippa's team. This category is not subject to any requirements or acceptance criteria.

US-F-C1: As a Sales Manager, I need a metric for successful and failed API calls to understand how clients are doing with testing the platform.

5.3.4 Will Not Have

Will not have are features that are considered to be of low priority and not essential for the specific time frame in question, or simply are not feasible currently. In this project, the user stories were initially based on feedback from the Sales and Customer Success stakeholders. However, after consulting with the Developer stakeholder, I determined that these stories were not feasible within the Klippa platform given the available data.

US-F-W1: As a Customer Success Manager, I need to compare actual usage against contract commitments to identify discrepancies and address them proactively.

US-F-W2: As a Customer Success Manager, I need to track detailed engagement metrics such as the number of processed documents and average processing time per document to better understand client usage patterns.

US-F-W3: As a Customer Success Manager, I need visualizations that include a line marking the total usage promised on the contract to quickly assess whether current usage is above or below expectations.

US-F-W4: As a Sales Manager, I need to track accuracy scores versus the number of processed documents to understand client activity on the Klippa platform.

6 Design

This chapter offers a comprehensive overview of the system design and architecture of the statistical interface developed for Klippa's Doc-Horizon platform. It delves into an in-depth analysis of the conceptual framework, modular organisation, and execution flow of the project, with a particular focus on the key components and their interactions.

The chapter is structured into several subsections, each of which provides a detailed account of a specific aspect of the system design. These include an overview of the system at a high level, considerations regarding scalability and maintenance, a description of the user interface, an analysis of the conceptual view, an examination of the module view, and an investigation of the execution view. The objective of this chapter is to provide a comprehensive examination of the system's architecture and functionality, with a particular focus on elucidating the design choices made to ensure an efficient, scalable, and maintainable solution for visualising and analysing usage data.

6.1 High-Level Overview

The front end of the application has been developed using Angular 16 and TypeScript. The codebase is structured into multiple modules, each of which encapsulates code related to a specific service or entity, such as projects or organisations. These modules are structured with separate folders for components, pages, and services, and additional folders for other types of files such as resolvers, modals, or layouts when necessary.

The primary objective of this project was to focus on the usage module, which encompasses the fundamental components, pages, and services related to usage statistics. The usage module, in conjunction with the project and organisation modules, communicates with the back end via GraphQL queries. These queries are integrated within the module services (e.g., `organization-usage.service` and `project-usage.service`) and facilitate data exchange between the front-end and back-end.

The application's back-end is developed using the Go programming language, which is responsible for processing business logic and handling requests from the front end, and the database employed is ClickHouse. While the primary focus of the development process was on the front-end, there were instances when interactions with the back-end were required for the purposes of data seeding and to gain insight into the backend processes, but no new additions to Klippa back-end were necessary.

The modular and scalable architecture of the statistical interface for Klippa's DocHorizon platform ensures that it is well-equipped to handle future enhancements and increased data loads. This provides a robust and efficient solution for visualising and analysing platform usage data.

6.2 User Interface

The user interface of the statistical interface for Klippa's DocHorizon platform has been meticulously designed in accordance with a dashboard approach. This approach is commonly employed in front-end design to create a unified space where users can access a multitude of data and functionalities in a coherent and visually appealing manner.

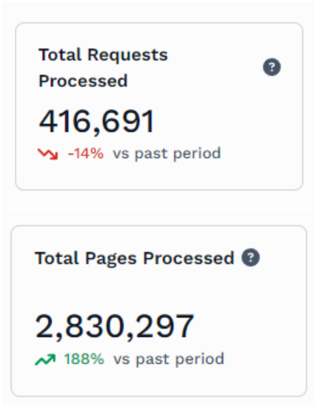
6.2.1 UI/UX Principles

The design of the interface was guided by the product designer stakeholder who provided designs on Figma, as explained in earlier chapters, and several key UI/UX principles aimed at enhancing usability and user experience. The layout is uncluttered and straightforward, facilitating the navigation of different sections and the retrieval of pertinent information. The design elements, including colour schemes, typography, and spacing, were unified to ensure a consistent and seamless user experience. Furthermore, responsive design principles were employed to guarantee that the interface is accessible and functional across a range of devices and screen sizes.

6.2.2 Component Layout

As previously stated in chapter 4.2, the interface is comprised of four principal components, each fulfilling a distinct function:

- **Filters Section:** This section allows users to customize their data views by applying a variety of filters, including project, service type, metric, and date range. The filters facilitate the real-time updating of displayed data.
- **General Statistics:** Key metrics are presented in a readily accessible card format, offering users a concise overview of pertinent data such as the total number of processed documents or API calls. These statistics cards are dynamically updated in accordance with the applied filters.
- **Chart Container:** This section provides dynamic visualisations of the data, supporting line and bar charts. The chart container allows users to customise their data view with specific chart options, thereby facilitating the identification of trends and patterns.
- **Projects Table:** This section provides a comprehensive overview of all projects within an organisation, accompanied by pertinent statistical data for each project. The projects table allows users to efficiently compare and evaluate the usage of different projects.


Total Requests Processed

416,691

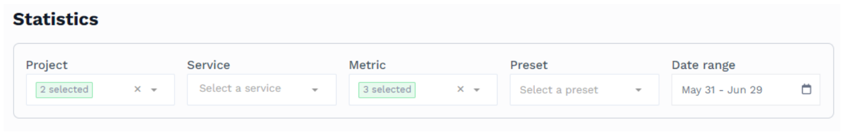
-14% vs past period

Total Pages Processed

2,830,297

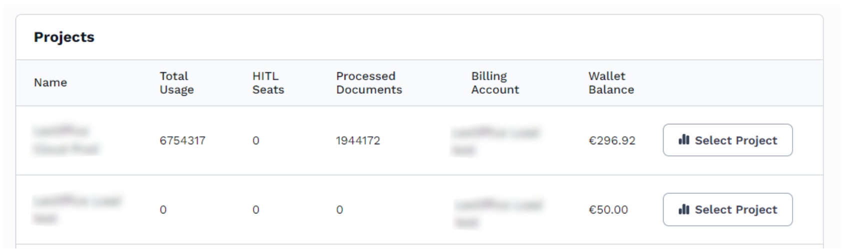
188% vs past period

Figure 4: General statistics cards


Statistics

Project	Service	Metric	Preset	Date range
2 selected	Select a service	3 selected	Select a preset	May 31 - Jun 29

Figure 5: Filters section


Projects

Name	Total Usage	HITL Seats	Processed Documents	Billing Account	Wallet Balance	
[redacted]	6754317	0	1944172	[redacted]	€296.92	Select Project
[redacted]	0	0	0	[redacted]	€50.00	Select Project

Figure 6: Project's table

6.2.3 Interactivity Features

The interface incorporates several interactive features to enhance user engagement and data exploration. These include:

- **Tooltips:** Provide additional information when users hover over data points in the charts, or in the general statistics cards.
- **Dropdown Menus:** Allow users to select different chart types, data intervals, and grouping options.
- **Zooming:** Users can zoom into specific areas of the chart by selecting a range with their mouse.
- **Export Functionality:** Enables users to export the displayed data in various formats.
- **Filter Interaction:** Users can interact with filters to dynamically update the data displayed in the charts and tables.
- **Project Selection Button:** Allows users to select a project from the projects table and view data for a specific project in the chart.

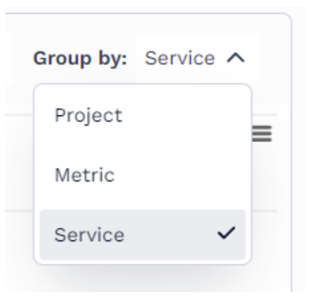


Figure 7: Chart dropdown menu

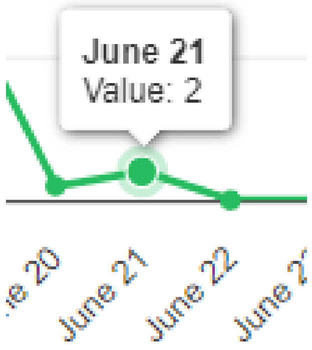


Figure 8: Chart tooltip

6.2.4 Integration with Highcharts

Highcharts was integrated into the Angular components by importing the relevant libraries and utilizing the `Highcharts.chart()` function within the `usage-chart-container`. This function takes in all necessary data and configuration options and renders the chart within the specified HTML element. This integration allows for seamless visualisation of data with extensive customization options, ensuring that the charts meet the specific requirements of the interface and can easily be extended to support even more functionalities.

6.3 Conceptual View

The conceptual view provides a high-level overview of the system's architecture and functionality. It features a diagram illustrating the major components of the system and their interactions. Additionally, it outlines the system's key features and capabilities, explaining how they address the user's needs. In AngularJS and in my project, services, components, and pages play an important role in the MVC architecture [9]:

- **Services**: Services in Angular are used to encapsulate reusable business logic and data manipulation functions. They provide a way to share data and functionality across different parts of an application. Services fit into the MVC architecture by serving as the model or data layer, handling data manipulation and business logic separately from the view and controller.

- **Components**: Components in Angular are reusable UI elements that get data from the services and display it. They promote code reusability and maintainability by breaking down the user interface into smaller, self-contained units. Components fit into the MVC architecture by representing the view aspects of the application, encapsulating the presentation logic and user interface elements.

- **Pages**: In Angular, pages are typically represented by different routes or states within the application. Each page corresponds to a specific view and often incorporates substantial controller logic. Pages fit into the MVC architecture by representing the controller aspect, handling the business logic, user interactions, and interactions with the model.

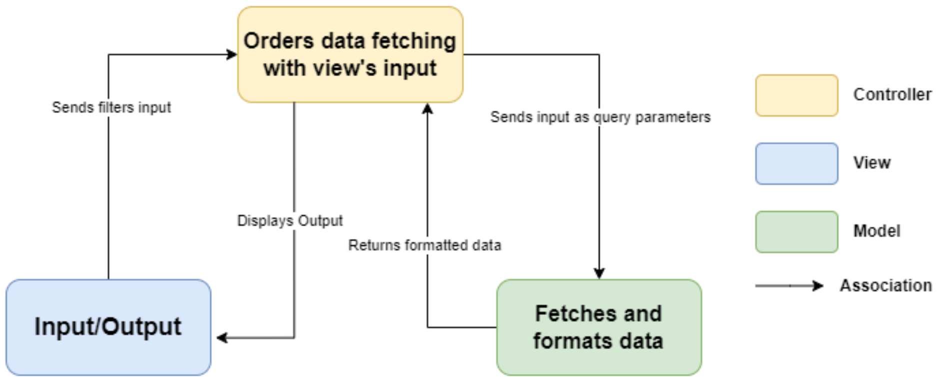


Figure 9: Conceptual View MVC Architecture

6.3.1 MVC Architecture

In this project, as part of the Klippa codebase, the Model-View-Controller (MVC) architectural pattern was adopted to ensure a clear separation of concerns and enhance maintainability.

The **Model** comprises the services such as `usage.service`, `transform-data.service`, `project-usage.service`, and `organization-usage.service`. The aforementioned services are responsible for the execution of business logic, the retrieval of data from the backend via GraphQL, and the transformation of data.

The **View** is implemented through Angular components and the html files of the pages, including `usage-stat-item`, `organization-usage.page.html`, `usage-chart-container`, and other related components. These elements are responsible for the rendering of the user interface and the presentation of data to the user.

The **Controller** aspect is embodied in the TypeScript files associated with the pages, particularly `organization-usage.page.ts`, which manage user interactions, invoke services to fetch and process data, and update the view accordingly.

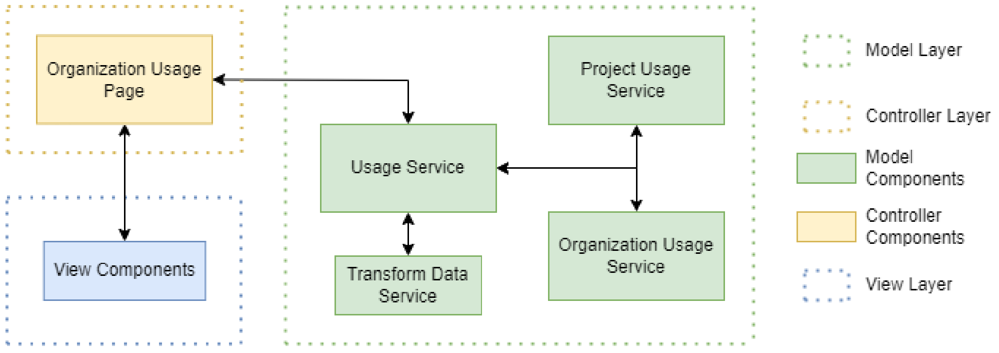


Figure 10: Conceptual Model Diagram

By structuring the project in this manner, we ensure that each layer of the application is dedicated to a specific aspect of functionality, thereby promoting modularity and facilitating easier debugging and enhancements.

6.4 Module View

In the Klippa codebase, the project is structured into modules based on functionalities or services. This organisational structure facilitates the management of the codebase's complexity by grouping related functionalities together. My project primarily involved working within the `usage` module, with occasional interactions with the `project` and `organization` modules. It is important to note that this overview is simplified due to confidentiality constraints. Consequently, there are many other components and interfaces within the Klippa codebase that I utilised but cannot specify.

6.4.1 Usage Module

The `usage` module is the primary focus of my work. It includes several key components and services that are integral to the project's functionality:

Pages:

- `organization-usage.page`: This is the main page where the logic for fetching and rendering usage data is implemented. The system handles user interactions, initiates the fetching of data from external services, processes the data, and updates the view.

Components:

- `usage-chart-container`: Responsible for rendering usage data in a chart format.
- `usage-stat-item`: Displays key statistical metrics. The `organization-usage.page` contains more than one statistical item, which, depending on the selected services, may vary in number and may be up to four in total. Collectively, these items constitute the general statistics section.
- `date-range-picker`: Allows users to select a date range for filtering data. One-fifth of the filters which collectively constitute the filters section.
- `project-picker`: Enables users to select a specific project. One-fifth of the filters that together make up the filters section.
- `project-service-picker`: Allows users to select a specific service within a project. One out of five filters within the filters section.
- `usage-metric-code-picker`: Enables selection of specific usage metrics. One-fifth of the filters that comprise the filters section.
- `preset-picker`: Allows users to choose predefined filters combinations. One-fifth of the filters that constitute the entire filters section.
- `usage-projects-table`: Displays a table of projects with their respective usage data.
- `usage-dropdown-button`: Provides dropdown functionality for various UI elements within the chart container.

Services:

- `usage.service`: A centralised service that coordinates the fetching and processing of data. The system calls upon the services of the project or organisation in question when required, and redirects the query response data to the `transform-data-service` in order to ensure that the data is correctly formatted and ready to be rendered into a chart.

- `transform-data.service`: Handles data transformation and formatting tasks to ensure the data is in the correct format for display.

6.4.2 Project and Organization Modules

While my primary focus was the `usage` module, we also interacted with services in the `project` and `organization` modules. These modules include their own sets of components and services, but we specifically used:

Services:

- `project-usage.service`: Fetches usage data specific to projects.
- `organization-usage.service`: Fetches usage data specific to organizations.

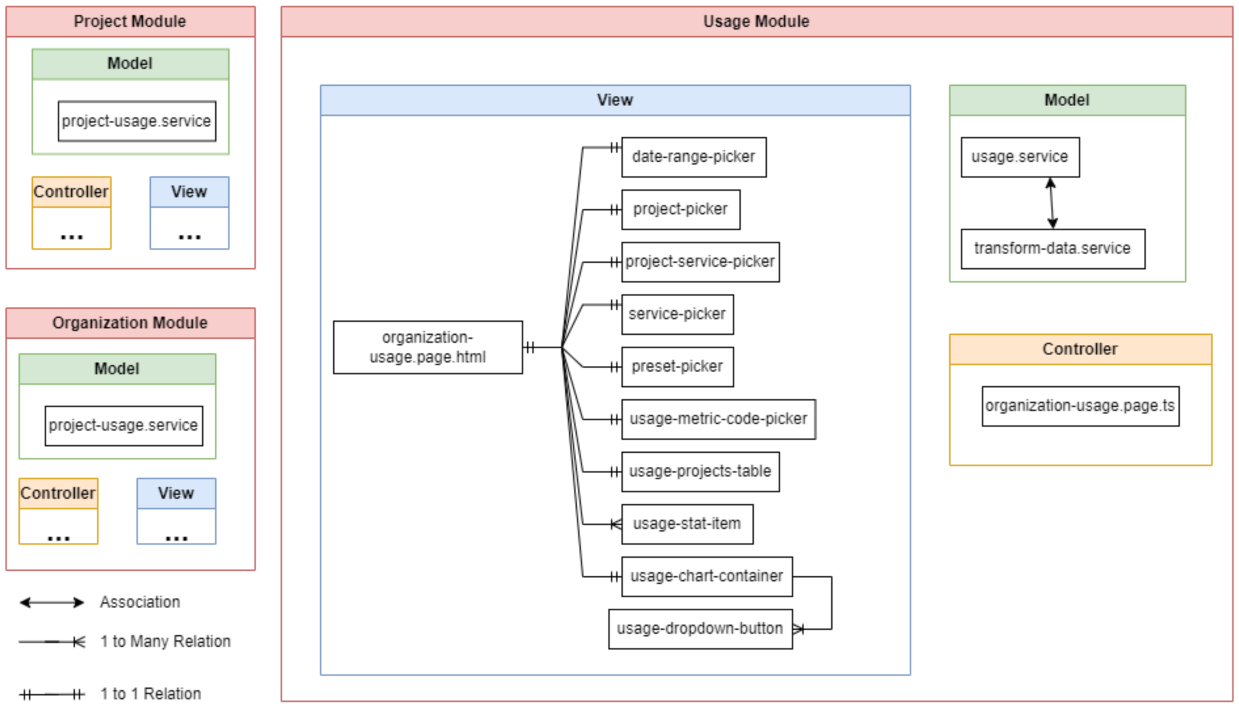


Figure 11: Module View Diagram

In this modular structure, the `organization-usage.page.ts` acts as the central point for handling logic and data flow. It fetches data using the `usage.service`, which in turn interacts with either the `project-usage.service` or the `organization-usage.service` to retrieve data from the backend. The data is then transformed by the `transform-data.service` before being passed back to the `organization-usage.page.html` for display in the various components.

This modular approach ensures that each functionality is encapsulated within its respective module, promoting code reusability, maintainability, and a clear separation of concerns. The confidentiality constraints mean that while we cannot detail all the components and interfaces used, this overview captures the primary structure and interactions within the Klippa codebase.

6.5 Execution View

The execution view offers a comprehensive explanation of the system's operational dynamics at runtime. The objective is to provide an in-depth analysis of the interactions between the system's components, processes, and resources, with a particular focus on how the system responds to various inputs and events.

The execution flow in the Klippa codebase, particularly within the `usage` module, is initiated by user interactions and proceeds through several stages of data fetching, processing, and presentation. Below is a step-by-step explanation of this execution flow:

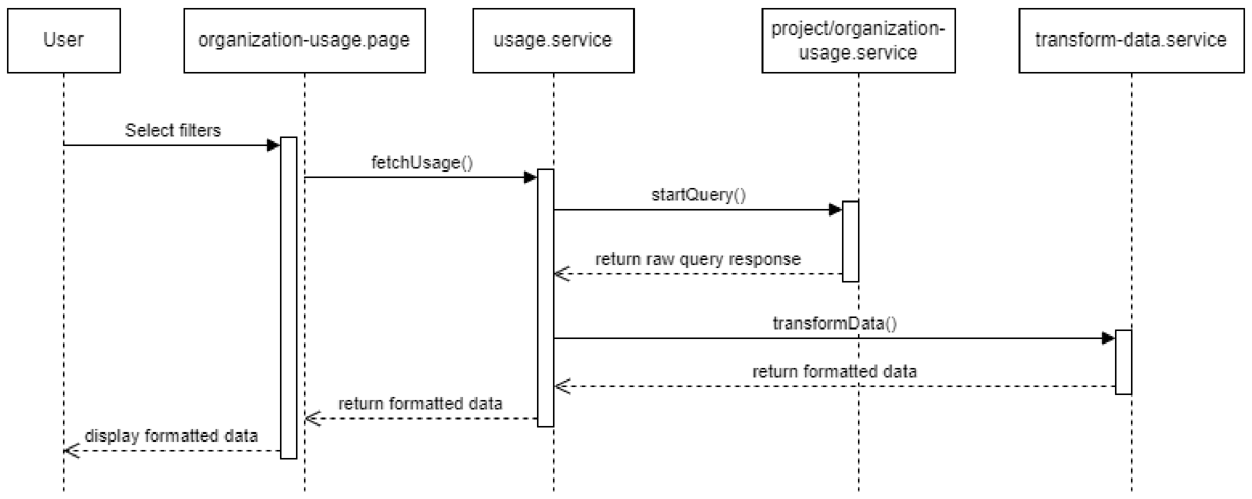


Figure 12: Execution View Diagram

1. **Step 1: User Interaction**
   - The user interacts with the filters section on the `organization-usage.page.html`. They select various filters such as date range, project, service, and usage metrics.
2. **Step 2: Data Fetching**
   - Based on the selected filters, the `organization-usage.page` invokes the `fetchUsage()` function.
   - The `fetchUsage()` function determines whether the data request pertains to a project or an organization. It then calls the appropriate service (`project-usage.service` or `organization-usage.service`) through the `usage.service`.
   - The selected service makes a GraphQL call to the backend, retrieving the required usage data.
3. **Step 3: Data Processing**
   - The retrieved data is passed back to the `usage.service`, which then forwards it to the `transform-data.service`.
   - The `transform-data.service` formats and processes the raw data, structuring it into a format suitable for presentation.
4. **Step 4: Data Presentation**
   - The formatted data is returned to the `organization-usage.page.html`.
   - The `organization-usage.page.html` updates the relevant UI components with the new data. For example, the data is passed to the `usage-chart-container` to render the updated usage charts and to other components like `usage-stat-item` and `usage-projects-table` for displaying key metrics and detailed project data.

This detailed execution flow ensures that the system responds efficiently to user inputs and provides a seamless and interactive user experience. By structuring the execution process in this manner, the application achieves a clear separation of responsibilities, allowing for easier maintenance and scalability.

6.6 Scalability and Maintenance

The system has been designed with scalability and maintenance in mind. Wherever feasible, components from Klippa's existing front-end codebase were reused, and new components were developed in accordance with Klippa's established code standards and guidelines. This guarantees that the newly developed components can be readily scaled or reused for other components within the front-end. Furthermore, all code underwent rigorous code reviews by other front-end developers at Klippa, with the objective of maintaining high standards of code quality and ensuring adherence to best practices.

7 Implementation

The Implementation chapter provides a comprehensive and detailed account of the systematic approach taken to develop the statistical interface for Klippa's Doc-Horizon platform. This chapter will set forth the methodologies adopted, and delineate the phases of the development process.

7.1 Methodologies

To further clarify my role at Klippa and my involvement in this project, I was a member of the Doc-Horizon development team. Although I was involved in the team's development process, participating in team sprints and deployment procedures, I worked independently on this project, with the exception of code reviews, where someone else had to review my code.

The Klippa code base is structured according to a modular approach, wherein each module encompasses a set of components, services, and pages. My principal focus was on the usage module, which encompasses all code related to usage statistics. However, it should be noted that some classes were located in other modules as well.

The Doc-Horizon team employs the use of GitLab for the purposes of version control. The management of tasks is conducted through the use of issue boards, wherein features or tasks are recorded as tickets. A new branch is created for each ticket, thus enabling the isolation of development and the independent review of each feature. Upon completion of a feature, a merge request (MR) would be created with the objective of merging the branch into the primary development branch. Subsequently, the MR would be subjected to a comprehensive code review by other developers. Once all feedback has been addressed and approval granted, the branch is merged into the main development branch. This initiates the process anew with the next ticket.

By following this structured approach, I was able to guarantee a seamless and productive development process, with the continuous integration and delivery of new features while maintaining the highest standards of code quality through the implementation of peer reviews and incremental enhancements. The aforementioned methodology proved instrumental in the successful development of the statistical interface for Klippa's Doc-Horizon platform.

7.2 Development Process

As a member of the Doc-Horizon team, I was integrated into these sprints, during which each sprint had defined goals and deliverables. This iterative approach proved invaluable in facilitating continuous feedback and improvements throughout the development cycle.

During the internship period, I participated in six team sprints. These sprints are detailed as follows:

1. **Sprint 59:** April 15th to April 19th.
2. **Sprint 60:** April 22nd to May 3rd.
3. **Sprint 61:** May 6th to May 17th.
4. **Sprint 62:** May 20th to May 31st.
5. **Sprint 63:** June 3rd to June 14th.
6. **Sprint 64:** June 17th to June 28th.

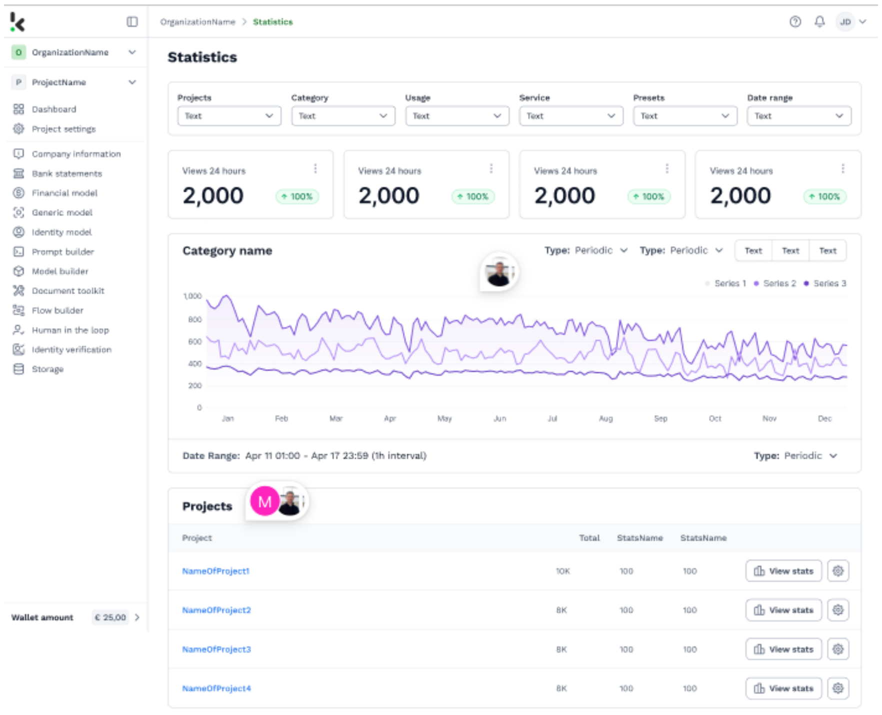
We can further categorize these sprints into three distinct development phases:

1. **Phase 1:** Onboarding, learning the technology stack, and selecting a charting library. This phase encompasses sprints 59 and 60.
2. **Phase 2:** Development of the Minimum Viable Product (MVP). This phase includes sprints 61 and 62.
3. **Phase 3:** Gathering feedback and improving the MVP. This phase consists of sprints 63 and 64.

7.2.1 Phase 1: Onboarding & Technology stack

In the initial phase of the project, the first few days were allocated to the onboarding process at the company. This phase entailed familiarising myself with the development workflow, becoming acquainted with my colleagues, integrating into the team's sprint cycle, and focusing on acquiring foundational knowledge in Angular and GraphQL. Furthermore, it was of the utmost importance to gain a comprehensive understanding of the functionality of the Doc-Horizon platform, including its operational specifics and the particulars of the UsageTracking data being measured.

Once I had established myself at the company and gained an understanding of the fundamentals, I was provided with design guidance for the statistical interface by the Product Designer of Klippa. The design, created in Figma, provided a framework and transparent criteria for the statistical interface, which I used to develop user stories that served as the initial set of requirements for the development of the MVP of the interface (5.3.1). The user stories ensured that the design and functionality were aligned with the product designer's vision and Klippa's expectations. The aforementioned process spanned the entirety of Sprint 59 and a few days from Sprint 60.



Project	Total	StatsName	StatsName	
NameOfProject1	10K	100	100	View stats
NameOfProject2	8K	100	100	View stats
NameOfProject3	8K	100	100	View stats
NameOfProject4	8K	100	100	View stats

Figure 13: Figma Design

The subsequent stage of the process entailed an investigation into the existing typescript libraries for data visualisation. This constituted a part of sprint 60, and the specifics of which will be elucidated in the next section.

7.2.2 Phase 1: Selection of an effective library

A structured and thorough approach was employed to identify the most suitable charting library for the statistical interface. This selection process was of critical importance in order to ensure that the library met both the immediate functional requirements and the long-term scalability requirements for Klippa's Doc-Horizon platform. In order to ascertain the specific requirements of the charting library, interviews were conducted with 2 front-end developers from the Doc-Horizon team, which led to the following requirements.

- **License:** The developers are willing to utilise libraries with a paid commercial licence if the functionality provided justifies the cost of the licence.
- **Software source:** The library should be accessible for download via Yarn, a package manager for JavaScript packages, and should ideally have no complex external dependencies.
- **Compatibility:** The Klippa environment is an Angular 16 application, and thus the library must be compatible with this version of Angular.
- **Stability:** It is imperative that the library be a stable, production-ready version and not a beta version.

- **Maintenance:** It is recommended that the library is maintained on an ongoing basis, preferably with a large user base and community.
- **Documentation:** It is essential that the documentation is comprehensive and includes illustrative examples and instructions.
- **Graph types:** For the current implementation, line and bar charts are sufficient; however, the possibility of using additional graphs, if necessary, would be beneficial.
- **Interactivity:** It is crucial to be able to interact with the chart in an effective manner, for instance by accessing tooltips over data points or using zooming functionalities.

Based on the requirements provided by the developers, a selection process was conducted in accordance with the multi-phase approach outlined by Jadhav and Sonar [8], which comprises the following steps:

Step 1, Determining the Need: According to the interviews conducted and the nature of my project, the need of this library is to provide dynamic and interactive visualizations from large datasets within the proposed interface of this project, the platform usage statistics.

Step 2, Shortlisting Candidates: To ensure that the most appropriate charting library was selected for the project, an extensive search was conducted for potential candidates. The search primarily involved researching popular Angular charting libraries through various sources such as Google searches, developer forums, and recommendations from Doc-Horizon developers. Key criteria for the shortlist included compatibility with Angular 16, availability via Yarn, strong community support and alignment with the identified requirements. After thorough research, the following charting libraries were shortlisted:

1. C3
2. Chart.js
3. Chartist
4. Dc.js
5. HighCharts
6. ApexCharts.js
7. amCharts
8. AnyChart

Step 3, Eliminating Unsuitable Options: Based on the project's specific requirements, three charting libraries were eliminated. C3 was discarded due to limited support for Angular and less active maintenance, last updated 7 years ago. Chartist was excluded due to its limited interactivity features and infrequent updates, with the last update being 2 years ago. Finally, Dc.js was also excluded due to its smaller community and basic interactivity features, and also due to infrequent updates, with the last update being 3 years ago. The remaining candidates, which met all the necessary criteria, were:

1. Chart.js
2. HighCharts
3. ApexCharts.js
4. amCharts
5. AnyChart

Step 4, Evaluating remaining packages: After shortlisting candidates, I had to find the most suitable libraries among the remaining 5 candidates. Since all candidates at this point met the requirements, I evaluated the remaining candidates based on the number of downloads in the past year. This data was obtained from the npmtrends.com website [18].

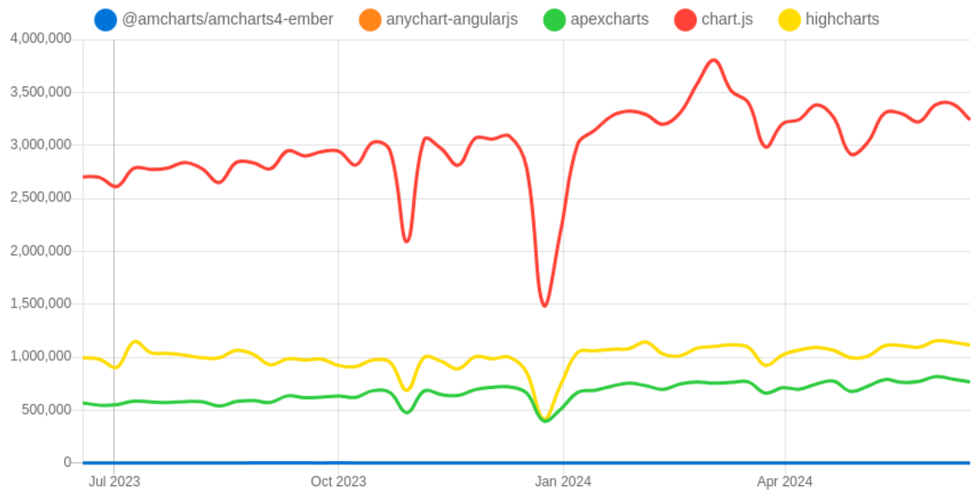


Figure 14: Downloads over the past year

Looking at the chart, it became evident that only Chart.js, and HighCharts, warranted further consideration. This conclusion is based on their significantly higher monthly download figures, with Chart.js averaging approximately 3,000,000 downloads, and HighCharts around 1,000,000 downloads.

In contrast, all other candidates had monthly downloads well below the 500,000 threshold, indicating relatively lower usage and relevance. Despite ApexCharts having approximately 500,000 downloads, ranking third, it was also eliminated from consideration due to the already promising candidates of Chart.js and HighCharts.

Step 5, Trial use: In order to further evaluate the suitability of the remaining candidates, namely HighCharts and Chart.js, a prototype integration was conducted. This stage of the evaluation process involved integrating each library into a prototype Angular 16 application. The objective was to assess the ease of integration, the quality of the documentation, and the level of community support. The process was as follows:

- **HighCharts:** The incorporation of HighCharts into the Angular 16 application was accomplished with smooth integration. The documentation offered comprehensive guidance and illustrative examples, including a complete library showcasing each available chart type, which greatly facilitated a straightforward integration process. HighCharts extensive configuration options allowed for a high degree of customisation and adaptability to the project's requirements. One minor issue was encountered during the integration, but a quick search on Google led to an immediate solution on Stack Overflow, demonstrating the strong community support. Overall, HighCharts offered an seamless integration with a rich set of features.
- **Chart.js:** The integration of Chart.js was relatively straightforward due to its well-documented guides and examples. However, during the integration process, it became apparent that Chart.js required more manual configuration and customization to achieve the same level of functionality and visual appeal as HighCharts. Additionally, while Chart.js offers a wide range of basic charts, it lacks some advanced features and built-in interactivity options that HighCharts provides out of the box. This increased the development time and complexity, making it less efficient for rapid prototyping and deployment.

While both libraries demonstrated strong capabilities and ease of integration, HighCharts ultimately proved to be a better choice for this project. The deciding factors were its extensive feature set and documentation, together with a strong community support. Furthermore, both candidates were presented to two Doc-Horizon front-end developers, who both concurred with the aforementioned assessment. Consequently, Chart.js was eliminated from further consideration, and HighCharts was selected for final integration into the project.

Step 6, Negotiating contracts: As Highcharts requires a paid licence for commercial use, I sought and received confirmation from my supervisor that this licence was acceptable.

Step 7, Purchasing and Implementing: I initiated the integration of the library into Klippa's code, and my supervisor subsequently acquired the necessary license. With the library selected and approved, and the MVP requirements defined, I was ready to begin coding, which I started in the final days of sprint 60.

7.2.3 Phase 2: MVP

Phase 2 encompassed two sprints, Sprint 61 and Sprint 62, which were dedicated to the implementation of all the user stories identified as "Must Have" in order to produce the MVP. The aforementioned MVP was subsequently deployed to the Klippa development environment on May 31st. Throughout this phase, I worked on a local branch, `feature/statistics-mvp`, which served as my primary development branch. In order to implement the various features, new branches were created from the main branch and subsequently merged back once the feature development was complete. Each merge request I made was subjected to a code review by a colleague, who was responsible for providing feedback and approving the merge.

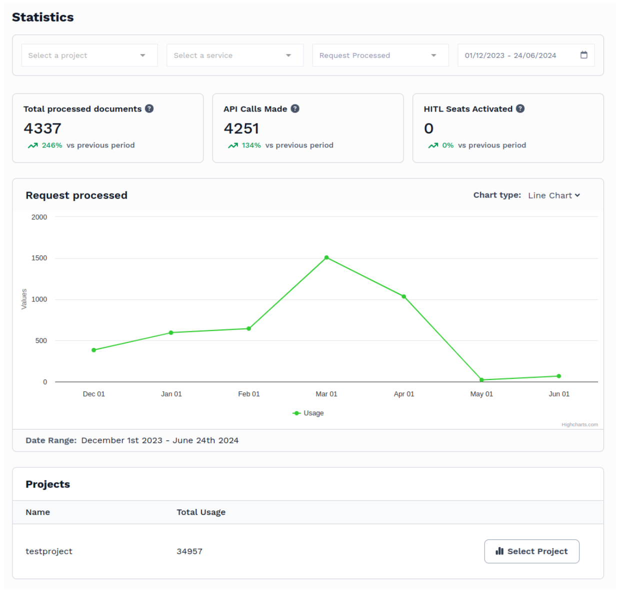
Firstly, new GraphQL queries were constructed in order to retrieve the usage data for both organisations and projects. At this stage of the process, a single query was constructed for each potential combination of filters, resulting in three queries for the project and three for the organisation.

- `GetUsageTrackingByProject`: A general query pertaining to the usage of data across projects, without the application of any filters.
- `GetUsageTrackingByMetricCodeByProject`: This query was designed to filter usage data based on a single selectable metric code.
- `GetUsageTrackingByMetricCodeByServiceTypeByProject`: Designed to filter one metric code and one service type
- `GetUsageTrackingByOrganization`: Identical to the aforementioned project query, but at the organisational level.
- `GetUsageTrackingByMetricCodeByOrganization`: Identical to the aforementioned project query, but at the organisational level.
- `GetUsageTrackingByMetricCodeByServiceTypeByOrganization`: Identical to the aforementioned project query, but at the organisational level.

The subsequent stage of the process involved the implementation of the "Must have" User Stories (5.3.1), in order to initiate the development of the various components that collectively constituted the new page in the Doc-Horizon platform, namely the statistics page, which satisfied the user story **US-F-M1**. The remaining user stories were satisfied by the following components:

- `usage-chart-container`: **US-F-M4**, **US-F-M6**
- `usage-stat-item`: **US-F-M3**, **US-F-M7**
- Filters pickers: `date-range-picker, project-picker , project-service-picker, usage-metric-code-picker`. Which together formed the filters section and satisfied **US-F-M2**, **US-F-M8**
- `usage-projects-table`: **US-F-M5**

This phase constituted a rigorous examination of the knowledge acquired during phase 1, requiring the application of Angular and GraphQL fundamentals within the extensive Klippa codebase. At the outset, identifying the appropriate modules and files for coding, as well as ensuring the code's reusability and coherence with Klippa's existing codebase, presented considerable challenges. It took me a few days to become confident in my ability to run and debug the code without encountering multiple errors at each line of code. Concurrently, I encountered numerous new concepts, necessitating the use of online resources, such as Google, to gain a deeper understanding of specific Angular rules, functions, and syntax. This approach, while demanding, proved an excellent method for consolidating foundational knowledge and acquiring additional practical skills.



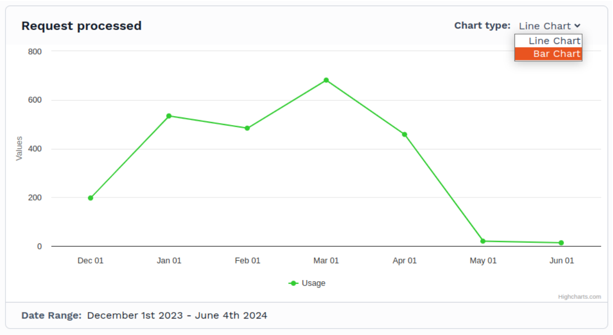
Total processed documents	API Calls Made	HITL Seats Activated
4337	4251	0
246% vs previous period	134% vs previous period	0% vs previous period

Date Range: December 1st 2023 - June 24th 2024

Projects

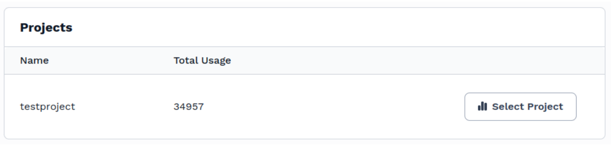
Name	Total Usage
testproject	34957

Figure 15: MVP General overview



Date Range: December 1st 2023 - June 4th 2024

Figure 16: MVP Chart Container



Projects

Name	Total Usage
testproject	34957

Figure 17: MVP Projects Table

7.2.4 Phase 3: Improvements

Upon the successful deployment of the MVP to the Doc-Horizon platform, the primary objective of Sprint 63 was to gather feedback from key stakeholders and incorporate this feedback into subsequent iterations of the interface. After conducting the interviews, sprint 63 and 64 served for implementing the received feedback and finalizing the interface.

Sprint 63: In order to achieve this objective, I began the Sprint 63 by conducting interviews with representatives from the Sales and Customer Success teams, both of whom are integral to the platform's usage and client interaction.

The Customer Success team emphasized the importance of improvements in the interface's clarity and functionality to better track client usage patterns, which led to the creation of user story **US-NF-S1**. Additionally, they stressed the need for filters that allow the selection of multiple metrics and services simultaneously, facilitating comparison between metrics and services, leading to the creation of user story **US-F-S2**. These enhancements aimed to provide a clearer and more detailed understanding of client activities on the platform, thereby improving the user experience and the interface's overall effectiveness.

The Sales team provided valuable input regarding the potential for identifying sales opportunities and understanding client behavior through usage metrics. They highlighted the need for metrics that enable a more granular understanding of client activities, such as hourly aggregation options, leading to the creation of user story **US-F-S4**. They also suggested the inclusion of metrics for successful and failed API calls to better understand client testing activities, which was addressed in user story **US-F-C1**. Additionally, they proposed features that allow viewing multiple projects simultaneously on the chart, enabling a comparative analysis of different clients' usage patterns, reflected in user story **US-F-S5**. Finally, they suggested including billing account information on the projects table to better manage client accounts, which resulted in user story **US-F-S6**.

The feedback obtained from these interviews served as a direct catalyst for the creation of new user stories and subsequent enhancements to the MVP. The user stories were then classified according to their priority and feasibility within the remaining project timeframe and available resources. This process was a pivotal point in the process of refining the interface to better align with the needs of the stakeholders and enhance its overall functionality.

With a new user stories at hand, I initiated the implementation of new functionalities. During the 63rd sprint, I implemented the following:

1. In accordance with **US-F-S8**, the date range component was enhanced to include buttons for predefined ranges, thereby improving user convenience and efficiency in selecting date intervals.

2. As specified in **US-F-S4**, an interval chart option was implemented, allowing users to select the interval by hour, day, month, or year, which facilitates a more granular analysis of usage data.

3. Following **US-F-S6**, additional metrics were incorporated into the project table, including billing account information, wallet balance, processed documents, HITL seats, and total usage, providing a comprehensive overview of project-specific data.

4. In response to **US-F-S5** and **US-F-S2**, a 'group by' chart option was developed, enabling users to select and display multiple projects, metrics, and services simultaneously, and group them within the chart by projects, metrics, and services.

5. In alignment with **US-F-S9**, the efficiency of data queries was significantly improved by reducing the number of queries to two general ones (one for projects and one for the organization), and by grouping the fetched data on the backend to minimize the number of data points retrieved.

Sprint 64: The primary objective of Sprint 64 was to finalize the project by implementing the remaining ”Should Have” user stories based on the feedback gathered during Sprint 63. This sprint also involved refining the interface and ensuring that all functionalities were seamlessly integrated.

During Sprint 64, I focused on the following implementations:

1. In line with **US-F-S7**, general statistics cards were dynamically linked to the selected service, ensuring that the statistics displayed are relevant to the user's current selection.
2. To enhance user convenience, **US-F-S10** involved adding presets to the filters, allowing users to quickly apply commonly used filter configurations.
3. As specified in **US-F-S3**, export options were added to the chart, enabling users to export the displayed data in various formats, thereby facilitating data sharing and analysis.
4. In accordance with **US-F-S1**, variable names were revised to ensure clear and consistent naming conventions throughout the interface, improving overall readability and maintainability of the code.

With the completion of these tasks, the interface was significantly improved in terms of functionality, usability, and clarity. This sprint marked the final phase of development and the final week of my internship, during which I also began drafting the thesis, documenting the entire development process, and reflecting on the project's outcomes and lessons learned.

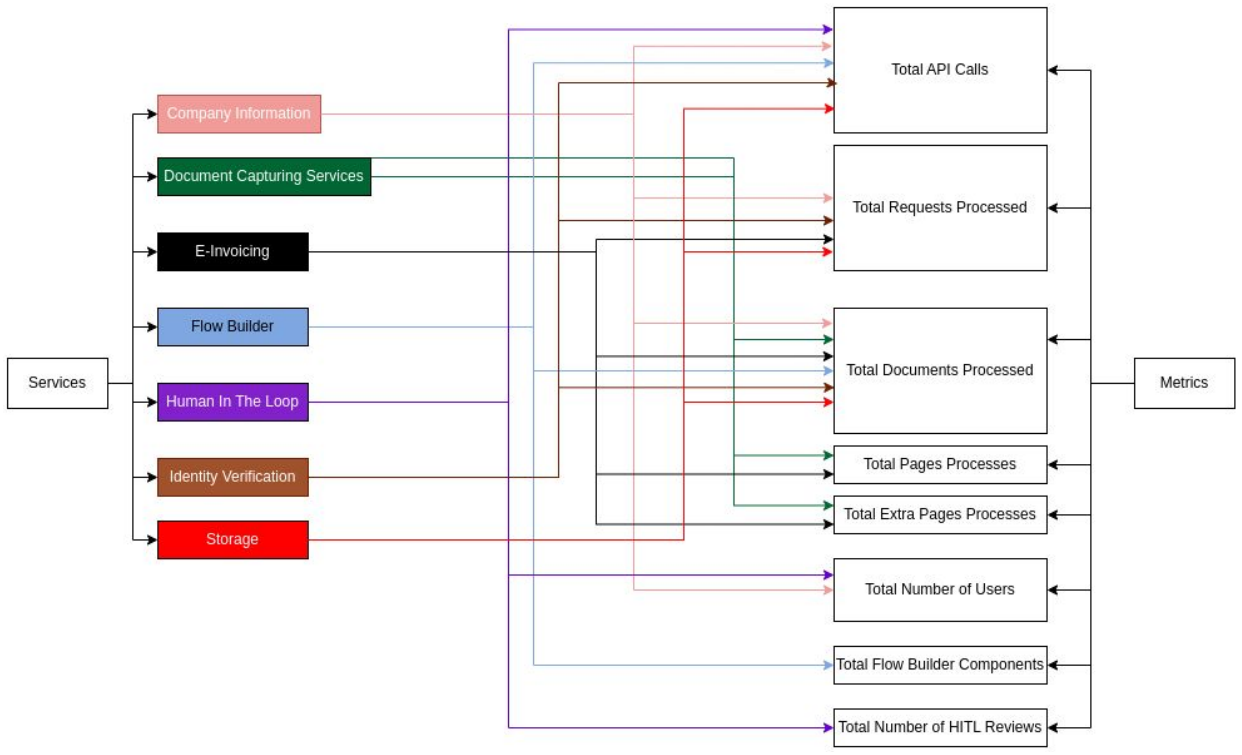
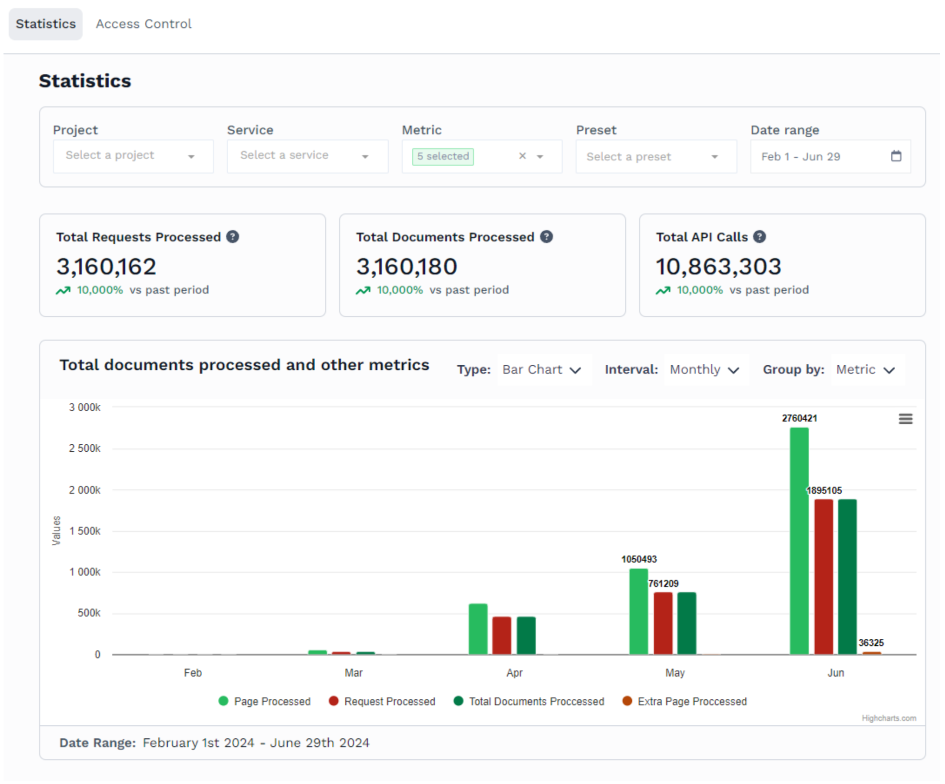


Figure 18: **US-F-S7,** Connection between statistic cards displayed and selected services



Statistics Access Control

Statistics

Project	Service	Metric	Preset	Date range
Select a project	Select a service	5 selected	Select a preset	Feb 1 - Jun 29

Total Requests Processed
3,160,162
10,000% vs past period

Total Documents Processed
3,160,180
10,000% vs past period

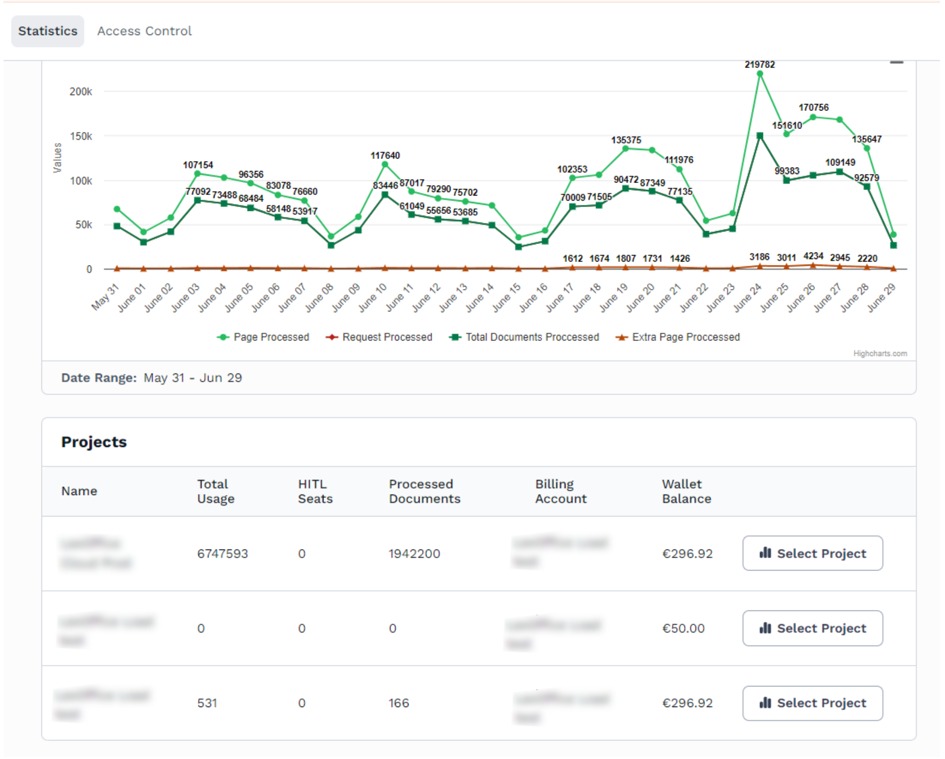
Total API Calls
10,863,303
10,000% vs past period

Total documents processed and other metrics Type: Bar Chart Interval: Monthly Group by: Metric

● Page Processed ● Request Processed ● Total Documents Proccessed ● Extra Page Proccessed

Date Range: February 1st 2024 - June 29th 2024

Figure 19: Final product with bar chart



Statistics Access Control

Page Processed — Request Processed — Total Documents Proccessed — Extra Page Proccessed

Date Range: May 31 - Jun 29

Projects

Name	Total Usage	HITL Seats	Processed Documents	Billing Account	Wallet Balance	
[redacted]	6747593	0	1942200	[redacted]	€296.92	Select Project
[redacted]	0	0	0	[redacted]	€50.00	Select Project
[redacted]	531	0	166	[redacted]	€296.92	Select Project

Figure 20: Final Product with line chart and projects table

8 Results, Discussion and Testing

Following the conclusion of my 11-week internship, the final product was successfully deployed onto the Klippa platform. It is now imperative to evaluate the project's quality and performance. Although my supervisor and the stakeholders at Klippa have expressed satisfaction with the product, it is essential to substantiate this feedback with concrete metrics. This chapter will assess the project's quality and performance independently to determine whether the stakeholders' verbal satisfaction accurately reflects the project's success.

8.1 Software Quality

According to Galin [5], Software quality can be defined as the extent to which a software product meets the established requirements and accurately represents the needs, wants, and expectations of the stakeholders. It encompasses the fulfilment of functional and technical requirements, adherence to coding standards, usability, reliability, performance, and security aspects.

Software Quality Assurance (SQA) activities play a pivotal role in ensuring software quality by meticulously planning and implementing processes that define and assess the adequacy of software development processes. The activities of SQA, including requirements analysis, testing, code reviews, and process audits, facilitate the identification and resolution of potential issues, the minimisation of software defects, and the enhancement of overall product quality [5].

Based on these definitions, I can evaluate this project quality by conducting the following SQA activities:

1. Requirements analysis
2. Code reviews & Testing
3. Stakeholder satisfaction evaluation

8.1.1 Requirements analysis

The process of SQA for requirements analysis, consisted in first reviewing all user stories to ensure clarity, completeness, consistency, and alignment with stakeholder needs. This was conducted by verifying that each user story is well-defined, specific, and covers a single functionality essential for the software product.

Once I had reviewed all user stories, it was time to ensure that all requirements and acceptance criteria were met. As I detailed in chapter 5, all requirements were met by the different components of the interface. Below is a detailed overview which components fulfilled the requirements.

- **Statistics page:** US-M1-RQ-F1, US-S1-RQ-F1, US-S1-RQ-F2

- **Filter's section:** US-M2-RQ-F1, US-M2-RQ-F2, US-M8-RQ-F1, US-M8-RQ-F2, US-S2-RQ-F1, US-S8-RQ-F1, US-S8-RQ-F2, US-S10-RQ-F1, US-S10-RQ-F2, US-10-RQ-NF2
- **General Statistics:** US-M3-RQ-F1, US-M3-RQ-F2, US-M3-RQ-F3, US-M7-RQ-F1, US-M7-RQ-F2, US-S7-RQ-F1, US-S7-RQ-F2, US-S7-RQ-F2
- **Chart container:** US-M4-RQ-F1, US-M4-RQ-F2, US-M6-RQ-F1, US-M6-RQ-F2, US-S3-RQ-F1, US-S3-RQ-F2, US-S4-RQ-F1, US-S4-RQ-F2, US-S5-RQ-F1, US-S5-RQ-F2
- **Project's table:** US-M5-RQ-F1, US-M5-RQ-F2, US-S6-RQ-F1

8.1.2 Code reviews & Testing

In the context of code review and testing, the SQA activities at Klippa involved a process where my merge requests underwent meticulous reviews by other front-end developers, as outlined in the methodology section. Each line of code was subjected to a meticulous examination to ascertain its compliance with quality standards, the absence of errors in its implementation, its maintainability, and its extensibility.

The reviewers played a pivotal role in ensuring that only the highest quality code was integrated into the codebase. Furthermore, the code review process at Klippa was meticulously aligned with the company's coding guidelines, emphasising consistency, best practices, and the overall enhancement of software quality.

This process ensured that all my merge requests to the main development branch met the requisite code quality standards. In the event that this was not the case, I would be provided with feedback so that I could adapt my code and ensure its correctness.

8.1.3 Stakeholder satisfaction evaluation

The concept of stakeholder satisfaction can be defined as the level of approval or contentment that stakeholders have regarding the project's outcomes and processes. In the context of this project, stakeholder satisfaction is intrinsically linked to the fulfilment of the defined user stories, which encapsulate all the requirements and expectations provided by the stakeholders throughout the internship.

The user stories serve as a concrete and measurable representation of stakeholder needs and objectives. By assessing the number and categories of user stories that have been successfully completed, I can gauge the extent to which the project has met these needs. Specifically, during the course of this project, a total of 23 user stories were identified, encompassing all the requirements set forth by the stakeholders. Out of these, 18 user stories were completed, including all the "Must Have" and "Should Have" criteria.

Weighting User Stories: To provide a more accurate measure of stakeholder satisfaction, it is essential to consider the weight of each user story category:

- **Must Have:** Critical requirements that are essential for the project's success. These carry the highest weight of **4** points.
- **Should Have:** Important but not critical requirements. These carry a weight of **3** points.
- **Could Have:** Desirable but non-essential requirements. These carry a weight of **2** points.
- **Won't Have:** Requirements that are acknowledged but will not be implemented. These carry the lowest weight of 1 point.

Calculation of Weighted Satisfaction: The preceding data has been subjected to a weighting process, the outcome of which is presented below.

- **Must Have:** All completed (8 out of 8). $8 \times 4 = 32$
- **Should Have:** All completed (10 out of 10). $10 \times 3 = 30$
- **Could Have:** None completed (0 out of 1). $0 \times 2 = 0$
- **Won't Have:** None completed (0 out of 4). $0 \times 1 = 0$

The total possible points were calculated as follows: $(8 \times 4) + (10 \times 3) + (1 \times 2) + (4 \times 1)$, resulting in a sum of 68 points. The achieved points were determined to be $32 + 30 + 0 + 0$, totaling 62 points. Consequently, the weighted satisfaction score was computed by dividing the achieved points by the total possible points and multiplying by 100, yielding approximately 91.2%.

This weighted approach offers a more precise measure of stakeholder satisfaction by accounting for the varying levels of importance of each user story category. Using this method, I determined that stakeholder satisfaction reached 91.2%, which is an excellent result and demonstrates that the project successfully met stakeholder expectations.

8.2 Software Performance

Defining a software engineering project performance involves establishing criteria that reflect the efficiency, effectiveness, and success of the project, and this has always been an historical complex problem, according to Junior and Meira [10]. Nonetheless, performance can be measured through various indicators such as project timelines, resource allocation, collaboration efficiency, and overall project success [3]

This has brought many questions about how I can measure the efficiency and productivity of this project, and the conclusion was that I had to define a set of key performance indicators (KPIs) that accurately reflect Klippa goals for this project.

As outlined by Mannila [16], Key Performance Indicators (KPIs) are quantifiable metrics that play a crucial role in evaluating the efficiency, effectiveness, and success of an organization or specific project. These indicators provide objective information essential for making informed decisions that positively influence both business and technical performance. Effective KPIs should be measurable, aligned with the organization's strategic objectives, and facilitate continuous feedback for process improvement. By ensuring that KPIs are consistent with the project's goals and measurable against set targets, organizations can accurately assess their progress. Importantly, KPIs must be designed to prevent undesired behaviors and remain in harmony with the organization's overarching objectives.

8.2.1 Applying KPIs to this Project

To assess the development performance of this project, I used several GitLab metrics that provide valuable insight into development throughput, efficiency and overall progress. These metrics were chosen for their objectivity, reusability, and ability to provide a clear view of the development process, thereby adding significant value to the organisation. The specific metrics used are:

- **Merge Requests (MRs) Merged:** This metric tracks the number of merge requests that have been successfully merged into the development branch. It provides insight into development throughput and the frequency of feature completions or bug fixes. This metric is particularly valuable because it directly measures productivity and the continuous delivery of improvements. Over the course of the 11-week internship, a total of 16 MRs were successfully merged, with an average of approximately 6 MRs per month and a peak of 10 MRs in June. Since the initial MR to the main development branch was merged in May, the total of 16 MRs demonstrates a continuous deployment and an upward learning curve throughout the internship. This is a positive indicator of progress and development.

- **Time to Merge:** This measures the average time taken to merge a merge request from its creation. It is an indicator of the efficiency of the review process, the speed with which new code is integrated, and it highlights the effectiveness of the collaborative process. A lower merge time indicates timely code review and integration, which reduces cycle time and facilitates faster delivery of updates. The average time to merge a merge request was 2 days. This indicates a relatively efficient code review and integration process.

- **Commits per Month:** This illustrates the number of commits made over a month, providing insight into the frequency of development activity and code contributions. It highlights the consistency and regularity of code contributions, which is essential for maintaining momentum in the development process. This metric helps in understanding the developer's engagement and the steady progress of the project. The commits graph shows a total of 138 commits made during the internship, with a monthly average of around 46 commits.

- **User Story Completion:** This KPI quantifies the number of user stories completed over a specified period. It is crucial for assessing progress against the planned backlog and understanding the functionality delivered, while also ensuring that development efforts are aligned with the user requirements and project objectives. Throughout the project, 18/23 User stories had its requirements and acceptance criteria fulfilled, including all "Must Have" and "Should Have" demonstrating the alignment of development activities with the project goals.

In conclusion, the metrics indicate a well-managed and efficient development process, characterized by consistent activity, prompt code integration, and successful implementation of planned features. These KPIs provide a comprehensive view of the project's progress and the effectiveness of the development practices employed.

8.2.2 Discussion of KPI Results

In order to evaluate the KPIs defined for this project, it is crucial to consider my personal learning curve, especially since I commenced the internship with no prior knowledge of the technologies used at Klippa. An analysis of the KPIs in isolation would not provide a comprehensive understanding of my development performance without accounting for the significant learning process involved.

Upon examination of the evaluated KPIs, a trend emerges that mirrors my learning progression in Angular. Consequently, an assessment of my development performance must also consider the general learning curve associated with Angular, particularly for individuals with a background similar to mine, a pre-graduate student with limited prior JavaScript knowledge.

According to academic references, the learning curve can be steep but manageable with the appropriate resources and guidance (Grant, 2014; [6]). Common challenges include performance issues and the complexity of directives, which can be particularly daunting for novice developers (Ramos et al., 2016; [17]).

Given this context, it is necessary to evaluate my learning experience with Angular against these academic insights. The upward trend observed in my development metrics, such as the number of commits and merge requests per month, suggests a correlation between my increasing proficiency in Angular and my overall productivity.

Figures 17 and 18 illustrate my development activity through the number of merge requests merged and commits made over the internship period, were I can clearly see an upwards trend.

Merge Requests Analytics: The merge request analytics graph demonstrates a progressive increase in the number of merged requests from April to June, with an average merge time of 2 days. Initially, in April, the low number of merged requests signifies the nascent stage of my learning process, predominantly focused on acquiring fundamental knowledge of Angular and GraphQL. As my proficiency with these technologies improved, a notable rise in merged requests is observed, doubling from April to May to a total of four. This increase is indicative of enhanced competence and growing familiarity with the technology stack, reflecting the beginning of more regular and effective code contributions. In June, the number of merged requests more than doubled compared to May, reaching a peak of ten. This surge aligns with the consolidation phase of my learning, where my comprehensive understanding of Angular and GraphQL facilitated higher productivity and efficient implementation of the MVP feedback. The overall upward trajectory in merged requests underscores a direct correlation between my increasing expertise and the ability to implement features and fixes with greater efficacy.

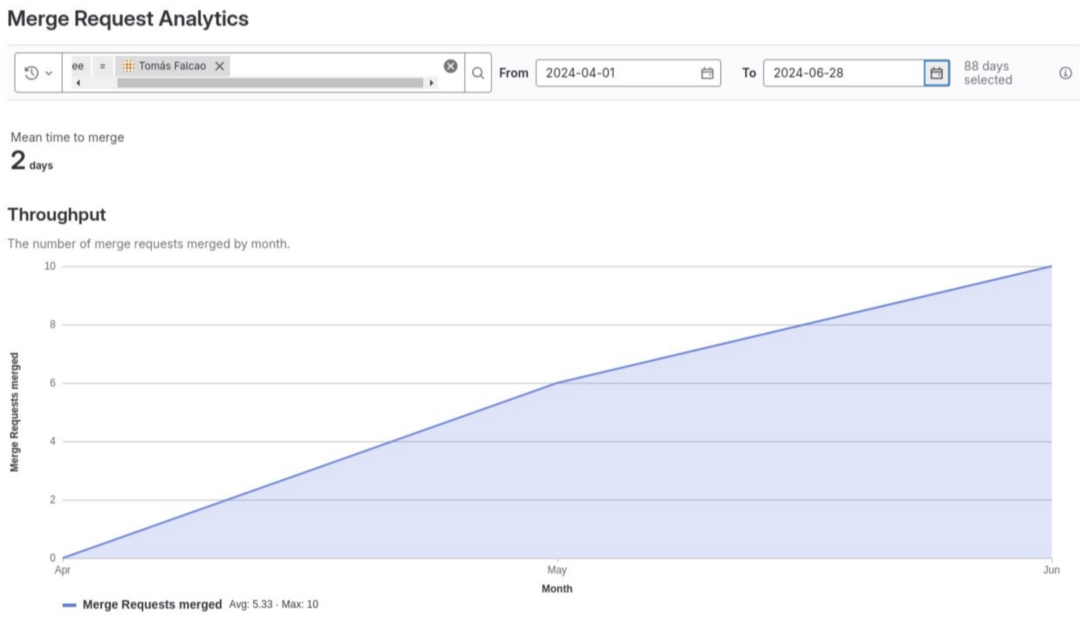


Figure 21: Merge Request Analytics

Time to Merge: The metric of an average of 2 days to merge a merge request is a significant indicator of the project's efficiency and effectiveness. A short merge time is beneficial for several reasons. Firstly, it suggests a streamlined and prompt code review process, where changes are quickly evaluated and integrated, thereby reducing the cycle time and enabling faster delivery of features and fixes. This is crucial in maintaining a continuous integration and continuous delivery (CI/CD) pipeline, which is essential for modern software development practices. Secondly, shorter merge times minimize the risk of merge conflicts and integration issues, as the codebase remains more up-to-date with frequent integrations. This helps in maintaining code quality and stability. Overall, an average merge time of 2 days demonstrates a well-functioning development process, fostering both productivity and high-quality code integration.

Commits per Month: The commits per month graph reveals a total of 138 commits over the course of the internship, averaging 46 commits per month. In April, the number of commits was relatively low, consistent with the initial phase of the learning curve, mirroring the trend observed in the merge requests analytics. As my familiarity with Angular and GraphQL grew, there was a marked increase in commit frequency, peaking in May and subsequently stabilizing in June with a slight decrease. This decline in June, despite the increased number of merged requests, suggests enhanced code efficiency, where fewer commits were required to achieve more substantial outputs. This pattern indicates that by June, my learning curve had reached its apex, reflecting a significant improvement in code efficiency as fewer commits yielded more comprehensive and feature-rich code implementations. This progression in commit activity signifies my advancing capability to contribute more effectively and efficiently to the codebase, demonstrating a positive and steep learning trajectory.

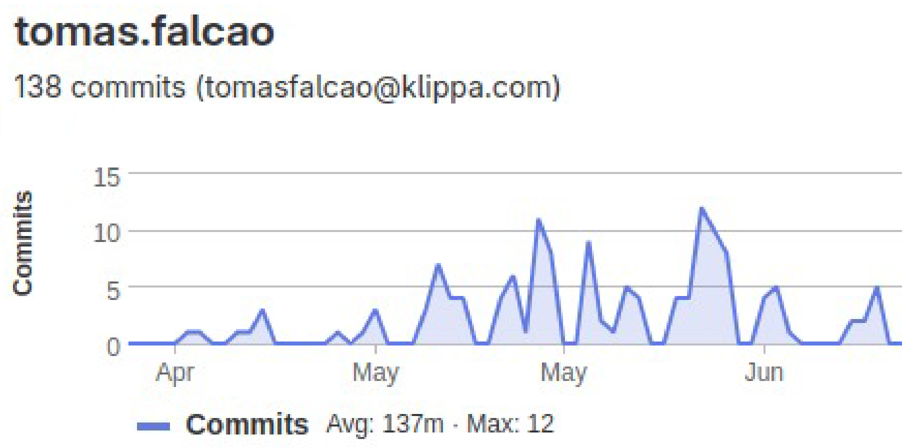


Figure 22: Commits per Month

User story completion: As previously discussed in Chapter 6.1.3, it is evident that all feasible stakeholder requirements were successfully completed and implemented into the project. All user stories identified as "must have" and "should have" were successfully fulfilled, which resulted in 91.2% stakeholder satisfaction. This indicates that stakeholders are highly satisfied with the final product.

Correlation with Learning Curve: The analysis of these KPIs against my learning curve reveals a clear correlation between my development metrics and my proficiency in Angular and GraphQL. The initial phase of low activity aligns with the steep learning curve typically associated with mastering new technologies. The subsequent rise in commits and merged requests reflects my growing competence and familiarity with Angular and GraphQL.

Given these observations, it is evident that my increasing proficiency in Angular and GraphQL positively impacted my productivity and efficiency, as illustrated by the KPIs. I can see a clear upward trend in code and features deployed, so I can also conclude that this learning experience was successful, as I was able to meet the project requirements while maintaining an upward learning and development curve. The consistent activity and prompt code integration further affirm the successful implementation of the project's planned features and objectives. This comprehensive evaluation of the KPIs, in conjunction with my learning curve, underscores the overall success of this project performance.

9 Conclusion

The thesis project was centred upon the development of a statistical interface for Klippa's Doc-Horizon platform, with the objective of enhancing the user experience through effective data visualisation and analysis. The study highlighted the necessity for intuitive interfaces in document management solutions to bridge the gap between complex data sets and user-friendly visualisations. The developed interface enables users to tailor their data views dynamically, facilitating improved decision-making through detailed insights into usage patterns.

The project was divided into several stages, beginning with a comprehensive review of existing interfaces, including Azure Monitor and Google Cloud Platform Monitoring Tool. The review facilitated the identification of optimal practices and informed the development of the new interface, ensuring that it met the specific needs of Klippa's stakeholders. The new interface incorporated a number of key features, including a filters section for the creation of customised data views, a general statistics section for the display of key metrics, dynamic chart visualisations, and a projects table for the presentation of comprehensive project overviews.

The system's architecture was designed with scalability and maintainability in mind, utilising a modular approach with Angular for the front-end, Go for the back-end, and clickhouse for the database. This configuration guaranteed the efficient handling of data and the provision of real-time updates. The project adhered to UI/UX principles, resulting in an uncluttered, responsive, and visually appealing interface.

The implementation of the new system presented a number of challenges, including the need to learn new technologies and integrate various components within the existing Klippa codebase. In order to address these challenges, the project adopted an industry-as-laboratory approach, which allowed for iterative development and continuous feedback from stakeholders. This approach proved to be crucial in refining the interface and adding functionalities such as extensive chart customisation and general statistics updates based on selected services.

In conclusion, the developed statistical interface successfully enhances the DocHorizon platform by providing detailed, customisable, and user-friendly data visualisations. This project not only improved the platform's functionality but also contributed to the broader field of data visualisation and user interface design. Future work could focus on further enhancing customisation options and integrating additional data sources to provide even more comprehensive insights for users.

10 Acknowledgments

I would like to express my deepest gratitude to Klaas Jelmer Boskma for his exceptional supervision throughout my internship. His guidance ensured that I was well-integrated within Klippa and that the project was progressing according to plan. His support and mentorship were invaluable to the success of this project.

Furthermore, I am profoundly grateful to Jan van der Molen and Oliver Strik for their unwavering assistance and for imparting their extensive knowledge of Angular and front-end development. Their consistent feedback and readiness to answer all my questions significantly contributed to my learning and the project's advancement.

This internship marked my first professional experience in the industry and proved to be an extraordinary learning journey. I developed both professionally and personally, thanks to the warm and welcoming environment at Klippa. I am thankful to everyone at Klippa for the opportunity to work on such an intriguing and impactful project. It was immensely rewarding to see my contributions being utilized and adding real-world value. I am excited about the next adventure and the continued growth that lies ahead.

Finally, I would like to express my gratitude to Andrea Capiluppi for his supervision and assessment of my thesis.

References

[1] Microsoft Azure Monitor Overview . https://learn.microsoft.com/en-us/azure/azure-monitor/overview. Acessed on 17/03/2023.

[2] Google clout platform documentation. https://cloud.google.com/bigquery/docs. Accessed on 04/07/2024.

[3] Sufajar Butsianto, Donny Muda Priyangan, Febri Dolis Herdiani, Budiman Budiman, and Yuliana Mose. Evaluation of the effectiveness of technology-based project management systems for software development. *Global International Journal of Innovative Research*, 1(2):175–181, 2024.

[4] Tomás Carrellan Esteves Leote Falcão. Developing a statistics interface for the klippa dochorizon application, July 2024. Research Proposal.

[5] Daniel Galin. *Software quality: concepts and practice*. John Wiley & Sons, 2018.

[6] Andrew Grant. *Beginning AngularJS*. Apress, 2014.

[7] Shreeram Hudda, Ritika Mahajan, and Sarvesh Chopra. Prioritization of user-stories in agile environment. *Indian Journal of Science and Technology*, 9(10.17485), 2016.

[8] A. S. Jadhav and R. M. Sonar. Evaluating and Selecting Software Packages: A Review. *Information and Software Technology*, 51(3):555–563, 2009. Accessed on 15/03/2024.

[9] Madhuri A Jadhav, Balkrishna R Sawant, and Anushree Deshmukh. Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, 6(3):2876–2879, 2015.

[10] Gibeon Soares de Aquino Junior and Silvio Romero de Lemos Meira. Towards effective productivity measurement in software projects. In *2009 Fourth International Conference on Software Engineering Advances*, pages 241–249, 2009.

[11] Klippa. Dochorizon, 2024.

[12] Klippa. Flow builder, 2024.

[13] Klippa. Human in the loop, 2024.

[14] Klippa. Identity verification, 2024.

[15] Klippa. Ocr, 2024.

[16] Jukka Mannila. Key performance indicators in agile software development. 2013.

[17] Miguel Ramos, Marco Tulio Valente, Ricardo Terra, and Gustavo Santos. Angularjs in the wild: A survey with 460 developers. In *Proceedings of the 7th International Workshop on Evaluation and Usability of Programming Languages and Tools*, pages 9–16, 2016.

[18] NPM Trends. Npm trends: Compare npm package download statistics, 2024. Accessed: 2024-07-02.

[19] Zsuzsa Varvasovszky and Ruairí Brugha. A stakeholder analysis. *Health policy and planning*, 15(3):338–345, 2000.