jamk

Web application upgrade to new modern technology

Case Tarmo volunteer enrolment service

Jouni Helenius

Master’s thesis
November 2022
Technology, communication and transport
Master's Degree Programme in Information Technology,
Full Stack Software Development

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Helenius Jouni

Web application upgrade to new modern technology

Jyväskylä: JAMK University of Applied Sciences, November 2022, 115 pages

Full Stack Software Development

Permission for web publication: Yes

Language of publication: English

Tiivistelmä

Juhannuskonferenssi on Suomen mittakaavassa kesän suurimpien tapahtumien joukossa, joka toteutetaan muutamien palkattujen henkilöiden ja muutaman tuhannen vapaaehtoisen yhteistyönä juhannuksena Keuruulla. Opinnäytetyössä vertaillaan suosituimpien JavaScript ohjelmistokehysten (framework) sopivuutta Isokirja-Opiston Tarmo talkoolaisjärjestelmän uudistukseen. Tarmo -järjestelmä on kehitetty vapaaehtoistyönä 2000-luvun alussa ja sitä on käytetty siitä lähtien Keuruulla pidettyjen Helluntaiseurakuntien Juhannus Konferenssin järjestelyissä. Yhtenä kulmakivenä juhannuskonferenssien järjestämisessä on ollut Tarmo -talkoolaisjärjestelmä, jonka avulla hallitaan talkoolaisjärjestelyjä.

Teknologiakatsauksessa keskitytään suosituimpiin JavaScript ohjelmistokehyksien Angular, React ja Vue historiaan, kehittymiseen, ominaisuuksiin ja tulevaisuuteen. Katsauksen tuloksien pohjalta valittiin Tarmo ilmoittautumisen uudistukseen React -ohjelmistokehys, sen suosion ja ominaisuuksien takia.

Konstruktiivisessa tutkimusosiossa toteutetaan React -ohjelmistokehyksellä Tarmo -järjestelmän ilmoittautumisosio Juhannuskonferenssia 2022 varten. Työssä tutustutaan vanhan ympäristön toiminnallisuuteen ja vaatimuksiin. Sen perusteella suunniteltiin selainpuolen arkkitehtuuri yhteistyössä palvelinpuolen osaajien kanssa. Tämän jälkeen suunniteltiin käyttöliittymä (UI) ja rakennettiin prototyyppi ilmoittautumisen käyttöliittymästä ja sen jälkeen toteutettiin käyttöliittymän ohjelmointi, testaus ja käyttöönotto kevään 2022 aikana.

Pohdintaosiossa käydään lävitse teknologia valintaa ja pohditaan erilaisia näkemyksiä niistä. Keskusteluosuudessa käydään lävitse projektin toteutumista, haasteita ja parannus-/ kehitysideoita.

jamk

Description

Abstract

The Midsummer Conference (Juhannuskonferenssi) is one of the biggest events of the summer at Finland. The Conference organization consist of a few paid people and over thousand volunteers at Keuruu. The thesis compares the suitability of the most popular front-end JavaScript frameworks for the reform of Isokirja's Tarmo volunteer management system. The Tarmo application was developed by a voluntary team in the early 2000s and it has been used since then in the volunteer arrangements of the Pentecostal Midsummer Conference at Keuruu. One of the cornerstones in the organization of midsummer conferences has been the Tarmo application, which is used to manage volunteer arrangements.

The technology review focuses on the history, development, features and future of the most popular JavaScript front-end frameworks Angular, React and Vue. Based on the results of the review, the front-end framework React was chosen to the renewal of enrolment, due to its popularity and features.

In the constructive research section, Tarmo enrolment application is implemented using the React framework. The work introduces the functionality and requirements of the old Tarmo environment. Based on that, the browser-side architecture was designed in cooperation with the architecture experts. After that, the user interface (UI) was designed and a prototype of it was built, and then programming, testing and implementation of the enrolment application was implemented during spring 2022.

In the Conclusion section concludes the choice of technology and consider different views on them. The discussion section analyses the implementation of the project, challenges, and improvement/development ideas.

Keywords/tags (subjects)

JavaScript, Framework, web application, front-end, react, angular, vue

Miscellaneous (Confidential information)

-

Contents

Figures

Tables

1 Preface (esipuhe)

I arrived in January 2020 to start Full Stack Software Development studies at JAMK. I knew something about programming terminology, but I had not done any real programming in the years. I have had study hard to learn programming and gain coding experience in studies. It was worth it.

The thesis has been a big leap for me as a software developer. The topic has been quite comprehensive because I wanted to learn as much as possible about everything. Hopefully, this thesis gives a kind of insight into the renewal of the web application to modern technologies and its challenges.

This season, the Tarmo system was voluntarily developed by Sampo Antila, Riku Kuusisto and myself. In the winter and spring, we held development “coding” camps for a few weeks at Hyvinkää. During them, things progressed well, and the work was engaging. Big thanks to Sampo for all the support and guidance. I have learned so much from you! Thanks also to Riku, the cooperation was smooth and helpful! Thanks for letting me be part of the team.

Big thanks to the supervisor of the thesis Jari Hautamäki, for his guidance and encouragement.

2 Introduction

Every year, the Pentecostal Midsummer Festival is organized in Isokirja college at Keuruu city. The number of visitors to the Midsummer Conference has been 20,000 - 30,000 / visitor per year, belonging to the big summer events in Finland. The volunteer organization has thousands of volunteer helpers.

The first conference was held in Tampere in 1945 and there were about 6,000 participants. The Pentecostal revival then founded the Bible College in Paimio in 1952 (Lundelin, n.d.), from there it moved 1959 to Katinala at Hattula. In early 80's, the Bible College moved five kilometers to better facilities in Lehijärvi, Hattula. In 1990, the Bible College moved to Keuruu, when the name was changed to Isokirja College. (‘Iso Kirja’, 2021; *Juhannuskonferenssi*, n.d.)

The management of a large volunteer organization became challenging with papers and took many resources handle invitations and check-in process. The Chancellery Committee began to become familiar with the challenges and consider what would be a solution for conference. A decision was made to begin developing computer software for conference needs in the early 2000s. Web application development started, and it was named “Tarmo”. Tarmo application is providing operations and practical arrangements of conference. Over the years, Tarmo has gained new features and has served well for almost 20 years.

A few years ago, the decision was made to upgrade Tarmo to modern technologies. The old web environment would require a lot of development work to keep it in trim, but architecturally time has passed it by. That is why it has been decided to reprogram the whole system from the scratch. The design and project management and programming work is done through DataCodex Oy as voluntary work. DataCodex receives a small compensation from IsoKirja for the costs of the cloud environment. The architecture and backend platform reform project has been started by volunteers. Currently there are three volunteers in the coding team, and the work is progressing slowly but surely. Tarmo consists of many components, and they will be renewed one at a time. It's hard to lock in a schedule when the creators are at their own jobs and use their limited free time for this project.

The thesis is divided into three entities, literature review (technology review), design and implementation. The scope of thesis is to find a suitable technology stack for the front-end platform implementation and to implement the first step of Tarmo project with this selected technology. The literature review goes through the most common JavaScript frameworks comparing them from different angles and trying to rank them.

3 Thesis research target and methods

The thesis research section approaches the challenge of upgrading old application to new technologies. Understanding subject is a major point. Research's technical depth is only one dimension of target. Technology research ascertain the result, what certify finest selection of new technologies. The user experience (UX) research collects feedback from users and operators. Based on technology – and user experience -research binds boundaries of the design.

3.1 Research objectives

The thesis research will focus discover solutions and technologies that can lead to optimal solutions and technologies. Most of users have experience of old Tarmo web application. For research project, this is valuable option to gain real user experience and feedback from real occasions.

An old Tarmo software was designed on the beginning of 2000s with up-to-date software development practices. It was not known in the design phase, what functionalities should be added in the future. Architecture has also changed radically in almost 20 years. In the past, server-based implementation was relied upon. In terms of security, the original implementation was challenging because each component must be secured with firewalls and security updates. Over the years, the server hardware has had to be replaced many times and operating system has evolved to newer versions. Third party applications and libraries has altered rapidly. Maintaining the application and environment has been laborious.

The back-end server architecture has been redesigned to support microservice architecture. All services are available via REST APIs. Burden of application's history induce stress to initiate application reformation from the beginning. The programming language will be switched to other and

development tools utilize common open-source components. The guideline of design new applications to Iso Kirja institute contains one main principle, services should run on the cloud environment. This simplifies maintenance processes and provide cost savings when there is no activity on

The thesis will answer the following research questions:

- What is suitable technology stack to produce web application?
- How to create professional code with tools to avoid issues?
- How to ensure good functionality also on mobile devices?

3.2 Research methodology

The thesis research methodology choice was solid. This chapter goes through the research methods and justifies the choice for this thesis.

The case study methodology is very common research methodology. It is very popular on the business and applies to cases like research company's product, service, operations, or process. Objective will be obtained profound cognitions of study case. Case study is applied to develop processes, products, operations, or process to preferable. This method is not suitable on this thesis as this more constructive thesis. (Ojasalo et al., n.d., Chapter 3.2 Tapaustutkimus)

Service Design methodology denotes service innovations, evolution, and design. Service experience design is user-driven and seeking user needs and provider's business goals. Service experience portions are service touchpoints, service moments and customer journey. This research contains small fraction from service Design methodology to constructive method. (Ojasalo et al., n.d., Chapter 3.5 Palvelumuotoilu)

Innovation methodologies discover ideas to new goods, products or services or improves existing goods, products, or services. Theoretical research science seeks novel findings and inventions, but not innovate them to goods, products, or services. This is difference between innovation and theoretical research. This innovation method is not suitable on this thesis as this more constructive thesis. (Ojasalo et al., n.d., Chapter 3.6 Innovaatioiden tuottaminen)

The foresight methodology attempts to obtain information to sight future trends. People are interested in knowing what the future tenders. Company or organization needs a plan how to reach future, it requires knowledge and a script. How to increase knowledge about the future? The future can be studied scientifically. Future study creates different models where different variables are taken into observation. This innovation method is not suitable on this thesis as this more constructive thesis. (Ojasalo et al., n.d., Chapter 3.7 Ennakointi)

Constructive research methodology is used when research is intended to create, for example, a new product, system, or plan. This is a right choice for this thesis and in the next section it will be covered in more detail. The research was enriched by creating three prototypes with different framework techniques in order to compare the practical experiences of all of them in this thesis. (Ojasalo et al., n.d., Chapter 3.4 Konstruktiivinen tutkimus)

3.2.1 Selected research method: Constructive method

The thesis research will focus discover solutions and technologies that can lead to optimal solutions and technologies. Constructive research methodology is used when research is intended to create, for example, a new product, system, or plan. The result of constructive research is a solution that allows for a new or improved solution. Research what acquire scientific theoretical knowledge can be execute by constructive research. Company or organisation obtains a neutral and theoretical solution based on knowledge from the research. (Ojasalo et al., n.d., pp. 65–71)

The topic of the thesis is *Web application upgrade to new modern technology* meets the features of constructive research. The renewal of Tarmo-system is entire. The back-end components will reproduce to support the microservice architecture. Back-end architecture is not in scope of thesis. Research focuses on the user interface (UI) and its implementation. The old software code is not used to create the user interface. The features of the user interface are listed and redesigned. The design takes into consideration as a whole user interface, to which new components can be added with ease in the future. The implementation of the user interface is divided into subprojects. Of the subprojects, only volunteer enrolment (Talkooilmoittauminen) is part of the programming work of the thesis.

The process of constructive method constitutes of six or seven phases depending on the literature. This study utilizes process model (figure 1) what is presented in book of Kehittämistyön menetelmät – Uudenlaista osaamista liiketoimintaan (Ojasalo et al., n.d., p. 67).

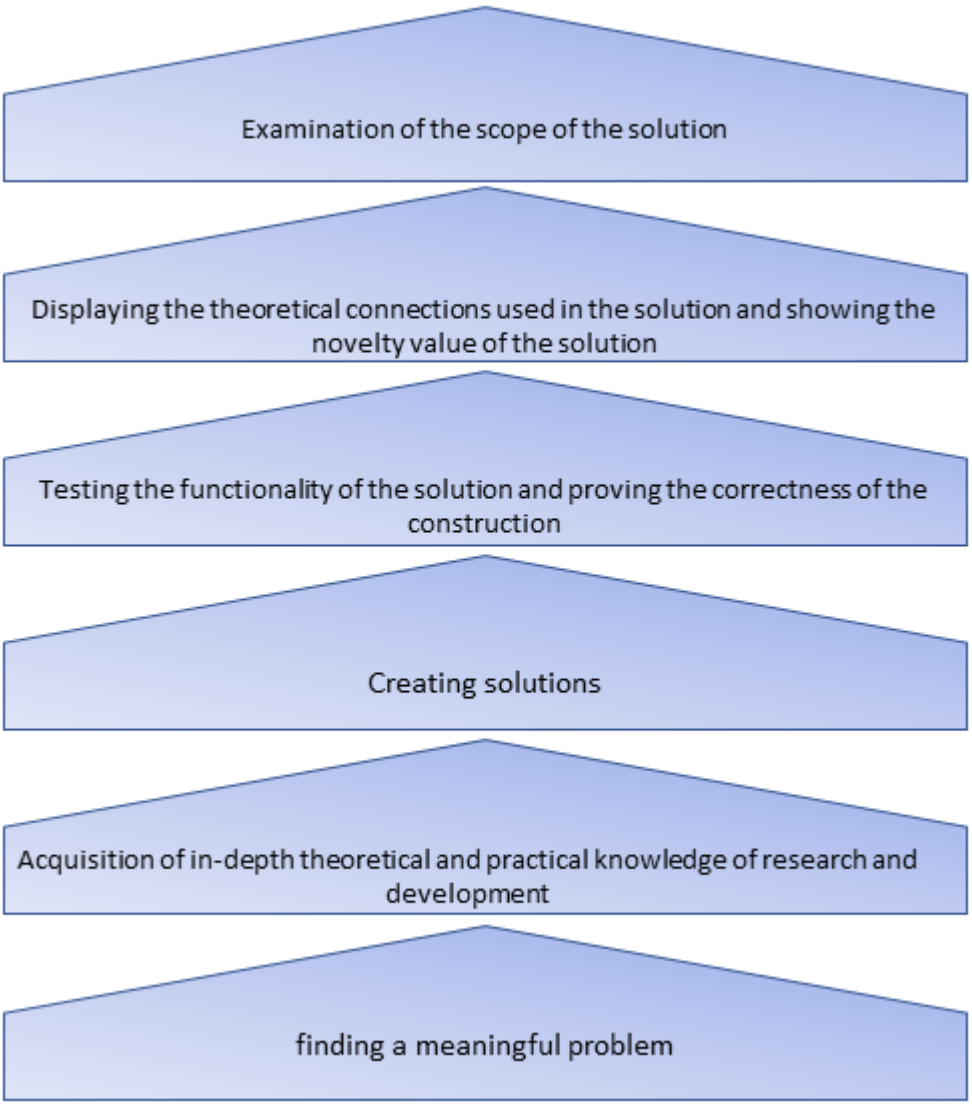


Figure 1: Constructive research process

Constructive research requires a **meaningful problem** to determine. Problem must be solvable and entire must not be too large. This study has an interesting topic which is limited to a suitable size. To support constructive research, usage data are clarified through a questionnaire. (Ojasalo et al., n.d., pp. 65–71)

Constructive research requires, acquisition of in-depth theoretical and practical knowledge of research and development. Understanding a problem or challenge requires skills to gather information from a versatile of sources. In many cases, this already requires familiarity with the topic or an in-depth acquaintance with the topic, software development attempts to take advantage of existing open-source applications. (Ojasalo et al., n.d., pp. 65–71)

Constructive research uses achieved knowledge to determine and build out a suitable solution. Solution must go through the requirements and functionalities. At this stage, a few different solutions may result. A few different prototypes are being developed to help you choose the right solution. (Ojasalo et al., n.d., pp. 65–71)

Constructive research uses testing the functionality of the solution and proving the correctness of the construction. Solutions are often complex and require a lot of testing. It would be desirable for automated test cases to be completed alongside the development. In this way, the basic functionality can be tested without manual steps. The end user tests, must be performed carefully and gather feedback to analyse sensation, experience and challenges of end user. In this way, the product can be improved. (Ojasalo et al., n.d., pp. 65–71)

Constructive research displays the theoretical connections used in the solution and showing the novelty value of the solution. Presentation and justification of the solution theory with evaluating and demonstrating the novelty value. (Ojasalo et al., n.d., pp. 65–71)

Constructive research examines of the scope of the solution. It is possible to expand the solution to different markets, if seeing the potential for expansion. Research like this requires special knowledge of different market areas and their requirements and special features. (Ojasalo et al., n.d., pp. 65–71)

3.3 Previous research

The purpose of this thesis is to provide new information about the choice of JavaScript Framework. Few studies have been done focus of on the choice of framework. Many international theses are behind a commercial license, and only studies with the keywords of JavaScript, Framework

of the years 2021-2022 from the Theseus system. Among these, those that comprise with comparation of JavaScript frameworks in a similar way. Several theses, framework was chosen by the subscriber of thesis and choice of framework was not justified in any way.

The study of Single-page application architecture (Karppanen, 2022) is one of these that deals with the same topic as this study, but with a different perspective. It covers JavaScript frameworks through the Single Page Application architecture and compares them with basic example programs.

The study of “Modernit verkkosovelluskehykset ja kirjastot” (Kemppainen, 2022) deals with almost the same subject matter as this thesis, it implemented an example program with the React and Angular frameworks. This research also looks into the labor market situation in Finland with regard to these frameworks.

JavaScript ja siitä syntyneet alustat (Miettinen, 2021)deal with the same topic, but from a completely different angle, i.e. through the development of JavaScript. This thesis dives deep into the world of JavaScript, and from there the JavaScript frameworks discussed in the study also pass through.

This study goes through JavaScript frameworks from history to the present day, giving an overall picture of the most common JavaScript frameworks and comparing them from different angles. Especially research dives deeply into the React framework. Information and views from many different sources have been collected in the material, and non-academic blog posts are also included.

4 Technology review

The technology review is literature review which focuses only on the front-end modern technologies. Front end refers to a user interface (UI) implemented with web technologies. Today, a large part of user interfaces is made using web technologies like HTML, CSS, and JavaScript. The advantages are their easy maintainability compared to native software. This technology review enters the options for modern technology to implement the front-end user interface. The review

clarifies the most popular JavaScript frameworks regarding their popularity and features. The literature review is based on literature, statistics of usage and feedback. The topic is comprehensive, and matters can be done in many ways to achieve the same result. This review introduces three currently most popular JavaScript libraries: Angular, React, and Vue.

4.1 JavaScript

JavaScript is a very common programming language. This is because it is used on the web browsers and nowadays also backend can be programmed with it. Originally JavaScript was an open, cross-platform object scripting programming language, when it was introduced by Netscape and Sun in 1995 (*Press Release´: NETSCAPE AND SUN ANNOUNCE JAVASCRIPT, THE OPEN, CROSS-PLATFORM OBJECT SCRIPTING LANGUAGE FOR ENTERPRISE NETWORKS AND THE INTERNET*, 1995). The standards of JavaScript have been evolved over the years. Initially, developments were more irregular and standard editions did not come often. 5th Edition was released December 2009 and 5.1 Edition was released June 2011. Four years later, 6th standard edition was published, and the naming was changed. It is called ES6/ECMAScript 2015. This standard version included many new features to JavaScript as Default Parameters, Template Literals, Multi-line Strings, Destructuring Assignment, Enhanced Object Literals, Arrow Functions, Promises, Classes, Modules and other improvements. Since then, it has become a new standard every year and has contained fewer changes. Currently, the latest adopted standard is ECMAScript 2021. (*JavaScript | MDN*, n.d.; Pollock, 2019; Talsania, 2018; T.J. Crowder, n.d.)

Document Object Model (DOM)

JavaScript enables front-end development from the browser side. It allows the use of APIs to manipulate HTML and XML files on the fly at the end of the browser. This API is called Document Object Model (DOM) and it defines how access to logical structure of documents in the web page. The standard of DOM is called Web APIs and specifications are in Mozilla Developer Network (MDN). The Browsers do not support all standards, and this poses challenges for developers. One recommended website is http://caniuse.com to check the operation of the property in browsers (*Web APIs | MDN*, n.d.).

WebAPIs (DOM) allow you to do things in the browser, but it's inconvenient to operate via APIs. Because of this, it has become different JavaScript libraries that provide an easier way to implement API calls. One of the best-known libraries is the JQuery API, the first version was released in 2006. It is small, fast, and feature-rich, allowing HTML manipulation, event handling, animation, and AJAX calls. This can do a lot, but modern JavaScript frameworks offer even more (*Web APIs | MDN*, n.d.).

The web development refers to the development of Internet sites on the Internet or intranet. This includes web design, web content editing, client-side / server-side programming and many other tasks. Programming or scripting enables the functionality of a web page. Often, JavaScript framework is used to make a web application. These have pre-built functionalities that make development easier and faster. These libraries utilizes on Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript (*What Is Web Development?*, n.d.).

(Chen, 2021)

4.1.1 Statically typed JavaScript implementations

JavaScript is very popular dynamically typed programming language, which means on runtime executes code assign dynamically values to variables and runtime does not check is value valid for assign. Programming with JavaScript is easy when you do not have to take care about on the types and contents of the variables. This can cause issues running JavaScript program if the data is not the right type, it is undefined, or error handling is completely missing or incomplete.

The programming languages have two type systems of static or dynamic. The static type system checks the types in the compile-time and dynamic distinguish the types in the run-time. Java and C# languages are static type languages. It was seen as important to upgrade JavaScript language to support static types. Three companies, Google, Microsoft and Meta saw possibility to invest static type systems for JavaScript. Google release Closure compiler which was meant be a tool making JavaScript download faster, allows more readable code with static types, more efficient JavaScript parse, analyze, removes dead code, and minifies code. It never gained popularity and is now used by a small margin. Google also launched open-source Dart language in 2011. Dart is class-based,

object-oriented programming language and supporting JavaScript compilation with style of C-language. Dart is meant to use web application and mobile application programming. It is quite new programming language and Google uses it in their applications like Gmail. The running code is precompiled into JavaScript, and all is compatible with all major browsers. Microsoft released TypeScript on 1st October 2012, which gained great popularity among developers. (Mikkonen, n.d., p. 48; 'TypeScript vs Dart | Top 7 Most Useful Differences You Need to Know', 2018)

TypeScript

Vanilla JavaScript is high-level programming language supporting ECMAScript standards, but it does not include types, interfaces, generics, and decorators. Microsoft has developed a TypeScript language which is strongly typed and make builds by default using ECMAScript 2015 features, but support also older ES versions. Typescript offers all JavaScript features and deliver additional layer of TypeScript type system to top of JavaScript. TypeScript focuses on producing safe and predictable code what can executed by any JavaScript engine. Static typing is main feature of TypeScript like other languages such as C# and Java and it makes JavaScript more familiar of them. (Nasserzadeh, 2020; *Why You Should Use TypeScript*, n.d.)

Benefits of TypeScript

The study "To Type or Not to Type: Quantifying Detectable Bugs in JavaScript" has examined the benefits of switching to static type language. The conclusion of study was, *"In this paper, we evaluated the code quality benefits that static type systems provide to JavaScript codebases. The results are encouraging; we found that using Flow or TypeScript could have prevented 15% of the public bugs for public projects on GitHub"* (Gao et al., 2017).

Boris Cherny, author of Programming TypeScript (O'Reilly) is a TypeScript expert and interviewed by Nate Black on Episode 384 of "Software Engineering Radio". He justifies the benefits of TypeScript, TypeScript enables larger teams JavaScript projects, wider code bases, and across different terminal devices. Large technology companies have millions of lines of code in their projects, with

numerous engineers implementing new features in which case TypeScript could potentially prevent code breakdown. TypeScript improves code quality because annotations are at the same time code documentation. (Black, 2020)

Betterprogramming.pub article: 10 Reasons to Use TypeScript Over Vanilla by Ali Nasserzadeh (Nasserzadeh, 2020). He is a frontend engineer at Google and in this article, he explains 10 reasons why you should use TypeScript. Below is a summary of the main points from the article.

The risk of bugs in code will be lower because TypeScript won't compile unworking code. Coder can make mistakes with typo in the variables or type error of variable. JavaScript doesn't recognize these errors and compile code and code has a bug.

Fail-Fast Principle means that there are often static type failures or other errors in the new code base and the compiler immediately reports them. This way, types are included in the code from the start of code.

More Information / Documentation in Codebase, the code is easier to read and understand, because the variables are typed according to the content, and that makes code more readable for other coders.

TypeScript Has More Features Than JavaScript, In the beginning, TypeScript offered features that were not available in older JavaScript versions, but now they have been included also in JavaScript. These features are interfaces, namespaces, generics, abstract classes, data modifiers, optionals, function overloading, decorators, type utils, and the "readonly" keyword.

Better Refactoring and Tools, Refactor's TypeScript code is much easier than non-typed JavaScript. Because of this, the developer can use the IDE to refactor variable names and change interface, class and enum on the fly.

It's Used by Many Big Companies, Has a Big Community, and Is the Primary Language of Angular, Microsoft has developed open-source TypeScript, which is widely used by developers and has

been developed a lot over the years. Angular 2 was a big step because it was the first framework to use TypeScript by default.

Conclusion, JavaScript allows the programmer to write bad code, but TypeScript reduces possible bugs in the code and makes the code easier to read. It's worth using TypeScript already, and probably in the future JavaScript will have the same features.

4.2 Browsers

The modern web technologies require browsers that support JavaScript functionalities. JavaScript allows you to dynamically manage via DOM the properties of element or to create or delete elements to execute modern web pages to a browser window. Browsers do not support all DOM features at the same time, so developers need to test the functionality for all main browsers.

Most browsers are based on Google's open-source Chromium web browser, which are Chrome, Microsoft Edge, Opera, Brave, Samsung Internet and many others, in them, the properties are quite close to each other. Market share of Chromium based browsers is approaching 80%. (*The Chromium Projects*, 2021).

Table 1: Global market share of the desktop browsers November 2021

Chrome	Safari	Edge	Firefox	Opera	IE
66.35 %	9.82 %	9.53 %	8.34 %	2.84 %	1.13 %

The worldwide market share of desktop web browsers (Table 1) shows that Google's Chrome browser has become the dominant browser with a market share of over 66 percent. This has certainly led to Google's commitment to the development of the web standards, their visibility in their own search engine and also the weak features of competing browsers at the time. (*Browser Market Share Worldwide*, 2021):

Table 2: Finland's market share of the desktop browsers November 2021

Chrome	Safari	Edge	Firefox	Opera	IE
69.61 %	7.96 %	7.18 %	10.85 %	2.47 %	1.12 %

In Finland, Google Chrome's market share (Table:2) is even bigger than its international market share. Apple's Safari has a smaller market share in Finland. Firefox is a little more popular than the average in the world. (*Desktop Browser Market Share Finland*, 2021).

Google Chrome is the clear market leader in the world, but the next three browsers are in the popularity at almost the same percentages and rankings may vary from month to month. The trend is that the popularity of the Microsoft Edge browser has risen due of popularity of Windows 11 and Microsoft's pressure to customers to use the Edge browser instead of Google Chrome (Brandon Hill published, 2021; Parsons, 2021).

4.3 JavaScript frameworks

JavaScript have many useful libraries available, and these are popular with developers. It is easier to use a pre-developed and tested library than to do it yourself and maintain it. The development work also will speed up when developer does not have to do everything itself. Difference of the library and framework are in the scope of the software. The library is smaller containing specific functionalities, the framework is like an umbrella for range of functionalities. One of the most popular the open-source library is JQuery, published in 2006 and it is used for DOM manipulation etc... JQuery is a library. This technology review does not cover JQuery library but concentrates into modern JavaScript frameworks.

Single Page Application (SPA)

Web developers have had a dream for years to make web applications that behave like a native application. Various actors have tried to implement this in many ways: IFrames, Java applets, Adobe Flash and Microsoft Silverlight. Each of "standards" has tried to make the environment to reach the desktop application experience in a web browser. None of these succeeded in that perfectly and have almost disappeared from use. A Single Page Application (SPA) enables this without

any additional browser plugin or learning a new programming language, using only browser JavaScript, using HTML and Cascading Style Sheets to implement the web experience. (*Chapter 1. What Is a Single-Page Application?*, n.d., Chapter 1 What is a single-page application?; *Masterworkshop SPA Design and Architecture: Understanding Single-Page Web Applications*, n.d.; Flanagan, 2006, p. 497)

The idea of a SPA-type architecture came about in the early 2000s, when the XMLHttpRequest (XHR) API implemented in major browsers was obtained through a new kind of thought model. Developers started to combine JavaScript, HTML and CSS code in a new way and that way started to be called Ajax, i.e. Asynchronous JavaScript. These became the cornerstones of so-called modern web development; the Document Object Model (DOM) was dynamically manipulated with JavaScript and the web page in browser was modified on the fly with CSS. SPA idea is that the JavaScript code is loaded into the browser and browser executes the JavaScript code inside and retrieves data from e.g., micro services API interfaces and browser render view of component to DOM. Presentation Layer logic is moved from server side to browser side. In SPA architecture views are not complete HTML page. Page view dynamic components are created with JavaScript and displayed via DOM without browser refreshes. The JavaScript frameworks included in this technology review are implemented with SPA architecture. (*Chapter 1. What Is a Single-Page Application?*, n.d.; *Masterworkshop SPA Design and Architecture: Understanding Single-Page Web Applications*, n.d., Chapter 1.1 SPA in a Nutshell; Flanagan, 2006, p. 497)

The most popular JavaScript frameworks

The popularity of JavaScript frameworks can be measured by various metrics. These are open-source products, so their popularity can be measured through NPM downloads, GitHub stars and the Stack Overflow Trending service.

Tanguy Krotoff (Tkrotoff) has GitHub project called FrontendFrameworksPopularity.md and contains links to various results in popularity. The site has collected survey and trend results from various sources such as Stack Overflow, State of JS, JetBrains, NPM downloads, NPM packages dependencies, GitHub repositories dependencies, Reddit metrics, Twitter metrics, Hacker News Hiring Trends, Google trends, YouTube trends, Similar Tech market share & web usage statistics,

BuiltWith Technology Lookup and GitHub stars. It gives a good overview of the popularity of the frameworks. In the next paragraphs, get to know some piece of these results in more depth. (Tanguy Krotoff, 2021)

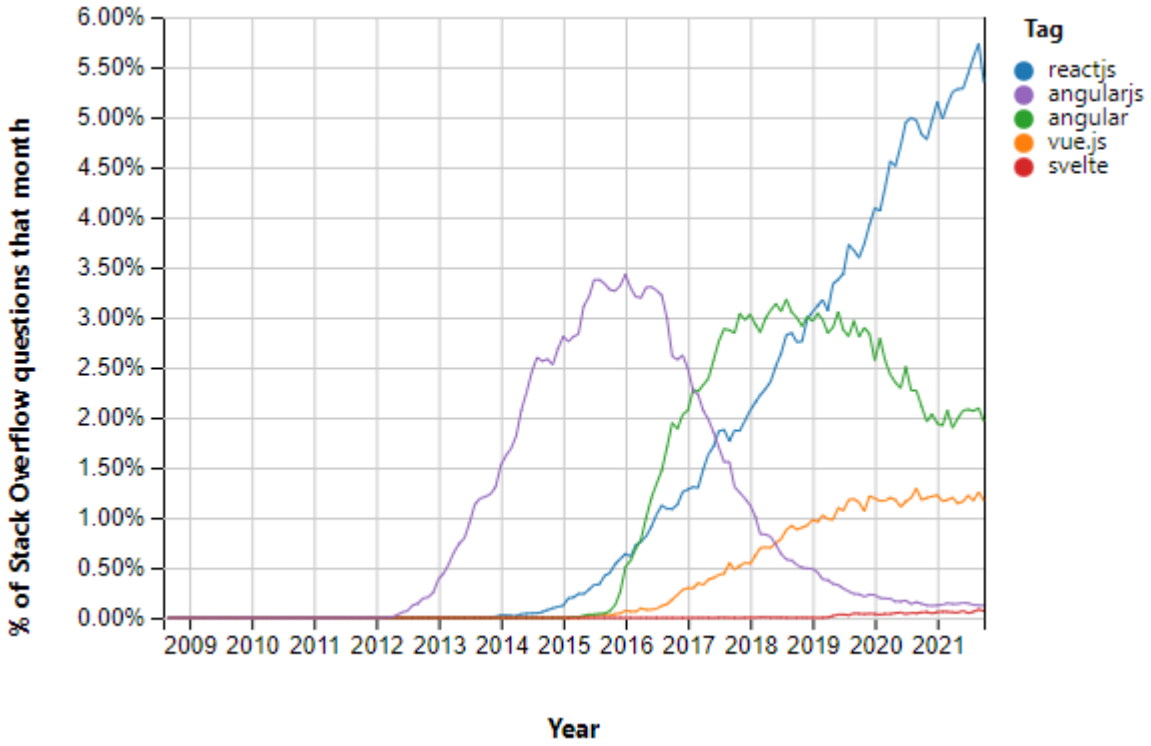


Figure 2 : Trending JavaScript Frameworks: Stack Overflow questions that month (1.12.2021) (*Stack Overflow Trends*, n.d.)

Stack Overflow Trending JavaScript Frameworks chart (Figure 2) gives an indication of how popular the frameworks are and how many questions the developers have had of them. This chart clearly shows, for example, the popularity of the Angular framework in the early 2010s, when AngularJS was released and was the most popular framework at that time. Architecturally, Angular had structural limitations and the redesigned Angular 2 framework was released in 2014, but the number of questions started to increase at the end of 2015. The statistic also clearly shows the increase in popularity of the React framework and its rise to the most popular software framework. The number of questions also shows that the popularity of the Angular software framework has waned over the years. The questions do not necessarily give a true trend of the framework's popularity, many questions can refer about its problems and questions caused by poor documentation.

It is also important for the developer that answers about the software framework can be found on Stack Overflow. (*Stack Overflow Trends*, n.d.)

Sacha Greif, a designer, developer, and entrepreneur created State Of JavaScript site. He has told *"I got tired of all this talk of "JavaScript fatigue", so I created a survey to get more data about recent trends in the JavaScript community"* (*Sacha Greif*, n.d.) . This site collects JavaScript related usage statistics. Following trend figures of JavaScript Front-end Frameworks are collected from rankings of front-end frameworks(*The State of JS 2021*, n.d.). (*Sacha Greif*, n.d.)

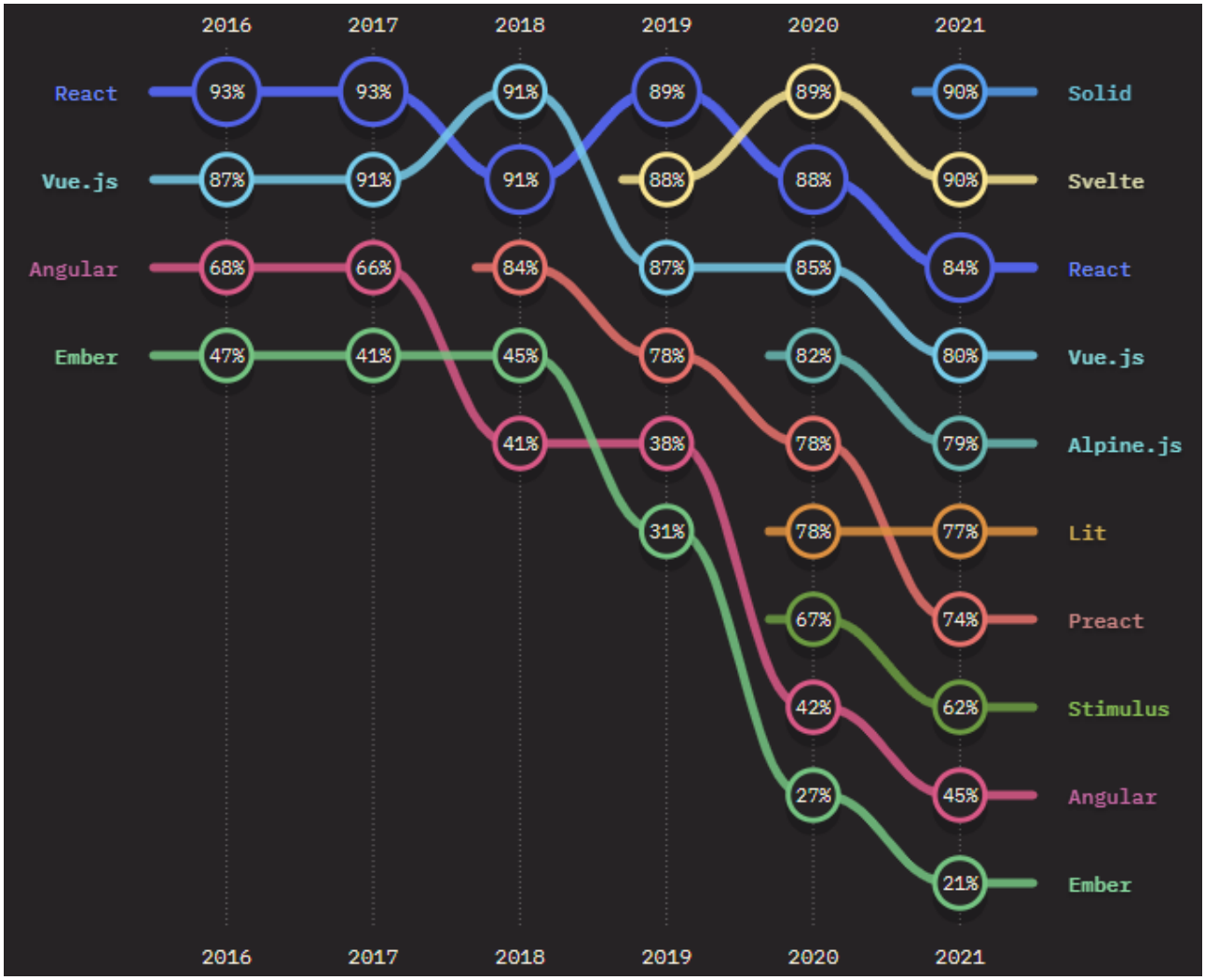


Figure 3: Stateofjs.com Satisfaction of Front-end frameworks

Satisfaction of the front-end frameworks chart (Figure 3) shows the developer satisfaction framework. It is calculated as follows: would use again / (would use again + would not use again). For

the first time this year, a Solid framework is included, which is partly React-like in nature. The biggest difference is that it does not have a Virtual DOM like React has and it's also light-weight. The biggest surprise here is that how Angular has dropped the list surprisingly down. Angular, it can be stated that the statistics show the challenges arising from the development of Angular and the problems of the development community. This and the fact that development with Angular has a long learning curve are probably the main reasons for the decrease in trust. (*The State of JS 2021*, n.d.)

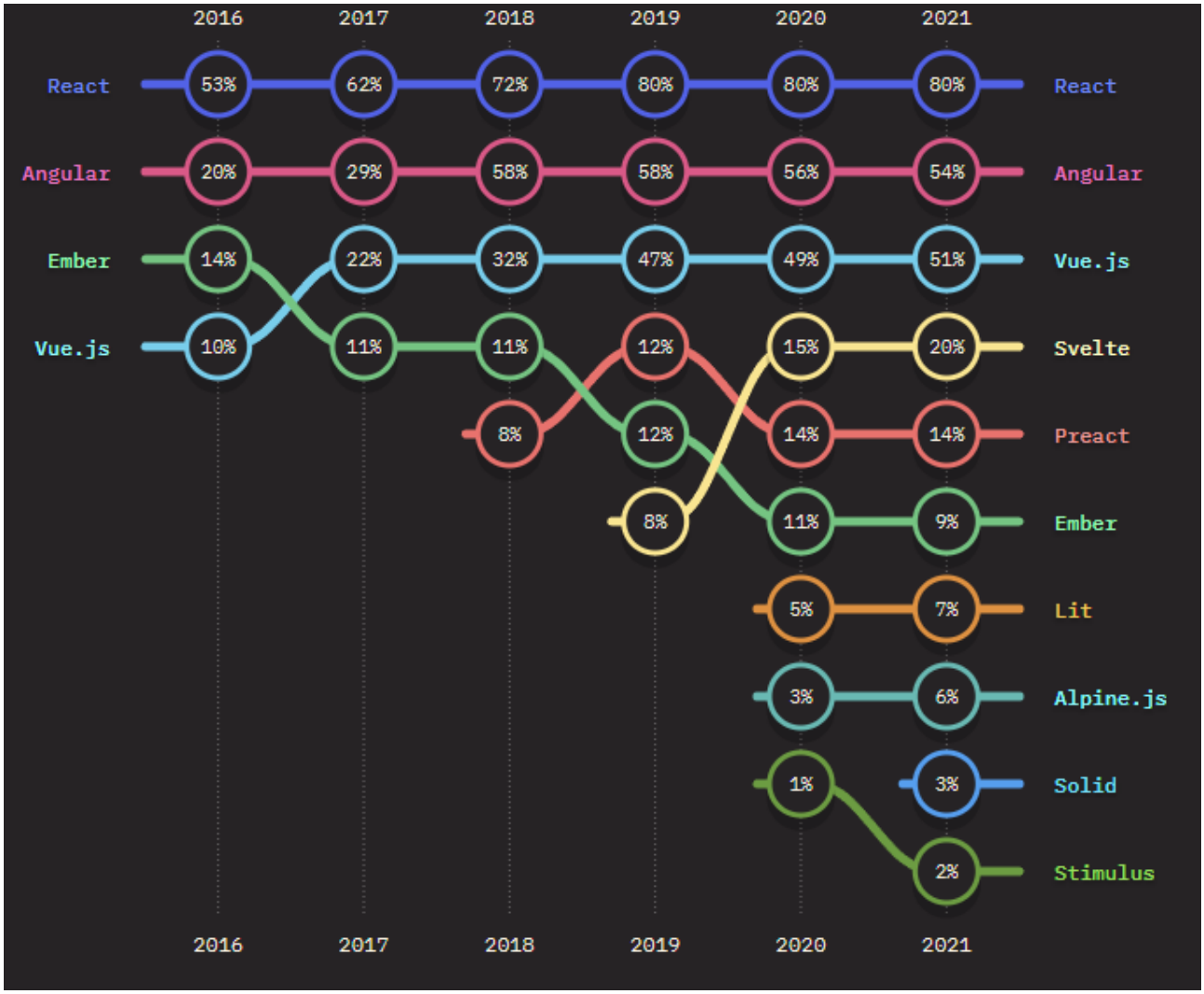


Figure 4: Stateofjs.com Usage of front-end frameworks (*The State of JS 2021*, n.d.)

Usage of the front-end frameworks chart (Figure 4) shows the use of software frameworks in projects. This statistic does not include the original AngularJS framework, because new features are no longer being developed for it. According to this statistic, React is the most popular platform to implement applications and Angular is the second most popular, although it has lost a little popularity. Vue.js has almost caught up with Angular. Among the frameworks outside of this study, the Svelte framework is on the rise, but still far from the three most popular frameworks. (*The State of JS 2021*, n.d.)

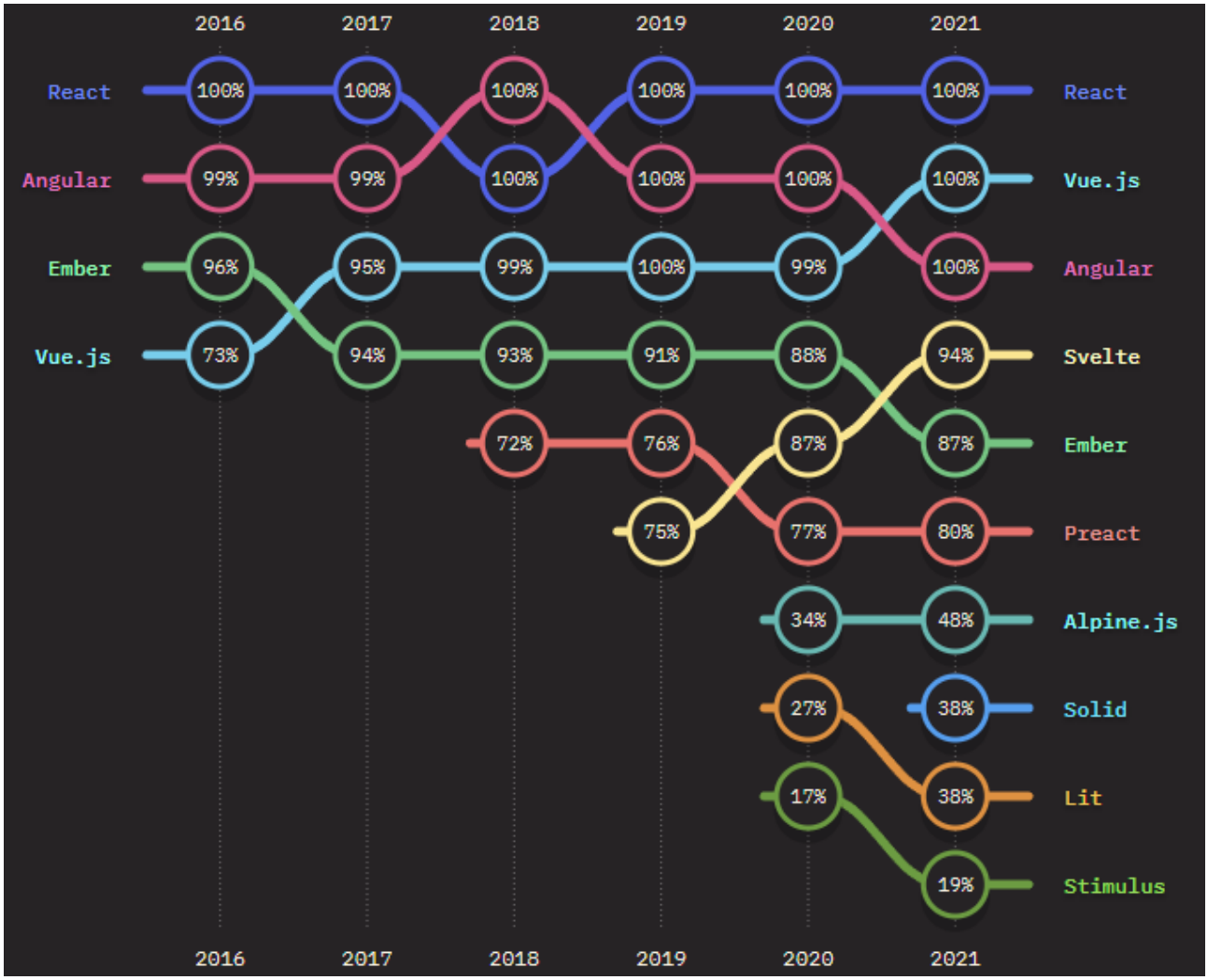


Figure 5: Stateofjs.com awarness of Front-end frameworks

Awareness of the front-end frameworks chart (Figure 6) shows how commonly frameworks are known in the developer's world. In practice, the three biggest ones are known among all developers. Svelte is also known too almost everyone. (*The State of JS 2021*, n.d.)

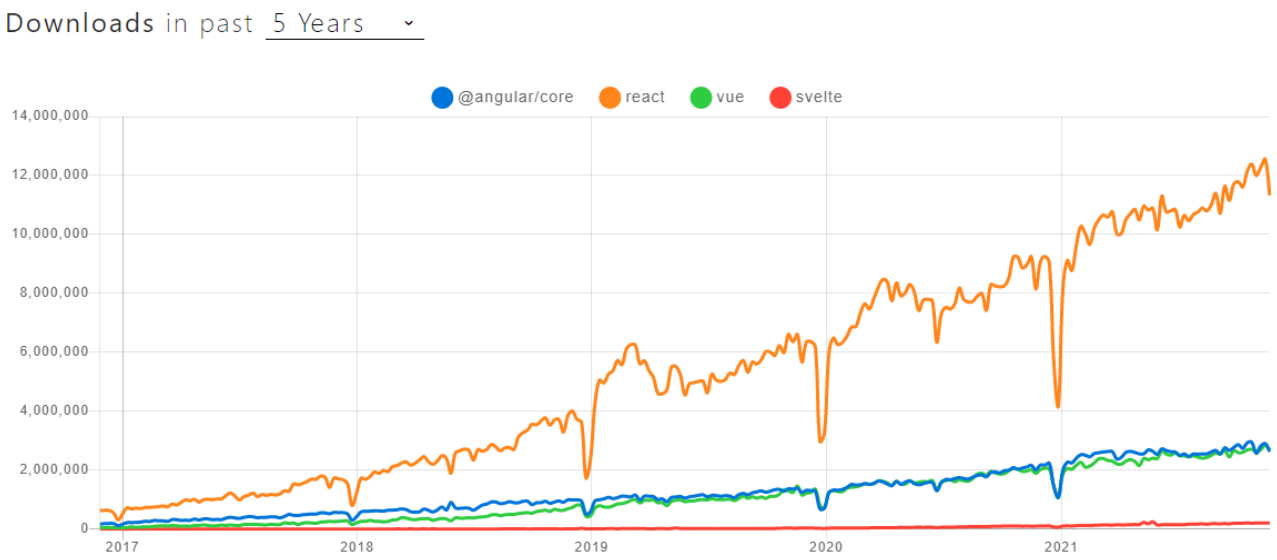


Figure 6: NpmTrends downloads of frameworks in past five years (*@angular/Core vs React vs Svelte vs Vue | Npm Trends*, 2021)

Npmtrends.com web site provides trends of usage of NPM packages chart (Figure 8). The Default Package Manager for JavaScript is Node Package Manager (NPM) and it is bundled into the JavaScript runtime environment of Node.js. It can be used to fetch JavaScript libraries into software projects, and it is very popular among developers. Because of this, there are good statistics on number of downloads and you can see the popularity of frameworks from them. It can be clearly seen from this statistic that the number of downloads of the React framework is multiple times compared to other frameworks. Angular and Vue are in the same number of downloads. (*@angular/Core vs React vs Svelte vs Vue | Npm Trends*, 2021)

GitHub is an internet hosting service for software development with Git version control platform. Github is acquired by Microsoft for US$7.5 billion in 2018 (Sawers, 2018). It has become one of the world's most popular version control platforms and offers services implemented to GitHub accounts for commercial use. GitHub is also the place for most of open-source project's source

codes. Because of this, it is possible to get good measurable data from GitHub like popularity based on the stars, forks, watchers etc. of the projects.

Table 3: Github activities of frameworks

Framework	Forks	Watchers	Open issues	Stars
React	36200	6700	670	178000
Vue	30800	6200	322	191000
Angular	20500	3100	1700	78000

The table (3) indicates activities of frameworks. Logged users use starring to find a repository or topic again later. Vue leads in stars but React follows about 10,000 stars behind and Angular is clearly behind of both. The situation can be explained by the fact that Vue users are more active and are often used in smaller projects due to its easy adoption. Angular's situation may be explained by the fact that it is used in larger, more complex projects and is more challenging to adopt. Number of forks indicates how many copies has been initiated from the original repository (*About Forks*, 2021). Amount of watchers indicates GitHub users who are following activity in a repository awaiting notifications of changes, but they are not collaborators of project (*Activity*, 2021). React leads in forks and watchers, Vue comes in second and Angular is clearly behind of these frameworks. This result combines with the popularity of the frameworks. Amount of open issues indicates bugs, work and ideas are not handled yet, this can signal that framework has bugs and project ability to fix problems in the reasonable time (*About Issues*, 2021). (*GitHub - Angular/Angular: The Modern Web Developer's Platform*, n.d.; *GitHub - Facebook/React: A Declarative, Efficient, and Flexible JavaScript Library for Building User Interfaces.*, n.d.; *GitHub - Vuejs/Core: 🖖 Vue.Js Is a Progressive, Incrementally-Adoptable JavaScript Framework for Building UI on the Web.*, n.d.)

4.4 React

React (also known as React.js or ReactJS) is a free and an open-source client-side component based JavaScript library for building user interfaces where certain features are added via community packages. React is called as a library and not a framework because it only contains the view layer from the MVC (Model,View and Controller) architecture. This is not a problem because React is very flexible and numerous open-source components have been developed for it to implement functions missing from the React library. This also means that the so-called React library alone cannot be used to implement a complete application. With React library and additional components has capabilities to build modern reactive user interfaces for the web.

Software Engineer Jordan Walke from Facebook created React, he launched early prototype called FaxJS in 2011 and React was launched to open-source by Facebook at JSConf US in May 2013 (*The History of React.Js on a Timeline*, 2018). First released open-source version was 0.3.0 and version numbering changed after 0.14.1 (29 October 2015) to 15.0.0 (7 April 2016). The current release is now 18.2.0 and in the open source era, 99 versions of React have been released. React version 18 has been released in spring 2022 and version 17.0.2 has been used in the programming part of this thesis. Meta (formerly Facebook) is maintaining open-source React. (*React Vs. Angular Vs. Vue*, 2019; Vyas, 2022a)

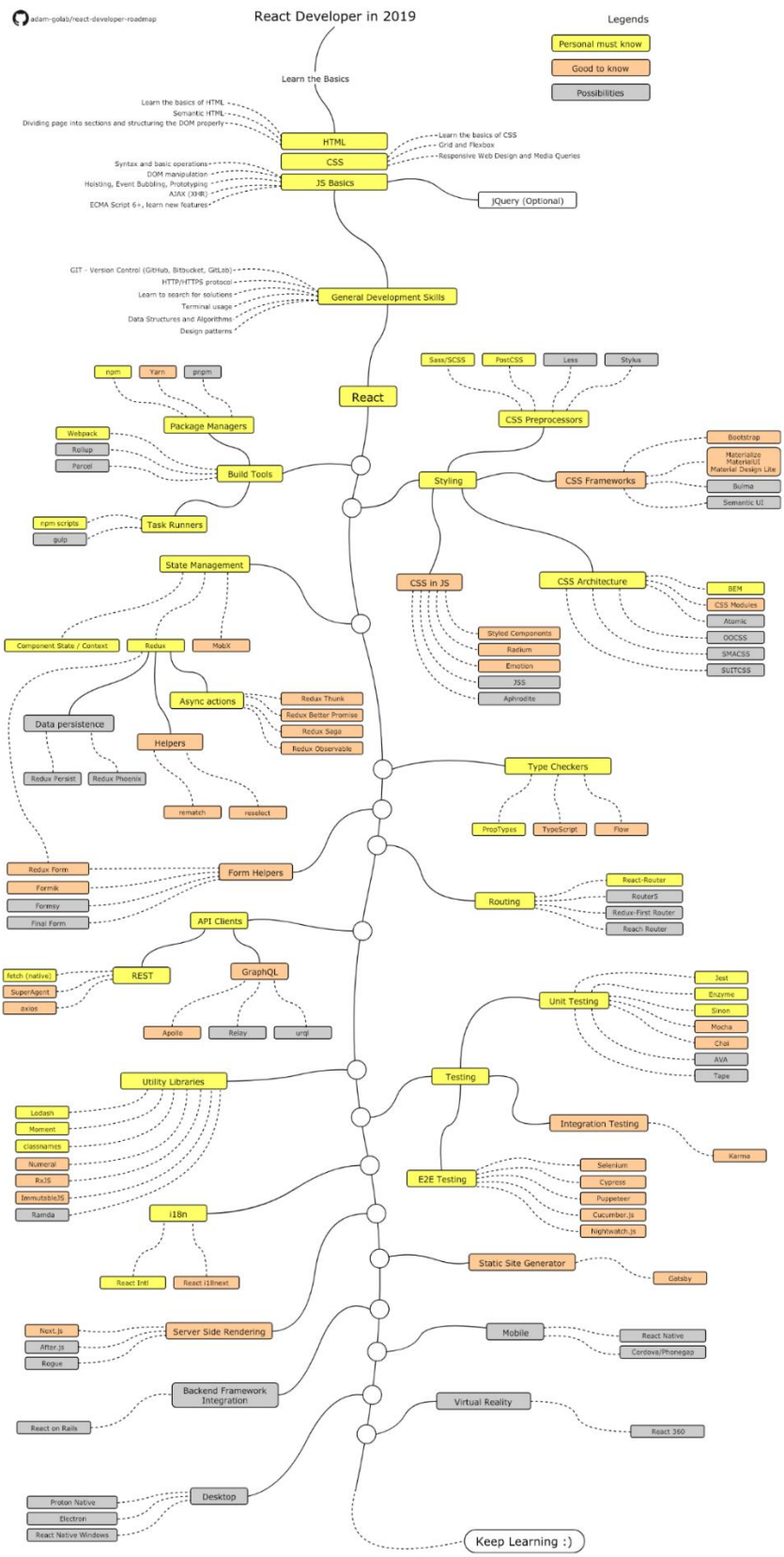


Figure 7:React Developer Roadmap by Adam Golab(Gołąb, 2018/2022)

A React developer must know the basics of information technology, communication, web development and at least master JavaScript, but it would be better to develop in TypeScript while avoiding traditional JavaScript bugs. Adam Golab has GitHub project called react-developer-roadmap (Figure 10) it gives brief information what React developer should know. React software developer needs to understand the basics of web development, which includes the basics of HTML, CSS and JavaScript. It is good for a developer to know how to use GIT version control with basic commands to be able to work in a software team. Understanding of network traffic and the http -/https protocol is nowadays a basic requirement of developer ability to use package managers is an important skill, styling of UI, developer need to understand and handle basics of the CSS architecture also it would be good to master a CSS framework such as MUI or Bootstrap. React ecosystem; it is good to master React-Router, from the State Management side, it is good to master React's Context or Redux, for Form management, it is good to master Formik, for handling Http/https connections, it is good to master JavaScript's built-in fetch or the external Axios library. (Gołąb, 2018/2021)

Use cases

React is most popular JavaScript framework what enables Single Page Application (SPA) functionalities on web application. Meta uses it in its own services such as Facebook, Instagram, and WhatsApp in its products. Other big web sites using React include, for example, Airbnb, Atlassian, Cloudflare, Dropbox, BBC, Imgur, Netflix, Paypal, Periscope, Reddit, Salesforce, Tesla and so on. (*10 Famous React Websites*, n.d.; Academy, 2016)

JavaScript and TypeScript in React

React uses "vanilla" JavaScript by default, but it is also option to use TypeScript in the project. Create React App installation application supports both JavaScript and TypeScript to configure project automatically with correct dependencies according to the project settings given by the coder. This is where React differs from other frameworks, as they are implemented directly with TypeScript. In the TypeScript project, the files are named so that those containing JSX syntax file extension is .tsx and files containing pure TypeScript file extension is .ts. (*Create React App*, n.d.)

Scripting language and rendering elements

React has its own way of combining UI templates and JavaScript logic. It is called JavaScript Syntax Extension. JSX uses attributes in HTML elements in the same way as regular HTML. JSX uses the camelCase naming convention, where words are separated by an uppercase letter and the first word starts with a lowercase letter. In attributes, for example, HTML attribute called class changes to className in JSX, because class is a reserved word in JavaScript. (*Handling Events – React*, n.d.)

```
const name = Niilo Nokkela;
const element = <h1>Hello, {name}</h1>;
```

With it, UI elements can be used in variables and use dynamic content inside variable from another sources like variable.

```
function getGreeting(user) {
    if (user) {
      return <h1>Hello, {formatName(user)}!</h1>;
    }
    return <h1>Hello, Stranger.</h1>;
  }
```

JavaScript comparison logic can be embedded into the HTML code in JSX syntax. (*Introducing JSX – React*, n.d.)

Components and Props

Components are conceptually like JavaScript functions. They receive inputs (called props) and return a React element that is rendered on the screen. You can use a function or an ES6 class in the component definition. The Function and Class Components method can be used in React development. According to good React programming, we try to make as many reusable components as possible. In this case, not so much code is generated and troubleshooting is easier. (*Components and Props – React*, n.d.)

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
  }
```

This function is a valid React component accepting props and returning a React element.

```
class Welcome extends React.Component {
    render() {
      return <h1>Hello, {this.props.name}</h1>;
    }
  }
```

This class is equivalent to the previous function and works similarly in terms of React. Props locates in the class level and are accessible from this word.

User-defined components can also represent in JSX syntax.

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
  }

  const root = ReactDOM.createRoot(document.getElementById('root'));
  const element = <Welcome name="Sara" />;
  root.render(element);
```

Handling events

It has been mentioned above that HTML attributes conform to camelCase naming convention.

```
<button onclick="activateLasers()">
    Activate Lasers
</button>
```

This example is pure HTML button with onclick event.

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

This example is a React JSX button with onClick event.

Virtual DOM

React has a concept called the virtual DOM (VDOM) which works by keeping the lightweight representation of the user interface in memory and updates it to the real DOM using the ReactDOM library. React provides a declarative API with algorithmics how and when render updates only changed DOM elements and this process is called reconciliation. (*React Vs. Angular Vs. Vue*, 2019; *Reconciliation – React*, n.d.; *Virtual DOM and Internals – React*, n.d.)

Hooks

Hooks are first introduced in version 16.8. They are used for state and components Lifecycle properties without writing a class. The component hierarchy does not need to be changed again when reusing Hook's logic. (*Introducing Hooks – React*, n.d.)

Testing

Create-react-app setups react project with React testing library and Jest as test runner. React is recommending use Jest JavaScript test runner that allows the access DOM via jsdom and React Testing Library is set of helpers for testing React components. React component testing is similar than other JavaScript code. React components can be tested in a few different ways, by rendering the entire component in a tree simplified test environment and analysing the results, or by running a complete application in a realistic browser environment, also what is called end-to-test tests. (*Testing Overview – React*, n.d.)

Future

React release 18 has been announced in the spring of 2022 and there have been fixes and improvements in a couple of newer versions. React is a safe choice for development work because Meta uses it for its own products, so the development and maintenance work continues there.

Additional useful third-party libraries

React is just a UI library, so it lacks the features of a complete framework that, for example, Angular offers a wide set of features. For this, third-party libraries must be used to implement these functions such as router, http-fetch, forms, state management and so on.

4.4.1 UI CSS frameworks for React use

Different libraries are available for React to create the UI. This section goes through a variety of alternative libraries. React is commonly used around the world that there is room for many different approaches of UI libraries. The most popular are Material Design based MUI, Ant Design and Bootstrap based react-bootstrap and reactstrap libraries.

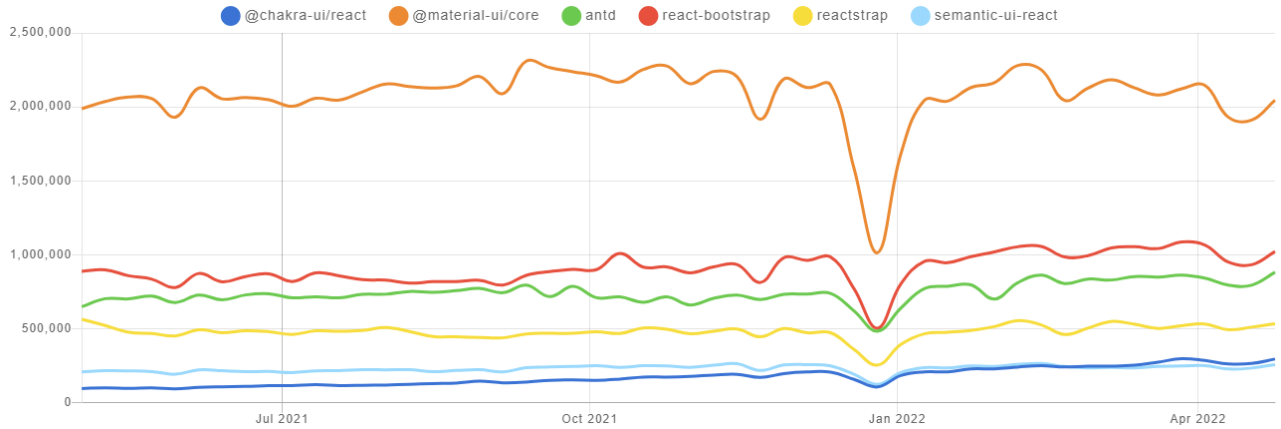


Figure 8: NPM trends of React UI libraries (*@chakra-Ui/React vs @material-Ui/Core vs Antd vs Evergreen-Ui vs Grommet vs Onsenui vs React-Bootstrap vs React-Suite vs Reactstrap vs Semantic-Ui | Npm Trends*, n.d.)

Downloads of React UI libraries chart (Figure 8) shows the popularity of UI libraries. MUI (old Material-UI) is clearly the most popular UI library. Bootstrap-based UI libraries react-bootstrap and reactstrap are in the top four. The third is Antd UI components, which are completely TypeScript based.

Table 4: React UI libraries statistics

Framework	Unpacked size	Forks	Watch-ers	Open is-	Stars	Weekly down-
Ant Design	48.2MB	33.7k	167	769	79.8k	877 172
Bulma	1.32MB	3.9k	623	141	45.5k	207 808
Chakra UI	66.5kB	2.2k	191	84	25.5	297,647
Evergreen-UI	7.33MB	775	120	26	11.6k	12 708
Fluent UI React	58.2kB	2.2k	280	693	13.2k	108 591
Grommet	6.88MB	955	130	150	7.9k	24 967
Material UI v5	9.18MB	26.9k	1.4k	923	77.8k	1 185 809 v5
Onsen	10.3MB	1k	319	145	8.6k	38 350

React-Bootstrap	1.37MB	3.3k	436	134	20.7k	1 026 586
React Bulma Components	959KB	126	24	10	1.1k	5 075
react-onsenui	773kB	*	*	*	*	1 722
Reactstrap	3.35MB	1.3k	173	251	10.3k	535 396
React Suite	13 MB			116		30 782
Semantic UI React	3.02MB	5.1k	1.3k	940	50k	259 558
Tailwind CSS	4.2MB	2.8k	567	17	56.2k	2 842 083

* Same GitHub project than OnsenUI

(*Ant Design*, 2015/2022; *Antd*, n.d.; *Bulma*, n.d.-a; *Chakra-Ui/Chakra-Ui: ⚡ Simple, Modular & Accessible UI Components for Your React Applications*, n.d.; *@chakra-Ui/React*, n.d.; *Evergreen-Ui*, n.d.; *Fluent UI Web*, 2016/2022; *@fluentui/React-Hooks*, n.d.; *Grommet*, n.d.; *Grommet*, 2015/2022; *Issues · Semantic-Org/Semantic-UI-React*, n.d.; John, 2017/2022; *MUI Core*, 2014/2022; *@mui/Material*, n.d.; *Onsen UI - Cross-Platform Hybrid App and PWA Framework*, 2013/2022a; *Onsen UI - Cross-Platform Hybrid App and PWA Framework*, 2013/2022b; *Onsenui*, n.d.; *React-Bootstrap*, n.d.; *React-Bootstrap/React-Bootstrap: Bootstrap Components Built with React*, n.d.; *React-Bulma-Components*, n.d.; *Reactstrap*, n.d.; *Reactstrap*, 2016/2022; *Rsuite*, n.d.; *Rsuite/Rsuite*, 2016/2022; *Segmentio/Evergreen*, 2017/2022; *Semantic-Ui-React*, n.d.; *Tailwindcss*, n.d.; *Tailwindlabs/Tailwindcss*, 2017/2022; Thomas, 2016/2022)

The following paragraphs contain summaries of the UI libraries presented in the previous table (Table: 4).

Ant Design

Ant Design is very popular UI-library especially in China market. Ant Design claims to be the second most popular UI library but depends how popularity is counted. Ant Design is complete and flexible design UI-library with plenty of components to build web applications. There is big site's using Ant Design like Alibaba, Tencent, Baidu etc.

Bulma

Bulma is an open-source CSS framework based on the CSS Flexible Box Layout from CSS standard. It provides ready-to-use frontend components to build responsive web interfaces and the design has been based on modularity and designed with a mobile mindset first. Bulma is not a JavaScript library that would be like a framework, but it is just that it is basically only a single bulma.css file that is imported for use from code. It differs from other UI libraries in that it does not use JavaScript. (*Bulma*, n.d.-b)

There is also a React component named as React Bulma Components which is implemented to support the Bulma CSS library. React Bulma Components is not the most popular UI library, but it also has a clear user base of about 5 000 weekly downloads from npmjs.com

Chakra UI

Chakra UI is a simple, modular and accessible component library gives build to React application. Chakra UI provides similar UI components than MUI v5. It is quite popular UI library (*Chakra UI - A Simple, Modular and Accessible Component Library That Gives You the Building Blocks You Need to Build Your React Applications.*, n.d.)

evergreen-ui

Evergreen UI is an open-source UI library what offers enterprise-grade, flexible and composable UI components. Evergreen UI is an open-source project launched by Segment. The main idea has been to create a library that supports today with smart components that work right out of the box and the future is considered with smart design solutions. There are almost as many components available as competing libraries, but most of components do not have that many properties to modify usage of component. Evergreen UI is not the most popular UI library, but it also has a clear user base of about 12,000 weekly downloads from npmjs.com.

Fluent UI React

Fluent UI React (formerly Office UI Fabric React) is a collection of UI components for React environment to using the Fluent Design Language. It is an open-source, cross-platform design system to Web, Windows, IOS, Android and MacOS platforms. Microsoft use it in Office 365 web applications, Dynamics, Azure DevOps etc. (*Microsoft Design*, n.d.)

Grommet

Grommet is a UI framework that provides a component-based, accessibility, and mobile-first approach to making an interface. Accessibility is an important part of Grommet and supports the W3C's Specification Web Content Accessibility Guidelines (WCAG) 2.1. Grommet provides pre-built components like other UI libraries. Evergreen UI is not the most popular UI library, but it also has a clear user base of about 25 000 weekly downloads from npmjs.com.

Material UI v5

Material UI v5 is based on Google's Material Design language version 2.0. Material Design trusts on grid-based layouts and other components that implement a similar experience in Android, iOS, and web environments. Material UI brand is changed in version 5 and Material UI new name is MUI. This change is made because people would not connect MUI directly to Google because Material name leads to Material Design too much. MUI is company what develops an open-source free version of MUI v5 version and commercial version is called MUI-X v5. Latest Material Design version 3.0 is published on Android platforms and support to other platforms is coming. MUI is the most popular UI library of about 3 million weekly downloads of v4 and v5 from npmjs.com.

Onsen

Onsen UI is an open-source framework specially designed for following design standards of mobile IOS and Android containing large set of rich UI components giving the feeling that it is a mobile application. It used for hybrid applications using Cordova or developing mobile web apps. Onsen UI uses the following web technologies HTML5, JavaScript and CSS. Onsen UI consists of three layers: (*Architecture*, n.d.; *Getting Started*, n.d.)

- CSS Components, written in next generation cssnext (*Cssnext - Use Tomorrow's CSS Syntax, Today.*, n.d.)
- Web Components, written in native JavaScript
- Framework Bindings, supports popular frameworks like AngularJS, Angular 2+,React and Vue.js

React version is called Onsen UI - React Components for Cordova/PhoneGap hybrid apps (react-onsenui).

React-Bootstrap

React Bootstrap provides Bootstrap 5 components built with React using functions and hooks supporting TypeScript language. Older version 1.x supports Bootstrap 4. Bootstrap is very popular an open-source CSS framework of mobile first front-end web development bases on HTML, CSS and optionally JavaScript design templates of interface components. It has different versions with Framework bindings to AngularJS, Angular, React, Vue.js etc. support frameworks. React-bootstrap is most popular UI library of Bootstrap for React and a second popular UI libraries for React with over a million weekly downloads.

Reactstrap

Reactstrap v9 provides Bootstrap 5 components built with React using class components supporting TypeScript language. Older version v8 supports Bootstrap 4. Reactstrap is second popular UI library of Bootstrap for React and a third popular UI library for React with over half a million weekly downloads.

React Suite

React Suite is a set of React UI component libraries for enterprise system products. It supports TypeScript and provides support for Electron. React Suite is popular in the Chinese speaking market and that is why there is not so much English language material available on the internet. The product pages have documentation in English. React Suite is not the most popular UI library, but it also has a clear user base of about 25 000 weekly downloads from npmjs.com.

Semantic UI React

Semantic UI React is official integration from Semantic UI. It is jQuery free implementation, declarative API provides features and prop validation, Augmentation enables rendering as another component inside in main component, shorthand props, subcomponents and auto controlled states. The Semantic UI is used in some projects by Amazon, Netflix, and Microsoft. Semantic UI React is fourth popular UI library, but it also has a clear user base of about 25 000 weekly downloads from npmjs.com.

Tailwind CSS

Tailwind CSS is a utility-first CSS framework packed with pre-defined classes. It provides pre-defined classes of CSS but does not delivery any out-of-the-box components. There is available over 500 professionally designed, fully responsive component examples from https://tailwindui.com/ site. Tailwind CSS is excellent solution for developers familiar with CSS, but if developer does not spend time to learn new CSS framework Tailwind CSS is not a good idea.

Tailwind CSS is easy to implement to the React project and therefore there is no separate package for the React environment. Install following dependencies tailwindcss, postcss and autoprefixer. Then generate configuration files with command npx tailwindcss init -p and configure path to template files. The last step is add tailwind directives base, components, utilities to index.css file. (Eschweiler, 2022)

4.4.2 Summary of React framework pros and cons

+ most popular JavaScript framework and ecosystem has huge amount available of third-party libraries
+ is easy to find support example from Stack Overflow and YouTube tutorials
+ easy to learn
+ TypeScript is also support
+ Meta (Facebook) is behind of open-source project
+ competent coders available

- Internal state management (Context) is limited and Redux is needed
- only “library” and have not built-in features like other frameworks

4.5 Angular

Angular (also known old version as Angular.js) is a free and an open-source client-side component-based JavaScript library for building scalable web applications which created and maintained by Google. Angular is widely used platform and framework create web applications. Angular was originally released in 2010 and was called AngularJS. AngularJS was developed Misko Heavery while working on Google. AngularJS created guidelines of modern web development and became one best single-page application platform and was very popular. AngularJS had some limitations, because of this it was rewritten in version 2. New Angular 2 was release on 14th of September 2016 and naming changed from AngularJS to plain Angular. Angular 2 support Typescript, ES5 and ES6 write Angular 2 code and old AngularJS was based on MVC (Model View Controller) architecture and it was replaced with service / controller architecture in Angular. Version 3 has been skipped and since then thirteen versions have become available, with version 14 scheduled to be released in June 2022. Angular is mentioned to be complex and its learning curve is slower than competing products because Angular has more built-in features and needs to know how to use the TypeScript programming language. (‘AngularJS’, 2022; Gudelli, 2022, p. 14; Hartman, 2020)

Features

Angular applications (figure 9) are modular, bases Angular’s own modularity system called NgModules. Application domain, a workflow or set of capabilities are based code of NgModules. NgModule defines scope of components, service providers and other code also they can import functionalities from other NgModules. During bootstrap, a root NgModule initializes a root component, but additional components can be included through the router or created through the template. A compilation context is shared to components belonging to NgModule. The view consists of a component and a template, and it can contain a view hierarchy, which allows you to create complex areas on the screen that can be managed as a unit by creating, editing, and destroying. A view is area of screen which component controls from screen. Component’s metadata is identified by

@Component decorator to the class. Metadata of component contains needed information to create and present component in its view. View is render with instructions from template which contains block of HTML code. The data binding is responsible pushing data values to HTML controls and collect changes of values from UI and updating values to back. Angular supports two-way binding mechanism. Angular's two-way binding tracks variable changes and follows up changes made by users and updates the correct value for the variable. Directives are giving instructions to DOM how render Angular templates. There are two different types of directives; structural and attribute and own directives can be defined using @Directive () decorator. Structural directives change the layout by adding, deleting, and replacing elements in the DOM. Attribute directives look like regular HTML attributes and alter the appearance of an existing element. A class with narrow, well-defined purpose and it should do specific task and do it well is called service. The injector is used to inject a service into a component allowing the component to access the service class. The Figure 9: Angular building blocks diagram below describes the Angular building blocks. (*Angular - Introduction to Angular Concepts*, n.d.; *Angular - Introduction to Components and Templates*, n.d.; *Angular - Introduction to Modules*, n.d.; *Angular - Introduction to Modules*, n.d.; *Angular - Introduction to Services and Dependency Injection*, n.d.)

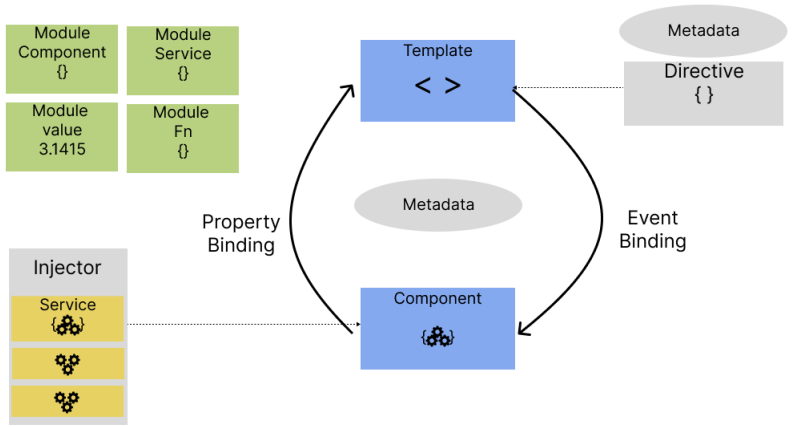


Figure 9: Angular building blocks diagram

Future

The Angular team releases major updates which contain new features and / or code optimization twice per year. Critical issues are fixed in minor updates. New Angular version 14 is released in June 2022 with following new features Strictly Typed Reactive Forms, Standalone components with optional NgModule and Extended diagnostics in compiler. (Gudelli, 2022)

Summary of Angular framework pros and cons

+ Second popular framework and feature rich framework has built-in services like http, forms…
+ Good solution for big enterprises
+ Native TypeScript support
+Predictable release cycle of release (2 per year) with LTS support
+ Google is behind of open-source project
- Performance is slowest of major front-end frameworks
- Learning curve is harder
- harder to find help from example from Stack Overflow
- Popularity is declining
- Community contentious
- harder to find skilled coders

4.6 Vue.js

Vue is an open-source JavaScript framework. It is used for building simply and easily front user interfaces (UI) and with standard building blocks like HTML, CSS and JavaScript. Vue is written in TypeScript and it has good TypeScript support.

History

Evan You has created Vue framework. He born at Wuxi in China. After high school he moved US for college. He got job of the prototyping in the browser from Google. At this time some Google's projects were using Angular framework dealing databinding and data driven DOM manipulation. This gave inspiration to him start project to build lightweight concept of the framework. The Vue project began on July 2013. He released first version 0.9 in GitHub on 25th of February 2014 with some core functionalities like declarative data binding, because he wanted share his outcome to

others. It was success and he was motivated to continue. He started the Patreon campaign to gain monetary support and after $4000 monthly fees he left Google and continued full-time on Vue. (*Between the Wires Interview with Evan You. | Between the Wires*, 2017; *First Week of Launching Vue.Js*, n.d.)

Features

Vue 3 should be faster, smaller, more maintainable, and easier to target native. This is main reason why composition API was introduced.

New Composition API is one of the most significant change of Vue 3. It is inspired by React Hooks, what allows function-based way of writing of components. It will not break anything in Vue 2 applications and Composition API is 100% compatible with old syntax and the options-based syntax. Vue 3 authors to Vue components with imported functions instead of declaring options. Reactivity API (ref() and reactive() etc.). (*Vue 3 – A Roundup of Infos about the New Version of Vue.Js - Made with Vue.Js*, n.d., p.)

Some changes from 2.x to 3.x

- Vue instances / app are now created with createApp()
- Data must now be a method
- Components, directives & third-party modules are registered on “app” instead Vue global object
- Class name changes v-enter is v-enter-form
- Router is created with createRouter()
- VueX store is now created with createStore()

New features 3.x

- Rewritten Virtua DOM for better performance
- Teleport component <teleport>
- Fragements
- Composition API will replace Options API
- Better Typescript support

Release Cycle

The life cycle management policy of Vue bases how new major version is available. Currently version 2.6 has already reached end of long term support (LTS) and end of life will be on 18th of September 2023. Vue does not have a fixed release cycle. Patch versions are released as needed and minor versions with new features are available between 3-6 months. Major version(s) are described on the road map of Vue. Major version are announced a head of time and will have early discussion phase and alpha /pre-release phases with release candidate (RC). (*Releases | Vue.Js*, n.d.)

Future

Ecosystem's open-source components combability of 3.x it took very long time and there are still components lacking support 3.0. Framework major breaking changes are stressful for third-party open-source projects and cause a need of rewrite code again. There is never a 100% common view of how the framework should be developed, and because of that there can also be a lot of resistance. The official version of Vue 3 (core) has been announced in September 2020, but default version of Vue changed to release 3 on 7th February, 2022. (*Vue 3 as the New Default | The Vue Point*, n.d.)

Evan You was interviewed at Vue Amsterdam 2022 in June. He sees that there have been challenges with communication and doing things in the development of Vue 3. Vue 3 was such a big change and broke the ecosystem for a long time, but the change will enable a better future. He mentioned about Vue's future.

"We probably won't do a "Vue 2 to 3" type of upgrade in the next five years because Vue 3 is a solid enough foundation to build upon for quite a while. We'll continue experimenting with the compilation strategy as the advantage of Vue is a really flexible reactivity system." (*Interview with Evan You - Insights About Vue 3 and Developer Experience*, n.d.)

Summary of Vue framework pros and cons

+ third popular framework
+ light-weight size

+ high performance and speed
+ easy learning curve
+ TypeScript support
+ Active community
+ popularity increasing
- no jobs opens for skilled developers
- Community slowness produces Vue.js 3 API compatible libraries.
- no big company behind of open-source project
- professional coders are observering Vue, but trust more React at this moment

4.7 Performance comparison

Frameworks are based on diverse architectures and have differences in burden of implementation and performance. The section below refers to a study of the performance of the frameworks.

Modern Web Frameworks: A Comparison of Rendering Performance article from March 2022

Risto Ollila, Niko Mäkitalo and Tommi Mikkonen have written an article, Modern Web Frameworks: A Comparison of Rendering Performance. It describes how the rendering strategies is implemented in the frameworks, Angular, React, Vue, Svelte and Blazor. These are the most used modern web frameworks. In the tests of frameworks, the same tests are performed to measure DOM synchronization results when test creating and updating DOM nodes and attributes of them. (Ollila et al., 2022a)

Rendering strategies

Frameworks have different rendering methods, React, Vue and Blazor are based on comparing two trees and calculating the minimum number of changes that need to be transferred from one tree to another. This method is solving the tree edit distance problem, otherwise it is called virtual DOM (vDOM)-based rendering. The second category are frameworks, Angular and Svelte which solve the tree edit distance problem implicitly. As in vDOM, each component defines a set of DOMs that should be rendered. Each component directly updates the portion of the DOM and component status updates as needed if there is a setup dirty flag on the data bindings. (Ollila et al., 2022b)

Performance Differences

All frameworks presented in article perform DOM updates render loop that goes through the entire component tree. Angular performs walks through the entire component tree at once, while checking all data bindings and changes have been made. Each update requires extra work on the subset of the component tree. In Angular handles manual definition for components what should not rendered. React and Blazor is using same walk-through subtree method to collect statues of the component, which initiates the render loop. Components can only permit to share their state with descendants. This ensures that the upper components cannot be affected by the change in the state of the lower component. In this way, all the components are gone through, but unnecessary work is done for those descendants that have not changes. Angular can also be used to manually specify which components should not re-rendered. Vue and Svelte only process the dirty component where the data has changed. In some cases, it must be possible to determine in advance which components are dirty. Both use a decency graph which contains all values of the components. The system will determine if the value has changed and will automatically setup a dirty flag on it. (Ollila et al., 2022c)

Table 5: Summary of factors affecting performance in the reviewed frameworks (Ollila et al., 2022c)

Framework	**Components Processed**	**Elements Processed**	**Virtual DOM**
Angular	All	Bindings only	No
React	Subtree of updated component	All	Yes
Vue	Dirty components only	Bindings only	Yes
Svelte	Dirty components only	Bindings only	Yes
Blazor	Subtree of updated component	All	Yes

Conclusions

The article reviewed Angular, React, Vue, Svelte and Blazor frameworks. The performance of the frameworks was measured by various tests. The test results shows that frameworks behave differently with how update the existing content. Results of the research, it can be concluded that in the

majority of tests, Vue is the fastest of the three, Angular is the slowest, but React is not much better. This is due to the DOM processing, where Vue only handles "dirty" components, this method clearly speeds up the screen refresh. Angular handles updating by updating all components and React updates the subtree of updated component. The rendering strategy affects the outcome a lot and Svelte was on average the fastest in the tests, Vue is in the second place, Angular and React are alternately slower on third and fourth place. Front-end application that needs a fast framework, the choice of three popular JavaScript Frameworks is Vue. (Ollila et al., 2022a)

4.8 Selection of JavaScript framework

Each of these JavaScript frameworks are designed differently and there is no such thing as a perfect framework. All of them have some defects or things that are not optimal, but each of these has been on the market for so long that the bugs / limitations of the new versions have been settled. All are good choice, but they have differences can be seen in different use cases. Because of this, they are used in various projects, and this is often also influenced by the organization's knowledge of the framework. Organization’s knowledge of that platform also influences the framework decision and how skilled people are available. Selection of JavaScript framework also has been considered in other international studies and articles. This section summarizes three different studies comparing JavaScript frameworks. Following studies and articles will be reviewed:

- **Study from March 2022**: RESEARCH AND ANALYSIS OF THE FRONTEND DEVELOPMENT FRAMEWORKS AND LIBRARIES WITH VOICE RECOGNITION IN REMOTE AREAS FOR DEVELOPING ECOMMERCE BUSINESS FOR PEOPLE OF REMOTE AREAS WHERE PEOPLE HAVE LESS OR NO KNOWLEDGE OF TECHNICAL DEVICES.
- **Article from July 2021:** Multi-Attribute Decision-Making Model for Ranking of Web Development Frameworks
- **Analysis:** Comparative Analysis on Front-End Frameworks for Web Applications by Rishi Vyas

Title: RESEARCH AND ANALYSIS OF THE FRONTEND DEVELOPMENT FRAMEWORKS AND LIBRARIES WITH VOICE RECOGNITION IN REMOTE AREAS FOR DEVELOPING ECOMMERCE BUSINESS FOR PEOPLE OF REMOTE AREAS WHERE PEOPLE HAVE LESS OR NO KNOWLEDGE OF TECHNICAL DEVICES study from March 2022.

The study compared different frameworks and justifications from their own point of view and more likely also from competence. It appears as a framework selection, and it is a good idea to take these results for analysis. *"In recent years, Angular has shown to be an outstanding framework. Many new and seasoned developers alike have been drawn to Angular because of its ease of use, developer friendliness, and potential for creating amazing applications. A cross-platform programming language, Angular is ideal for web development. Multiple operating systems can be used with it. Angular may be used to create a wide variety of apps. Desktop apps can be built using same Angular approaches as are used to build native apps for both web and mobile platforms, such as Windows, Mac, and Linux. Its possible to make native apps ."* (Prasad, 2022) The Conclusion chapter states following conclusions: *"There are three various front-end development frameworks and libraries described in this paper that can be used to create web applications, as well as a list of possible web app development solutions. React, Angular 2, and Vue are compared in a variety of ways, including data binding, language-based, technical support, volume, and performance, among other things. It's safe to say that Angular 2 has the most extensive set of capabilities and functions for large-scale commercial applications, notably in e-Commerce. React with Vue can be used for live streaming, blogging, and small and medium-sized apps. A UI framework is required for the development of a whole front-end section to demonstrate a professional UI design. Further investigation of front-end development methods and their working principles will be the focus of our future work."* (Prasad, 2022)

Title: Multi-Attribute Decision-Making Model for Ranking of Web Development Frameworks article from July 2021

This article enters into the possibilities of how rank web development frameworks. Article contains mathematical reflection how get most reliable list of ranked frameworks. This model compares twenty different web development frameworks with three different use cases where there are different emphases; What programming language is worth of choosing and studying, this information is iterated in GitHub and Stack Overflow. Following (figure9) shows rank results of JavaScript web development frameworks results of this study are in line with this study. (Borissova et al., 2021)

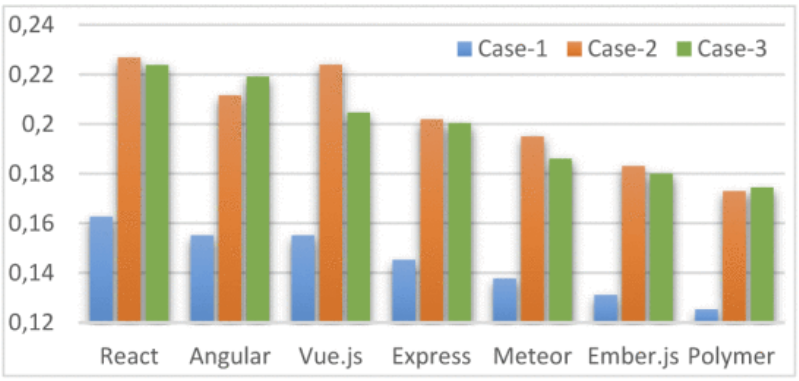


Figure 10: Ranking of web development frameworks based on JavaScript programming language (Borissova et al., 2021)

Title: Comparative Analysis on Front-End Frameworks for Web Applications by Rishi Vyas

International Journal for Research in Applied Science & Engineering Technology (IJRASET) released July 2022 Comparative Analysis on Front-End Frameworks for Web Applications by Rishi Vyas(Vyas, 2022a)**.**

All front-end developers have heard of Angular, React and Vue frameworks. Frameworks define the best use cases that can be used to implement projects using the Framework's features without implementing each feature directly through the DOM. Each of these frameworks is implemented differently and has different features. React is just a UI library, while the Angular framework offers many different services internally and Vue is a framework but does not contain so many features. A library is a collection of classes and functions, a Framework is more of a model. Web applications are important nowadays and are part of our everyday life.(Vyas, 2022a)

This study also came to the conclusion that Angular is robust and time-tested, React is flexible and fast, and Vue is simple and top-performing. All of these are safe choices for the project and the choice depends on the skills of the team. (Vyas, 2022b)

Summary of technology review

Reflection of these JavaScript Frameworks can be found in chapter 8 "Conclusions". It reflects on these and the choice of best of them and how difficult it is.

5 Design and planning phase

The planning and planning phase identify existing infrastructure and renovation needs. The environment has worked locally, and reforms have been postponed from year to year. This has created a compulsive situation where the old must be upgraded to a newer one to achieve data security. At the same time, the server and databases will be moved to the Azure cloud because investment of new hardware and maintaining server is more expensive than acquiring capacity from the cloud.

5.1.1 Old on-premises environment

The original environment has been implemented very efficiently, inexpensively, and as efficiently as possible. There is only one server in the environment that acts as a firewall, application server, and database. The server locates on premises at IK Opisto. The communication connection is implemented with an ADSL connection. The architecture is based on the LAMP stack and has added other services such as email, barcode reader and plastic card printing.(*What Is LAMP (Linux, Apache, MySQL, PHP)?*, n.d.)

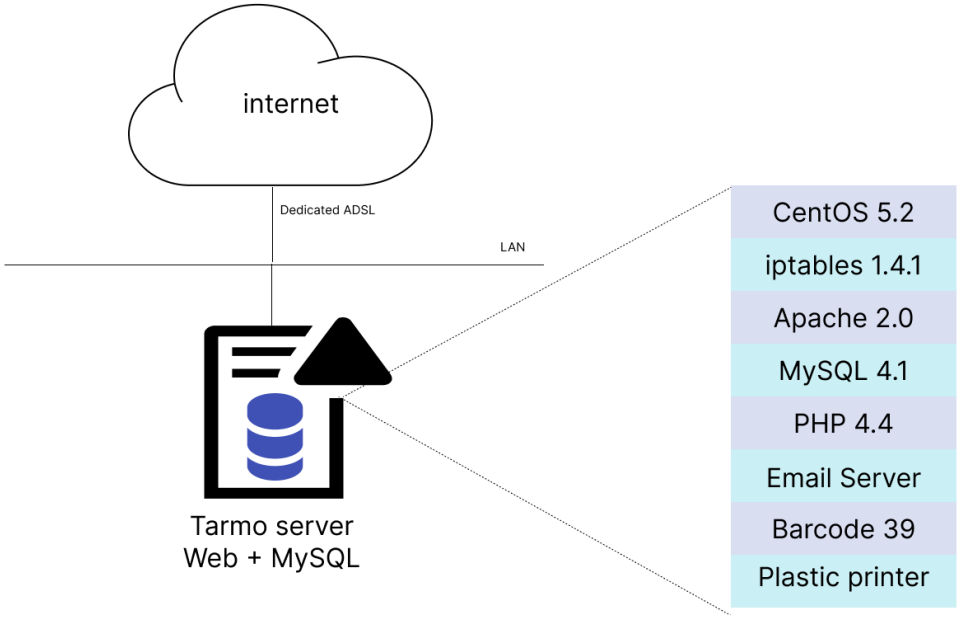


Figure 11:On premises Tarmo architecture

Architecture (figure 11) is based on the open-source LAMP stack, what is kind of software bundle what contains open-source components to build web application. Each letter of LAMP stands for four building blocks; L means Linux, A means Apache HTTP server, M means MySQL and PHP means PHP, Python or Perl. These components are available on common repositories of Linux distros.

Linux server operating system is currently CentOS 5.2. At the start of the project, the operating system was Debian (3.0 -Woody). It was the best option at the time and it could have continued to the present day by updating to the latest versions, but later at the time of the personnel changes, it was decided to replace with the CentOS version. This decision was wrong because CentOS is intended only testing and not for production use. Biggest challenges of CentOS upgrade (5.2) from release to next release (5.3). Upgrade path was missing and required reinstall CentOS 5.3 from scratch. This is laborious and operating system and component upgrades must be done manually. Environment is complex and built from many separate components. The project hasn't done upgrade this change decades, and this caused a big problem because security updates were not available after the end of lifetime support. Project is in a situation where there is something to be done and we cannot continue in the current situation.

Data communication security is enhanced by the iptables 1.4.1 firewall, which allows only the necessary traffic to the server. Server's firewall may have helped that external attacks have not reached services. The situation cannot be silenced and major changes must be made.

Apache HTTP server 2.0 is open-source HTTP server for UNIX and Windows platforms. Apache is commonly used web server. The market share is approx. 31,5 % of world's web servers (*Usage Statistics and Market Share of Apache, April 2022*, n.d.). Apache 2.0 is released on 6th of April 2002 and last maintenance fix is released on 10th of July 2013. Newer OpenSSL library requires newer Apache release and upgrade is must. (*[Announcement] Apache HTTP Server 2.0.65 Released - Apache Mail Archives*, n.d.; *'Official Release: Apache 2.0.35 Is Now GA' - MARC*, n.d.)

MySQL 4.1 is open-source relational database management system (RDBMS). Version 4.1 has been available from October 2004 (*What Is LAMP (Linux, Apache, MySQL, PHP)?*, n.d.). It is very old and has been unsupported many years.

PHP version PHP 4.4 has been released on 11th of July 2005, and latest version 4.4.9 is released 7th of August 2008 (*PHP: PHP 4.4.0 Release Announcement*, n.d.). This old version of PHP causes security issues, and CentOS have not support for newer Apache and OpenSSL what are supporting TLS1.2. This big change of Apache & SSL & PHP is required to make SSL certificates work in browser contexts because browser security enhancements prevent the use of TLS 1.0 and TLS 1.1 in encrypted traffic. TLS 1.2 was published ten years ago to avoid old TLS versions security concerns. (*TLS 1.0 and TLS 1.1 - Chrome Platform Status*, n.d.)

Because of the challenges mentioned above, the project was forced to make changes. There were two options, continue the old way and get a new server iron for your local engine room. Another option is to move services to the cloud. The decision was easy and ended up using cloud services from Microsoft's Azure cloud service. The main reason of selection of Azure cloud, is that Datacodex Oy uses the Azure cloud in other services. They have good foundations how to setup new environment and maintain it. Project has allocated Azure resources for project usage and capacity is increased depending on the load.

5.1.2 New cloud architecture

DataCodex has experience setting up an environment in the Azure cloud. They have decades of experience in programming and providing services. As an example, they have created web application called Sähköpaimen, which is a member registration system of the Finnish Pentecostal Church.

The new cloud architecture is based on their best experience of the Azure environment and has been implemented cost-effectively. Security is one of the most important features and has been a guiding factor in environment design. Services are classified according to whether they are accessible from the public Internet or only from internal networks. On the public internet side, there is an Azure firewall that blocks extra traffic from the internet side. In front of the internal network is also the Azure Firewall, which allows the API to communicate with resources such as Azure SQL, the Tarmo-Legacy API, and blob Storage.

Tarmo 3 pilviarkkitehtuuri

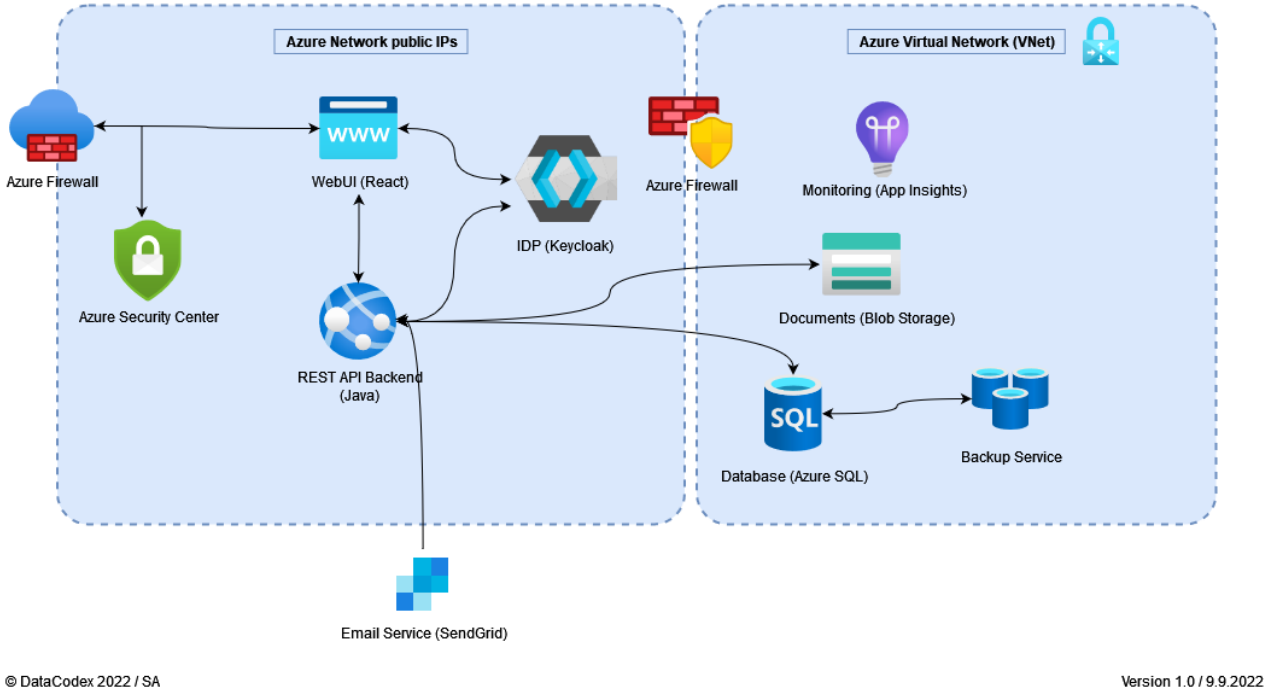


Figure 12: Tarmo cloud architecture

Components of architecture of the above figure (11) is explained in the following paragraphs.

Azure Firewall

Azure Firewall is a managed network security service to protect resources of Azure Virtual Networks. Access rights of network traffic from is defined in firewall, only https connections are opened to resources webserver and REST API server. Only necessary traffic has been opened to the resources of the private network, such as the database, email sending and Blob Storage, i.e. what the REST API backend needs.

Azure Security Center

Azure Secure Center is unified security management of cloud workloads. It collects events from Azure or log analytics to give tailored recommendations via analytics engine. It gives recommendations how increase security in cloud workloads. Recommendations are Microsoft based Microsoft best practices of the security. Security Center integrates natively with Microsoft Defender Advanced Threat Protection to protect Windows and Linux servers. (Datashield, n.d.)

In addition to this, the project services use the Microsoft Defender for Cloud product and the GitGuard product to check project versions and alert security threads.

WebUI (React)

WebUI represents component what contains container which has web server and enrolment React application. The project uses automatic deployments from GitHub projects.

REST API Backend (Java)

Rest API backend has new Tarmo API services. Enrollment service use API services and authorization (IDP) via this API. REST API is implemented with Java BootSprint.

Azure Virtual Network (VNet)

Azure Virtual Network (VNet) reprensents internal networks of where locates additional services like databases and traffic is limited from the public side.

Monitoring (App Insights)

Application Performance Monitoring (APM) is providing performance monitoring and is an extension of Azure Monitor. APM works proactively to understand performance of application or reactively to determine the cause of an incident from review of application execution data. The coder

can add lines of code to use Insights functionalities from SDK in error handling situations to transmit issue information to App Insights. (AaronMaxwell, n.d.)

Documents (Blob Storage)

Microsoft's object storage solution for the cloud is Azure Blob storage. Blob storage is used and optimized for big amounts of unstructured data. This enrolment service does not use Blob Storage, but next release of Tarmo will use it for example for print queues. (tamram, n.d.)

Database (Azure SQL)

Azure SQL is database server is database service in Azure cloud. In this project, the database is used only by the servers of the new Tarmo API. Database services are in the internal network (private), traffic is protected by a firewall and allowed traffic from API servers.

Backup Service

The databases are backup by Backup Service of Azure.

IDP (KeyCloack)

An identify provider IDP manages authentication services for services. IDP's task is to create, maintain and manage the identity information of users, services or systems and provides authentication to other service providers(application) within federation or distributed network. IDP is a reliable operator trusted by third party like users, server and they use it when authentication is needed. The project does not use the authentication services provided by Azure, project selected KeyCloack, an open-source identity and access management solution. It implements almost all standard IAM protocols (OAuth 2.0,OpenID and SAML) from out of the box and customizable aspects of product or module. (*IdP (Identity Provider) - Glossary - Hermes*, n.d.)

OAuth 2.0 (Open Authorization) is the industry standard authorization protocol. It is designed to grant access to a web site or application to resources hosted by other apps on behalf of a user.

Oauth 2.0 is not an authentication protocol, but it is an authorization protocol. An Access Token are used in OAuth 2.0. Access Token includes information about access information to the resource on behalf of the end user. Generally, the JSON Web Token (JWT) format is used, but OAuth does not specify the format to be used. An expiration date is included in Access Tokens for security reasons. In this environment, authorization with KeyCloak OAuth 2.0 is used. (*What Is OAuth 2.0 and What Does It Do for You?*, n.d.)

Email Service (SendGrid)

Twilio's SendGrid service is used to send emails from Tarmo REST API backend. During winter-spring time, it was used to send notification-, and invitation letters to previous years' supervisors and workers. The enrolment process of volunteer started by ordering a registration link into the email.

Project use GitHub integration, CI / CD build pipelines and DevOps from Azure. These features are included in the same pricing and the project will use these features to reduce delivery time.

5.2 Front-end Architecture

The new architecture will be created to support the shift from the old platform to the hybrid model and the ultimate new technology environment. The journey has begun, and the first stop is the so-called hybrid model. In this model, assistant enrollment is moved to the cloud and Legacy is made with new technologies, and an API is provided to convey information between the new ones. Planning will focus on this phase and other phases will come later. The architecture is designed to be as scalable as possible without the need to make any major architectural changes in the future.

The project implements Tarmo registration as a Single Page Application (SPA) using React and backend. React code is as simple as possible by using reusable components as much as possible. SPA acts as presentation layer and renders pages via DOM.

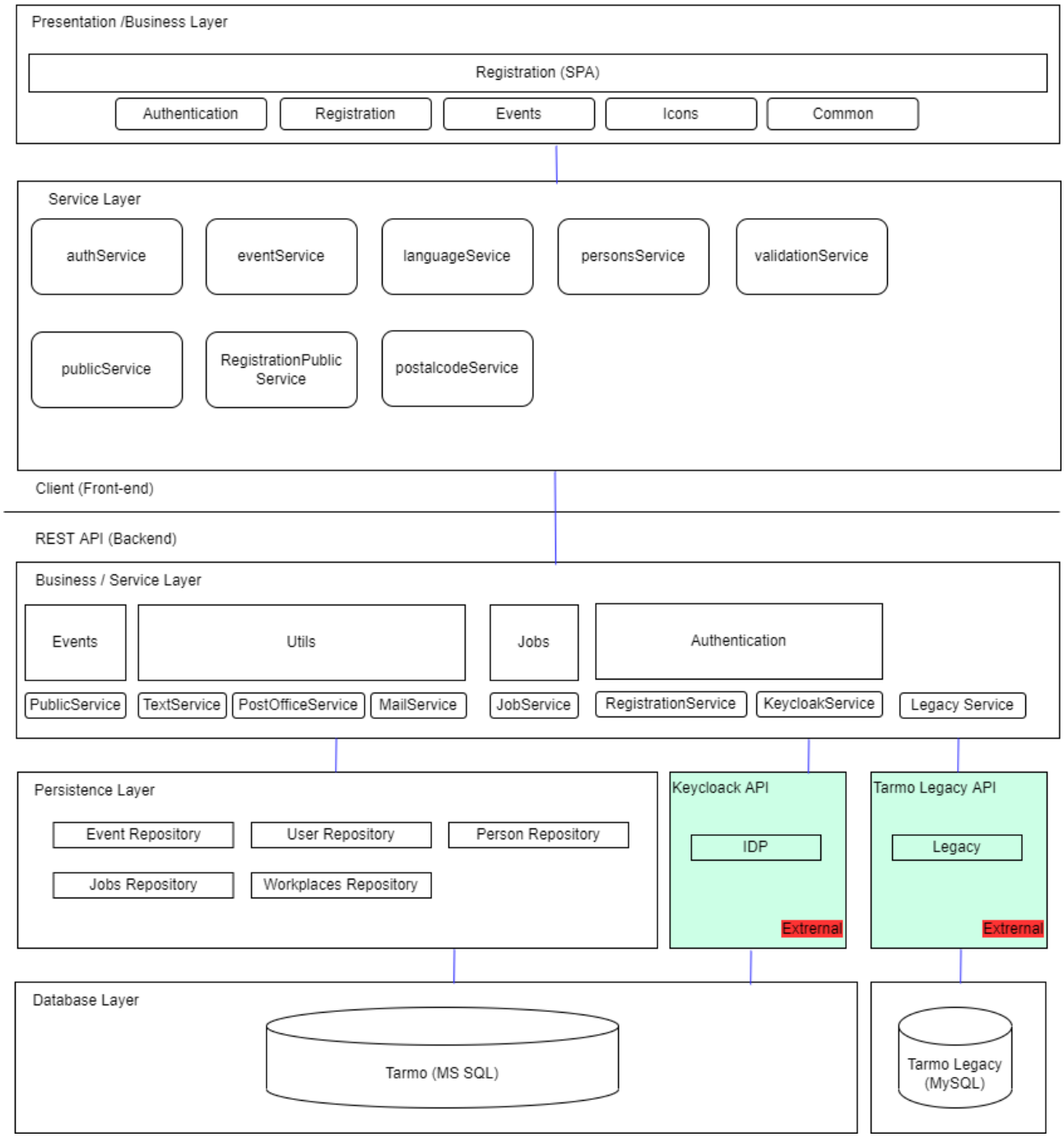


Figure 13:Tarmo Registration layout architecture

Client (Front-end)

The presentation and Business layers are together in the front-end. The functionalities are common and that is why they are described in the same rectangle. React application (SPA) is running in the presentation layer and ensure all necessary browser task like rendering. React uses external libraries for various functions like MUI for appearance of the UI, Axios for REST queries and Formik

for forms. React Router handles url redirects without traditional web HTML-page fetch from the www-server.

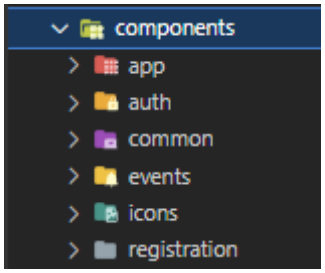


Figure 14: Tarmo Registration business layer components

The business layer consists of Tarmo Registration front-end components. The figure shows the main components. The business layer has components organized according to functionality of component. This makes it easier to understand functionality of software code.

The Services layer of the front-end contains logic that is called via the business layer. Services layer components communicate with backend services.

REST API (Backend)

The backend side has a combined business and service layer. The services are divided according to which service they are related to. The front-end communicates with these services. Related REST API interfaces are compiled under each business layer service. Service layer is provided by the microservice APIs. Part of the API works without authentication, and most of it requires a working token-based authentication. APIs belong to the business layer. APIs are sorted by functionality into the business layer, and they are documented in the Swagger service (https://tarmo-api.isokirja.fi/swagger-ui/index.html). Swagger is an automated service of Azure cloud to document REST APIs.

.

In Java programming, the Persistence layer is often called the repository layer. Both layers have the same task of guaranteeing that the business layer can reach the data and the cache. In Tarmo environment Persistence Layer provides authentication-related services through KeyCloak and Tarmo Legacy's services are available through the Tarmo Legacy API.

The database layer can also be called the storage layer. Layer's task is to store data and deliver it to the service layer. In general, the information is stored in databases, but the layer does not care where it is stored. In Tarmo environment, the new Tarmo database is MS SQL and the retiring lecagy Tarmo database is MySQL.

5.3 Supervisor registration flow

This chapter covers Supervisor Registration process. Flow is complicated, but in this way, an easy-to-use experience for the end user can be implemented without logging into the service. Identification is done with an e-mail address.

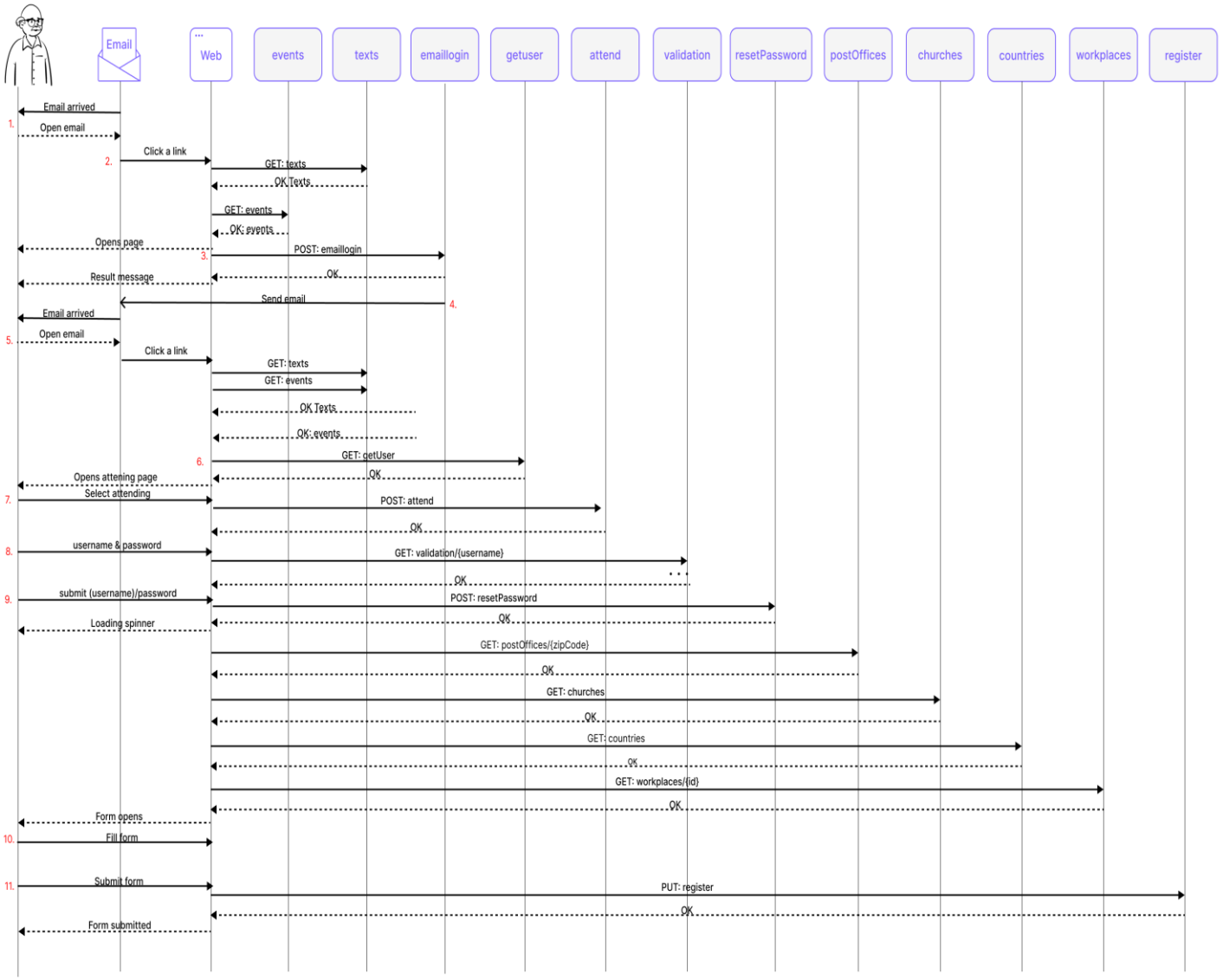


Figure 15:Supervisor registration flow

Supervisor Registration flow (figure 15)

The purpose of the registration process is to make the login process as simple and secure as possible. Supervisors have IDs for the old system and IDs has transferred to new platform, but for security reasons, passwords need to be changed to the most secure ones, and some users must change username due too short usernames. Because of this, we ended up using an email link to sign up. The user will receive an invitation letter if there is a link to register. The user's email address has been added to the link. By clicking this link, the system will order a login link from the emaillog API to its email address. The supervisor checks in to check if the user exists and only then sends a login

link to the email address. For security reasons, the correct error messages that the service cannot be used are not displayed. This is because the existence of user cannot be ascertained.

1. Supervisor gets invitation email from Iso Kirja Opisto. Email contains instructions and button what contains embedded link with email-address to page where registration link can be ordered.
2. Supervisor clicks link in email and registration link order page opens.
3. Supervisor press submit button with her/his email address. Front-end submits email address to emailLogin API.
4. Backend sends email to supervisor via Azure SendGrid.
5. Supervisor opens email and click button from email. Button contains embedded link with action token.
6. Front-end verifies action token with getUser API and gets related information of user. Front-end renders attending page.
7. Supervisor select attending and front-end post it to attend API.
8. If supervisor is attending, opens change of password and some circumstances also change username. Front-end validates is password valid and validates new username.
9. Supervisor submit new password.
10. Registration form opens.
11. Supervisor fill form and submits it. Front-end post information to Registration API and if registration was successfully, front-end displays successfully message error scenarios it displays error message

    * Note: This diagram does show flow between API to IDP (KeyCloack)

Registration-service

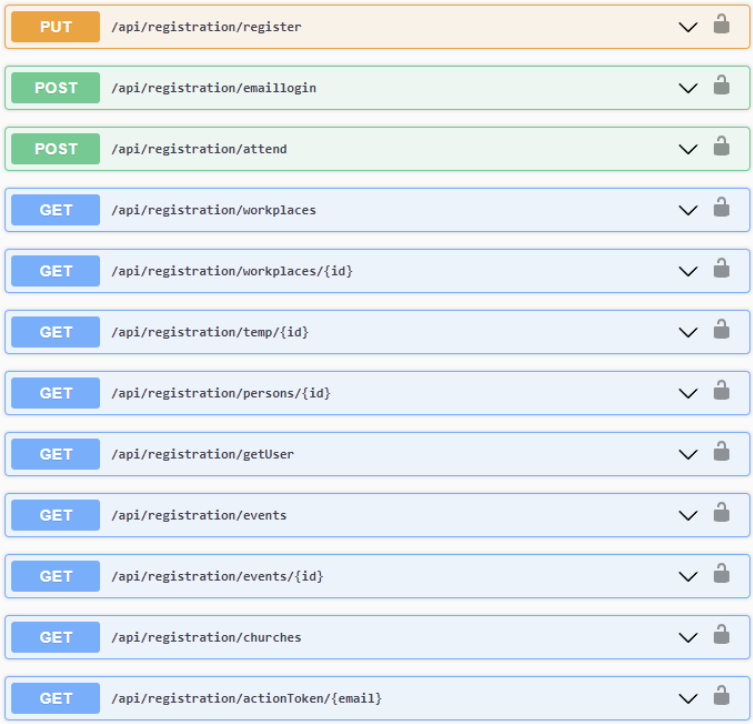


Figure 16: Tarmo-API registration-service

REST services related of registration (enrolment) are available from registration REST API (figure 16).

Job-service

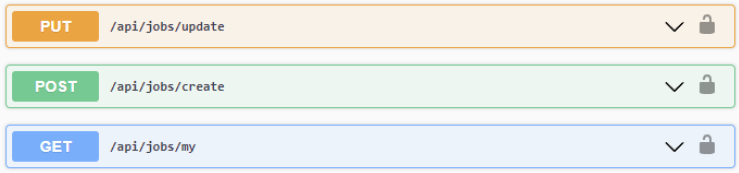


Figure 17:Tarmo-API job-service

Rest services related to jobs are available from jobs REST API (figure 17)

Public-service

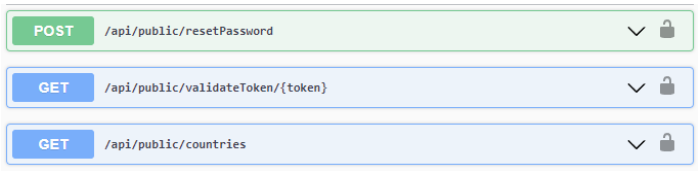
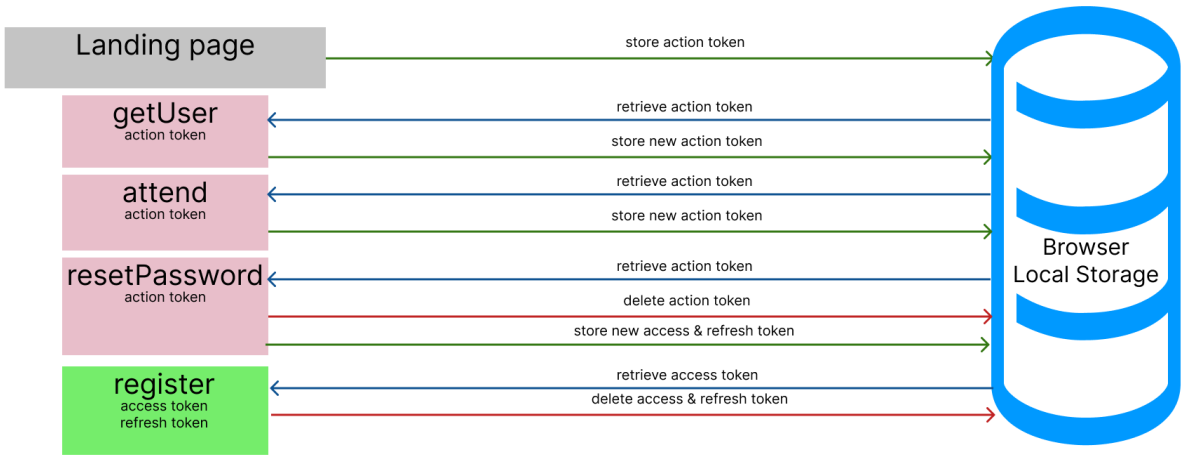


Figure 18: Tarmo-API public-service

Enrollment process includes necessary password reset of user due of better security and REST API services are available from public API (figure 18).

5.3.1 Authentication

The Emaillogin API sends a registration URL link to the email via Azure Sendgrid. Link contains information about the event, its role and is included in the action token. The URL syntax is https://address/{event}/{role}/{action_token}. This takes into attention the future that there may be many events going on at the same time.



The login page identifies the service, role, and action token and takes you to the right registration page. The Action token is stored in Local Storage so that it can be used in the next API call. The next API call is attend and it returns a new action token. Every action token is disposable and binded to use with the specified API. All users must change their password via resetpassword API. Resetpassword API returns access token and refresh token for secure connection if authorization

succussed. Error handling of front-end application check authorization status and navigate to the form or error page. Access token is used in register API and after successful submit tokens are cleared from browser's local storage.

First action token is valid 180 minutes (3 hours). Access token is valid for 180 seconds and Axios uses refresh token to obtain new access token.

5.4 Technologies

In the design phase, the technology stack used in the implementation is decided. I have limited experience in various software architectures and have no work experience in the field of software engineering. The project has more than 25 years of experience in software engineering. This best practice information is used to make decisions. The selected architectures are proven and used in professional projects.

Backend components

Backend services are also implemented with the best practices in terms of the group's expertise. The expertise of Java Spring Boot-based micro service programming is the reason for choosing it for the programming environment to implement API services. IDP is implemented with open-source KeyCloack software. The Microsoft Azure cloud service was chosen as the cloud service platform. This decision was made because DataCodex uses the Azure cloud environment to provide their services.

Front-end framework

The front-end framework could be one of these frameworks reviewed above; Angular, React and Vue and the choice of any of them would be good and fitting to this project. At this time, the project decided use React v17 with the TypeScript feature. The main reason is that the React experience can be found from team. React v16 has been used a lot in projects with good results. Usually, a coding project starts with a boiler plate project that has as many already implemented features

as possible. This time, the boiler plate project will initially require version upgrades, which are described in more detail later in this document.

UI framework

Today, UI framework needs to be able to scale from PC screens to tablets and mobile devices.

5.5 UI-Design

The design of the user interface must consider the different skills of the users and how smoothly they use the web software. Users range from over 80 to about 10 years old.

The design considers the fact that it is used from a computer and a mobile device. One important feature is the single sign-on with the authentication link, as most do not store the old system password. For security reasons, it was decided that everyone would change their password to be more secure when logging in to the system. There are pictures of the old user interface where the data was collected, but otherwise the planning starts from scratch.

The UI layout was chosen for the Material UI because it is widely used and is the latest version of MUI 5.0 that supports React version 17.x. It also scales well to the resolutions of mobile devices and allows easy use of different terminals.

5.5.1 UI Protyping

The project used software for prototyping that provides easy insight into what the output might look like. We had decided up using Figma software designed for making and prototyping vector graphics. It is available as a web-based and native application for Windows and MacOS machines. Android and iOS versions of the software are available for real-time prototyping on mobile devices.

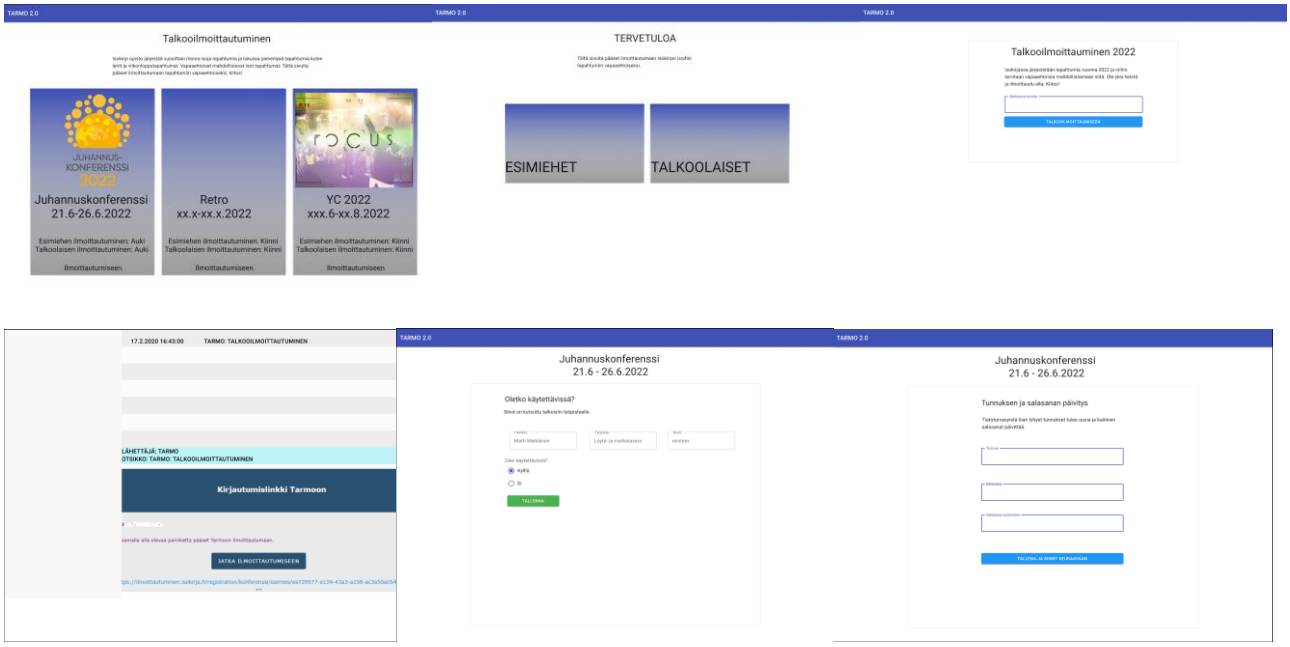
I was not aware of Tarmo system so well, I had only experience with the role of the end user to enroll volunteer. I had to investigate old documents and screenshots of the old system to gain knowledge of previous situation. This year’s plan is replaced old JuhannusKonferenssi enrolment

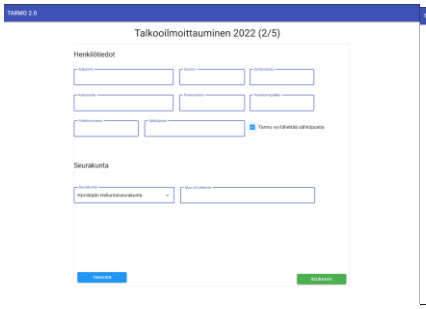
service and rest will be done in the future. The enrolling user interface is designed for this use case only, and the menu structures, etc. of the future Tarmo system will not be planned at this stage.

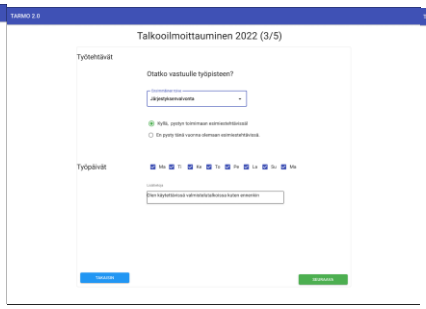
Project approached design from the perspective that we are agile to make changes to plans. Prototyping did not consider how APIs should work and their specs. For this reason, many different versions of these prototype diagrams were designed when challenges were encountered on design and implementation phase. Security suspicions of old system led to changes in flow. Users will be migrated from the old system and a decision was made that everyone will need to change new secure password and username long enough. From an architectural point of view, this process must be done at the beginning and that is why the change of ID and password came.

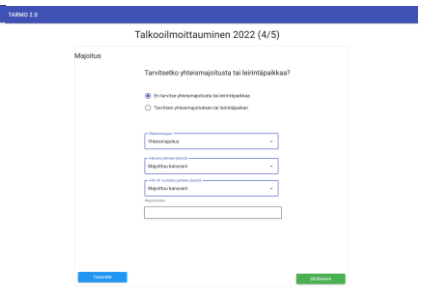
Many different versions were made in the design process to understand how the backend should work. I also decided to make changes to the operation of the user interface to get a workable service on the backend side.

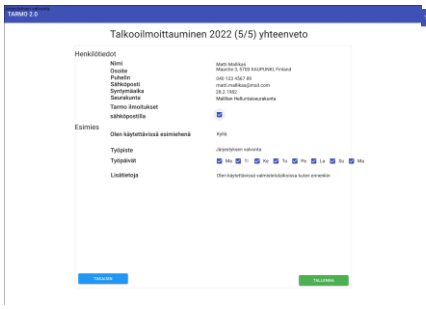
Link to Figma prototype https://www.figma.com/file/7XJUPg57J5pG8kOnWfU1On/Tarmo?node-id=2%3A2

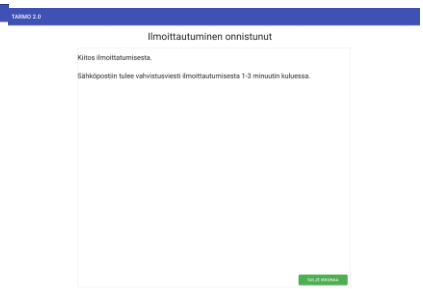



TARMO 2.0
Talkooilmoittauminen 2022 (2/5)
Henkilötiedot
Seurakunta
TAKAISIN
SEURAAVA


TARMO 2.0
Talkooilmoittauminen 2022 (3/5)
Työtehtävät
Otatko vastuulle työpisteen?
Työpäivät
TAKAISIN
SEURAAVA


TARMO 2.0
Talkooilmoittauminen 2022 (4/5)
Majoitus
Tarvitsetko yhteismajoitusta tai leirintäpaikkaa?
TAKAISIN
SEURAAVA


TARMO 2.0
Talkooilmoittauminen 2022 (5/5) yhteenveto
Henkilötiedot
Esimies
TAKAISIN
TALLENNA


TARMO 2.0
Ilmoittautuminen onnistunut
Kiitos ilmoittatumisesta.
Sähköpostiin tulee vahvistusviesti ilmoittautumisesta 1-3 minuutin kuluessa.
SULJE IKKUNA

6 Implementation

This section walks you through the progress of a software project. The project started in early February, a bit of a surprise when there was no certainty whether an event would be held. Before deployment, project need to migrate your old Tarmo Legacy Azure to the cloud and implement the Legacy API to forward enrollments to the Legacy side.

My experience of React coding was limited in the beginning the project. I have learnt React programming in Application Development course with vanilla JavaScript. Early discussion with project team I noticed that vanilla JavaScript is not enough for professional programming, and I had to study and become familiar with TypeScript programming. I attended Udemy's course called "Understanding TypeScript – 2022 Edition (*Learn TypeScript (Ditch JavaScript)*, n.d.). I got the basics of TypeScript programming to be able to program TypeScript React code for a project.

The project decided that use the boiler plate code from Datacodex's another project React-front codebase. This boiler plate uses older versions of React, React Router, Material UI and many other components. First task is upgrade boiler plate code to the latest versions.

6.1 Upgrade of boiler plate code

Upgrading is always a leap into the unknown and it's hard to know what's going to come against everything. Open-source software often has trouble upgrading because support lags behind. So the features and support of the different versions of React were examined. It was then decided which versions to try to deploy. The table (6) below shows which shoots have been updated to the latest versions.

Table 6: Boiler-plate code versions

Component	Orginal version	Upgraded version	Notes
typescript	4.03.05	4.05.04	
react	16.14.00	17.00.02	
react-dom	16.14.00	17.00.02	

react-scripts	4.00.03	4.00.03	Too many issues with 5.0.0 --> downgraded
@types/react	16.09.56	17.00.38	
@types/react-dom	16.09.09	17.00.11	
@testing-library/jest-dom	5.14.01	5.16.01	
@testing-library/react	12.00.00	12.01.02	
@testing-library/user-event	13.02.01	13.05.00	
@types/jest	26.00.24	27.04.00	
@material-ui/core	4.12.03		
material-ui/icons	4.11.02		
material-ui/lab	4.0.0-alpha.60		
material-ui/pickers	3.03.10		
material-ui-search-bar	1.00.00		
emotion/react		11.07.01	New MUI 5.0
emotion/styled		11.06.00	
mui/material		5.02.07	
types/node	16.04.13	16.11.19	
@mui/icons-material			
notistack	1.00.10	2.00.03	

6.1.1 React 17 and Typescript

React 17 is a stable version. It was released on October 20, 2020. At the implementation stage the new React 18 was still in beta. React version 17 was the right choice for this situation. The upgrade went well, but React-Scripts 5.0 caused problems. The node libraries used by React did not support all the security requirements for Typescript libraries and were not compatible. For this reason, it was decided to downgrade React-Scripts to version 4.0.

```
"dependencies": {
    "typescript": "^4.5.4",
     "react": "^17.0.2",
        "react-dom": "^17.0.2",
        "react-scripts": "5.0.0",}
    "devDependencies": {
```

```
        "@types/react": "^17.0.38",
        "@types/react-dom": "^17.0.11",}
```

The original software project also uses the Typescript language. The change is just a change of version from 4.3 to 4.5. This change was straightforward and did not cause any problems.

6.1.2 React Router

The React Router v6 update changes the syntax and needs some changes to code. The update is generally well documented, and the changes make sense. React Router v6 introduced a Routes component. It is kind of like Switch, but a lot more powerful. Routes are chosen based and best match is selected instead of being traversed in order. The new Route has few changes like exact is gone and descendant routes use trailing * in the path. (*React Router | Upgrading from V5*, n.d.)

React Router v5	**React Router v6**
<BrowserRouter></BrowserRouter>	<BrowserRouter></BrowserRouter>
<Switch></Switch>	<Routes></Routes>
<Route exact path="/"></Route>	<Route path="/" element={<Home />} />
const history = useHistory();	const navigate = useNavigate();

```
import React from 'react';
import { BrowserRouter } from 'react-router-dom';

const rootNode = document.getElementById('root');

ReactDOM.render(
    <React.StrictMode>
        <BrowserRouter>
            <App />
        </BrowserRouter>
    </React.StrictMode>,
    rootNode
);
```

In Index.tsx capsulates BrowserRouter functionality of React Router to all subcomponents. This is done one component higher than the Routes components. There were restrictions on placing both in the same component.

```
import React from "react";
import { Route, Routes } from "react-router-dom";

const AppRoutes: React.FC = () => {
  return (
    <Routes>
      <Route path="/" element={<RegistrationView />} />
      <Route path=":event_name" element={<EntranceSelection />} />
      <Route
        path=":event_name/:volunteer_role/:email"
        element={<EventView />}
      />
      <Route path=":event_name/:volunteer_role" element={<EventView />} />

      <Route
        path=":event_name/talkoolainen/ilmoittautuminen"
        element={<PreparationVolunteerPhase />}
      />
      <Route
        path=":event_name/:volunteer_role/ilmoittautuminen"
        element={<PreparationPhase />}
      />
      <Route
        path="registration/:event_name/:role/:action_token"
        element={<LandingPage />}
      />

      <Route path="*" element={<PageNotFound />} />
    </Routes>
  );
};
```

In AppRoutes.tsx contains Routes and Route components. In version 6, Routes component replaces the Switch component. In the route components, the syntax has changed; The Exact flag is no longer needed but the addresses are always exact and in other situations the wilcard * is used in the url, the component is replaced with element. This change allows data to be passed to the component through the element.

6.1.3 Material UI

Material UI 5.0 was released in 16th of September 2021. Our project had some experience of MUI 5.0 and this experience showed that it made sense to upgrade to it. As the version changed, the brand also changed from Material UI to MUI. In the new version, components <Stack>, <Autocomplete>, <Pagination>, <Skeleton>, <SpeedDial> and <ToggleButton> have been moved from the Material UI Lab to MUI core. MakeStyles has been replaced more simply by sx prop. Typescript support has been improved and many others changes. (*Introducing MUI Core v5.0 - MUI*, n.d.; Zaninotto, n.d.)

The update was done by following the MUI migration from v4 to v5 guide component one at a time (*Migration from v4 to v5 - Material UI*, n.d.).

After updating the MUI components, changes must be made to the code to enable MUI v5. The boiler plate code has been running Material UI v4 and numerous components are using the Material UI and require an update. A few components have minor changes and some props have been changed / mitigated.

First uninstall old Material UI v4 components, then install MUI v5 components.

```
npm install @mui/material @emotion/react @emotion/styled
npm install @mui/icons-material
npm install @mui/lab
npm i @mui/styles
```

Change imports from @material-ui to @mui/material. MUI v5 has changed the naming of packages and therefore each import must be changed. A few features have also been moved from the lab side to the core of MUI. A few components have minor changes to property names and values that were previously supported, such as "default", and have been removed from color property at least.

```
// Old syntax Material UI v4
import { Container, Grid, Typography } from '@material-ui/core';
import { Settings } from '@material-ui/icons';
```

```
import * as locales from '@material-ui/core/locale';
// new syntax Material UI v5
import { Button, IconButton, } from '@mui/material';
import AddIcon from '@mui/icons-material';
import * as locales from '@mui/material/locale';
```

6.2 Implementing enrolment service

After completing the specifications, work began on the software project. The work was done according to how the back-end APIs were completed. The schedule was already tight at the beginning and that was why the project was completed in two phases. In the first stage, a Supervisor Enrolment Service will be made. After that, a volunteer enrolment service will be implemented.

The intention was to plan execute to the result of the design, but a few times we had to go back to the design phase when there was a stopping issue or design limitation. These challenges were mainly due of depending on design of legacy-Tarmo and limitations of API implementation. As an example, the first plan of enrolment service was everything like changing password and submit form data is handled by a single API PUT message. This became the first design challenge and we had to return to the design meeting. Because of the change added new steps to registration flow and decided to redesign later the volunteer's phase. The reason for this change was as follows security limitation. Email login link contains an action token information for getUser API, but IDP requires login to get an access token. Login process is handled through the resetPassword API and after that we obtain the correct access token.

6.2.1 React Context

In this project, it was considered whether to use React's internal Context feature or to use an external Redux service to manage the facilities. It ended up using React Context functionality instead of Redux. The following reasons led to the conclusion; Redux creates unnecessary complexity vs. React Context, application does not have many states handling in the environment and Context feature handles this amount of a load. The context is used to manage the language texts of the user interface. They are loaded at once and accessed by the components through Context.

The language context is providing application UI's texts. It supports Multilanguage, but in this phase, UI is handling only Finnish language. React Context provides a way to pass data available the component tree without using props to pass data to subcomponents. In this project React Context of language is defined on *languageContext.ts* file. This file contains definitions for the LangString, CountryCode, and Language interfaces. The exported typed functions of LanguageText and LoadLanguageStrings are introduced for use. In LanguageContextInterface, the type defines the Context interface for language texts. According to this type, initial values are defined for the React Context.

```
export interface LangString {
    key: string;
    value: string;
}

export interface CountryCode {
    code: string;
    value: string;
}

export interface Language {
    lan: string;
    value: string;
}

export type LanguageTextFunc = (key?: string) => string;

export type LoadLanguageStrings = () => Promise<AxiosResponse<LangString>>;

export interface LanguageContextInterface {
    langStrings: LangString[];
    languageCodes: Array<Language>;
    countryCodes: Array<CountryCode>;
    T: LanguageTextFunc;
    setLanguageStrings: (strings: LangString[]) => void;
    setLanguage: (language: AppLanguage) => Promise<void>;
}

const initialState: LanguageContextInterface = {
    langStrings: Array<LangString>(),
    languageCodes: Array<Language>(),
    countryCodes: Array<CountryCode>(),
    T: (): string => '',
    setLanguageStrings: () => null,
    setLanguage: () => Promise.resolve()
};
```

```
const ctxt = React.createContext<LanguageContextInterface>(initialState);
ctxt.displayName = 'LanguageContext';

export const LanguageDefaultState = initialState;
export const LanguageContext = ctxt;
export const LanguageProvider = ctxt.Provider;
export const LanguageConsumer = ctxt.Consumer;
```

The code in App.tsx handles retrieving of language context. UseMemo hook from React is used with languageState function. This implementation is done with useMemo to reduce unnecessary computing in every render and returns memoized value, if it is available from memory otherwise it loads language text strings to memory. The useEffect hook is firing loading of language text strings.

```
const languageState: LanguageContextInterface =
useMemo<LanguageContextInterface>(() => {
  return {
    langStrings: initLangString(),
    languageCodes: LanguageService.getLanguageCodes(),
    countryCodes: LanguageService.getCountryCodes(),
    T: (key?: string): string => {
      if (!key) return "";

      const langStr = languageState.langStrings.find(
        (item: LangString) => item.key === key
      );
      return langStr ? langStr.value : key;
    },
    setLanguageStrings: (strings: LangString[]): void => {
      languageState.langStrings = strings;
    },
    setLanguage: async (newLanguage: AppLanguage): Promise<void> => {
      // Load translation strings
      const translations = await LanguageService.getLanguageStringsAsync(
        newLanguage
      );

      languageState.setLanguageStrings(translations);
      moment.locale(newLanguage);
      setStateLanguage(newLanguage);
    },
  };
}, []);

useEffect(() => {
```

```
// Load translation strings
const langPromise = languageState.setLanguage(language);

Promise.all([langPromise]).then(() => {
  setLoaded(true);
});

}, [language, languageState, authState, user, navigate, location]);
```

LanguageService.ts contains the LanguageService class to retrieve language texts. Function called LocalCache.cachedRequest is called with url parameter and call nested function. This checks is texts stored to localStorage and are not expired or retrieve text strings from Texts API.

```
import axios from 'axios';
import { AppLanguage } from 'types/common';
import { LangString, CountryCode, Language } from '../contexts/languageContext';
import LocalCache from './localCache';
import config from 'config/config';

class LanguageService {
    public async getLanguageStringsAsync(language: AppLanguage): Promise<Lang-
String[]> {
        const url = `texts/${language}`;
        if (config.STAGE !== 'production') {
            // reread language strings always on development
            const axiosInstance = axios.create(); // create separate instance to
avoid interceptors
            const response = await axiosInstance.get(url);
            return response.data;
        }

        return LocalCache.cachedRequest<LangString[]>(url, async () => {
            const axiosInstance = axios.create(); // create separate instance to
avoid interceptors
            const response = await axiosInstance.get(url);
            return response.data;
        });
    }

    public getCountryCodes(): CountryCode[] {
        return [
            { code: AppLanguage.Finnish, value: 'Finland' },
            { code: AppLanguage.Swedish, value: 'Sweden' },
            { code: AppLanguage.English, value: 'England' }
        ];
    }
```

```
    public getLanguageCodes(): Language[] {
        return [
            { lan: AppLanguage.Finnish, value: 'Finnish' },
            { lan: AppLanguage.Swedish, value: 'Swedish' },
            { lan: AppLanguage.English, value: 'English' }
        ];
    }
}
export default new LanguageService();
```

App.tsx returns JSX component what is wrapped with <LanguageProvider value={languageState}>. This provides language texts available to all subcomponents.

```
return (
    <LanguageProvider value={languageState}>
      <AuthProvider value={authState}>
        <CookiesProvider>
          <ThemeProvider theme={theme}>
            <LocalizationProvider dateAdapter={AdapterDateFns} locale={fi}>
              <AppLayout locale={language} />
            </LocalizationProvider>
          </ThemeProvider>
        </CookiesProvider>
      </AuthProvider>
    </LanguageProvider>
  );
```

This example shows how to use LanguageContext from component level. It needs imports of useContext from “react” and LanguageContext from “contexts/languageContext”. Text data is available via function call T (“Keyname”).

```
import {useContext} from "react";
import {LanguageContext} from "contexts/languageContext";
import {TextBox} from "components/TextBox"

const demoComponent=()=>{
const { T } = useContext(LanguageContext);
return (<TextBox label={T("Common_TextBox_Label")}>)
}
export default demoComponent;
```

6.2.2 Enrolment Form with Formik

Building forms might be very complex and due of this was decided to use the Formik library for building forms. It helps getting values from form and allows manipulation, validation and error messages and handling form submission. The intention was to build a form that might support all the different cases and not do the same thing many times. API returns personal information, possible workstation information, and Accommodation Information. This data comes from Tarmo Legacy and may contain gaps or errors. Before setting initial data for Formik, a few checks are executed and, for example, postal validation is done via API with zip code and it returns post office information and it is populated, if it was misconfigured or missing. For a new user, the first name and last name are initialized after they are created by the API.

The Formik form will be initialized with this data and the form is ready for use. Form is complex due of different use cases and validation depends on use cases. In some situation, extra values are populated into Formik values or subcomponent updates states in main level component.

6.2.3 Enrolment form validation

Formik is designed to build and manage complex form. It has extensive features of validation. Form-level validation is practical and have access to all the values of fields in the form and validate the dependencies of the other fields. Field-level validation is also supported via validate prop of <Field> or with useField hook. Formik validations can be synchronous or asynchronous. Formik also supports validation schema approach. Self-coded validators or 3rd party libraries can be used for validation. Formik authors have included yup library support and Formik has special configuration option called validationSchema which automatically transform validation errors to object and error status is available build business logic around error messages.

The enrolment form uses the Yup library for validation. The aim is to validate as much as possible to ensure the registration fields are as accurate as possible and the required information is specified. Each Formik step has its own validation schema that is defined inside the element.

```
<FormikStep
```

```
label={T("Enroll_Personal")}
validationSchema={getValidationStepPersonalSchema(T)}
>
<StepPersonalInformation
  event={event}
  isSupervisor={isSupervisor}
  countryList={countryList}
  responsibilityArea={securityArea}
  handleAge={(value) => setCurrentAge(value)}
/>
</FormikStep>
```

Yup validation is versatile and allows you to use built-in functionalities such as field min -and max lengths, required, matches field contents with regex functionality and test functionality allows to build own code to run validation. Versatile validation is easy to implement with yup's features. The example below has a small snippet of validation rules.

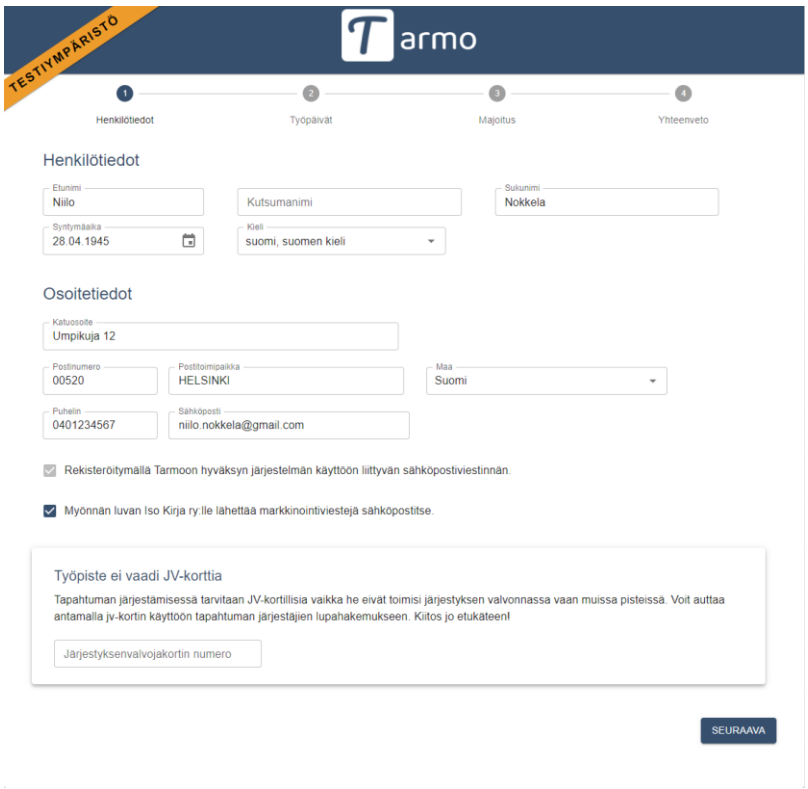
```
import * as Yup from “yup”

const getValidationStepPersonalSchema = (T: LanguageTextFunc) => {
    return Yup.object().shape({
      person: Yup.object().shape({
        firstName: Yup.string()
          .min(2, T("Common_Required"))
          .max(225, T("Common_Field_Long"))
          .required(T("Common_Required")),
        country: Yup.string()
          .max(225, T("Common_Field_Long"))
          .required(T("Common_Required"))
          .test("country", T("Registration_Country_Required"), (value) => {
            if (!value || value === null) {
              return false;
            } else {
              return true;
            }
          }),
        email: Yup.string()
          .email(T("Common_Email_Wrong_Type"))
          .max(255, T("Common_Field_Long"))
          .required(T("Common_Required")),
        birthDate: Yup.string().test(
          "birthDate",
          "syntymäaika on väärässä muodossa. Oikea muoto on:pp.mm.vvvv",
          (value) => isBirthDateValid(value)
        ),
```

```
      }),
    });
};
```

6.2.4 Web implementation

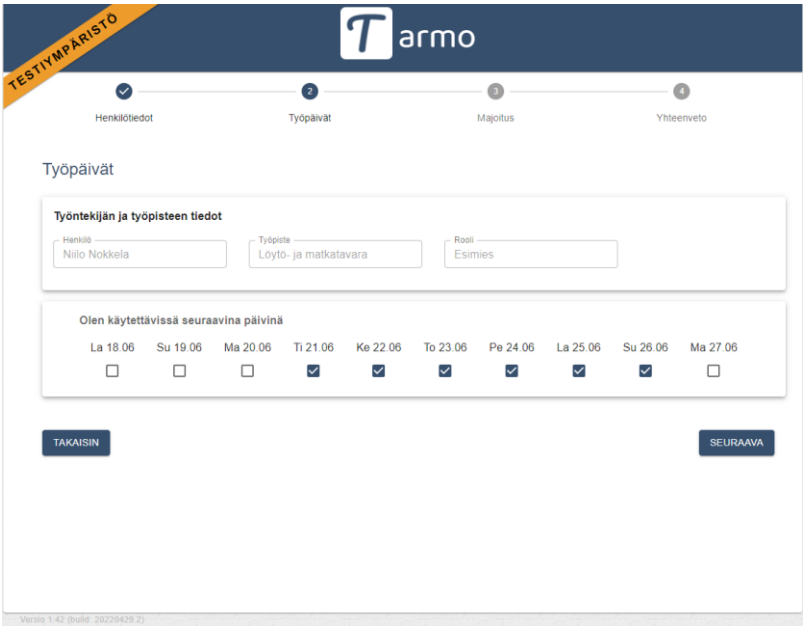
Attending page of supervisor shows the person's name, job location, and role. The user can choose whether he can attend the event. If the supervisor has already registered or abandoned to register, the notification text will be displayed accordingly.

The credentials page is showed to supervisors, volunteers, and new volunteers (a new user). Registration user always set a new secure password, and users with too short username must set a new a valid username. New registration user set both username and password. The validation service validates that the username fulfill requirements and username is not in use.



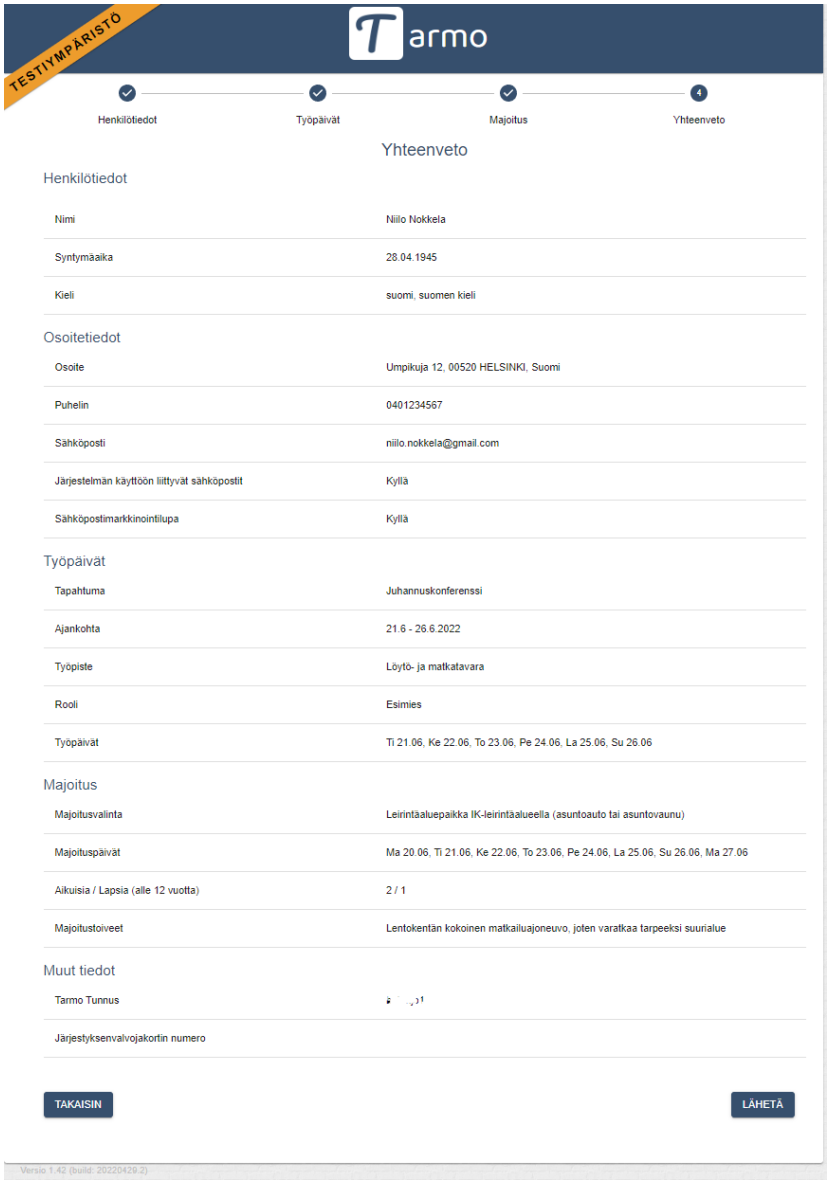
The personal information is showed to supervisors, volunteers, and new volunteers (a new user). The text fields of the form are rendered on the page dynamically of different conditions. Person who belongs security area gets to extra text fields of security officer card (Järjestyksenvalvojan

kortti) and SSN. This information must be delivered to authoritative. Under 18-year-old gets extra text fields of information about their guardians.



The workplace information page is showed in different way depending on is it about supervisor, volunteer with workplace or volunteer without workplace.

The accommodation page shows accommodation choices and collect necessary information. The choices are different for supervisor and volunteer.



The summary page shows an enrollment information before submitting it. Page is rendered dynamically depending fulfilled information. Submit button submits data to registration API.

7 Launch to production

The schedule was busy, and the project started too late. The implementation was divided into two separate phases, with the enrolment of supervisors to be implemented first and later volunteer

enrolment. The reform of Tarmo platform was not only frontend development project because almost everything has changed. All work is done by volunteers, so committing to schedules is difficult due to lack of time.

Pre-phase: Azure environment and Tarmo migration

Construction of a new Azure cloud environment began for Tarmo in February. All components of the lecagy Tarmo system will be transferred to it so that the old local environment can be dismantled unnecessarily. The Azure cloud was configured with network settings, resources for new services, database resources, CI / CD build pipelines for new services, and all this was done during February.

The biggest challenges appeared with the old environment server with very old versions and old libraries. Azure does not deliver such old operating systems and add-ons for virtual machines. Because of this, had to be renovated a lot of code changes to enable functioning and support of newer versions. At the same time, it was revealed that database queries are not optimized, and code had to be optimized for database queries because in the previous implementation, the server and the database were on the same server. In Azure, database services are in the network, and it provides more network latency than old implementation on the same server. The team has a desire to do quality work and that's why these optimizations were made.

Enrolment front-end application implementation started on February. Development progressed according to how the APIs were completed. Extra design meetings must be held a few times and the design plan changed a few times due to the challenges faced.

Phase 1: Enrolment of managers and supervisors

The enrolment application version 1.23 was first released on the production side on March 28th, and the next day was released version 1.24 with improved validation error messages. Invitations were senton 29th of March 2022 to supervisors via email. There is below a small screenshot of the invitation letter. It contained a lot of instructions for supervisors and at the bottom was a button "Tilaa kirjautumislinkki ilmoittautumiseen" from which supervisor subscribe disposable enrolment

link.

After email invitation a few bugs were reported from end users who could not log in or other error occurred. API had issue of aging timeout of action token, and it was quickly repaired when the fault was discovered. Some errors were due to incorrect user data and were fixed on the database side. Enrolment application had few layout and validation issues, but they were not showstopper issues. It was concluded that fixes will be released later branch of volunteer when volunteer enrolment will be opened. Volunteer enrolment will require more API features and legacy Tarmo API will need more features.

Phase 2: Enrolment of volunteers

Application development enrolment of volunteers was performed quickly in about three weeks. It uses same code base as much was reasonable and the necessary volunteer components were implemented. Testing of web application could not be completed on time, because the changes to the API side were not ready to the point where the entire chain could be tested. Testing performed by checking what data is submitted from the application to API. Our test users were populated with job information, and everything was ok. After the tests, a few bugs were fixed, and church information was removed from the form and code. After the tests, a few bugs of UI look

were fixed, and church information was removed from the form and code due GDPR regulations. Version 1.45 was moved to production.

Invitations were sent on 3rd of May 2022 to invite volunteers via email. This deployment has major issue, with production data migration where job's workplace Id missed. This caused error that, web enrolment could not display the workplace of invited user. This was due to a failed data migration and a fix was made to the API to create correct job information. Also, there was another issue with old users, if they were used same email between two users. A new database request unique email addresses and migration could not move user correctly. To this date, no other issues have been identified

8 Conclusion

The purpose of this research was found out best practices of programming and suitable framework to build front-end application with modern technology of new Tarmo front-end application.

Choosing the best technology stack is hard or an almost impossible task. The opinions of application developers vary and are often based on their experiences from work history. Sometimes these opinions have not been updated over the years and, at worst, have been able to turn a blind eye to new technologies. The view is influenced so much by which software languages one has started software development in and how developer gained knowledge about it. The choice of technology is a business decision and cannot be influenced by individual software developer. The developer has got a playground where he is allowed to implement himself but is not allowed to leave the playground. There have traditionally been two playgrounds in computer world. Developers of Microsoft-based services have programmed C# code with .Net dependencies in Microsoft environments and the web services are implemented to IIS web servers. Developers of Java language are using own technologies such as Boot Spring, Apache, Tomcat etc. in their implementations. The modern software technologies have one approach where front-end and backend environments relay same programming language. Software developer can code both back-end and front-end applications in the JavaScript programming language, Back-end servers in Node.js environments, and browsers. On the other hand, modern software development is also like picking cherries where the best implementations for different services are taken. In this our implementation of micro services is implemented in Java's Boot Spring.

8.1 Research question: What is suitable technology stack to produce web application?

I implemented the small prototypes separately with Angular and React in order to get a better picture of the differences between the frameworks. Due to lack of time, I didn't get around to implementing the prototype with the Vue framework. Below are my own observations of prototypes.

Prototype experiences

Angular and React frameworks were used for prototyping and prototype programming was started from the scratch to gain comprehension and experience. Starting the project with frameworks were clear and following installation instructions from framework website were achieved start-up framework projects. A Material Design-based UI library was installed into framework.

Implementation started with React and MUI v.5 versions because these are the most familiar to me. A prototype is a SPA application that retrieves data from the REST interfaces and displays it in tables and uses the forms feature to add and edit data. State status management is implemented with built-in components and no external components such as Redux are deployed.

React v18, MUI v.5.0

React is not a full feature framework because it does not have all the features packaged in the framework. Because of this, there are separate sublibraries for various intended use, the Axios component was used for http / https connections, and Formik forms were used to enter and edit data. Using sublibraries might cause problems with Framework version upgrades. State management can be done with the built-in React Context feature, which scales to a so-called mid-level load, but is not recommended for a site where updating the state information is heavy. For this, there is Redux, a Predictable State Container for JS Apps, and a Redux version of React is available. React Context is used in this prototype. It is complex to define but performs its function.

The MUI v5 was released in September 2021 and reformation were executed to it since the name change to MUI from Material UI. MUI is easy to use for a beginner and https://mui.com site is well informative contains example code snippets of MUI components. MUI Components are comprehensive and include many simplify features for setting CSS styles within components. Component has built-in features for component placement, and all definitions related to CSS styles can be easily defined through sx property.

Experiences

React's learning curve is gentle. It's easy to get involved and it's encouraging to feel that you know something. I had to learn React programming beginning of autumn 2021 for the assignment of the

Framework course. I learnt the basics of React using JavaScript, but it wasn't enough in Tarmo project, it was necessary to use TypeScript and I studied it at the end of 2021. This prototype was also implemented with TypeScript. Understanding of React gained more experience during implementation of the enrolment service. You can find solutions to React problems / challenges from Stack Overflow or elsewhere on the internet. In the beginnig it was more challenging to understand the answers, since most of them are made in vanilla JavaScript and I used TypeScript. When understanding of TypeScript increased, this was no longer a problem at all. Also, I am grateful to the project for forcing me to use TypeScript with React. Using TypeScript avoided a lot of bugs I was doing when I hadn't fully thought through all the options. In summary, React is easy to learn while not as versatile as its competitors. This also causes the need to use third-party library with React. There is a lot of material about React, tutorials, blogs, videos, GitHub projects, Stack Overflow questions where it is easy to learn more. According to my understanding, React is the most used JavaScript framework in Finland.

Angular v13, Angular Material v13

Angular is a full-featured framework and many sources complains that the learning threshold is the highest due to the TypeScript language and full feature framework. I can fully sign this statement because everything is so different in the world of React TypeScript. However, many things are understandable and make life easier to do a long run. State management is simpler thanks to the built-in Services model, and Angular doesn't need Redux to manage status states like React urgently needs. Angular has good assistance features like angular-cli that can be used to automatically create components. Angular's web pages are informative and you can find basic information about Angular there. I studied Angular course Udemy's Angular - The Complete Guide (2022 Edition) by Maximilian Schwarzmüller. It is an extensive package with over 35 hours of video lessons and additional exercises. I went through the features needed to make a prototype. Angular Prototype worked with some of the features and I will update it later.

Experiences

My experience with Angular years ago was from AngularJS and I have almost completely forgotten all of it. I had to start learning curve of Angular from almost zero. Initializing an Angular project

with the angular-cli was easy and with angular-cli creating component is easier and avoiding manual work.

Vue v3, Vuetify 3 Beta

In terms of time management, during this thesis I did not manage to study Vue 3 and implement the prototype with Vue.

Conclusion of technology review

The front-end JavaScript frameworks are suitable for modern web application development. Front-end frameworks have been on the market for so long that the features have developed and the major issues have been solved. Open-source software development differs from development work done with commercial products. Some situations the development open-source community reacts to challenges in hours, but sometimes development falls by the wayside of the power struggle. Open-source libraries and frameworks are dying, and new ones are emerging. It is important to monitor how the product's development and popularity behave. Using the most popular products often guarantees that support is found and that they develop further according to user feedback. I recommend that you should follow the development and make decisions accordingly.

The comparison of front-end framework can be compared with different development curves in timeline, such as Angular has been in star role many years in beginning 2010, but the star has been fading a bit. React has risen from Angular's shadow to become the new champion and Vue is like a rookie team that is playing almost a dream season and is slowly rising to perhaps challenge React. The coming years will show whether Vue will emerge as a real challenger to React at the enterprise level and whether Angular will somehow get a turbo boost. Or will the “black horse” become the new star in scene of front-end JavaScript Framework?

The technology review gives a good understanding of these three frameworks and there is a more detailed look at their differences. In principle, you won't make a big mistake if you choose one of these frameworks. Angular is at its best in large projects at enterprise level where there are a lot of coders and in organizations that already have expertise. React is popular in Finland and a big

part of new commercial projects are implemented with React, because its ecosystem contains various third-party libraries to fulfill need of projects. Vue is a good choice for small projects and those with a need for good performance. In terms of employment, it is worth choosing the one with the most employment opportunities, but on the other hand, experts are sought for all of these, and expert coders will certainly find work.

Choice of the framework

We chose one of the three most popular Framework, Angular, React, and Vue. The choice was difficult because all of these can be implemented in a web application with adequate features. The following arguments weighed on our decision. We had three arguments for the selection:

- Our experience of framework and successfully implemented solutions
- Popularity of framework
- Roadmap of framework
- Expertise of framework available in Finland
- Experiences of protypes

Our choice was the React Framework. It is the world's most popular JavaScript framework in every level and an evolving and maintained framework. React version 18 is released in 29th of March 2022, but the decision is based on version 17. Our team has deep knowledge of the React architecture and plenty of successfully implementations. Other competing frameworks Angular and Vue are also good choices if, for example, the company has the know-how and experience for these frameworks. Vue is especially interesting when there is a need for good performance of application.

8.2 Research question: How create professional code with tools to avoid issues?

This was my first professional software project, and it opened my eyes to see how professionals implements software projects. Our team was small and consisted of three members. I was main responsible of the front-end enrolment application. User interface pre-design protype was done with Figma software and MUI v5 templates. Prototype presents the functionality of user interface and gives an understanding of what we are doing. Development tasks are handled with Azure’s

DevOps tools. We allocated work tasks and bugs via DevOps work items. We did not use sprints when everyone did their best and the work was basically the content of one sprint. For this reason, sprints were not used. GitHub acted as our code repository and the build CI / CD functionality was implemented with Azure DevOps.

It was a surprise to me that professionals do not program with pure vanilla JavaScript instead they use language extension called TypeScript language. It requires more precise typing of the variables and prevents many fault situations when the code knows what is running and handle correctly situations where there are incorrect values or types in the variables. Before starting the project, I had to learn how to program TypeScript with JavaScript to fulfil requirements of the project team.

I had little experience from study projects with, but I hadn't been participated in the development through GitHub. I was insecure about what I was doing and many times I asked for advice for colleagues. I was advised to use Fork software on Windows, which is a graphical interface from GitHub. I started using it and I got along better with it.

8.3 Research question: What kind of experiences and lessons were learned from the project?

The project provided many good lessons for the next steps in the future. Probably most IT projects have similar problems as we had like understanding what we are doing, basic knowledge of the old system is incomplete, terrible hurry, design plan changes when team was performing coding tasks and data integrity in the old system caused interesting bugs. One finding is different that is not usually encountered in straightforward company IT-projects, because we work voluntarily and do not receive any compensation for our work.

The initial knowledge and understanding of the operating logic of the system was incomplete. I knew the main features of the Tarmo software, but I did not know the specific cases and I did not know exactly how the data was stored and used. There was not much documentation of the old Tarmo that I could have accessed and verified from documentation. There was a person involved in the project who had designed most of the system and he advised and steered me in the right direction in enrolment application development. At the beginning of the project, we decided that the data fields in the API interface are in English, although the database structure is still in Finnish.

This made it easier for me because I got to learn a little about the database structure because I translated references to the API code.

Like a normal IT project, we had an honourable time schedule. I started in December 2021 by learning TypeScript programming by enrolling to a TypeScript course. After that, I orientated to Material UI v5 features and update requirements. I did an exercise update for one project to gain an understanding of how to do it and what problems might be encountered. I was waiting simultaneously of start of the project. At the beginning of February, we really started working and realized requirements of schedule from conference committee, the enrolment of supervisors should be in production by the end of February. We were shocked that this schedule is really challenging because only of us I was able to do the full-time project and the others were busy with their full-time days.

We had three tracks of development Legacy Tarmo to Azure Cloud and Legacy Tarmo API, Tarmo-API and Tarmo Enrolment web application. Legacy Tarmo migration to Azure were several difficulties in old versions of libraries and code changes due of deprecated features of MySQL and other components. It was hard to estimate when issues are solved. After this, the Legacy Tarmo API had to be implemented, which acts as a bridge between the new Tarmo API and the old Tarmo. Tarmo API was implemented at the same time and features were added as the Legacy Tarmo API was supporting them. In front-end development, I first upgraded the boiler plate to the latest versions and the changes they required. I did everything slowly by hand without replacing automatically and carefully to learn as much as possible at the same time. My Front-end development was slow, and I had the learning process all the time. I had full days off dedicated for coding and the challenges on the legacy development gave more time for development of front-end. Otherwise, I would not have kept up with the pace of development. This was my first real software project, and everything was new to me, and I needed time to embrace things. I had more difficulty with coding enrolment of supervisors and volunteer part was already going well.

We used Azure’s DevOps tools to manage work items of tasks and bugs. It gives a good view of work tasks and bugs and are not forgotten in a hurry. I did a task or bug if it already had not been done. GitHub's commit should refer to the Task or bug number from DevOps. DevOps tools provide a better overview of a larger project and can be used to report time spent on work tasks.

There are three members in our project, so we were able to communicate directly. Not all of us had internalized the use of DevOps tools then communication was handled through Slack. While the project was running, we held weekly meetings where we went through the development situation and discussed the challenges and made plans. This accelerated the development work briskly. We also held week-long development days at Kaitaniemi campcenter in February and March. During face-to-face weeks, the development work proceeded faster and more efficiently. We have been using the Slack to discuss the project all the time and it has also contributed well to the development work.

Motivation is one of the most important characteristics in projects. Members of the volunteer projects must find motivation from somewhere because here is no financial benefit from the projects. We all have experience of the IsoKirja Juhannus -conference and faith as a unifying factor. We want to help the conference with our own contribution, and we can do it with software engineering skills. The challenge of volunteering is the adequacy of time when you should also get some salary to cover the cost of living. We estimated purchase of Tarmo software project from IT company would cost approximately EUR 300,000. Communities do not understand how big contributions volunteers can make without compensation. Software projects take a lot of time and volunteers do not have too much of it that in their normal daily lives, when they should find time for their loved ones as well. Because of this, a project can take years when you can't focus on doing it full time. This is exactly the situation where we are in in this rebuilding Tarmo project. We managed build phase one registration service and needed APIs. in a year 2023 Tarmo should be renewed, but we do not know if we have enough resources to do it in a year.

Volunteers could be found to build systems where there could be a business opportunity to implement a commercial product on the same model or set up a company to produce this product as a service. In our case, it is hard to find another customer that would need a system tailored to this event. There are also few events of this size in Finland, so commercializing this system is probably difficult. The community should also consider rewarding volunteers with some "rewards" to gain better motivation to make a product and reserve more time for work. Software project needs continues support, because there are many security vulnerabilities in the open-source components, and the administrator must update the software packages regularly and, if necessary, very quickly depending on severity of vulnerability.

8.4 Research question: How to ensure good functionality also on mobile devices?

The world has changed over the years and today most Tarmo enrolments are done with smart devices. That's why, when choosing the UI, we focused on features that seamlessly support everything from a computer's large screen to smart devices of different sizes. MUI 5.0 was chosen as UI library because it has good features for implementing UI that scale on different screen sizes. During the project, the user interface was carefully tested on different sized screens. Some of the tests were done with terminal devices, but also different size of displays tested using the Developer Tools of the Chrome browser.

The form was designed in such a way that the entities were small enough to fit on the screen of the computer screen without the need to scroll. On the web browser page, the page layout contains information in many columns, and on the smart device page layout consists of one column, using the width of the screen and scrolling down. In this way, the UI functionality can be made to work as well as possible on different types of terminal devices without implementing a dedicated mobile application.

There has been a wish from organization to implement the Tarmo mobile application. This would mean that the application should be implemented as separate a web application and the mobile application. We have initially discussed this, that the web application would be implemented with the React framework and the mobile application would be implemented with Flutter to Android and IOS devices. The amount of work in this way would be relatively large, and because of that, you could initially implement a mobile application with a limited scope, e.g. a mobile ID card for a volunteers to gain access food services and prove your identity.

We did not receive feedback on mobile usability from users. A few users did not understand that the authenticated session **doesn’t last forever**. **We haven’t been aware situation like this during design phase of the application. P**robably **next version of the application has counter clock feature at right corner of top screen to help user understand how much time is left to fill the form.**

Based on this research and implementation, It can be stated React framework and UI implemented with MUI 5.x UI library can be used to create a good user experience on different end devices. It's important to remember that nowadays major of users uses smart devices and they are

expecting brilliant user experience. Appearance and usability of the application gives an image of the organization's ability to implement applications.

9 Discussion

I have studied the basics of programming in the past, but in my work, I have only programmed scripting (PowerShell) and a little JavaScript functionality. JAMK's Full Stack programming studies have increased my skills and this project was my first real software project.

The project was a good learning place for me. I got to do a project with real experts. I learned a lot from them and got good instructions and answers to my questions. In project, we coded together for a few weeks from the same of premises, staying in a camp center were reserved for us. These sessions were great for software development, learning and know better each other.

The decision to keep normally the midsummer conference was decided in January-February 2022, and after it our team was informed that the enrolment of managers / supervisors must be opened right away. The management team of the conference had not realized that after two years break of the conference, there was a lot of updating due, because the old system was outdated, and the old system no longer worked in new web browsers. Team committed that the enrolment service of managers would be ready in a month with best effort from team. During the month, the old Tarmo system was supposed to be transferred to the Azure cloud service and at the same time updated to support a newer PHP version and Tarmo Legacy API had to be implemented to pass data between the Legacy Tarmo API and the new Tarmo API. Team had a lot of challenges with all areas amount of work, but team finally got the enrolment service up and running, initially for managers, 3 weeks late. A few weeks later, the service was opened for all volunteers to register to conference.

Myself, I only had experience with the Tarmo system as an end user, registering for the midsummer conference as a supervisor. I did not have a good knowledge in advance of Tarmo's architecture and functionality. Sampo Antila is founder of Tarmo and he clarified the architecture of the system and its functionality for me. During the project he supported me the whole time in coding front end. We had limited information in the planning phase and due of it we had to remake

changes to the plans on the fly in some special cases, such as how to act if the enrolment is left unfinished or authentication token expires or if user want to come again to make sure that he/she has registered or want to change enrolment information. I did not have the right knowledge how to handle hygiene -and order security control card (Järjestyksenvalvojan kortti) information. Requirements changed in the middle of programming process and the logic was also corrected during the coding process before going to live. The authentication logic also had to be changed after planning due of requirements of the new authentication system had different requirements than we had assumed in planning phase. It was resolved in such a way that each user is directed to change the password before filling in more detailed information for enrolment.

I had too optimistic view of project work and assumed too much based on the data from Tarmo legacy API and I left out some checks in first versions of front-end due of hurry timeschedule. This caused some bugs in my code when the Tarmo Legacy API provided special data in the fields, fields were completely missing, or end user filled something wrong to field. I would have saved time if I had carefully implemented all the features in order. One “traditional” bug was that, in accommodation details, the field has reserved 255 characters in the database and front-end code in first versions left unchecked the length of the field. In production, a bug was reported because some user had written about 1000 characters of additional information of accommondation. Some circumstances with Authentication caused situations where user could not authenticate correctly and I had only error message in API response and Authentication had not re-authentication possibility, in the web page code had to only print error on the screen and instruct user to contact administrative people. There were also challenges with authentication Tarmo username with Scandinavian characters and they had to be disabled in usernames. I also learnt that all validations should be work correctly, because end users can fill in anything in the fields or try a shortcut without filling in the data.

The assumption was that the Registration API returns information about successful data storage and front-end prints successful registration on the screen. In a few situations, we noticed that registration information had remained in the internal database of the Registration API and had not been transferred correctly to Tarmo Legacy API. We had some other design issues in this version of API that caused problems in special situations and they will be fixed before registration for next year's conference starts.

In front-end development, I had the biggest challenges in that first version where supervisors could sign up. The production version had a few bugs in validations and data processing. Corrected versions of them were announced. The version that supports the registration of all volunteers was more stable and there was no bug in it that would have required a new version to be made. That's why I'm glad that during this development project I had learned to make better and more fault-tolerant code.

I learned a lot from this project. One of the most important lessons was that you need to know what data is processed, how and how to manage error situations. Good design leads to better code. Now I had to fix and add code in the middle of the project to get the various functionalities to work. The code started to look like spaghetti code and needs optimization for next year. I had to implement some functionalities in the front-end when the API did not support them.

We used to communicate in Slack message channels and kept weekly status meetings in Google Meet. Slack is a convenient product for communication, although sometimes there can be misunderstandings when you do not know how to bring up the issue clearly enough in writing. Finding information from old Slack conversations is sometimes challenging. Azure DevOps Boards were used store work items. Features were made work item tasks and errors were made bug items. The code was stored on GitHub and the CI/CD pipeline was in the Azure cloud. Enrolment front-end code was made with Microsoft's Visual Code in the Windows 10 environment. GitHub's tool was the Fork software.

Rush has been one of the biggest problems in this project as well. While making architectural figures for my thesis, I have noticed that the overall picture has been blurry and it can be seen in the naming of blocks of services in code. In the future, we should go through the architecture one more time and optimize it with current findings. A clear architecture produce simpler code and it is easier for new programmers to get involved in the development work. This architecture should last for decades and support new services and ideas.

There are many ideas for improvement the enrolment section in front-end. There were a few CSS bugs in the code where the lines are not perfectly aligned and the layout could be improved and made to look better. In addition to successful enrolment, you could send a confirmation email

with information about the registration. A few data should be added to the registration API interface so that the front-end can dispense with a few hard-coded values and validation. React is the right choice of framework for Tarmo Backoffice use, because use of it will be browser-based for a long time. The mobile application could initially replace the work card of volunteer people but moving other Tarmo features to mobile could be too tedious.

9.1 Research ethics and quality

My responsibility was to research JavaScript frameworks of front-end, update React boiler plate code to the latest versions, plan the registration service with co-operation, plan the front-end, program, support in error correction and document this part for the thesis. My tutor was Sampo Antila and he supported me in those situations where I needed help on how to best implement features. He has more than 20 years of experience in software development, so I got a lot of good tips and guidance from him. He was overall responsible for this project and was responsible of the architecture, the operation of the Legacy API and the old Tarmo. Riku Kuusisto implemented the new Tarmo API and he was responsible for it and its interworking with the Legacy API.

Thesis has followed the instructions given in the JAMK reporting instructions and the research methods are generally known and used methods. Thesis has been tested for plagiarism and the warnings have been checked.

Needed materials of technology review were collected as possible were collected from scientific sources and product documentation, additional materials were used where the results of were in the same direction as scientific sources. This ensures quality of the research.

Thesis has been executed with the client's requirements and the produced documentation and software code have been stored in the locations specified by them. The software code uses open-source components and good programming practices. Security related materials are not described in this thesis.

The technology review is compiled from many different sources, and they are marked according to the APA standard. The technology overview covers front-end web development from history to the present day.

The conclusions of thesis have been compiled based on the technology review and experiences.

Thesis work has been done as a voluntary work and the author of it has not received any financial benefit from it. Author of the thesis does not own the rights to the software code he produces.

References

10 Famous React Websites. (n.d.). AnyforSoft. Retrieved 1 September 2022, from https://anyforsoft.com/blog/10-famous-websites-built-react-js/

AaronMaxwell. (n.d.). *Application Insights overview—Azure Monitor*. Retrieved 6 November 2022, from https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview

About issues. (2021). GitHub Docs. https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues

Academy, C. (2016, June 15). Top 32 Sites Built With ReactJS. *Medium*. https://medium.com/@co-deracademy/32-sites-built-with-reactjs-172e3a4bed81

Activity. (2021). GitHub Docs. https://docs.github.com/en/rest/reference/activity

@angular/core vs react vs svelte vs vue | npm trends. (2021). https://www.npmtrends.com/@angular/core-vs-react-vs-vue-vs-svelte

Angular—Introduction to Angular concepts. (n.d.). Retrieved 23 May 2022, from https://angular.io/guide/architecture

Angular—Introduction to components and templates. (n.d.). Retrieved 23 May 2022, from https://angular.io/guide/architecture-components#templates-and-views

Angular—Introduction to modules. (n.d.). Retrieved 23 May 2022, from https://angular.io/guide/architecture-modules

Angular—Introduction to services and dependency injection. (n.d.). Retrieved 23 May 2022, from https://angular.io/guide/architecture-services

AngularJS. (2022). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=AngularJS&oldid=1088337734

[Announcement] Apache HTTP Server 2.0.65 Released-Apache Mail Archives. (n.d.). Retrieved 22 April 2022, from https://lists.apache.org/thread/98t1yjk1c3ngzh8wdx55d1ct9vd1koov

Ant Design. (2022). [TypeScript]. Ant Design Team. https://github.com/ant-design/ant-design (Original work published 2015)

Antd. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/antd

Architecture. (n.d.). Onsen UI. Retrieved 2 May 2022, from https://onsen.io/v2/guide/architecture.html

Between the Wires interview with Evan You. | Between the Wires. (2017, June 3). https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11/03/evan-you/

Black, N. (2020). Boris Cherny on TypeScript. *IEEE Software*, *37*(2), 98–100. https://doi.org/10.1109/MS.2019.2958155

Borissova, D., Dimitrova, Z., Dimitrov, V., Yoshinov, R., Garvanova, M., & Garvanov, I. (2021). Multi-Attribute Decision-Making Model for Ranking of Web Development Frameworks. *2021 25th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, 3–8. https://doi.org/10.1109/CSCC53858.2021.00009

Brandon Hill published. (2021, November 16). *Microsoft Confirms Anticompetitive Edge Browser Behavior in Windows 11*. Tom's Hardware. https://www.tomshardware.com/news/microsoft-confirms-windows-11-edge-default-browser

Browser Market Share Worldwide. (2021). StatCounter Global Stats. https://gs.statcounter.com/browser-market-share/

Bulma. (n.d.-a). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/bulma

Bulma: Free, open source, and modern CSS framework based on Flexbox. (n.d.-b). Retrieved 2 May 2022, from https://bulma.io

Chakra UI - A simple, modular and accessible component library that gives you the building blocks you need to build your React applications. (n.d.). Chakra UI: Simple, Modular and Accessible UI Components for Your React Applications. Retrieved 29 April 2022, from https://chakra-ui.com

chakra-ui/chakra-ui: ⚡ Simple, Modular & Accessible UI Components for your React Applications. (n.d.). Retrieved 22 October 2022, from https://github.com/chakra-ui/chakra-ui

@chakra-ui/react. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/@chakra-ui/react

@chakra-ui/react vs @material-ui/core vs antd vs evergreen-ui vs grommet vs onsenui vs react-bootstrap vs react-suite vs reactstrap vs semantic-ui | npm trends. (n.d.). Retrieved 29 April 2022, from https://www.npmtrends.com/@material-ui/core-vs-antd-vs-evergreen-ui-vs-grommet-vs-onsenui-vs-react-bootstrap-vs-react-suite-vs-reactstrap-vs-semantic-ui-vs-@chakra-ui/react

Chapter 1. What is a single-page application? · SPA Design and Architecture: Understanding single-page web applications. (n.d.). Retrieved 12 September 2022, from https://livebook.manning.com/book/spa-design-and-architecture/chapter-1/

Chen, C.-H. (2021). Person Re-Identification Microservice over Artificial Intelligence Internet of Things Edge Computing Gateway. *Electronics (Basel)*, *10*(18), 2264-.

Components and Props – React. (n.d.). Retrieved 1 September 2022, from https://reactjs.org/docs/components-and-props.html

Create React App. (n.d.). Retrieved 1 September 2022, from https://create-react-app.dev/

cssnext—Use tomorrow's CSS syntax, today. (n.d.). Retrieved 2 May 2022, from https://cssnext.github.io/

Datashield. (n.d.). *Microsoft Azure Security Center | Core Features, Pricing*. Retrieved 5 November 2022, from https://www.datashieldprotect.com/blog/microsoft-azure-security-center-core-features-pricing

Desktop Browser Market Share Finland. (2021). StatCounter Global Stats. https://gs.statcounter.com/browser-market-share/desktop/finland/

Eschweiler, S. (2022, March 24). How To Use Tailwind CSS With React. *CodingTheSmartWay*. https://medium.com/codingthesmartway-com-blog/how-to-use-tailwind-css-with-react-9dd78bbdc0e0

Evergreen-ui. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/evergreen-ui

First Week of Launching Vue.js. (n.d.). Evan You. Retrieved 13 April 2022, from http://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/index.html

Flanagan, D. (2006). *JavaScript: The Definitive Guide: The Definitive Guide*. O'Reilly Media, Inc.

Fluent UI Web. (2022). [TypeScript]. Microsoft. https://github.com/microsoft/fluentui (Original work published 2016)

@fluentui/react-hooks. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/@fluentui/react-hooks

Gao, Z., Bird, C., & Barr, E. T. (2017). To Type or Not to Type: Quantifying Detectable Bugs in JavaScript. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 758–769. https://doi.org/10.1109/ICSE.2017.75

Getting Started. (n.d.). Onsen UI. Retrieved 2 May 2022, from https://onsen.io/v2/guide/

GitHub—Angular/angular: The modern web developer's platform. (n.d.). Retrieved 19 October 2022, from https://github.com/angular/angular

GitHub—Facebook/react: A declarative, efficient, and flexible JavaScript library for building user interfaces. (n.d.). Retrieved 19 October 2022, from https://github.com/facebook/react

GitHub—Vuejs/core: Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web. (n.d.). Retrieved 19 October 2022, from https://github.com/vuejs/core

Gołąb, A. (2021). *React Developer Roadmap* [JavaScript]. https://github.com/adam-golab/react-developer-roadmap (Original work published 2018)

Gołąb, A. (2022). *React Developer Roadmap* [JavaScript]. https://github.com/adam-golab/react-developer-roadmap (Original work published 2018)

Grommet. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/grommet

Grommet: Focus on the essential experience. (2022). [JavaScript]. grommet. https://github.com/grommet/grommet (Original work published 2015)

Gudelli, A. (2022, May 18). *Angular 14 version release*. Angular Wiki. https://www.angularjs-wiki.com/angular/angular-14-release/

Handling Events – React. (n.d.). Retrieved 1 September 2022, from https://reactjs.org/docs/handling-events.html

Hartman, J. (2020, January 25). *AngularJS vs Angular 2 vs Angular 4: What's the Difference?* https://www.guru99.com/angularjs-1-vs-2-vs-4-vs-5-difference.html

IdP (Identity Provider)—Glossary—Hermes. (n.d.). Retrieved 5 November 2022, from http://kb.mit.edu/confluence/display/glossary/IdP+(Identity+Provider)

Interview with Evan You—Insights About Vue 3 and Developer Experience. (n.d.). Retrieved 6 September 2022, from https://www.monterail.com/blog/interview-evan-you-vue3

Introducing Hooks – React. (n.d.). Retrieved 2 September 2022, from https://reactjs.org/docs/hooks-intro.html

Introducing JSX – React. (n.d.). Retrieved 1 September 2022, from https://reactjs.org/docs/introducing-jsx.html

Introducing MUI Core v5.0—MUI. (n.d.). Retrieved 26 April 2022, from https://mui.com/blog/mui-core-v5/

Iso Kirja. (2021). In *Wikipedia*. https://fi.wikipedia.org/w/index.php?title=Iso_Kirja&oldid=19880504

Issues · Semantic-Org/Semantic-UI-React. (n.d.). GitHub. Retrieved 22 October 2022, from https://github.com/Semantic-Org/Semantic-UI-React

JavaScript | MDN. (n.d.). Retrieved 10 November 2021, from https://developer.mozilla.org/en-US/docs/Web/JavaScript

John. (2022). *React Bulma Components* [JavaScript]. https://github.com/couds/react-bulma-components (Original work published 2017)

Juhannuskonferenssi. (n.d.). Juhannuskonferenssi. Retrieved 7 December 2021, from https://www.juhannuskonferenssi.fi/mediatiedote

Karppanen, T. (2022). *Single-page application -arkkitehtuuri* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. http://www.theseus.fi/handle/10024/754107

Kemppainen, H. (2022). *Modernit verkkosovelluskehykset ja kirjastot* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. http://www.theseus.fi/handle/10024/780084

Learn TypeScript (Ditch JavaScript). (n.d.). Udemy. Retrieved 27 April 2022, from https://www.udemy.com/course/understanding-typescript/

Lundelin, L. (n.d.). *Helluntaituulia Varsinais-Suomessa*.

Masterworkshop SPA Design and Architecture: Understanding Single-Page Web Applications. (n.d.). Retrieved 19 October 2022, from https://masterworkshop.skillport.com/skillportfe/main.action#summary/BOOKS/RW$8267:_ss_book:147239

Microsoft Design. (n.d.). Microsoft Design. Retrieved 2 May 2022, from https://www.microsoft.com/design/fluent/

Miettinen, T. (2021). *JavaScript ja siitä syntyneet alustat* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. http://www.theseus.fi/handle/10024/505831

Migration from v4 to v5—Material UI. (n.d.). Retrieved 26 April 2022, from https://mui.com/material-ui/guides/migration-v4/

Mikkonen, J. (n.d.). *Statically typed programming languages in the JavaScript ecosystem: A type system perspective*. 78.

MUI Core. (2022). [JavaScript]. MUI. https://github.com/mui/material-ui (Original work published 2014)

@mui/material. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/@mui/material

Nasserzadeh, A. (2020, February 11). *10 Reasons to Use TypeScript Over Vanilla JavaScript*. Medium. https://betterprogramming.pub/10-reasons-to-use-typescript-over-vanilla-javascript-a49256e527d3

'Official Release: Apache 2.0.35 is now GA'—MARC. (n.d.). Retrieved 22 April 2022, from https://marc.info/?l=apache-httpd-announce&m=101810732100356&w=2

Ojasalo, K., Moilanen, T., & Ritalahti, J. (n.d.). *Kehittämistyön menetelmät—Uudenlaista osaamista liiketoimintaan* (3.-4.painos,2015). Sanoma Pro Oy. Retrieved 8 September 2022, from https://www.ellibslibrary.com/book/978-952-63-2695-5/kehittamistyon-menetelmat

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022a). Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*, *21*(3), 789–813. https://doi.org/10.13052/jwe1540-9589.21311

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022b). Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*, *21*(3), 794–795. https://doi.org/10.13052/jwe1540-9589.21311

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022c). Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*, *21*(3), 795–798. https://doi.org/10.13052/jwe1540-9589.21311

Onsen UI - Cross-Platform Hybrid App and PWA Framework. (2022a). [JavaScript]. OnsenUI. https://github.com/OnsenUI/OnsenUI (Original work published 2013)

Onsen UI - Cross-Platform Hybrid App and PWA Framework. (2022b). [JavaScript]. OnsenUI. https://github.com/OnsenUI/OnsenUI (Original work published 2013)

Onsenui. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/onsenui

Parsons, J. (2021, December 3). Microsoft warns billions of Google Chrome users to stop using it now. *Metro*. https://metro.co.uk/2021/12/03/microsoft-warns-billions-of-google-chrome-users-to-stop-using-it-now-15708060/

PHP: PHP 4.4.0 Release Announcement. (n.d.). Retrieved 22 April 2022, from https://www.php.net/releases/4_4_0.php

Pollock, J. (2019). *JavaScript*.

Prasad, A. (2022). *RESEARCH AND ANALYSIS OF THE FRONTEND DEVELOPMENT FRAMEWORKS AND LIBRARIES WITH VOICE RECOGNITION IN REMOTE AREAS FOR DEVELOPING ECOMMERCE BUSINESS FOR PEOPLE OF REMOTE AREAS WHERE PEOPLE HAVE LESS OR NO KNOWLEDGE OF TECHNICAL DEVICES*. *6*(11), 6.

Press Release´: NETSCAPE AND SUN ANNOUNCE JAVASCRIPT, THE OPEN, CROSS-PLATFORM OBJECT SCRIPTING LANGUAGE FOR ENTERPRISE NETWORKS AND THE INTERNET. (1995, December 4). https://web.archive.org/web/20070916144913/http://wp.netscape.com/news-ref/pr/newsrelease67.html

React Router | Upgrading from v5. (n.d.). Retrieved 26 April 2022, from https://reactrouter.com/docs/en/v6/upgrading/v5

React Vs. Angular Vs. Vue. (2019, December 27). Simplilearn.Com. https://www.simplilearn.com/react-vs-angular-vs-vue-article

React-bootstrap. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/react-bootstrap

react-bootstrap/react-bootstrap: Bootstrap components built with React. (n.d.). Retrieved 22 October 2022, from https://github.com/react-bootstrap/react-bootstrap

React-bulma-components. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/react-bulma-components

Reactstrap. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/reactstrap

Reactstrap. (2022). [JavaScript]. reactstrap. https://github.com/reactstrap/reactstrap (Original work published 2016)

Reconciliation – React. (n.d.). Retrieved 2 September 2022, from https://reactjs.org/docs/reconciliation.html

Releases | Vue.js. (n.d.). Retrieved 13 April 2022, from https://vuejs.org/about/releases.html

Rsuite. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/rsuite

Rsuite/rsuite. (2022). [TypeScript]. React Suite. https://github.com/rsuite/rsuite (Original work published 2016)

Sacha Greif. (n.d.). Retrieved 24 May 2022, from https://sachagreif.com/

Sawers, P. (2018, June 4). Microsoft confirms it will acquire GitHub for $7.5 billion. *VentureBeat*. https://venturebeat.com/business/microsoft-confirms-it-will-acquire-github-for-7-5-billion/

Segmentio/evergreen. (2022). [JavaScript]. Segment. https://github.com/segmentio/evergreen (Original work published 2017)

Semantic-ui-react. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/semantic-ui-react

Stack Overflow Trends. (n.d.). Retrieved 1 December 2021, from https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular%2Csvelte%2Cangularjs

Tailwindcss. (n.d.). Npm. Retrieved 22 October 2022, from https://www.npmjs.com/package/tailwindcss

Tailwindlabs/tailwindcss. (2022). [JavaScript]. Tailwind Labs. https://github.com/tailwindlabs/tailwindcss (Original work published 2017)

Talsania, K. (2018, July 30). Top 10 ES6 Features Every Javascript Developer Must Know. *Medium*. https://medium.com/@kavisha.talsania/top-10-es6-features-every-javascript-developer-must-know-4c81ec54bbcd

tamram. (n.d.). *Introduction to Blob (object) storage—Azure Storage*. Retrieved 5 November 2022, from https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

Tanguy Krotoff. (2021). *Front-end frameworks popularity (React, Vue, Angular and Svelte)*. Gist. https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

Testing Overview – React. (n.d.). Retrieved 1 September 2022, from https://reactjs.org/docs/testing.html

The Chromium Projects. (2021). https://www.chromium.org/

The History of React.js on a Timeline. (2018, April 4). RisingStack Engineering. https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

The State of JS 2021: Front-end Frameworks. (n.d.). Retrieved 24 May 2022, from https://2021.stateofjs.com/en-US/libraries/front-end-frameworks/

Thomas, J. (2022). *Bulma* [CSS]. https://github.com/jgthms/bulma (Original work published 2016)

T.J. Crowder. (n.d.). *JavaScript: The New Toys*. Wrox Press © 2020. https://masterworkshop.skillport.com/skillportfe/main.action#summary/BOOKS/RW$17709:_ss_book:120314

TLS 1.0 and TLS 1.1—Chrome Platform Status. (n.d.). Retrieved 21 April 2022, from https://chromestatus.com/feature/5759116003770368

TypeScript vs Dart | Top 7 Most Useful Differences You Need to Know. (2018, August 5). *EDUCBA*. https://www.educba.com/typescript-vs-dart/

Usage Statistics and Market Share of Apache, April 2022. (n.d.). Retrieved 22 April 2022, from https://w3techs.com/technologies/details/ws-apache

Virtual DOM and Internals – React. (n.d.). Retrieved 2 September 2022, from https://reactjs.org/docs/faq-internals.html

Vue 3 – A roundup of infos about the new version of Vue.js—Made with Vue.js. (n.d.). Retrieved 13 April 2022, from https://madewithvuejs.com/blog/vue-3-roundup

Vue 3 as the New Default | The Vue Point. (n.d.). Retrieved 6 September 2022, from https://blog.vuejs.org/posts/vue-3-as-the-new-default.html

Vyas, R. (2022a). Comparative Analysis on Front-End Frameworks for Web Applications. *International Journal for Research in Applied Science and Engineering Technology*, *10*(7), 298–307. https://doi.org/10.22214/ijraset.2022.45260

Vyas, R. (2022b). Comparative Analysis on Front-End Frameworks for Web Applications. *International Journal for Research in Applied Science and Engineering Technology*, *10*(7), 298. https://doi.org/10.22214/ijraset.2022.45260

Web APIs | MDN. (n.d.). Retrieved 10 November 2021, from https://developer.mozilla.org/en-US/docs/Web/API

What is LAMP (Linux, Apache, MySQL, PHP)? - Definition from WhatIs.com. (n.d.). WhatIs.Com. Retrieved 22 April 2022, from https://www.techtarget.com/whatis/definition/LAMP-Linux-Apache-MySQL-PHP

What is OAuth 2.0 and what does it do for you? (n.d.). Auth0. Retrieved 5 November 2022, from https://auth0.com/intro-to-iam/what-is-oauth-2/

What is Web Development? - Definition from Techopedia. (n.d.). Techopedia.Com. Retrieved 10 November 2021, from http://www.techopedia.com/definition/23889/web-development

Why You Should Use TypeScript. (n.d.). Serokell Software Development Company. Retrieved 13 May 2022, from https://serokell.io/blog/why-typescript

Zaninotto, F. (n.d.). *Why You Should Upgrade To Material-UI V5*. Retrieved 26 April 2022, from https://marmelab.com//blog/2022/02/27/mui-v5-rocks.html