



UPPSALA
UNIVERSITET

Självständigt arbete i informationsteknologi
12 juni 2023

A tailored web-based system for managing and attracting customers

Carl Anton Gustavsson
Elis Indebetou
Filip von Knorring
Erik Larsson
Nils Persson
Ramez Shakarna



UPPSALA
UNIVERSITET

Institutionen för
informationsteknologi

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1, hus 10

Postadress:
Box 337
751 05 Uppsala

Hemsida:
<https://www.it.uu.se>

Abstract

A tailored web-based system for managing and attracting customers

Carl Anton Gustavsson
Elis Indebetou
Filip von Knorring
Erik Larsson
Nils Persson
Ramez Shakarna

In an effort to improve customer interactions and relationship management, a CRM system and website were developed for Minafribrev, a finance company. The CRM system was tailored to manage customer activities and store relevant information and notes, while the website serves as a platform for customers to sign up for services and access information. This project aimed to address the specific needs of the company and provide solutions for efficient customer management through the usage of these two tools.

Handledare: Ellen Arvidsson, Andreína Francisco, Diletta Goglia, Petter Terenius, and Carl Wallskog

Examinator: Andreína Francisco

Sammanfattning

Denna rapport redogör utvecklingen av ett kundhanteringssystem (CRM) och en webbsida för Minafribrev, ett finansbolag, i syfte att förbättra kundinteraktioner och relationshantering. Målet var att utveckla ett system som kunde användas för att effektivt hantera kundaktiviteter och att lagra relevant data som kontaktinformation, information om möten, och att hantera kundrelaterade uppgifter. CRM-systemet anpassades för att hantera dessa uppgifter, medan webbplatsen fungerar som en plattform för kunder att registrera sig för företagets tjänster och få tillgång till information.

För att lösa problemet utvecklades CRM-systemet och webbplatsen med användning av moderna webbt teknologier och verktyg, såsom Express, React, och Next.js. Dessa verktyg möjliggjorde en effektiv och säker implementering av systemet, samtidigt som de uppfyllde funktionella och säkerhetskrav som fastställdes av intressenterna.

För att fastställa att CRM systemet och Webbsidan uppfyllde de krav intressenterna hade användes två olika testverktyg, Lighthouse för webbplatsen och Cypress för CRM systemet. Resultaten visade att CRM-systemet och webbplatsen effektivt uppfyllde de specificerade kraven och att systemet hanterade kunddata på ett säkert sätt. Webbsidan uppnådde även väldigt bra resultat i utvärderingen, vilket indikerade på en god prestanda, tillgänglighet och sökmotoroptimering.

Dessa resultat visade att CRM-systemet och webbplatsen var användbara och effektiva lösningar för Minafribrev. Genom att använda dessa verktyg kan företaget förbättra sin kundhantering och erbjuda en bättre upplevelse för sina kunder.

Contents

1	Introduction	1
2	Background	2
2.1	CRM	2
2.2	Stakeholders	2
2.3	Laws and regulations	3
2.4	Usability	3
3	Purpose, aims, and motivation	3
3.1	Goals with the CRM	4
3.2	Goals with the website	4
3.3	Ethical aspects	4
3.4	Sustainability	5
3.5	Security	5
3.6	Delimitations	6
4	Related work	6
4.1	Other CRM tools	7
4.2	Other websites	7
5	Tools & Approaches	9
5.1	Programming language	9
5.1.1	Alternatives	10
5.1.2	Type safety vs. Development efficiency	11
5.2	Frontend application frameworks	12

5.2.1	Frontend framework	12
5.2.2	Meta framework	13
5.3	User interface libraries	14
5.3.1	Website UI library	14
5.3.2	CRM UI library	15
5.4	File browser	15
5.5	Backend	15
5.5.1	Database management system	16
5.5.2	Web server framework	16
5.5.3	Object data modeling library	17
5.5.4	REST API	17
5.6	Authentication provider	18
6	System structure	19
6.1	Architecture overview	19
6.2	REST API server	20
6.3	Database	21
6.4	Customer-facing web interface	22
6.5	Administrator-facing CRM web interface	23
7	Requirements & evaluation methods	24
7.1	Requirements	24
7.1.1	Functional requirements	24
7.1.2	Security requirements	25
7.2	Evaluation	25
7.2.1	End-to-end testing	26

7.2.2	Quality testing	26
7.2.3	Backend testing	27
7.2.4	Security checklist	28
8	Implementation of the system	29
8.1	Website	29
8.1.1	Navigation	29
8.1.2	Layout	30
8.1.3	Interactivity & tracking	32
8.2	CRM	35
8.2.1	Customers & Prospects page	36
8.2.2	Records page	37
8.2.3	Archives	38
8.2.4	Customer pages (Prospects, Individuals & Companies)	38
8.3	File browser	40
8.4	REST API server	41
8.4.1	The API layer: Controllers	41
8.4.2	The business logic layer: Services	44
8.4.3	The data access layer: Models	45
8.5	Database design	45
8.5.1	Users collection	46
8.5.2	Archives collection	46
8.5.3	Companies collection	47
8.5.4	Records collection	48
8.6	Integrating authentication: Auth0	49

9 Results	49
9.1 End-to-end testing	50
9.2 Backend testing	51
9.3 Quality testing	53
9.4 Security checklist	54
10 Discussion	55
10.1 Website	56
10.2 CRM	56
10.3 Backend system	57
11 Future work	58
11.1 Fullmaktskollen	58
11.2 Customer interface	59
11.3 Multiple employee accounts	59
12 Conclusions	59

1 Introduction

In today's digital age, companies must establish a strong online presence and cultivate good relationships with their customers to increase their chances of success [2] [3]. Developing digital tools such as a website, and customer engagement tools can be vital in achieving this goal. This project aims to contribute to a successful start for a new company by developing a Customer Relationship Management (CRM) system, and a website. In general, a CRM system is a software tool that can help businesses manage customer interactions more efficiently and improve their overall customer service, sales, and marketing efforts [26]. It allows for the storing of customer information and the analysis of customer behavior, which can be used to enhance relationships with customers.

These products were developed for Minafribrev, a company operating in the finance sector. The stakeholders had a clear vision of the design and functionality of the CRM and website, which enabled the creation of tailored solutions that met their needs.

The CRM system was primarily designed for managing customer interactions, keeping track of customer activities, and organizing customer-related notes and files. Although the system did not include features for predicting and analyzing customer behavior, it still provided valuable tools for improving customer relationship management for our stakeholders.

In addition to the CRM system, a website was also developed that serves as an online platform for Minafribrevs customers. When a customer signs up for the company's services they enter information like their phone number or email, which automatically gets stored in the CRM if they sign up for the service.

Declaration of division of labour The division of labour was evenly distributed amongst all members of the group. To ensure we met our project deadline, we divided the team into three groups: one for the CRM, one for the website, and one for the backend. Nils Persson, Erik Larsson, and Elis Indebetou led the CRM group, Filip von Knorring and Anton Gustavsson were in charge of the website, and Ramez Shakarna was responsible for the backend. While we were assigned specific responsibilities, team members were free to help out in other areas as needed. We based our decisions on previous experience and individual interests. The respective individuals mostly wrote the corresponding parts of the report, while the group cooperatively addressed and evenly distributed responsibilities for other parts.

2 Background

In this section, we will explore the concept of a CRM system and its benefits. We will also present the stakeholders involved in this project and their goals. Additionally, the relevant laws and regulations that govern data protection concerning the creation of a CRM system and a website will be presented. Lastly, we will touch upon the significance of usability in designing effective websites.

2.1 CRM

Administrative work can increase as a business expands and attracts more clients. This can decrease the efficiency of the workforce as less time is spent in contact with clients. One digital solution for automating some of the administrative work is a CRM system.

A CRM system can be thought of as an information system designed to help businesses realize their customer focus [35]. It provides businesses with a centralized platform for managing all of their customer-related files, information, and interactions. It also allows for better communication and cooperation within the company by providing access to customer data between employees. This ensures that all employees can access the latest notes and information about each customer.

The positive effects of a CRM may also extend to the customers themselves. Researchers suggest there is a positive relationship between how much information a business has about their customers and customer satisfaction [43]. This suggests that CRM applications may have a direct impact on customer satisfaction, which is another good reason to have a CRM system.

2.2 Stakeholders

The stakeholders for this project, Minafribrev, are a start-up working with *fribrev*¹. The stakeholders goal is to move these fribrev to better funds for their customers. To keep track of all their potential and current customers, a customized CRM was requested, which should be tailored to their needs. They also wanted a website where customers could contact them and find information about them.

¹A pension that is no longer receiving payments. For example, when employment is terminated, the related occupational pension ceases to receive payments from the employer. At this point, the pension becomes a *fribrev* [45].

2.3 Laws and regulations

Laws play a crucial role in regulating various aspects of our lives, and the digital realm is no different. Digital systems that store people's private information in Europe must comply with the General Data Protection Regulation (GDPR), which is the European nations' data protection law [28]. Similarly, a Swedish business that works in finance must comply with the Swedish financial supervisory authority [44]. One of their laws, SFS:2016:51 6c, states that an account-keeping institute must store documents for at least ten years [42]. If they are stored in a digital system, this data has to be retrievable to make sure the business complies with the law.

2.4 Usability

Usability is a crucial factor that can significantly influence a consumer's intentions. If a company is not well-known or if its intentions are uncertain, having good usability is crucial [37]. Various intentions may include encouraging users to register for or pay for a service.

To ensure a website's usability, various aspects can be accounted for, such as anticipation, consistency, reversibility, efficiency, and visibility. Anticipation refers to designing websites based on visitors' needs and requirements, while consistency involves designing the website in a manner that aligns with consumers' knowledge and skills. Reversibility ensures that users can easily undo tasks or actions, while efficiency involves simplifying processes and minimizing errors. Lastly, visibility is about ensuring transparency and not hiding information from the customers [37].

3 Purpose, aims, and motivation

The objective of this project was to create digital solutions that enable the stakeholders to effectively attract and oversee customers. Two different tools were requested, a CRM system and a website. The goal was to have them work together to increase work efficiency and get new customers into the system.

3.1 Goals with the CRM

The objective of the CRM system is to assist stakeholders in effectively managing their customer relationships. The stakeholders' primary objectives included the ability to view all customers, identify potential prospects, and access detailed customer information. Another request was to have a place where they could store notes about their customers, such as calls or recent meetings.

Another requested feature was the ability to associate individuals with their employers. By having this feature, it would be easier to identify potential customers in companies the stakeholders had previously worked with. Furthermore, the CRM system also needed a comprehensive file management system to store all relevant business-related documents and customer-related documents.

3.2 Goals with the website

The overall goal of the website was to serve as the primary interface between the company and its customers. The stakeholders established three key objectives for the website. Firstly, it needed to appear trustworthy since its users would need to give them power of attorney to move their pensions. Secondly, it needed to be easy to find the sign-up form as well as apply for the service. Lastly, the website needed to have an aesthetically pleasing appearance to give a good impression to its customers.

3.3 Ethical aspects

There were different ethical aspects to consider when designing the system. This is because of the level of integrity that is associated with insurance and banking businesses. The developed tools will be utilized for the storage of customer information and powers of attorney. It is therefore crucial that the system maintain a high level of security to comply with GDPR regulations and ensure the integrity of customer data. Five different ethical aspects were mainly considered. These were, in no particular order, **legal compliance**, **privacy**, **sustainability**, **equality** and **transparency**.

- Legal compliance means that the system must comply with laws and regulations and had to be taken into consideration throughout the design of the system.
- Privacy is crucial since the system handles individuals' private information, which means certain steps need to be taken to ensure customer privacy. This is explored further in Section 3.5.

- Sustainability is specific to insurance which this project is adjacent to and it means that pension services offered are suitable for the customer, not made to drain the customer by way of fees. Rather the services should ideally benefit the customer's savings and we are making this crystal clear to the customer.
- Equality relates to the fact that the website should be easy to use for everybody and designed with accessibility in mind.
- Lastly, transparency, meaning that it should be transparent to customers what they are agreeing to and how, for example, their data is stored and used.

3.4 Sustainability

The United Nations' sustainable development goals are 17 goals set to be reached by 2023, aiming to be a shared blueprint for peace and prosperity for the planet and its people [21]. These 17 global goals are further divided into different subgoals, and this project aims to work towards at least one of these subgoals. That is subgoal 8.10, *Strengthen the capacity of domestic financial institutions to encourage and expand access to banking, insurance, and financial services for all* [22].

This project aims to create software that makes it easier to manage and assist customers in moving their pension funds. By doing so, this software aims to expand access to banking services, granting people more freedom to select where they want their money invested. Consequently, this means that we contribute to subgoal 8.10 of expanding financial services accessibility.

3.5 Security

The CRM software stores files and information about users, which can be highly sensitive. This includes their social security number, email, powers of attorney, and bank statements with their total pension capital. Since this is data pertaining to Swedish customers in Europe the security has to obey the GDPR laws brought up in Section 2.3. The product must adhere to the regulations set by the Swedish Financial Supervisory Authority, Finansinspektionen, or FI for short, which were described in 2.3.

3.6 Delimitations

This report aims to provide a comprehensive analysis of the system's functionality and security, employing a variety of evaluation methods to assess different aspects of the system. However, it is important to note that certain delimitations were necessary due to time constraints and a substantial workload associated with the project's scope.

A significant delimitation pertains to user testing. User testing can provide valuable insights into the content and usability of a system, particularly for a customer-facing component like the website [7]. Through user testing, we could gain first-hand feedback about the user interface, navigation, content relevance, and overall user experience, which would be instrumental in enhancing the website's efficacy and user satisfaction.

However, due to the limited timeframe and the extensive tasks involved in the project, we were unable to conduct formal user testing. While we strived to adhere to best practices in web design and usability, we acknowledge that direct user feedback could offer additional perspectives and potentially highlight areas for improvement that we may not have identified.

Another critical area that was not explored in this report is penetration testing. Penetration testing is a vital aspect of evaluating a system's security [50]. It involves simulating cyber attacks to identify vulnerabilities that could potentially be exploited. This would have been a valuable addition to our security assessment, providing practical insights into the system's resilience against real-world attacks.

However, effective penetration testing requires a considerable amount of expertise, resources, and time. Therefore, despite recognizing its importance, we were unable to carry out any penetration testing for our system. This limitation is worth considering when interpreting the results of our security evaluation.

Despite these delimitations, we believe that the evaluation methods employed in this report provide a robust assessment of the system's performance, security, and compliance with the defined requirements. As the system evolves and expands, future work could include user testing to further refine and enhance the user experience.

4 Related work

There exists similar applications and websites which solve some of the same problems presented in this report on the market already. How these systems work and how they differ from ours will be discussed in this section.

4.1 Other CRM tools

One of the most popular and used CRM tools on the market today is Salesforce². Salesforce is a cloud-based CRM platform that provides a variety of tools and features for sales, marketing, and collaboration. Salesforce offers various subscription packages that include many existing templates that offer some customization for their customers' specific needs.

Another popular CRM software is called Scoro³. Scoro is, like Salesforce, a cloud-based tool that has a subscription-based model. Scoro differs from Salesforce by placing a greater emphasis on reporting and statistics. It provides businesses with real-time data that allows them to make informed decisions based on their customer interactions.

The main difference between these existing CRM tools and the one we developed is ownership. Like Salesforce and Scoro, many of the companies on the market today only sell subscriptions to their services, whereas our stakeholders will have full ownership and control of the system. This allows for greater customization and flexibility to tailor the CRM system to their specific needs without having to rely on pre-existing templates or features. Our system also has the additional benefit of being cost-effective to host and maintain due to its lightweight design while still providing the essential functionality required by Minafribrev at its current stage of business. In contrast, since most of the other CRM tools available today operate on a subscription model, using their services could result in significantly higher monthly costs.

4.2 Other websites

There are a few companies that have a similar business model and work in the same domain as Minafribrev. Most of these companies are banks that offer fribrev solutions as an addition to their already existing set of products. However, some have a very similar business model as our stakeholders. The business with the most similar model and website design is Svenska Fribrevsbolaget⁴, abbreviated as SFB. Similar to our website, the customer is greeted with a "call to action" form on their website. This is a typical way for websites to try to gain potential customers' interest as quickly as possible. SFB does this with a short text introduction to the product besides a sign-up form. Comparably, Minafribrev uses a similar slogan and introduction, with a button leading to a sign-up form. This similarity can be seen in Figure 1.

²salesforce.com/se/products/what-is-salesforce/

³scoro.com/crm-software/#resources

⁴svenskafribrevsbolaget.se

The image shows two website screenshots. The top screenshot is for **minafribrev.se**. It features a main heading "Samla dina fribrev och sänk avgiften" (Collect your tax-free allowances and reduce taxes). Below the heading is a paragraph explaining that when a job ends, the employer's contributions and the pension are added to a tax-free allowance. It also mentions that different insurance periods end, but the unpaid money remains, and there is an opportunity to influence savings by continuing to be active. There are two buttons: "Samla dina fribrev nu!" (Collect your tax-free allowances now!) and "Mer om fribrev" (More about tax-free allowances). To the right is an illustration of a piggy bank with a person sitting on it, holding a coin.

The bottom screenshot is for **Svenska Fribrevsbolaget**. It has a main heading "Samla din tjänstepension – sänk dina fondavgifter!" (Collect your pension – reduce your fund fees!). Below this is a paragraph stating that by collecting your pension, you can reduce fund fees, get a better overview of dispersed pension money, and have the opportunity for a higher future pension. To the right is a form titled "Ta första steget mot en högre framtida pension!" (Take the first step towards a higher future pension!). The form asks for E-post*, Telefon*, and Personnummer (ÅÅÅÅMMDD-XXXX)*. There is a checkbox for "Jag godkänner att SFB får spara och behandla mina personuppgifter." (I agree that SFB can store and process my personal data.) and a button "Sammanställ mina fribrev!" (Collect my tax-free allowances!). At the bottom, it says "Identifiera dig med BankID" and "Utmärkt 4.5 av 5 ★ Trustpilot".

Figure 1 Comparison of Minafribrev (top) and Svenska Fribrevsbolaget (bottom) and their homepages

As previously mentioned, call-to-action buttons are a proven way of getting user interaction and commitment [41]. The visitors first impressions of the websites are critical for both companies, as they aim to generate a substantial number of leads that can be followed up on later.

Looking further at SFB, there is a clear difference in content compared to Minafribrev. SFB has various sections on the front page as well as further reading on subsequent pages, which results in more text and clutter. This can be compared to Minafribrev, where there are fewer text sections on the main page, focusing on concise ideas. Minafribrev has fewer additional pages, more figures, and imagery, as well as interactive pension calculator sliders.

In addition to Svenska Fribrevsbolaget, attempts have been made at a number of the major banks, like SEB ⁵, to offer a product that will help customers manage their private pensions and move the money to the bank in question. These sites are simpler and have some informational text and a link that navigates the customer to Fullmaktskollen⁶. This grants the banks access to collect and find all the customers pension funds, making it easier to move them.

5 Tools & Approaches

In this section, we outline the methods, tools, and techniques used to develop the CRM and website for the stakeholders. We have divided the section into frontend frameworks, backend technologies, and alternatives and justifications for the chosen methods. A primary focus of the discussion is the choice of JavaScript as the programming language for this project, as well as the decision to use the MERN⁷ technology stack (MongoDB⁸, Express.js⁹, React¹⁰, and Node.js¹¹).

5.1 Programming language

For this project, we have chosen JavaScript as the primary programming language for developing the website, the CRM web application, and the backend. JavaScript is a widely used [24], versatile programming language and it is the standard for website interactivity [4]. Its widespread use and adoption ensure a rich ecosystem of tools, libraries, and frameworks, which can significantly enhance the development process. Moreover, JavaScript's extensive community support and comprehensive documentation make it a practical choice for this project. A key advantage of JavaScript is its native support across all web browsers [8], allowing for seamless integration and compatibility.

⁵seb.se

⁶fullmaktskollen.se/in-english/

⁷mongodb.com/mern-stack

⁸mongodb.com

⁹expressjs.com/

¹⁰react.dev

¹¹nodejs.dev/en

5.1.1 Alternatives

There are alternative programming languages that have been considered for developing the website and the CRM web application, such as PHP¹², Python¹³, and Ruby¹⁴.

- **PHP:** PHP is a popular server-side scripting language, primarily used for web development. It offers a wide range of frameworks and libraries and has a large user community [23]. However, PHP is primarily used for server-side tasks, while JavaScript is more versatile, being suitable for both client-side and server-side tasks, making it a better choice for this project.
- **Python:** Python is a versatile and powerful programming language, known for its readability and ease of use [19]. Python offers several web development frameworks, such as Django and Flask, which could have been used for this project.
- **Ruby:** Ruby is an object-oriented programming language that emphasizes simplicity and productivity. Ruby on Rails¹⁵, a popular web development framework built on Ruby, offers a wealth of features for rapid application development, such as a built-in Object-relational mapping (ORM), automatic code generation, and a Model-View-Controller (MVC) architecture.

Despite the availability of the aforementioned alternative programming languages, we chose JavaScript for several reasons:

- **Native browser support:** JavaScript runs in all modern web browsers, ensuring seamless integration and compatibility across different platforms and devices.
- **Versatility:** JavaScript is suitable for both client-side and server-side tasks, making it a good choice for full-stack web development projects.
- **Ecosystem:** JavaScript's extensive ecosystem of tools, libraries, and frameworks, such as React.js, Node.js, and Express.js, facilitates a streamlined development process and allows for the creation of a robust and efficient system.

JavaScript was chosen as the primary programming language for this project due to its flexibility, versatility, and native browser support. While PHP, Python, and Ruby

¹²php.net/index.php

¹³python.org

¹⁴ruby-lang.org/en

¹⁵rubyonrails.org/

have their advantages, they each possess limitations that make JavaScript a more suitable choice. PHP lacks JavaScript's versatility for both client-side and server-side tasks, leading to a more complex development process [48]. Python, although easy to read and use, can have slower performance in web applications compared to JavaScript, particularly when handling large amounts of data or in real-time scenarios [9]. Lastly, Ruby's performance compared to JavaScript is also lacking where it can perform up to 4 times slower [15]. In summary, JavaScript's unique combination of features allows for a streamlined development process, making it the ideal choice for developing the website and CRM web application.

5.1.2 Type safety vs. Development efficiency

There is an extension to JavaScript called TypeScript¹⁶, which could have been considered for this project. TypeScript is a superset of JavaScript that adds optional static typing, resulting in improved type safety and other benefits associated with typed languages. While TypeScript offers these advantages, the decision to use JavaScript instead was based on the following considerations.

JavaScript allows for faster development compared to TypeScript [20], as it eliminates the need for setting up and maintaining type definitions. This advantage enables the team to deliver the project within a shorter timeframe and ultimately contributes to overall efficiency. Additionally, the team members were already familiar with JavaScript, while TypeScript may require some team members to undergo additional learning and familiarization. Opting for JavaScript ensures that all developers can contribute effectively without the need for extensive training in TypeScript.

Compatibility with external libraries is another factor to consider. For libraries without built-in TypeScript support, developers either lose the purpose of TypeScript by ignoring types when writing code to interact with these libraries, or they have to spend time writing types and interfaces for these external libraries. Choosing JavaScript simplifies the development and deployment process and ensures seamless compatibility with external libraries.

While we acknowledge the benefits of TypeScript, we believe that the advantages offered by JavaScript, such as speed of development, widespread use, and compatibility, make it a more suitable choice for this project.

¹⁶typescriptlang.org

5.2 Frontend application frameworks

In this section, we discuss the various frontend application frameworks, meta frameworks, and UI libraries that were considered for this project. We also provide reasoning for the choice of React.js, Next.js¹⁷, React Bootstrap¹⁸, and Material-UI¹⁹.

5.2.1 Frontend framework

There are several popular frontend frameworks available, such as React.js, Vue.js²⁰, and Angular²¹. Each framework offers its unique set of features and advantages, depending on the requirements and goals of the project.

- **React.js:** is a popular and versatile JavaScript library for building user interfaces, known for its high performance, easy maintainability, and component-based architecture.
- **Vue.js:** is another popular choice, a progressive JavaScript framework that focuses on approachability and flexibility, offering a gentle learning curve and adaptability to various project structures.
- **Angular:** is a robust, feature-rich framework developed by Google that is designed for building large-scale, complex web applications. It enforces a strict structure, which can help maintain consistency and scalability in large projects.

In this project, we chose React.js as the framework because of React.js's ability to develop complex and interactive web applications efficiently. The component-based architecture of React.js promotes code reusability and modularity, making it an excellent choice for developing scalable and maintainable systems. Furthermore, React.js is the framework with the largest and most active community [1], ensuring a wealth of resources and third-party libraries that can be utilized to speed up development.

We decided against using Vue.js because, while it is lightweight and fast, its ecosystem may not be as mature as React.js. This means that there might be fewer resources and

¹⁷nextjs.org/learn/foundations/about-nextjs/what-is-nextjs

¹⁸react-bootstrap.github.io/

¹⁹mui.com/material-ui/getting-started/overview/

²⁰vuejs.org

²¹angular.io/features

external libraries available, which could slow down the development process. Additionally, the team is already familiar with React.js, allowing us to leverage our existing knowledge and further accelerate development.

Angular was not chosen due to its steep learning curve and monolithic structure, which may not be suitable for the project's needs. Angular is more suited for large-scale, enterprise-grade applications with dynamic content, whereas our project requires a more versatile and lightweight solution [33].

5.2.2 Meta framework

In addition to the primary frontend framework, there are so-called meta frameworks built on top of these frameworks. A meta-framework is a higher-level framework built on top of a primary frontend framework, providing additional features, optimizations, and capabilities that enhance the development process and extend the functionality of the underlying framework [25]. Next.js is a powerful framework built on top of React.js, designed to facilitate server-rendered React applications. It offers excellent Search Engine Optimization (SEO) capabilities and efficient static site generation. SEO is the process of enhancing a website's visibility to search engines, which is crucial for attracting organic traffic. Websites that are easily discoverable by search engines and rank well in search results can attract more visitors and potential customers [29]. As the website is customer-facing, it is crucial to ensure that it is optimized for SEO. Next.js provides server-side rendering, which ensures that all content is readily available for search engine crawlers, resulting in better indexing and ranking.

While Next.js is a powerful meta-framework, it may not be the best choice for the entire codebase, which includes both the website and the CRM system. Next.js introduces an additional layer of complexity to the development process, which can be a significant hurdle when the underlying UI system is already fairly complex. Given the complexity of the CRM system, adding the overhead of a meta-framework like Next.js might not be ideal. Instead, focusing on the primary frontend framework, such as React.js, will allow the development team to work more efficiently and streamline the process.

Furthermore, the additional features provided by Next.js are not required for the CRM system. CRM systems are typically used internally by businesses and do not require the same level of SEO optimization as customer-facing websites. Because the CRM is not intended to be found by search engines or to rank high in search results, the SEO benefits provided by Next.js are unnecessary. However, for the customer-facing website, the use of Next.js was deemed advantageous due to its features that cater to improved user experience and SEO.

5.3 User interface libraries

The use of user interface (UI) libraries offers several advantages in the development process, such as faster development, consistency, and well-designed systems that have been tested and refined. These libraries provide pre-built components and design systems that ensure a cohesive and visually appealing user interface. Furthermore, it is essential to select UI libraries that are compatible with the chosen frontend framework, in this case, React.js.

There are numerous popular UI libraries available for React.js [13], such as React Bootstrap²², Material-UI²³, Ant Design²⁴, and Semantic UI²⁵. Each library offers its unique features and benefits, it is therefore vital to choose the one that best aligns with the project's specific needs and requirements.

5.3.1 Website UI library

We chose React Bootstrap for the customer-facing website due to its focus on responsiveness and adaptability to various screen sizes. As the website is customer-facing, it is crucial to ensure that it provides an optimal user experience across different devices, including desktops, laptops, tablets, and smartphones. React Bootstrap's responsive components and grid system allow for the creation of a user interface that automatically adapts to the screen size, ensuring that the content is always presented in a visually appealing and easily navigable manner. By leveraging the capabilities of React Bootstrap, we can develop a responsive, consistent, and visually appealing design for the customer-facing website.

For custom UI design we also made use of the tailwind library for faster development [16] and a smoother and more tailored way of styling [17] than what could be achieved with pre-built components. Tailwind comes with pre-defined css styling classes rather than fully featured components. This in turn enables the developer to apply as many of these classes as needed to achieve a certain design.

²²react-bootstrap.github.io

²³mui.com

²⁴ant.design

²⁵semantic-ui.com

5.3.2 CRM UI library

For the administrator-facing CRM system, we have chosen Material-UI, a popular React UI framework that offers a comprehensive set of pre-built components for user interface design and data visualization. While Material-UI may not be as focused on responsiveness as React Bootstrap, it provides extensive data visualization capabilities that are highly valuable for the CRM system. Since the users of the CRM will primarily be using their laptops when interacting with it, we can make certain assumptions about the screen sizes they will be using, which allows us to prioritize data visualization and functionality over responsiveness. Material-UI's components enable us to create a professional and user-friendly interface, allowing the users to efficiently manage customer relationships and gain insights from the data presented. By choosing Material-UI for the CRM system, a powerful tool that meets the specific needs of the stakeholders while maintaining a visually appealing and functional interface can be developed.

5.4 File browser

To effectively manage files and folders related to customers, the CRM system required a file browser. To fulfill this requirement, we opted to use Chonky ²⁶, a file browser library for React. Chonky operates as a UI shell, providing a presentation layer for files and folders. The decision to use Chonky was driven by the fact that we were unable to find any other suitable options. Other systems featured a file browser, however, they were fully integrated with a backend and required a monthly subscription. Given the stakeholders' preference to avoid ongoing costs, Chonky was used to aid the implementation of the file browser.

5.5 Backend

In this section, we discuss the architecture and technologies used in the backend of the application. Our primary focus is on building a robust and scalable infrastructure. We have chosen a NoSQL²⁷ database, MongoDB, for its flexibility and seamless integration with the chosen web server framework, Node.js, and Express.js. Mongoose²⁸ is used as an Object Data Modeling library to maintain data consistency and integrity while working with MongoDB. To facilitate communication between the frontend and backend

²⁶chonky.io

²⁷mongodb.com/nosql-explained

²⁸mongoosejs.com

components, a Representational State Transfer (REST) API²⁹ was chosen, built using Express.js, providing a standardized and scalable architecture for data exchange.

5.5.1 Database management system

When deciding on a database management system for the CRM, various factors such as scalability, flexibility, and compatibility with the chosen web server framework were considered. SQL databases, such as MySQL³⁰ and PostgreSQL³¹, and NoSQL databases, like MongoDB and Firebase³² were evaluated in this process.

SQL databases store data in tables with predefined schemas, which enforce a structured approach and strong data consistency. In contrast, NoSQL databases offer more flexible data storage, such as key-value, document, or graph stores, allowing for the handling of diverse and unstructured data. This flexibility and scalability make NoSQL databases a better fit for managing large volumes of diverse data [11].

After considering the available NoSQL databases, MongoDB was chosen for its seamless integration with Node.js and Express.js, ease of use, and excellent community support. MongoDB is a document-oriented database that stores data in a flexible, JSON-like format [34]. Firebase, on the other hand, poses the risk of vendor lock-in. This risk is due to the lack of self-hosting options and its dependence on the Google ecosystem, which hurts the potential of migrating the database in the future if deemed necessary. Furthermore, Firebase, unlike MongoDB, can not be self-hosted or deployed with a variety of managed services. However, it is essential to be mindful of the security implications when working with a NoSQL database like MongoDB, which requires proper configuration, access controls, and data validation to mitigate potential risks.

5.5.2 Web server framework

For the web server framework, we have chosen Node.js, a JavaScript runtime built on Chrome's V8 JavaScript engine [49], and Express.js, a popular web application framework for Node.js. This combination enables us to build a scalable and high-performance backend, capable of handling multiple simultaneous connections and efficiently processing data.

Without Express.js, the web server functionality would have needed to be built from

²⁹aws.amazon.com/what-is/restful-api

³⁰mysql.com

³¹postgresql.org

³²firebase.google.com

scratch, which would have been time-consuming and error-prone. Express.js simplifies the development process by providing a wide range of built-in features, middleware, and best practices that streamline routing, error handling, and data processing. This allowed for more focus on implementing the CRM's core features and functionalities [40].

5.5.3 Object data modeling library

Using an Object Data Modeling (ODM) library, such as Mongoose, is crucial for managing data relationships, schema validation, and streamlining the development process when working with MongoDB and Node.js. Mongoose provides a structured way to define and manipulate data models, enhancing security by validating data before it is saved in the database. This helps with maintaining data consistency and integrity, as well as aids in safeguarding against security vulnerabilities such as NoSQL injection attacks.

By choosing Mongoose as the ODM for the CRM backend, a more secure and efficient way to work with MongoDB can be ensured, while benefiting from the flexibility and scalability of a NoSQL database.

5.5.4 REST API

To facilitate communication between the frontend and backend components of the CRM, we chose to use a REST API. A REST API provides a standardized architecture for building web services, allowing different components to interact with each other using HTTP methods (such as GET, POST, PUT, and DELETE) [5] and exchanging data in various formats (e.g., JSON, XML).

The REST API was chosen for the following reasons:

- **Simplicity:** REST APIs are easy to understand and implement, as they use familiar HTTP methods and rely on standard conventions.
- **Scalability:** REST APIs are stateless, meaning that each request from the client to the server must contain all the necessary information to process the request. This stateless nature enables the API to scale easily and handle a large number of simultaneous connections.
- **Compatibility:** REST APIs can be accessed by various clients, regardless of the technology or platform they are built on, making it a flexible and adaptable solution for the CRM system.

- **Maintainability:** The modular architecture of REST APIs allows for easier maintenance, as changes in one part of the system do not necessarily impact the other parts.

Using Express.js as the web server framework, the REST API was built to handle different types of requests from the frontend, process the data, and interact with the MongoDB database as needed. This approach enables seamless integration of the frontend and backend components while ensuring efficient and secure data exchange between them.

5.6 Authentication provider

To protect against unauthorized system access, such as access to the CRM, we needed a way to authenticate and authorize the administrators. The identity provider Auth0³³ was chosen for this purpose. Auth0 provides a platform to implement secure authentication and authorization with many features such as multi-factor authentication, user management, and monitoring. The platform is managed by Okta³⁴, which was declared as the industry leader for Access Management in [10]. By using Auth0, we can delegate a critical and sensitive part of the system to industry experts and simultaneously reduce development time. Furthermore, it allows us to keep our REST API stateless by not storing authentication states on the server, such as sessions. There are many alternatives to Auth0, however, we deemed Auth0 to be the best fit for our system due to the following reasons:

- **Seamless integration:** Auth0 provides Software Development Kits (SDKs) for both Node.js and React.js, which allows us to easily integrate it with our chosen tech stack.
- **Affordability:** A free tier is available in Auth0 which includes all the necessary functionality for our use case.
- **Quality:** As previously mentioned, Auth0 is managed by Okta which is deemed to be an industry leader, and the platform contains high-quality documentation and is easy to use.

³³auth0.com

³⁴okta.com

6 System structure

In this section, the overall system structure is presented. The system consists of four main parts, which are referred to as *nodes*. In [46], a *node* is defined as "a point of intersection/connection within a data communication network". The nodes are:

- Customer-facing web interface
- Administrator-facing CRM web interface
- REST API server
- Database

In the following subsections, each part and the communication between the nodes are elaborated upon in more detail.

6.1 Architecture overview

The system consists of four nodes that communicate with each other, with the REST API server being the central point, as demonstrated in Figure 2. Notably, the two web interfaces exist completely independently and only communicate with the server when a service is needed. For instance, the customer interface will send a request to the server when a customer signs up. The administrator interface will interact with the server whenever a change is made in the CRM. However, the two web interfaces will never directly interact with each other or with the database. The server interacts directly with all the nodes, acting as the glue, without having to know where any of the parts reside. In essence, the system is designed to allow for hosting each part in a separate place and still allow for communication between them.

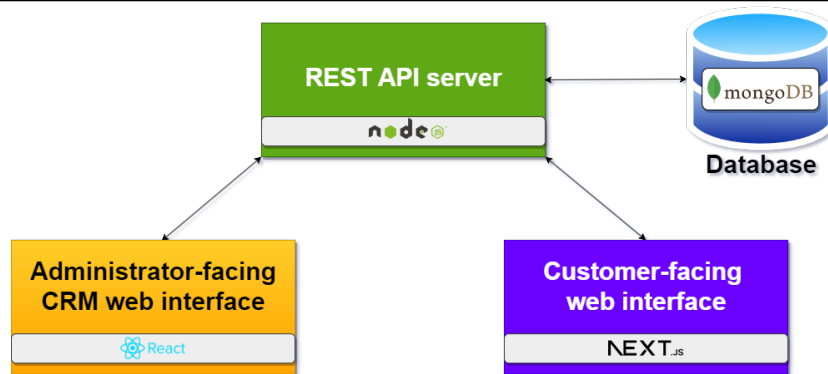


Figure 2 Communication between the Administrator-facing CRM web interface, REST API server, and Customer-facing web interface. Both the CRM and Customer-facing web interfaces communicate with the same REST API server, and the REST API server communicates with the database.

6.2 REST API server

The backend server was built using a *service-oriented architecture* with Node.js, Express.js, and Mongoose. Service-oriented architecture is defined as "an approach for building distributed systems that deliver application functionality as services to either end-user applications or other services" [39]. This architecture is beneficial as it offers efficient maintenance and great adaptability [30]. The server consisted of three layers: an API layer *Controllers*, a business logic layer *Services*, and a data access layer *Models*. This architecture is illustrated in Figure 3. The API layer handles the requests, processes the data, forwards the data to the business logic layer, and sends responses back to the client. In the business logic layer, the requested services are performed by translating the request and data into an action, which is forwarded to the data access layer. All database operations such as queries and insertions are handled by the data access layer and the result is sent back to the business logic layer.

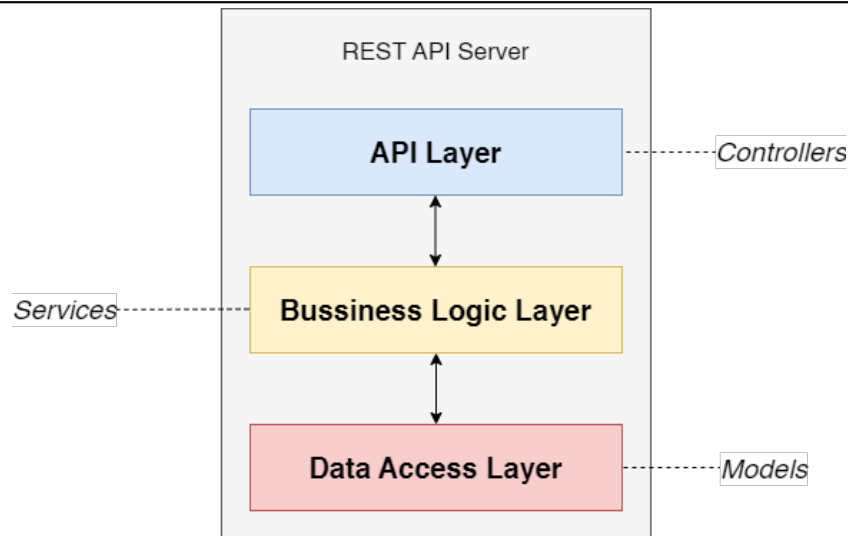


Figure 3 Three-layer architecture in the REST API server. The API Layer communicates with the Business Logic Layer and the Business Logic Layer communicates with the Data Access Layer.

6.3 Database

The database stores all the necessary data for the system to function. This includes customer data and data manually added by the CRM administrators. The data is divided into *collections*, where each *document* represents one instance of the *collection*. In MongoDB, a *collection* represents a group of *documents*[36], similar to how tables are used to organize data in a relational database. A *document* represents one record that contains fields of data [38]. In this case, the data was divided into four collections, as shown in Figure 4: *Users*, *Archives*, *Companies*, and *Records*. The *Users* collection holds user data, such as their name, phone number, and email. Uploaded files and folders for both users and administrators can be found in the *Archives* collection. Company information, such as organization number, the associated users, and contact persons, are stored in the *Companies* collection. Finally, the *Records* collection contains information about assignments and notes associated with customers or companies. The detailed implementation of the collections is presented in Section 8.5.

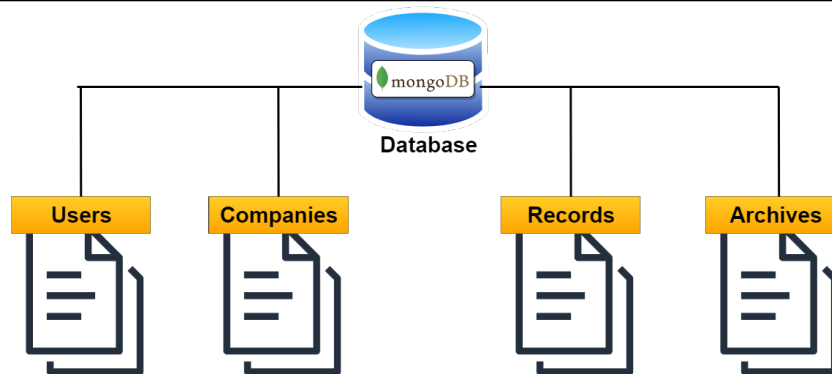


Figure 4 Database structure with the four collections Users, Archives, Records, and Companies.

6.4 Customer-facing web interface

The customer-facing web interface is a presentational layer for the customers. It presents information about the company, the service they offer and allows users to sign up to be contacted by the company. The interface was divided into *pages*, *sections*, and *navigation*. Each page consists of one or more sections, and navigation consists of the header and footer, as shown in Figure 5.

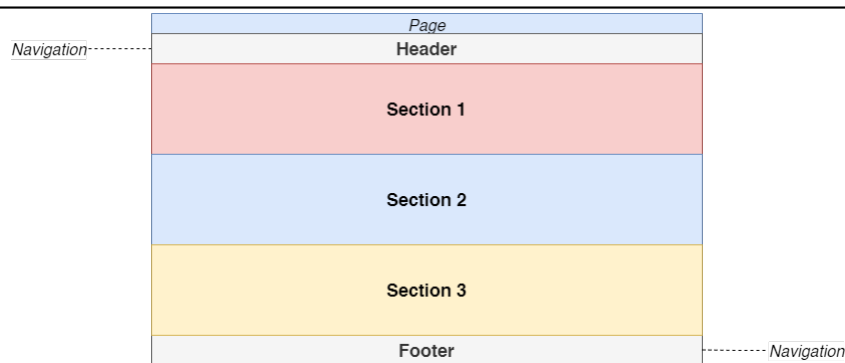


Figure 5 Structure of a page in the customer-facing web interface. A page consists of multiple sections and the two navigation parts Header and Footer.

6.5 Administrator-facing CRM web interface

The administrator-facing web interface is the presentational layer for the users of the CRM. It provides pages where the users can find all the customers, records, and files stored in the CRM. The interface has two navigation components, one that is always visible and navigates between the main pages of the CRM as shown in Figure 6. The other navigation component appears when viewing a customer and allows navigation between the information, records, and archive of a customer. This is seen in Figure 7.

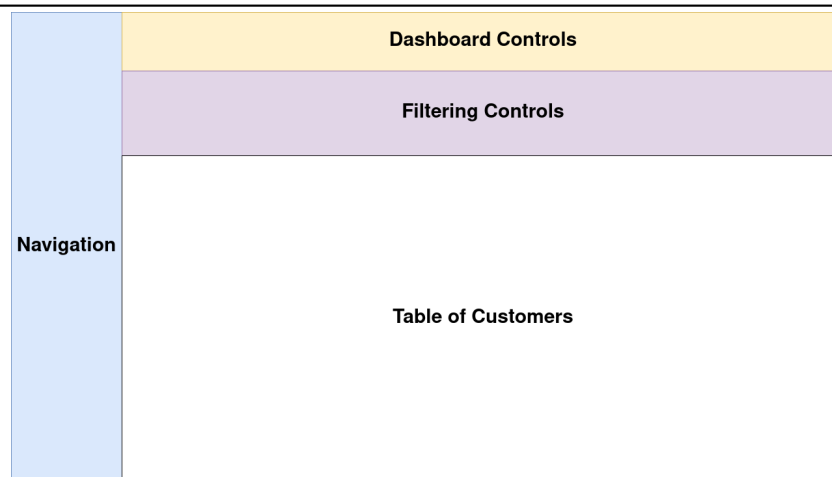


Figure 6 Structure of a main page in the administrator-facing CRM. This figure shows the structure when navigating between the main tabs.

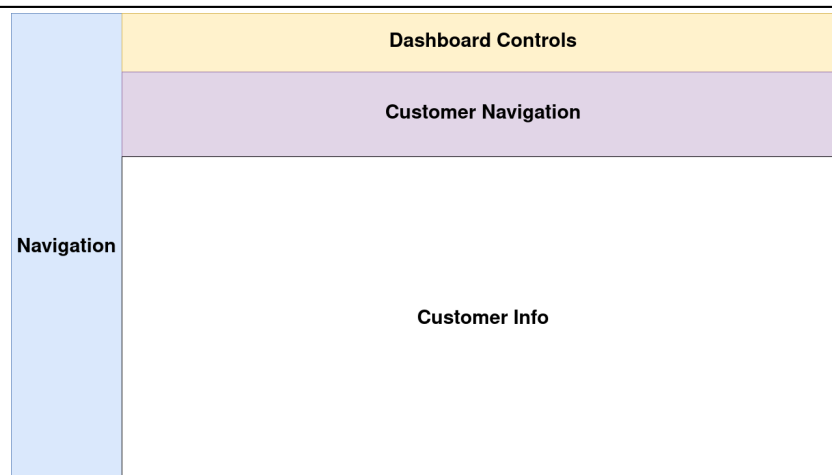


Figure 7 Structure of a customer page in the administrator-facing CRM. This figure shows the structure when looking at customer information.

7 Requirements & evaluation methods

In this section, we will be focusing on the functional and security requirements outlined by the stakeholders. We will discuss the technologies, methodologies, and evaluation techniques employed to ensure that the system meets the specified requirements and adheres to the necessary regulations.

7.1 Requirements

Based on the stakeholder's input, the following functional and security requirements have been identified for the CRM system and the website.

7.1.1 Functional requirements

The functional requirements for the CRM system and website are designed to ensure that the system is user-friendly, efficient, and meets the stakeholder's expectations. These requirements outline the essential features and capabilities that the system must possess to provide value to its users and facilitate effective customer management. The following list provides an overview of the functional requirements for this project:

- The CRM should allow users to easily view and filter all customers in the database using categories, dates, and other relevant criteria.
- The CRM should support both companies and individuals as customers.
- Each customer should have a dedicated page displaying and allowing navigation through all stored information about them.
- Each customer should have a records tab where users can add and view notes and records related to the customer.
- Each customer should have an archives tab where all files stored related to the customer are kept.
- Individuals should be able to be added to companies in the CRM system.
- A general archive page should be available for storing general files for the stakeholder.

- When an individual sign up through the homepage, they should be added as a prospect with the same functionality as a customer, although sorted differently. Prospects can be upgraded to customers within the CRM system.
- The website should offer easy navigation and access to information for users.
- Users should be able to register through a contact form on the website.

7.1.2 Security requirements

Given the sensitive nature of the private customer information stored in the system, our stakeholders emphasized the importance of implementing robust security measures to ensure real-world usage of the CRM and website. Protection against security breaches are crucial to maintaining the trust and confidence of users in the system. To address these concerns, the following security requirements have been identified:

- The system should ensure the confidentiality, integrity, and availability of customer data. The system should be designed to maintain the secrecy of customer data and prevent unauthorized changes.
- Access controls should be in place to restrict unauthorized access to customer data. By implementing stringent access controls, the system should ensure that only authorized individuals can access certain data.
- Encryption techniques should be used to protect sensitive customer information.

The technologies used in the system, such as Auth0 and MongoDB, inherently provide strong security features, including data encryption and robust access controls. To evaluate how these features are utilized, and how potential security vulnerabilities are avoided, the OWASP Node.js security cheat sheet was used, as described in Section 7.2.4.

7.2 Evaluation

The goal of the chosen evaluation methods is to prove that the system meets the functional and security requirements. It provides an objective assessment of the system's overall performance, capability, and robustness. This process allows for the identification and addressing of potential issues or shortcomings, thereby promoting a high-quality, secure, and efficient system.

7.2.1 End-to-end testing

End-to-end testing involves testing the entire system, including all its components, to ensure that it functions as expected and meets the specified requirements [27]. In this project, end-to-end testing was performed to verify that the CRM system and website fulfilled the functional requirements provided by the stakeholder. To conduct this testing, the Cypress³⁵ test framework was utilized. Cypress is a modern, powerful, and easy-to-use testing framework designed for end-to-end testing of web applications [31]. Some of the benefits of using Cypress for this project include:

- Real-time reloading: Cypress automatically reloads tests whenever changes are made to the test code, making it easier to iterate on tests quickly.
- Time-travel debugging: Cypress provides the ability to view snapshots of the application's state at each step of a test, allowing for easier debugging and understanding of test failures.
- Simple API: Cypress offers a simple API for writing and organizing tests, making it more accessible for developers who may not have extensive experience writing tests.

The test cases were designed to cover all functional requirements, including customer and prospect management, filtering, record and file handling, and website navigation and registration. By leveraging the capabilities of the Cypress test framework, we can ensure that the system functions as expected and meets the specified requirements while streamlining the testing process.

Cypress was deemed specifically useful for testing the CRM as it allows for comprehensive end-to-end testing. Tests in Cypress are organized in various test files or modules, known as 'specs'. Each of these test specs is specifically designed to target certain aspects of the functional requirements stated in Section 7.1.1.

7.2.2 Quality testing

When evaluating a website, one method is running tests in the browser that mimic actual user interaction and evaluate a simulated user experience. User experience and website performance are critical to attracting customers, making these types of tests useful.

We evaluated the system using the testing tool Lighthouse³⁶, which is built into the

³⁵cypress.io

³⁶developer.chrome.com/docs/lighthouse

Google Chrome browser³⁷. Lighthouse evaluates the site based on what Google³⁸ have decided are good and useful attributes of a website. It evaluates these in an artificial user environment known as lab data [32].

Lighthouse scores the website from 1-100 in four different categories:

- **Performance:** Lighthouse measures the performance of web pages, evaluating factors such as page load speed, resource optimization, and responsiveness. It provides insights into areas that can be optimized to enhance the overall speed and efficiency of the website.
- **Accessibility:** Lighthouse assesses the accessibility of web pages, examining various aspects such as the proper use of semantic HTML, inclusion of alt text for images, and keyboard navigability. It identifies areas where accessibility improvements can be made to ensure that websites are usable and accessible to all users, including those with disabilities.
- **SEO:** See Section 5.2.2 for a more detailed explanation of SEO.
- **Best Practices:** Lighthouse evaluates whether web pages adhere to industry best practices and standards. It checks for proper implementation of security measures, valid HTML markup, and efficient use of code.

The goal for the audit is to reach a score of 90 or above in each category. According to Google, this is indicative of good performance in the respective category.

7.2.3 Backend testing

To ensure the reliability and stability of the backend, thorough testing is essential. Jest³⁹ has been selected as the testing framework for the backend components of the CRM system and the website. Jest is a popular and widely used JavaScript testing framework that is easy to set up and provides powerful features, such as snapshot testing, parallel test execution, and seamless integration with other tools and libraries.

Jest allows for the creation and execution of unit, integration, and functional tests, providing comprehensive coverage of the backend components. By utilizing Jest, we can ensure that the backend services perform as expected and can handle various scenarios

³⁷google.com/intl/sv/chrome

³⁸google.se

³⁹jestjs.io

and edge cases. This approach helps us maintain the stability and performance of the system, even as new features are added or existing components are modified.

7.2.4 Security checklist

To evaluate the security of the CRM system and website, specifically focusing on the Node.js server application, the OWASP (Open Web Application Security Project)⁴⁰ Node.js Security Cheat Sheet⁴¹ was used. This checklist provides a comprehensive set of guidelines for testing the security of Node.js server applications, ensuring that they adhere to best practices and are protected against common vulnerabilities [12].

- The CRM system and website's Node.js server application were tested against the OWASP Node.js Security Cheat Sheet to identify and mitigate potential security vulnerabilities.
- Security testing will include checks for authentication, authorization, data validation, session management, secure communication, error handling, and other relevant security aspects specific to Node.js server applications.
- The use of the OWASP Node.js Security Cheat Sheet will help ensure that the system complies with GDPR and effectively protects against security breaches.

While the OWASP checklist provides a comprehensive guide for security practices, reaching a certain number of checklist items does not guarantee a completely secure system. In essence, security is not a binary state where a system is either secure or not. Instead, the goal is to minimize vulnerabilities and ensure that the system is resilient against as many potential threats as possible. With 24 items on the OWASP checklist, this project aims to meet as many of these criteria as possible. However, it is essential to understand that each checked item merely increases the system's overall security, reducing its vulnerability to specific threats, however, it does not necessarily imply absolute security. Every system has potential vulnerabilities, and the goal is to mitigate them effectively.

⁴⁰owasp.org

⁴¹cheatsheetseries.owasp.org/cheatsheets

8 Implementation of the system

This section presents a comprehensive overview of the implementation process for the various components of the system. The development and integration of each component have been carried out with the primary goal of providing a seamless and efficient customer relationship management experience for Minafribrev. The following subsections detail the design and implementation of each component.

8.1 Website

The implementation of the website focuses on meeting the stakeholder's key objectives, including trustworthiness, effortless sign-up, and an aesthetically pleasing appearance. To achieve these goals, we have divided the website implementation into three main components: navigation, layout, and interactivity. Each component utilizes specific technologies and strategies to deliver a seamless and user-friendly experience.

8.1.1 Navigation

To provide easy navigation and access to information, the website employs Next.js. Next.js enables seamless navigation between pages by implementing client-side routing with optimized prefetching. This ensures that users can navigate through the website smoothly and efficiently, without encountering delays or unexpected behavior.

Next.js uses a file-based routing system, where each file in the *pages/* directory represents a unique route on the website. This approach simplifies the routing configuration and allows developers to create new pages by simply adding a new file to the *pages/* directory. For example, the *pages/index.js* file within the *pages/* directory corresponds to the main page of the website, accessible at "https://www.minafribrev.se", see Figure 8.

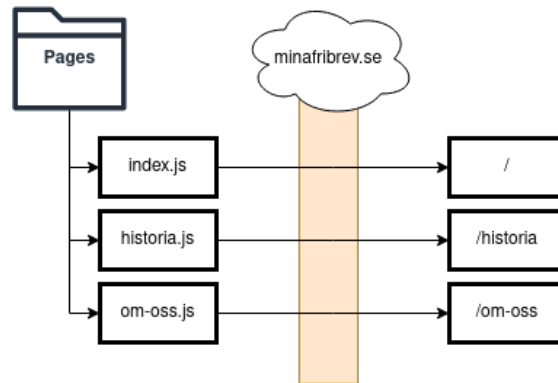


Figure 8 Website navigation, translation from file to a route.

This allows for a streamlined implementation of the navigation bar where HTML anchor elements can be used to direct the user to the correct route. This can be seen in Figure 9.



Figure 9 Website navigation user interface.

By utilizing Next.js for navigation, the website achieves a high level of responsiveness and a streamlined user experience, aligning with the stakeholders' requirements for easy navigation.

8.1.2 Layout

The website's layout is structured using Bootstrap. The layout is divided into three primary sections: the navigation bar, the main content area, and the footer. The navigation bar provides quick access to essential pages, while the main content area displays relevant information in a clear and structured manner. The footer contains additional links and information, such as contact details and legal information. The navigation bar and main content are shown in Figure 10.

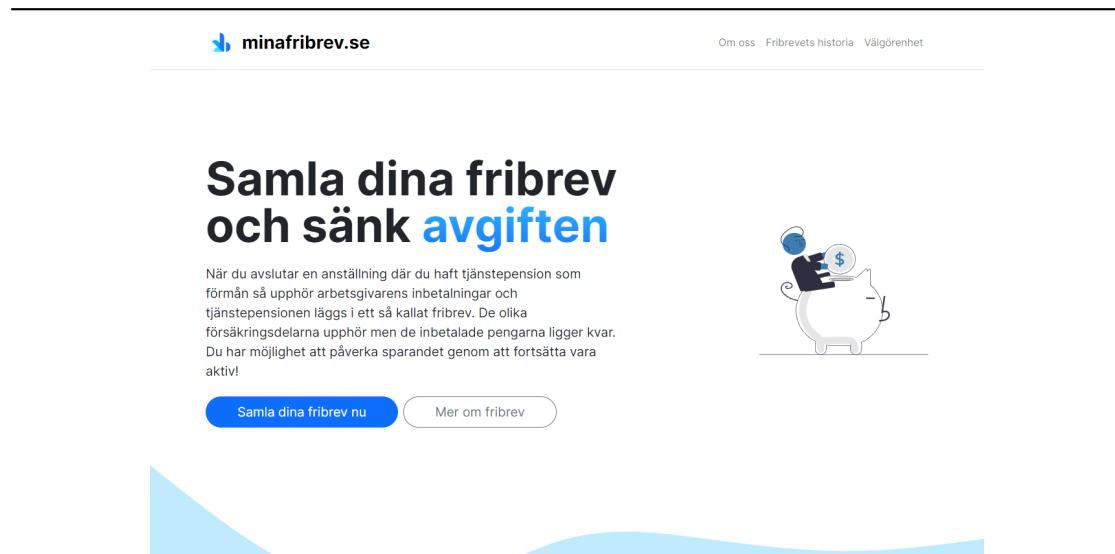


Figure 10 Layout of the landing page when displayed on a desktop size screen.

Bootstrap's grid system, as well as tailwind's grid and flexbox system, is utilized within the main content area to create a responsive and adaptive layout. This system allows for easy arrangement and alignment of content elements, ensuring that the website maintains a visually appealing appearance on various devices and screen sizes as seen in Figure 11.

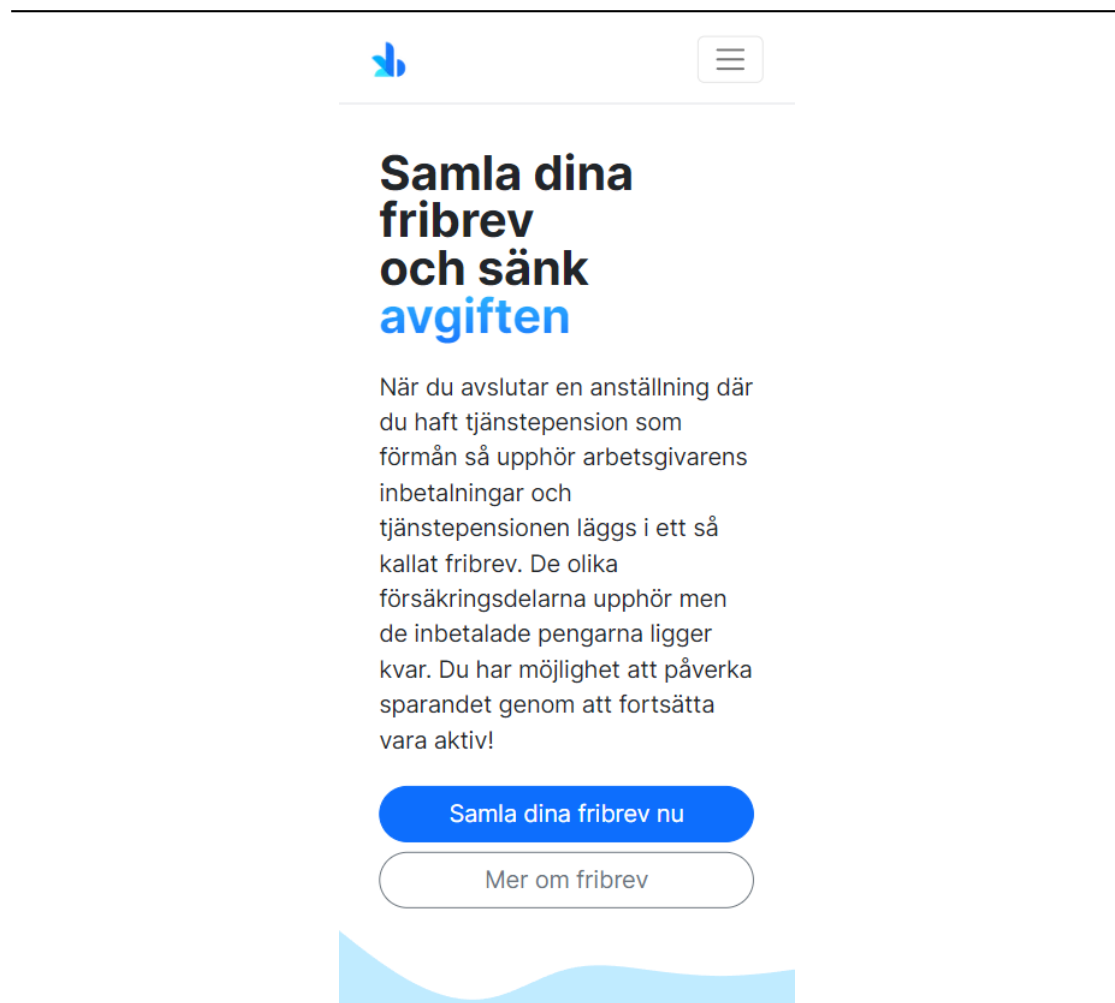


Figure 11 Layout of the landing page when displayed on a phone size screen.

By using Bootstrap for the layout, the website can effectively deliver a consistent experience for users on a wide range of devices.

8.1.3 Interactivity & tracking

The website's interactivity enhances the user experience by providing dynamic elements and real-time responses to user actions. One notable interactive feature is the contact form, which allows users to register on the website. When the form is submitted, a POST request is sent to the API, which processes the registration data and adds the new user to the CRM as a prospect. The contact form features a simple user interface where

the user is presented with several validated text inputs in conjunction with a slider and an array of buttons to select the preferred time of contact, see Figure 12.

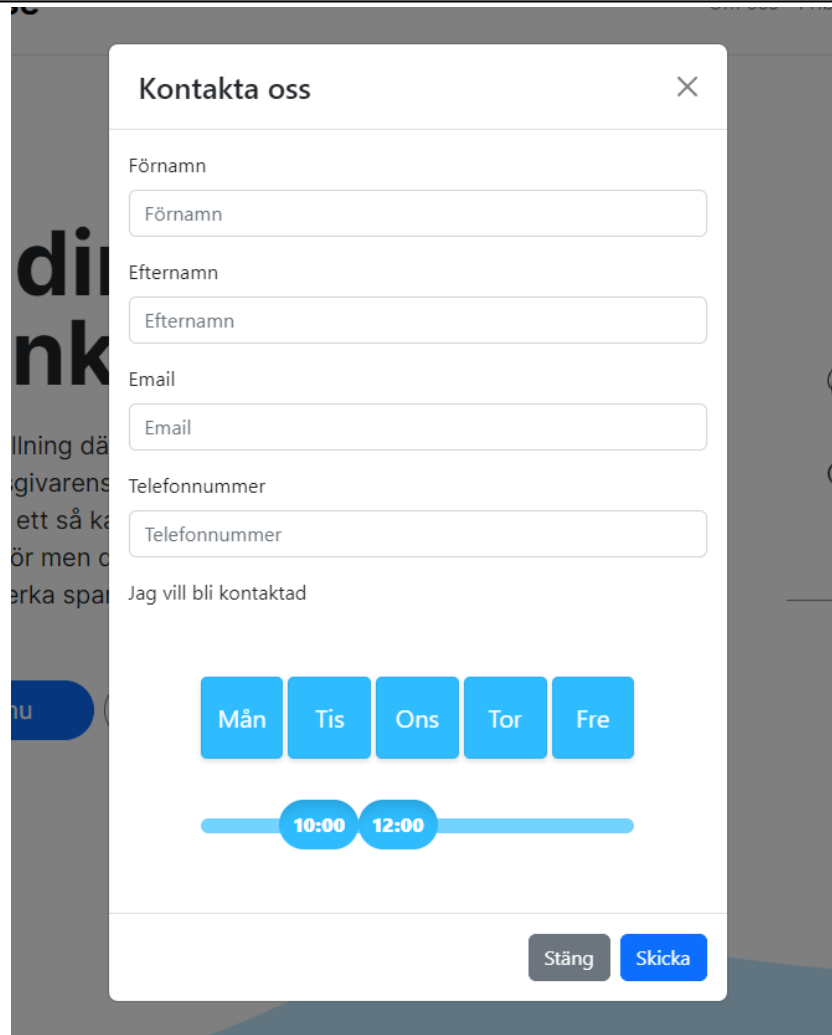


Figure 12 Contact form for signing up for the service on the website.

Another interactive feature of the website is the savings calculator, which aims to engage users and demonstrate the potential savings they can achieve by using the stakeholder's service. The calculator allows users to input the number of days left until their retirement using a slider, and select their income type from a list. The calculator then uses these inputs to estimate the user's potential savings. The user interface of the savings calculator is presented in Figure 13.

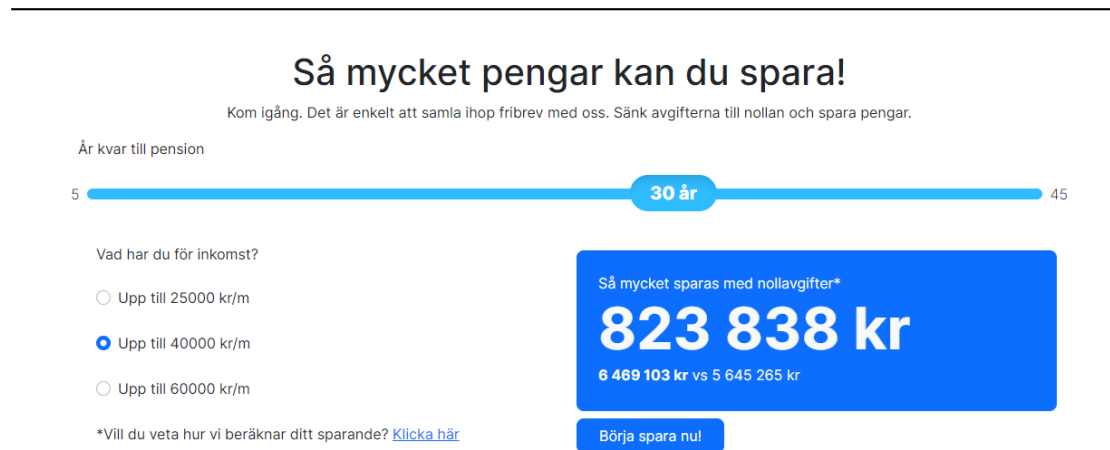


Figure 13 Calculator to inform users of the amount of money that can be saved using the stakeholder's service.

To measure the effectiveness of the website and better understand the needs of the users, the stakeholders expressed the desire to track user visits and conversions. To achieve this, Google Analytics⁴² was implemented as a tracking tool. Google Analytics provides insights on user behavior, such as the number of users visiting the website, the pages they visit, the duration of their stay, and whether they complete desired actions, such as signing up for the service or visiting other pages.

To ensure compliance with privacy regulations and provide transparency to users, a cookie consent pop-up was added to the website. This pop-up is shown to first-time visitors, asking them to accept or decline the use of cookies. If the user accepts cookies, Google Analytics is loaded, and their preferences are saved for future visits. Conversely, if the user declines, Google Analytics is not activated, and no information is gathered or stored about the user in their web browser. The cookie consent pop-up can be seen in Figure 14.

⁴²analytics.google.com/analytics/web

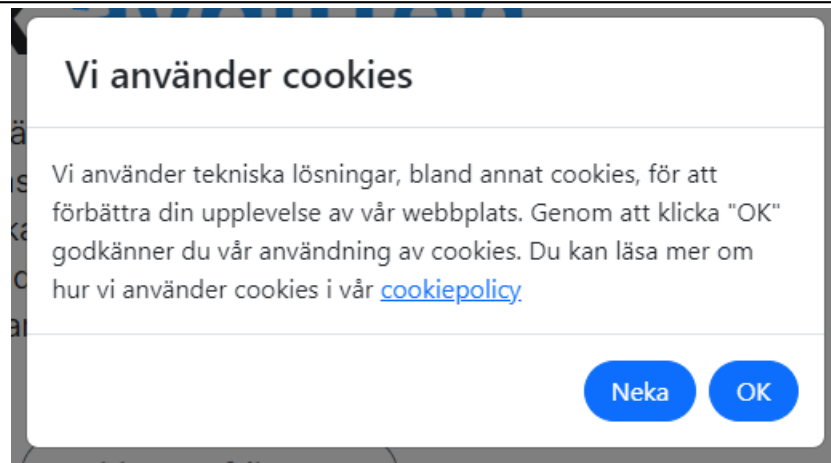


Figure 14 Cookie consent pop-up shown on the first-time visit to the website.

The implementation of Google Analytics, combined with the interactive features of the website, such as the contact form and savings calculator, provides valuable insights to our stakeholders on user behavior and preferences. These insights can be used to further improve the website and tailor the services offered to better meet the needs of the target audience.

8.2 CRM

The CRM has multiple pages that are built together with components. Each page is associated with a distinct navigation route, which serves as an identifier for the system to determine the appropriate page to render. The CRM has four different main pages named Customers, Prospects, Records, and Archives. These pages are consistently accessible through fixed routes, eliminating the need for component re-rendering during navigation. Instead, the unique route assigned to each path instructs the program on which page to render.

To access the CRM, logging in is required. The login page is the standard page supplied by Auth0, described more in Section 8.6. The design of this page can be seen in Figure 15. When the provided authentication details are correct, the user is redirected to the Customers page. This page is described in greater detail in the following subsection. To the left of every page is the navigation bar, as seen in Figure 15.

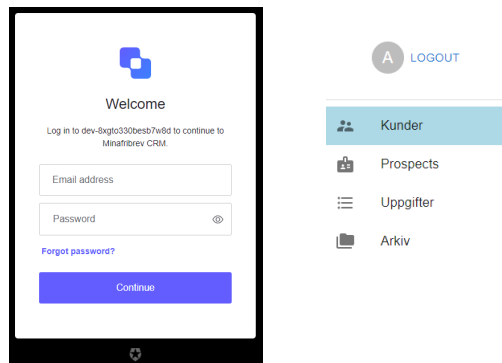


Figure 15 The login screen and navigation bar of the system. The login screen is shown to the left, and the navigation bar is shown to the right.

8.2.1 Customers & Prospects page

The Customers and Prospects pages are designed identically, with the only difference being what information is shown on the two pages. On the Prospects page, only individuals who have voluntarily registered to be contacted are displayed. In contrast, the Customers page exclusively showcases the acquired customer base. This differentiation ensures that the Prospects page presents only potential leads who have expressed interest, while the Customers page exclusively focuses on existing customers who have completed the conversion process. When accessing these pages, a GET request is sent to the server which retrieves all the customers and stores them in an array. The array contains an object consisting of two keys, customers and companies, and their corresponding value is an array consisting of multiple objects. The division between customers and companies was established on the premise that companies themselves can be customers. The essential information is then displayed in a table, such as name, email, and phone number, see Figure 16. The table was created using the DataGrid component from Material-UI. It provided comprehensive searching functionality based on the fields in the component. This was beneficial as it provided a user-friendly and effortless method for quickly finding specific customers. Additionally, a way to filter by company or user was also implemented.

Kunder

Filtrera

Privatpersoner

Företag

Q Sok... DENSITY EXPORT LAGG TILL NY KUND

Typ	Namn	Telefon/orignummer	Kontakt	Senast uppdaterad ↓	Fullmakt
Person	John Carters	+46760503839	johncarter@gmail.com	10:37 04/05/23	✕
Företag	Snerikes	45834594329	Marko. K	11:10 27/04/23	
Företag	Volvo AB	34359873458	Bred. S	11:04 27/04/23	
Företag	Minafribrev.se	09876567890	Markus. G. Vidar. W	18:14 26/04/23	
Företag	Chess.com	09876543211	Martin. B	18:14 26/04/23	
Person	Daniel Ek	0777770000	ceo@spotify.com	15:42 26/04/23	✓
Person	Ramez Shakama	0734568930	ramez.shakama@gmail.com	10:58 26/04/23	✓
Person	Anton Gustavsson	0735468936	anton.gustavsson@gmail.com	10:58 26/04/23	✓

Rader per sida 100 1-19 of 19

Figure 16 The Customers page in the CRM.

8.2.2 Records page

The Records page displays all the records on both customers and companies. Similar to the Customers page, a GET request is sent to the server when accessing this page, which returns an array of objects with all the records. These record objects consist of all the information tied to them, such as their title, their status, and the corresponding identifier of the individual associated with the record. Some of this data is then displayed in the data grid shown in Figure 17. The page also features a filter section where the records can be filtered based on different parameters. The records can, for example, be filtered based on their status, their type, or when they are scheduled.

☰ Uppgifter

Filtrera

Sok personer

Sok företag

Uppgiftstyp
Möte, Samtal, Uppgift

Status
Ny, Pågående, Stängd

Startdatum från

Startdatum till

Slutdatum från

Slutdatum till

Sök...

DENSITY

EXPORT

SKAPA UPPGIFT

Startdatum	Slutdatum	Titel	Status	Uppgiftstyp	Uppgift på
2023-04-05	2023-04-29	Videosamtal	Stängd	Samtal	User
2023-04-27	2023-04-28	Titel	Pågående	Möte	User
2023-04-01	2023-04-28	Titel	Pågående	Samtal	User
2023-04-05	2023-04-29	Möte med Vid.n	Pågående	Möte	Company
2023-04-07	2023-04-27	Längre möte med kunden	Stängd	Möte	User
2023-04-01	2023-04-30	En titel	Pågående	Samtal	User

Rader per sida 100 1-6 of 6

Figure 17 The Records page in the CRM.

8.2.3 Archives

The Archives page of the CRM system features a file browser that allows users to conveniently store general documents and notes. A detailed discussion on the functionality of the file browser will be presented in Section 8.3, and an image of the page can be seen in Figure 18.

↑ root

Search 2 items

Create folder Actions Upload files Options

demoFolder	May 4, 2023, 3:27 PM
ganttchart (2).png	Apr 27, 2023, 3:51 PM

Figure 18 The Archives page in the CRM.

8.2.4 Customer pages (Prospects, Individuals & Companies)

When browsing through the Customers and Prospects pages, you can easily click on a specific individual or company. This action takes you to a dedicated navigation link,

such as /customers/id or /company/id, depending on your selection. The chosen individual's or company's unique identifier (id) is then stored in the customer or company object. With this information, a fetch method is triggered to retrieve all the relevant details about the selected person or organization. Another fetch method is employed to ensure the system remains updated with the latest information, especially when multiple users are simultaneously using the system. The data associated with a company or individual is conveniently organized into three separate pages: an information page, a records page containing information tied to that particular person or company, and an archive page. To provide seamless navigation, a user-friendly navigation bar is positioned at the top of the page, enabling effortless switching between these different sections.

The information page displays general user information such as name, email, social security number, and power of attorney if it is an individual. The information is displayed in Material-UI components called Textfields. This component serves the purpose of enabling users to modify or input new information, if necessary. To ensure that the correct data is sent to the server if changes are made, validation methods have been implemented. For instance, a notable validation requirement is verifying that the social security number adheres to the prescribed format for a valid SSN in Sweden. A visual representation of what the user sees when the validation fails can be seen in Figure 19.

Figure 19 The information page about a customer/prospect.

The company page is structured and implemented in the same manner as the customer's/prospects information page, however, the difference is that the information displayed is about a company, as shown in Figure 20. A DataGrid component is also implemented

on this page to display all individuals connected to a certain company. The DataGrid also provides access to the individual page. Furthermore, it is also possible to connect a user to a company and if this connection is made, the user will also show up as an employee of the company on the company page.

Figure 20 The information page about a company.

The records page for a specific individual or company looks and functions the same as the main Records page described in Section 8.2.2. The difference is that on this page only records tied to the selected individual or company are displayed.

The archive page for an individual is a standard file browser where the administrators can store documents. It is the same component as the main archive page, see Section 8.2.3. The only difference is what files are stored in the file browser. As mentioned above, the main archive is for storing company-related documents and this one stores data related to a specific individual. See Figure 18 for reference.

8.3 File browser

The file browser had to change the structure of the data between the frontend and the backend. In Chonky, which is the frontend, files, and folders were organized using parent and children relationships. However, the backend needs a path for each file and folder. So, the data gets converted from the parent/children structure to the path structure to make everything work together smoothly.

Upon opening the file browser, a fetch request is initiated to retrieve the specific ID's data. Once the data is successfully fetched from the backend, it is then converted to the appropriate format for the frontend to use.

For the file browser to work with the backend, we utilized fetch requests. If the response

from the fetch request did not result in an error, the frontend code was allowed to execute. This way the interface updated quickly, and the File Browser did not have to be fetched from the backend every time an action was made to it.

8.4 REST API server

In Section 6.2, the overall structure for the REST API server was briefly described, including that it consists of a three-layer architecture: an API layer *Controllers*, a business logic layer *Services*, and a data access layer *Models*. This architecture was also illustrated in Figure 3. In this section, we detail each layer's implementation and justify the design.

8.4.1 The API layer: Controllers

Controllers are responsible for handling requests by mapping them to the right service and sending responses back to the client. To distinguish between different types of requests, the *HTTP request method* is used. The HTTP protocol defines multiple HTTP request methods that are used to indicate the type of operation the client wants to perform [6]. Our API utilized the request method *GET* to retrieve data, the *POST* method to submit new data, the *PUT* method to update existing data, and the *DELETE* method to remove data. The separation of different resources, such as customer data and records, is achieved through different *endpoints*, as shown in Figure 21. According to [47], an *API endpoint* is defined as "the provider-side end of a communication channel and a specification of where the API endpoints are located so that APIs can be accessed by API clients". Our API endpoints are prefixed with */api*, followed by the resource they represent. We have separated the resources into four main endpoints: */api/users* for users, */api/companies* for companies, */api/archives* for files, and */api/records* for records. Consequently, each of these main endpoints has its own controller.

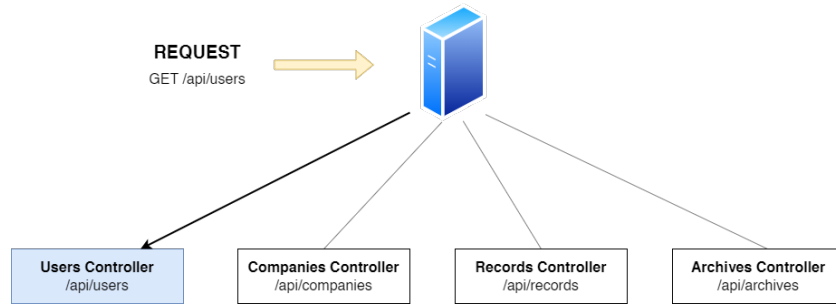


Figure 21 An example illustrating how a request is mapped to a controller using the endpoint. The server receives a GET request to the endpoint /api/users, which is mapped to the Users controller.

Once a request has been mapped to the appropriate controller, the data is processed inside the controller by extracting any information needed to forward to the service. This data comes from the request parameter or request body. For some GET requests, for instance, when retrieving an entire collection, no data is needed and the service is called with no parameters. Figure 22 provides an illustration of the process.

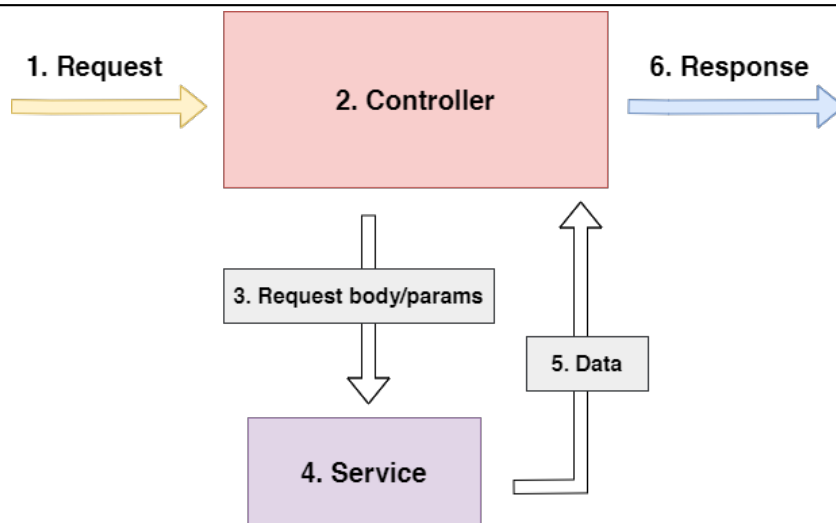


Figure 22 Illustration of the Controller process. The Controller processes the request, extracts the request body/params, passes it down to the appropriate service, and sends a response back to the client.

The format of responses is universal: a *status code* indicating the result of the request, and a JSON response body containing a *message* and a *data* field. In the message field,

additional information about the response is provided; if the request failed validation, an error is displayed. Similarly, if a resource is updated, created, or removed, the message field is populated with "Resource X updated/created/removed successfully", depending on the modified resource and the operation performed. For successful GET requests, the message field is omitted. The data field is used to return data relevant to the request and is included in all responses except error responses. An example response of a successful PUT request that updates an existing user is shown in Figure 23.



```
RESPONSE
status: 200

{
  "message": "User updated successfully",
  "data": {
    "_id": "642be42715470579b39c9267",
    "email": "janedoe@gmail.com",
    "name": {
      "first": "Jane",
      "last": "Doe"
    },
    "ssn": "555555555",
    "phone": "+46760503839",
    "power_of_attorney": false,
    "is_customer": true,
    "updatedAt": "2023-05-04T08:37:26.227Z",
  }
}
```

snappify.com

Figure 23 Example JSON response of a successful PUT request that updates an existing user. The status code is 200, indicating a successful response. In the response body, the message field is affirming the performed operation, and the data field is an object with the user's updated data.

When it comes to error handling, a try-catch block⁴³ is placed inside each controller function that catches errors. When an error occurs, it is logged and a standard response

⁴³developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch

is sent. The response has the status code 500, indicating a server error, and a message in the response body informing that an internal server error has occurred. However, the try-catch block is meant to be the last line of defense against unexpected errors to prevent the server from crashing. Naturally, our approach is to avoid errors from occurring in the first place, as an error being thrown in our case means a function was called inappropriately without enough validation. Furthermore, unaccounted errors are not sent back to the client with a detailed message. Since a detailed error message in the case of an unexpected error could expose too many internal details of the server and thus raise a security concern, a vague "internal server error" is returned instead. The decision to return a vague error was based on OWASP's REST API security recommendations [14].

8.4.2 The business logic layer: Services

Services translate requests into actions by communicating with Models (data access layer). All business logic is handled by Services, including data validation. Essentially, Services are the bridge between the database and the API layer, decoupling the business logic from the communication protocol of the server and database implementation. A Service has a single main responsibility. When a Service is called, the input data is validated and split into one or more actions that are delegated to the appropriate Model, illustrated in Figure 24. Similar to Controllers, they are divided into four main services: *Users* for users, *Companies* for companies, *Records* for records, and *Archives* for files.

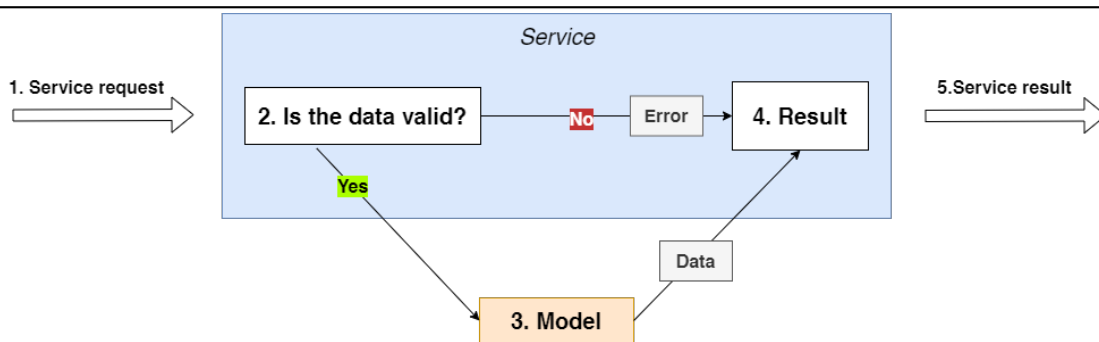


Figure 24 The Service process for handling data and returning results. The input data undergoes validation and is subsequently delegated to the appropriate Model if valid, while returning an error result in case of validation failure.

8.4.3 The data access layer: Models

Models perform database operations and also include data *schemas* to enforce data integrity. Schemas define the shape of a document, such as the allowed fields, their data types, and constraints. The schemas and database operations are implemented using Mongoose. In each Model, methods for creating, reading, updating, and removing data are defined. When creating or updating a document, Mongoose validates the supplied data against the defined schema for the collection. If a constraint is violated, an error is thrown. The four Models and schemas in the data access layer are *Users*, *Companies*, *Records*, and *Archives*. Naturally, the schemas are defined according to the database design presented in Section 8.5. By utilizing schemas, we gain another security layer and enforce data integrity.

8.5 Database design

As mentioned in Section 6.3, the database consists of four collections: Users, Archives, Companies, and Records. This was also illustrated in Figure 4. In the following subsections, the exact implementation of each collection is provided and explained.

Disclaimer

Before we delve into the specifics of the database design, it is important to address potential security concerns related to exposing the names and types of the fields in our document-based database, MongoDB.

In a traditional SQL database, knowing the structure of the database (i.e., table names, column names, and their types) can pose a security risk as it could be exploited in SQL injection attacks [18]. However, MongoDB, being a NoSQL database, is not vulnerable to such attacks.

While it is crucial to protect the content of the database, revealing the schema – the names and types of fields – does not pose a direct security threat. This information does not provide an attacker with any actionable means to exploit the system or gain unauthorized access to sensitive data.

Furthermore, it is vital to note that before an attacker could even attempt to exploit the database schema or perform any kind of attack on the database, they would first need to bypass the authentication system.

8.5.1 Users collection

The Users collection consists of documents, one for each user. These documents include several fields, such as email, name, social security number (SSN), phone number, company, power of attorney, and preferred time of contact. All fields in a user document including their data types are shown in Figure 25.

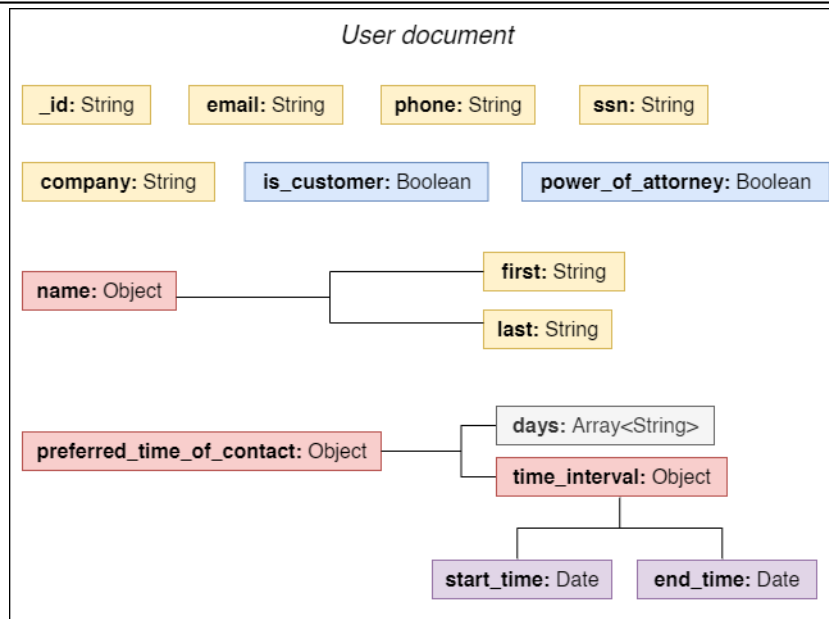


Figure 25 Data fields in a User document. The bold part represents the name of the field, followed by the data type.

8.5.2 Archives collection

The Archives collection contains both files and folders. Consequently, the model for the data depends on whether the document belongs to a file or a folder. Most of the fields are mutual, such as *path*, *name*, and *belongs_to*. The exact models for files and folders are presented in Figures 26 and 27, respectively.

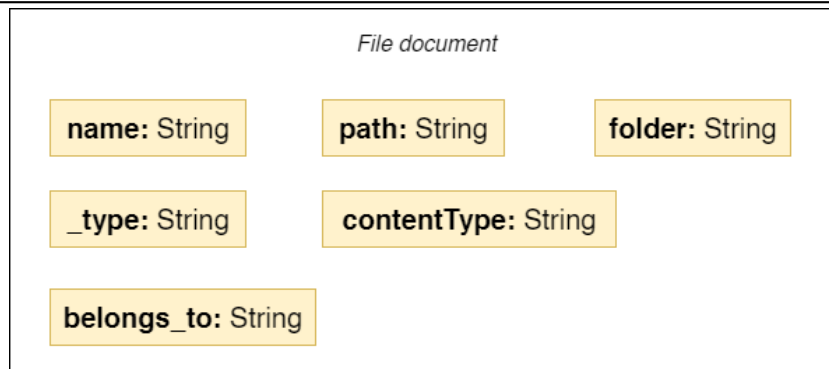


Figure 26 Data fields in a File document. The bold part represents the name of the field, followed by the data type.

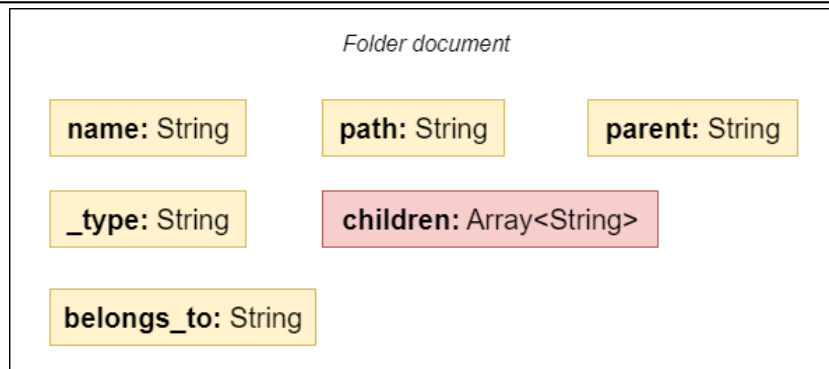


Figure 27 Data fields in a Folder document. The bold part represents the name of the field, followed by the data type.

8.5.3 Companies collection

The Companies collection stores information about companies, such as organization number, name, contact persons, and employees. In Figure 28, the fields in a company document and their data types are provided.

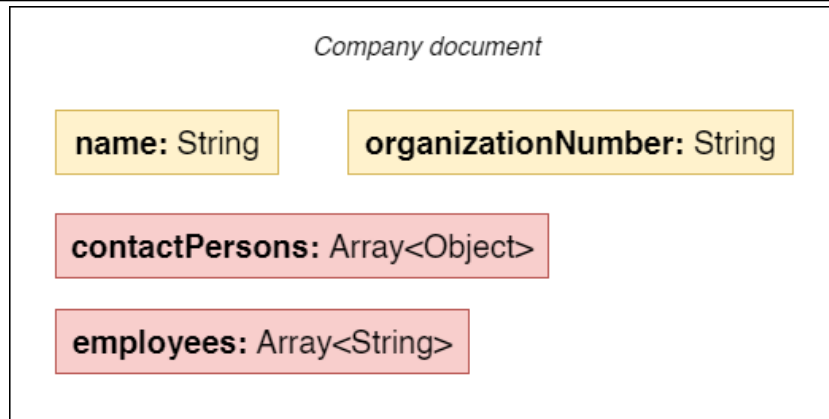


Figure 28 Data fields in a Company document. The bold part represents the name of the field, followed by the data type.

8.5.4 Records collection

Records hold information about an event related to a company or customer, added by the administrators. Each Record can have notes associated with it, as well as a category, status, title, description, and date. All fields in a Record document, alongside their data types, are shown in Figure 29.

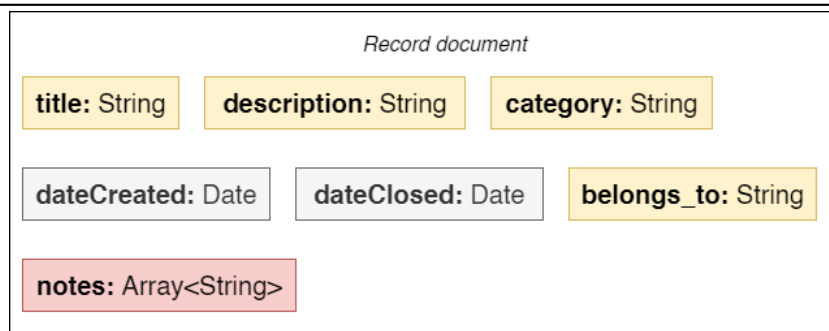


Figure 29 Data fields in a Record document. The bold part represents the name of the field, followed by the data type.

8.6 Integrating authentication: Auth0

To prevent unauthorized access to the CRM and REST API, the identity provider Auth0 was used, as described in Section 5.6. For our purposes, any user that exists in the administrator database is granted full access to the system. This design was created in compliance with the preferences and demands of the stakeholders. Administrators are added and managed on Auth0's platform, and the integration of the authentication consisted of two steps. First, the login page in the CRM was configured to redirect to Auth0's supplied login page. Second, a middleware was added to each API endpoint. The middleware verifies the authentication and authorization of an administrator, granting or denying access based on the results. The process is illustrated in Figure 30. All functionality such as logging in, logging out, and checking if a user is authenticated was provided by Auth0. Our work merely consisted of integrating the functions and handling the results of the actions.

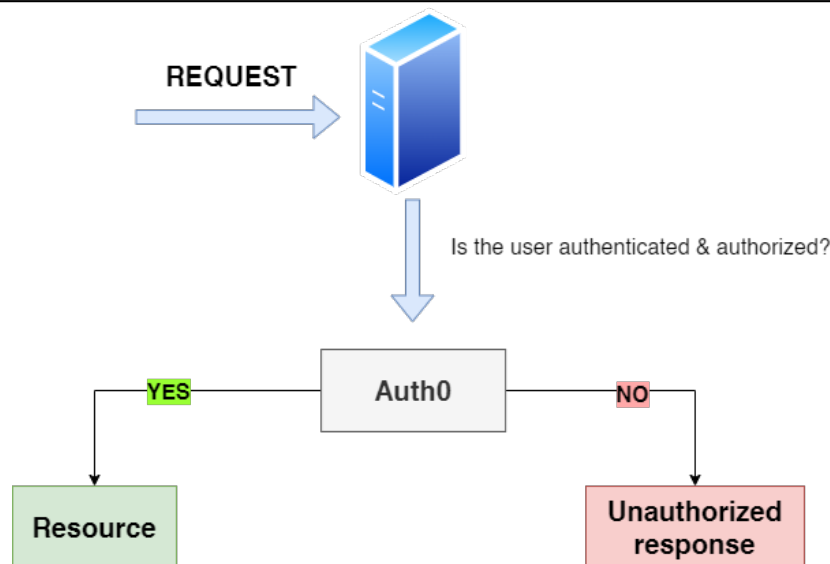


Figure 30 Access verification process in the REST API. The server employs Auth0 to validate the authentication and authorization status, granting access to the requested resource accordingly.

9 Results

This section presents the results obtained from the evaluation methods and metrics outlined in Section 7.2. These evaluations were designed to ensure that the developed

system meets the functional and security requirements set by the stakeholders, as well as to assess the overall quality of the website and CRM application. The evaluation results will provide valuable insights into the system's performance, accessibility, usability, and security, allowing us to gauge its success in addressing the needs of the stakeholders and end-users. The implication of the results is discussed in Section 10.

9.1 End-to-end testing

End-to-end tests were performed on the CRM using Cypress, as described in Section 7.2.1, to check the basic functionality of the website and also to ensure that the functional requirements were met. The requirements were defined in Section 7.1.1.

The test modules are enumerated in the ensuing list, followed by the test results in Figure 31.

- **Authentication**
- **Basic routing**
- **Creating and Deleting a Customers and Companies**
- **Adding a Customer to a Company as an Employee**
- **File-system**
- **Filtering Customers and Companies**

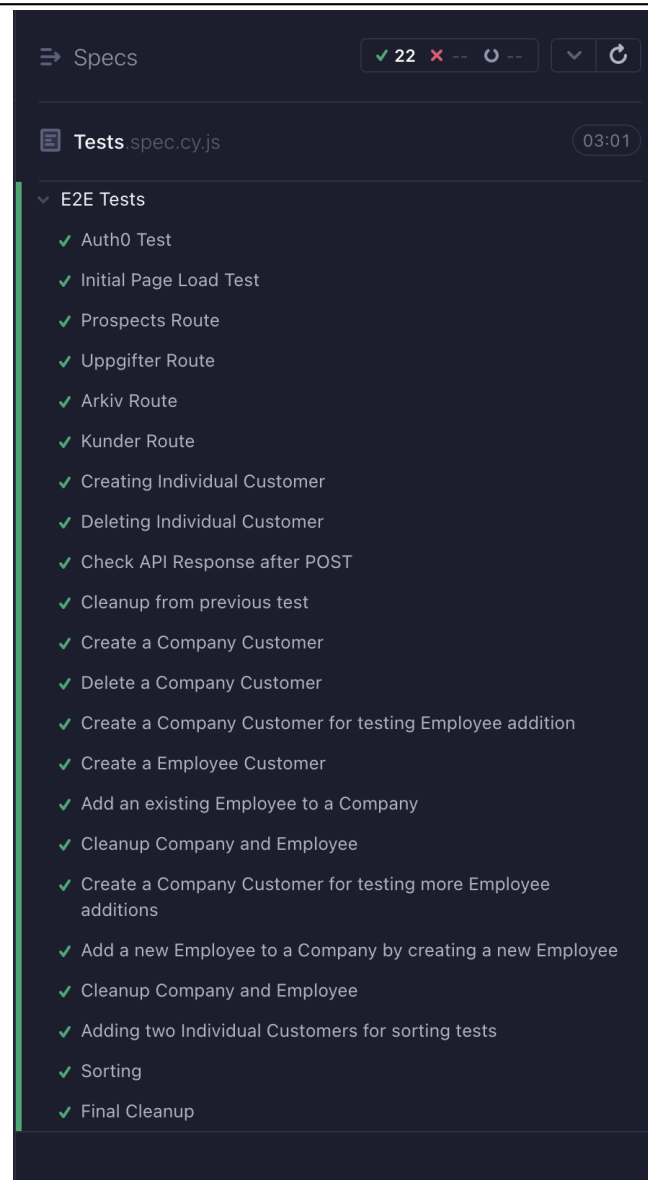


Figure 31 Results from the Cypress tests. The 22 passed tests are shown.

9.2 Backend testing

The backend was tested with Jest to ensure it performs as intended, as described in Section 7.2.3. In total, 103 tests were conducted and all the tests passed, as shown in Figure 32. The tests consisted of API calls to each endpoint and assessed error handling,

response outputs, and database operations. These evaluations covered approximately 72% of all the backend code. A more detailed result of the code coverage is provided in Table 1.

```

PASS __tests__/users/createProspect.test.js
PASS __tests__/records/updateNote.test.js
PASS __tests__/users/deleteUser.test.js
PASS __tests__/records/getRecordsFrom.test.js
PASS __tests__/users/updateUser.test.js
PASS __tests__/records/createRecord.test.js
PASS __tests__/companies/updateCompany.test.js
PASS __tests__/users/createUser.test.js
PASS __tests__/companies/deleteCompany.test.js
PASS __tests__/companies/getCompany.test.js
PASS __tests__/records/createNote.test.js
PASS __tests__/users/getUser.test.js
PASS __tests__/records/deleteRecord.test.js
PASS __tests__/records/getRecordById.test.js
PASS __tests__/companies/createCompany.test.js
PASS __tests__/records/deleteNote.test.js
PASS __tests__/records/updateRecord.test.js
PASS __tests__/archives/createFile.test.js
PASS __tests__/archives/moveFile.test.js
PASS __tests__/archives/deleteFolder.test.js
PASS __tests__/archives/getArchive.test.js
PASS __tests__/records/getRecords.test.js
PASS __tests__/companies/getCompanies.test.js
PASS __tests__/users/getUsers.test.js
PASS __tests__/users/getProspects.test.js
PASS __tests__/users/getUnemployedUsers.test.js

Test Suites: 26 passed, 26 total
Tests:       103 passed, 103 total
Snapshots:   0 total
Time:        160.948 s, estimated 164 s

```

Figure 32 Backend test results conducted with Jest. The result shows 103 passed tests.

Resource	Statements	Branches	Lines	Functions
All files	73.4%	53.2%	72.4%	73.5%
Controllers	74.1%	25%	75%	74.1%
Services	65.6%	57.4%	63.3%	65.6%
Models	70.2%	60.1%	64.7%	70.7%

Table 1 Backend tests code coverage results. The percentage values represent the code coverage.

9.3 Quality testing

A Lighthouse audit was conducted to assess the quality of the website. The website achieved a perfect score of 100 in all categories: Performance, Accessibility, Best Practices, and SEO as seen in Figure 33.

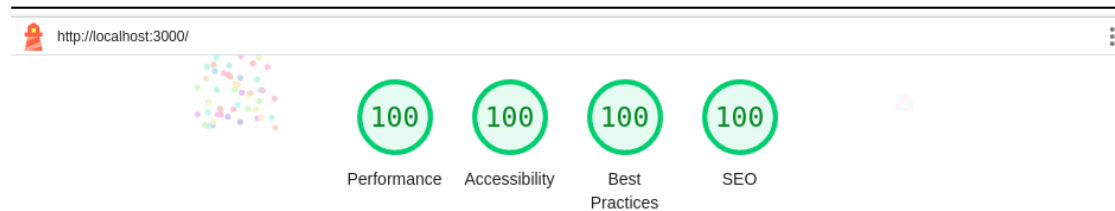


Figure 33 Scores from Lighthouse audit performed on the website.

Additionally, the Lighthouse audit provided a detailed analysis of the website's performance metrics. The report from lighthouse can be found in Figure 34.

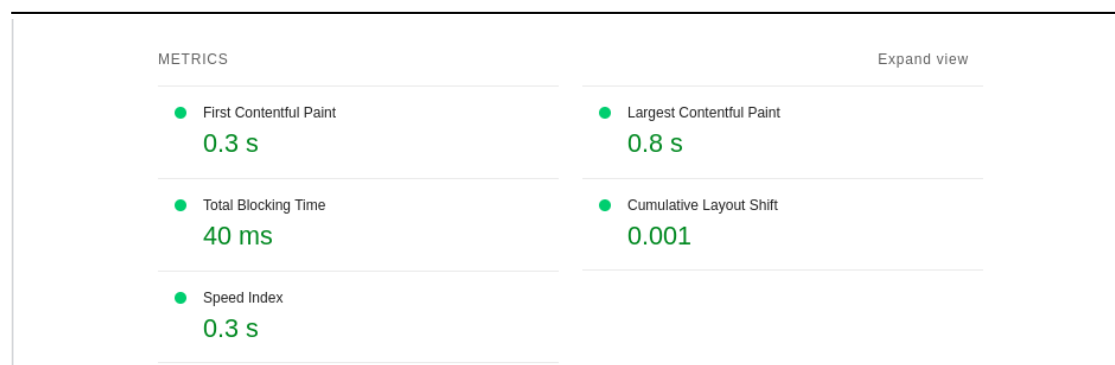


Figure 34 Performance metrics from Lighthouse audit performed on the website.

- **First Contentful Paint (FCP): 0.3s** - FCP measures the time it takes for the first piece of content to be rendered on the screen. A fast FCP of 0.3s indicates that users can quickly start engaging with the website content.
- **Largest Contentful Paint (LCP): 0.8s** - LCP measures the time it takes for the largest piece of content to be rendered on the screen. An LCP of 0.8s demonstrates that users can rapidly access the most relevant information.
- **Total Blocking Time (TBT): 40ms** - TBT quantifies the total duration during which the main thread is blocked and unresponsive to user input. A low TBT

of 40ms indicates that users can interact with the website without any significant delays.

- Cumulative Layout Shift (CLS): 0.001 - CLS measures the visual stability of the website by capturing unexpected layout shifts. A minimal CLS of 0.001 signifies that the website's layout is stable and does not disrupt the user experience.
- Speed Index: 0.3s - Speed Index measures the average time it takes for the visible parts of the page to be displayed. A low Speed Index of 0.3s indicates that users can quickly view and interact with the website's content.

9.4 Security checklist

This section presents the results of an audit performed on the server using the OWASP Node.js Security Cheat Sheet, as summarized in Table 2. For the convenience of the readers, the detailed explanations for each recommendation are not included in this text. Instead, comprehensive information about each item can be found directly on the OWASP website⁴⁴.

⁴⁴cheatsheetsseries.owasp.org/cheatsheets/Nodejs_Security_Cheat_Sheet.html

Recommendation	Status
APPLICATION SECURITY	12/14
Use flat Promise chains	✓
Set request size limits	✓
Do not block the event loop	✓
Perform input validation	✓
Perform output escaping	✓
Perform application activity logging	×
Monitor the event loop	×
Take precautions against brute-forcing	✓
Use Anti-CSRF tokens	✓
Remove unnecessary routes	✓
Prevent HTTP Parameter Pollution	✓
Only return what is necessary	✓
Use object property descriptors	✓
Use access control lists	✓
ERROR & EXCEPTION HANDLING	3/3
Handle uncaughtException	✓
Listen to errors when using EventEmitter	✓
Handle errors in asynchronous calls	✓
SERVER SECURITY	2/2
Set cookie flags appropriately	✓
Use appropriate security headers	✓
PLATFORM SECURITY	5/5
Keep your packages up-to-date	✓
Do not use dangerous functions	✓
Stay away from evil regexes	✓
Run security linters	✓
Use strict mode	✓
Adhere to general application security principles	✓
TOTAL	22/24

Table 2 Results from OWASP Node.js Security Cheat Sheet audit

10 Discussion

In this section, the evaluation results and their implications will be discussed. The website, CRM, and backend will be discussed separately. There are no results that provide proof of performance, security, or robustness for all nodes in the system working con-

currently. Therefore, a proper evaluation of the system as a whole can not be established. See Section 9 for the concrete results.

10.1 Website

The website serves as the primary interface between Minafribrev and their customers, making it essential to create a user-friendly and visually appealing platform. As seen in the previous section, the quality testing results showed perfect scores in Performance, Accessibility, Best Practices, and SEO. These scores both fulfill our set goals and indicate that the website is performing well according to Lighthouse metrics. However, it is important to note that some of these categories, such as accessibility, may be misleading. As described in Section 7.2.2, accessibility measures, for example, the proper use of HTML and the inclusion of alternative text for images. It does not, however, measure how a real user would interact and perceive the website. Due to time constraints, real user tests could not be performed; therefore, only the Lighthouse metrics and their implications can be discussed.

The website's high performance score demonstrates that it loads quickly and efficiently, ensuring a positive user experience. A fast-loading website can significantly impact user retention and conversion rates, making this metric especially important for the success of Minafribrev's online presence.

The perfect score in Accessibility highlights the website's ability to cater to a wide range of users, including those with disabilities or using assistive technologies. This attribute is crucial in providing equal access to Minafribrev's services and ensuring that all potential customers can effortlessly navigate the website.

The Best Practices and SEO scores showcase the website's adherence to industry standards and its ability to rank well in search engines. This aspect is critical for attracting new customers and increasing the visibility of Minafribrev's services.

10.2 CRM

The CRM system forms a core part of the overall architecture and plays an essential role in effectively managing customer relationships. The CRM system was designed and built to meet the specific needs of the stakeholders, with a focus on handling customer data, records, and files effectively.

The results from the end-to-end testing indicated that the CRM system satisfied all func-

tional requirements without any errors, demonstrating its reliability and effectiveness. The system is designed to support both companies and individuals as customers, providing a wide range of functionality to filter, categorize, and manage customer information. It is noteworthy, however, that these tests could be exposed to bias given that the development team was responsible for writing both the code for the CRM and the tests. Additionally, the tests were primarily written to check the fulfillment of functional requirements, implying the potential existence of bugs in edge cases or non-intended usage scenarios.

10.3 Backend system

The backend system provides a solid foundation for the overall architecture, ensuring smooth operation of the CRM system and the website. Testing revealed that the server logic was relatively robust and independent of the frontend, offering significant advantages for future adaptability and scalability. The flexibility of the backend design allows for the possibility of developing new frontends on top of the existing backend infrastructure, ensuring that the system can evolve to meet the changing needs and requirements of the stakeholders over time. However, the data presented in Table 1 indicates that there is scope for improvement in terms of code coverage. Across all areas, the code coverage fell below 80%, suggesting inadequate testing in certain portions of the code. This deficiency is particularly evident in the code coverage for branches, which revealed a significant number of uncovered cases. On a positive note, the system successfully passed 103 tests, providing coverage for approximately 73.4% of the entire codebase. While this result is satisfactory to some extent and implies a reasonable level of robustness, further efforts are required to enhance code coverage and ensure comprehensive testing.

In terms of security, the system was designed with a significant emphasis on adhering to the OWASP checklist to safeguard against common security vulnerabilities. However, it must be recognized that no system can be completely immune from security threats. Digital security is an ongoing endeavor, with developers continuously working to keep up with evolving methods of exploitation. Therefore, even a system that meets all current security recommendations can potentially become vulnerable over time.

Two significant points from the OWASP checklist, namely "*Perform application activity logging*" and "*Monitor the event loop*", were not implemented in this project. These points pertain to the monitoring aspects of an application, which we recognize as critical for ensuring system security. However, due to time constraints, we could not incorporate any sophisticated logging mechanisms. Although the backend does log any encountered endpoint errors in the console, it does not store them in a database nor does it notify an

administrator. Furthermore, these two OWASP points suggest monitoring all events, not merely errors. Therefore, for optimal security, the development of a comprehensive logging system, or the integration of a suitable pre-existing one, would be necessary.

Furthermore, our security evaluation had limitations, most notably the lack of penetration testing, as outlined in Section 3.6. Despite achieving 22 out of the 24 items on the OWASP checklist, implying a robust defense against known vulnerabilities, it does not guarantee protection against unknown vulnerabilities or future threats. Security must therefore be viewed as an ongoing commitment, requiring continuous monitoring, updating, and improvements to maintain the integrity of the system.

11 Future work

During the initial phase of the project, we engaged in extensive discussions with the stakeholders to ascertain their envisioned outcomes for the final product and their expectations from our team. However, due to time constraints, certain ideas proposed by the stakeholders had to be deferred. In this section, we will address a selection of the ideas that were raised for the ideal end product.

11.1 Fullmaktskollen

One substantial future improvement to the website would be to implement Fullmaktskollen⁴⁵, which is a tool used to manage powers of attorney for pensions and life assurance policies. It allows users to both sign and revoke the powers of attorney. The original intention was to integrate Fullmaktskollen into the website, allowing users to grant the stakeholders access to handle their insurance paperwork when signing up for their services. This tool would've made the workload easier for the company since they currently must contact the customer to receive a power of attorney. Considering this constraint, the CRM system is not designed for storing and managing the powers of attorney efficiently. Without access to Fullmaktskollen's service, it was difficult to plan for future integration. However, the powers of attorney can still be stored as uploaded documents in the archives part of the CRM.

⁴⁵fullmaktskollen.se/in-english/

11.2 Customer interface

At present, the only interface provided for the customers is the website described in Section 8.1. However, in order to enhance the customer experience further, it would be advisable to develop a dedicated platform specifically for the customers. On the platform, they would then be able to sign in and access their pension funds directly. This new platform would allow customers to monitor and manage their investments, providing them with greater control and flexibility.

11.3 Multiple employee accounts

Another functionality that would significantly enhance the usability of the CRM is the capability to track individual employee accounts, roles, and activities. Presently, all actions are treated uniformly as belonging to a single user. However, as the business expands and the number of CRM users increases, this limitation will become increasingly evident. It will become challenging to assign customers to specific employees and effectively monitor unattended potential leads. By implementing a feature to track different employee accounts, roles, and activities, these issues can be addressed, thereby enhancing the overall efficiency and effectiveness of the CRM.

12 Conclusions

It can be concluded that the website and CRM system have successfully met the requirements set both by the stakeholders and by us. The CRM system we have built has the capabilities for managing customer information and interactions, including keeping notes and storing customer files, which is crucial for any business that values customer relationship management. On the other hand, the website can effectively provide information about the company's offerings and allows users to sign up for their services, which is essential for customer acquisition and retention.

Furthermore, it is important to note that while there may be room for improvement in the future, the stakeholders expressed satisfaction with the functionality of both products. The website and CRM system can serve their intended purposes effectively and efficiently, and if implemented can provide a solid foundation for the company's online presence and customer management. It does however need more features, as described in future work, for the product to reach its maximum potential.

From a broader perspective, the successful implementation of the website and CRM

system showcases the importance of stakeholder communication. It also highlights the value of using modern technologies and tools in building a robust and efficient system that meets the needs of the stakeholders.

References

- [1] angular vs react vs vue | npm trends. Accessed 2023-04-24. [Online]. Available: <https://npmtrends.com/angular-vs-react-vs-vue>
- [2] An empirical analysis of the value of complete information for ecrm models. Accessed 2023-04-25. [Online]. Available: <https://www.jstor.org/stable/25148730>
- [3] Establishing quality online presence for veterinary practices. Accessed 2023-04-25. [Online]. Available: <https://www.magonlinelibrary.com/doi/abs/10.12968/vetn.2013.4.2.64>
- [4] HTML standard. Accessed 2023-04-17. [Online]. Available: <https://html.spec.whatwg.org/multipage/webappapis.html#scripting>
- [5] HTTP methods - REST API tutorial. Accessed 2023-04-17. [Online]. Available: <https://restfulapi.net/http-methods/>
- [6] Http request methods. Accessed 2023-05-02. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
- [7] The importance of user testing: What every business should know. [Online]. Available: <https://marvelapp.com/blog/importance-user-testing/>
- [8] JavaScript versions. Accessed 2023-04-17. [Online]. Available: https://www.w3schools.com/js/js_versions.asp
- [9] Javascript vs python: which is better for web development and rich text editors | TinyMCE. Accessed 2023-04-24. [Online]. Available: <https://www.tiny.cloud/blog/python-vs-javascript/>
- [10] Magic quadrant for access management. Accessed 2023-05-09. [Online]. Available: <https://www.gartner.com/doc/reprints?id=12BLL9EIV&ct=221104&st=sb>
- [11] NoSQL vs SQL databases | MongoDB. Accessed 2023-04-17. [Online]. Available: <https://www.mongodb.com/nosql-explained/nosql-vs-sql>
- [12] OWASP secure coding practices-quick reference guide | OWASP foundation. Accessed 2023-04-27. [Online]. Available: <https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/>
- [13] React UI components libraries: Our top picks for 2023. Accessed 2023-04-17. [Online]. Available: <https://kinsta.com/blog/react-components-library/>

- [14] “REST Security - OWASP Cheat Sheet Series,” accessed 2023-05-09. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html
- [15] Ruby vs JavaScript. Accessed 2023-04-24. [Online]. Available: <https://www.boottrails.com/blog/ruby-vs-javascript/>
- [16] To tail or not to tail. Accessed 2023-05-02. [Online]. Available: <https://www.telerik.com/blogs/tailwind-not-tailwind-question>
- [17] To tail or not to tail. Accessed 2023-05-02. [Online]. Available: <https://blog.logrocket.com/comparing-tailwind-css-bootstrap-time-ditch-ui-kits/>
- [18] Top 10 2013-a4-insecure direct object references - OWASP. [Online]. Available: https://wiki.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References
- [19] Top advantages of python over other programming languages. Accessed 2023-05-25. [Online]. Available: <https://www.edoxi.com/studyhub-detail/advantages-of-python-over-other-programming-languages>
- [20] TypeScript vs. JavaScript for building large apps | proxify.io. Accessed 2023-04-17. [Online]. Available: <https://proxify.io/articles/typescript-vs-javascript>
- [21] United nations sustainable development goals. Accessed 2023-04-17. [Online]. Available: <https://sdgs.un.org/goals>
- [22] United nations sustainable development goals, goal 8. Accessed 2023-04-17. [Online]. Available: <https://sdgs.un.org/goals/goal8>
- [23] Usage statistics and market share of PHP for websites, april 2023. Accessed 2023-04-17. [Online]. Available: <https://w3techs.com/technologies/details/pl-php>
- [24] Usage statistics of JavaScript as client-side programming language on websites, april 2023. Accessed 2023-04-17. [Online]. Available: <https://w3techs.com/technologies/details/cp-javascript>
- [25] What is a JavaScript meta-framework? - the lean software boutique. Accessed 2023-04-17. [Online]. Available: <https://www.ombulabs.com/blog/javascript/what-is-a-javascript-meta-framework.html>
- [26] What is crm? the complete crm guide. Accessed 2023-04-20. [Online]. Available: <https://www.oracle.com/se/cx/what-is-crm/>

-
- [27] What is end to end testing? guide to e2e testing | functionize. Accessed 2023-04-27. [Online]. Available: <https://www.functionize.com/automated-testing/end-to-end-e2e-testing>
- [28] What is gdpr, the eu:s new data protection law. Accessed 2023-04-20. [Online]. Available: <https://gdpr.eu/what-is-gdpr/>
- [29] What is SEO - search engine optimization? Accessed 2023-05-15. [Online]. Available: <https://searchengineland.com/guide/what-is-seo>
- [30] What is SOA? - service-oriented architecture explained - AWS. Accessed 2023-05-15. [Online]. Available: <https://aws.amazon.com/what-is/service-oriented-architecture/>
- [31] Why cypress? | cypress documentation. Accessed 2023-04-27. [Online]. Available: <https://docs.cypress.io/guides/overview/why-cypress>
- [32] Why lab and field data can be different (and what to do about it). Accessed 2023-05-02. [Online]. Available: <https://web.dev/lab-and-field-data-differences/>
- [33] P. Borrelli. Angular vs. react vs. vue.js: Comparing performance. Accessed 2023-04-24. [Online]. Available: <https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/>
- [34] Document data format: BSON — java sync. Accessed 2023-04-17. [Online]. Available: <https://www.mongodb.com/docs/drivers/java/sync/current/fundamentals/data-formats/document-data-format-bson/>
- [35] C. Bull, “Strategic issues in customer relationship management (crm) implementation,” *Business Process Management Journal*, vol. 9, no. 5, pp. 592–602, 2003. [Online]. Available: <https://doi.org/10.1108/14637150310496703>
- [36] "Collections". MongoDB. Accessed 2023-04-17. [Online]. Available: <https://www.mongodb.com/docs/compass/current/collections/>
- [37] M. G. Daniel Belanche, Luis V.Casalo, “Website usability, consumer satisfaction and the intention to use a website: The moderating effect of perceived risk,” *Journal of Retailing and Consumer Services*, vol. 19, no. 1, pp. 124–132, Jan. 2012, publisher: Pergamon. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0969698911001160>
- [38] "Documents". MongoDB. Accessed 2023-04-17. [Online]. Available: <https://www.mongodb.com/docs/compass/current/documents/>

- [39] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling, *IBM: Patterns: service-oriented architecture and web services.*, Jan. 2004, accessed 2023-05-08.
- [40] Express JS development - benefits for enterprise applications. Accessed 2023-04-17. [Online]. Available: <https://www.rlogical.com/blog/what-are-the-benefits-of-using-express-js-for-developing-enterprise-applications/>
- [41] R. C. Handayani, B. Purwandari, I. Solichah, and P. Prima, “The impact of instagram “call-to-action” buttons on customers’ impulse buying,” in *2018 2ND INTERNATIONAL CONFERENCE ON BUSINESS AND INFORMATION MANAGEMENT (ICBIM 2018)*, 2018, pp. 50–56, accessed 2023-04-25.
- [42] W. Kluwer, “Svensk författningssamling,” p. 5, 2016, accessed 2023-04-17. [Online]. Available: <http://rkrattsdb.gov.se/SFSdoc/16/160051.PDF>
- [43] S. Mithas, M. S. Krishnan, and C. Fornell, “Why do customer relationship management applications affect customer satisfaction?” *Journal of marketing*, vol. 69, no. 4, pp. 201–209, 2005, accessed 2023-04-24. [Online]. Available: <https://journals.sagepub.com/doi/pdf/10.1509/jmkg.2005.69.4.201>
- [44] G. offices of Sweden, “Swedish financial supervisory authority,” *Business Process Management Journal*, p. 1, 2015, accessed 2023-04-20. [Online]. Available: <https://www.government.se/government-agencies/swedish-financial-supervisory-authority-finansinspektionen/>
- [45] pensionsmyndigheten. (2022) Pensionsordlista och pensionsmyndighetens förkortningar. pensionsmyndigheten. Accessed 2023-04-12. [Online]. Available: <https://www.pensionsmyndigheten.se/forsta-din-pension/om-pensionssystemet/pensionsordlista>
- [46] M. Rouse. Node. Accessed 2023-04-25. [Online]. Available: <https://www.techopedia.com/definition/5307/node>
- [47] A. Singjai and U. Zdun, “Conformance assessment of architectural design decisions on api endpoint designs derived from domain models,” *Journal of Systems and Software*, vol. 193, p. 111433, 2022, accessed 2023-04-25. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0164121222001352>
- [48] D. O. Tuama. PHP vs JavaScript: When to use. Accessed 2022-10-17. [Online]. Available: <https://codeinstitute.net/se/blog/php-vs-javascript/>

- [49] The v8 JavaScript engine. Accessed 2023-04-17. [Online]. Available: <https://nodejs.dev/en/learn/the-v8-javascript-engine/>
- [50] Vaultes. Why penetration testing is important - vaultes enterprise solutions. [Online]. Available: <https://www.vaultes.com/why-penetration-testing-is-important/>