UNIVERSITY of SALZBURG

PORSCHE INFORMATIK

SabisApp:

# The Comparison of the Popular Web Frameworks for UX Design and UI Implementation (Angular vs. Vue.js)

## BACHELOR THESIS

submitted in partial fulfillment of the requirements for the degree of Bachelor of Science (B.Sc.) in Computer Science

by Ganna Shulika
Registration Number: 01123666

to the Department of Computer Science
at the Faculty of Natural Sciences
at the Paris Lodron University of Salzburg

Supervisors: Univ.-Prof. Dr. Wolfgang Pree, Robert Behmüller

Salzburg, April 23, 2020

Porsche Informatik GmbH. Louise-Piëch-Straße 9, 5020 Salzburg

•

Department of Computer Science,Faculty of Natural Sciences Paris Lodron University of Salzburg Jakob-Haringer-Straße 2 , A-5020 Salzburg , +43 (0) 662 8044-6300
www.cosy.sbg.ac.at or informatik.uni-salzburg.at

# Statement of Authentication

I hereby declare that I have written the present thesis independently, without assistance from external parties and without use of other resources than those indicated. The ideas taken directly or indirectly from external sources (including electronic sources) are duly acknowledged in the text.

Salzburg, April 23, 2020

Ganna Shulika

# Abstract

Nowadays, with constantly growing complex web technologies a huge number of JavaScript Frameworks emerged to simplify the development process. But it is still not always clear what framework to choose.
SabisApp was a new project at Porsche Informatik GmbH. started in 2019, but lacked detailed specification and user interface design. It was also not clear what framework to use for the front-end part of the application. The discussions, web research and the specialists available in the company lead to the problem of choosing between Angular and Vue.js frameworks.
This document describes the new re-/design and specification process of the new web application's GUI (surveys, observations, user tests, evaluation, mockups design) and the comparison criteria overview of two frameworks. For our comparison we used different comparison criteria found during the literature research, talking to the experienced developers, web application architects of Porsche Informatik GmbH and searching throngs the experienced developer's blogs on the Internet. Finally after analysing what framework theoretically fitts our purposes we implemented two projects with Angular and Vue.js frameworks, compared the results and made the final decision. Angular framework was chosen for SabisApp to be more suitable for the implementation due to is security, modularity, maintanability, compatibility, popularity, large community, simplicity to intergrade Material design library, well structured, predefined by default source code hierarchy and TypeScript to support type safe anf object oriented programming.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

In the early days of JavaScript, it was out of the question to develop complete applications with it. Its function was primarily to check, for example, form entries in the browser. However, with the increasing popularity of the language and the triumph of open source, JavaScript is the medium of choice for the development of web applications. Over the last 10 years web pages have become more dynamic and powerful thanks to JavaScript.

A web application is a technically complex and voluminous software product that implements large-scale business logic and complex interactions with the end user. In this connection, the question arises of maximally simplifying its implementation and subsequent support. To simplify and unify the code, web application developers typically use assistive tools such as frameworks.

A lot of code has been moved from the server side to the client side in the browsers. This made a problem of having thousands of code lines of JavaScript connecting to the various HTML and CSS files with no formal organisation.This is why JavaScript Frameworks like Angular.js, Angular, Vue.js, React and others were developed.

Frameworks are software products that simplify the creation and maintenance of technically complex or loaded projects. As a rule, a framework contains only basic software modules, and all project-specific components are implemented by a developer on their basis. This ensures not only high development speed, but also greater productivity and reliability of solutions.

## 1.2 The Research Project

When starting a new web application project, a developer is faced with a choice: which JavaScript framework to choose? Currently there are dozens of JavaScript frameworks.

There are differences between them, and sometimes quite substantial. How is the question answered today with which technology a web application should be implemented? The development of web applications in the form of multi page applications (MPAs) using the description languages HTML, CSS and JavaScript is increasingly being replaced by the increasing number of single page application (SPA) frameworks. SPAs are applications which have only one page to be loaded into the browser, but different components to be reloaded when needed but it gives the experience of multiple pages which load quicker as by MPA. This trend has been going on for 10 years now, which has resulted in a multitude of different frameworks with different approaches. As different as the approaches between the frameworks are, the common goal is the same: high-performance web applications with a good user experience.

As complexity increases, SPAs inevitably need to support some form of modularization. The complete application can be implemented with the help of individual components, which increases functionality and reusability.

These individual components enable dynamic reloading of the content during the run time. They can either update and accept new values or can be completely replaced. These update options do not require a new initialization of the page and therefore offer an improved user experience.

Furthermore, the application can be fully interacted with and actions can be triggered. The actions are processed in real time like in a desktop application and the interface reacts dynamically to the event. Some advantages over MPAs are already clear from the description.

The criteria for choosing the right framework should take into account both the specifics of the project and the requirements dictated a the company, e.g.:

- the need for long-term support for the project;
- adding new functionality;
- search for personnel to work with the project;
- the possibility of using it for other projects of the company
- openness of the framework to guarantee its security for end user
- and others.

## 1.3 Goals

The main goal of this thesis is to compare two of the most used front-end web frameworks (Vue.js and Angular) to make a correct choice for the future development of a web application SabisApp (client side) according to the kind of the application (should

be SPA), features and GUI components (the features and components are still to be specified, designed and discussed. Our design is to be done with the help of user experience engineering methods: user needs, groups analysis, ideation techniques, prototyping and design evaluation. After the final design is done there will be created two projects with the identical features and GUI components using Vue.js and Angular frameworks. The frameworks and the projects will be compared according the comparison categories found in the literature. After the theoretical and practical analysis and comparison is done we will make the final decision which front-end web framework to use for SabisApp. The objectives of this research are:
1. To analyse the application's requirements, user groups, user needs,
2. to design, test, analyse and improve GUI mock-ups for the app development,
3. to define the features and the architecture for the future application,
4. to research and make an overview of the comparison criteria of web frameworks Angular and Vue.js, to make a choice of the framework according to the relevant comparison criteria,
5 to implement two projects with vue.js and Angular and to compare the results;
7. to make a final choice according the literature comparative analysis and the implementation results.

## 1.4 Scope and Limitations

As the project was closed due to the lack of experienced specialists in the team and the leaving one of the key developers of the Sabis Systems. The research was limited only for developing several micro services GUI pages prototypes of the application.

## 1.5 Outline

This document is structured as follows:

- Chapter 2:
  The first chapter provides a background information on the web application to be developed. It describes the usage, kind of application, the features, the UX/UI (re)/design of SabisApp using contextual enquiry, online questionnaires, mock-ups and prototypes (Pencil, Justinmind Prototyping Tool) and the application's architecture.

- Chapter 3:
  This chapter gives a short overview of existing front-end web frameworks (Literature research), the criteria for comparing the frameworks and the survey for making a framework choice according to the relevant aspects described.

- Chapter 4:
  In the fourth chapter the implementation process of two projects with the identical

GUI components using two web frameworks (Vue.js and Angular) will be presented. It describes the results of the comparison of the implemented projects (code, architecture and look&feel).

- Chapter 5:
  Chapter 5 presents the results of the research.

# Chapter 2

# SabisApp

This chapter describes a planing of the features, GUI design and technical requirements for a future web application project SabisApp. We will make a short overview of the GUI design process steps and the research methods we used for the SabisApp GUI design. The results and discussions will be presented in the final project plan.

## 2.1 Background

SabisApp is a mobile web application to be developed. It contains of a small part of SabisWeb and SABIS systems. SabisWeb is an internal web application of Porsche Informatik GmbH. for managing the company's employees' training and education. This system sends and receives data from/to SABIS, a desktop application written in Delphi, which basically visualises and manages its database data for every course, trainer, trainee and so on. SabisApp will be used by trainers, trainees and the training responsible persons to have an overview of the courses, timetable for all the booked events, to be able to chat with the trainers and the course group, to do small quizzes for the courses and see their scores, to give feedback and evaluate courses. The trainers will be able to save the attendance lists.

There are seven basic micro-services to be developed: "Chat", "Events Calendar", "Feedbacks", "Score", "Courses Planer", "Micro Learnings" and "Participants List". The first GUI design ideas and the architecture were presented in the document "SabisAppVorschlag.pdf" [1]. The first four implemented micro-services were "Score", "Courses Planer", "Micro Learnings" and "Participants List". Unfortunately, it was impossible to test the ready functionality with the users because of the not executable code and only several available images that represent the design of the future app. Our team which was assigned to this project, except one person, was not familiar with the concepts and processes of the Sabis- systems and we lacked information to start the design and imple-

[1] https://github.com/shulikaga/Bachelor_project_ux_part/blob/master/SabisAppVorschlag.pdf

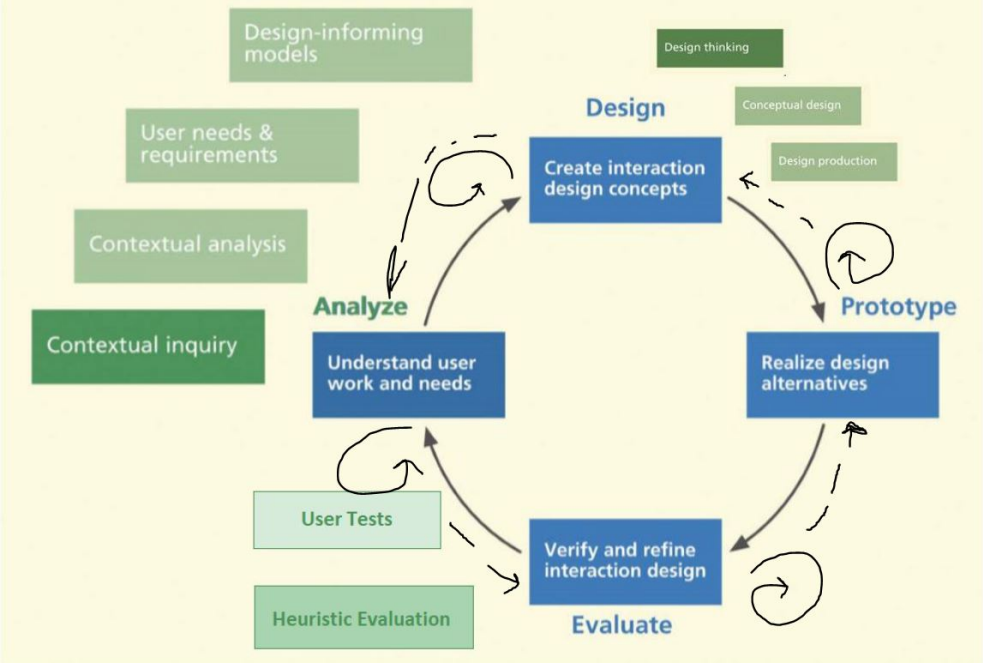


Figure 2.1: UX Design Process

mentation of the GUI. That is why, we put the images from the "SabisAppVorschlag.pdf" document into the first mockup [2] to start the UX (re-)design process.

## 2.2 UX Design Process

Rex Hartson and Pardha S. Pyla in the text book "The UX Book" describe usability experience and user interface design process as a lifecycle of four basic steps: analysis, design, prototyping and evaluation. The whole life cycle and every step can be iterated if needed [8]. We summed up all the methods for each step into one Figure to have a compact overview (See Figure 2.1).

Analysis step is done to understand user work, expectations, problems, frustrations and needs while using the user interface. This step contains of such research methods as: contextual enquiry [3], contextual analysis, user needs and requirements analysis, the development of the design-informing models [4] like personas [5] and flow models (work flow).

[2] `https://shulikaga.github.io/Bachelor_project_mockups_part/`

[3] **Contextual enquiry** is a research method for a user-centered design in which a researcher observes and interviews a user while interacting with a user interface.

[4] **Design-informing models** are design-oriented constructs, they do not appear directly in the design, but are the descriptions of user work, needs, the raw data that are turned into actionable items to inspire the future design.

[5] As archetypal users, **personas** represent the goals and needs of the target group and make it possible to make informed decisions in the development of user-friendly products right from the start.

The design step creates interaction design concepts. It involves design thinking techniques such as brainstorming, imagining, and reflecting on possible outcomes, conceptual design and design production.

The results of the prototyping step are low-fidelity mockups and high-fidelity prototypes which will be evaluated, analysed, and redesigned till the final design fulfill the user's needs.

During the evaluation step the interaction design will be verified and refined. The methods of the design evaluation are: user tests, like online questionnaires and heuristic evaluations [6][8].

## 2.3 SabisApp GUI Re-/Design Process

We began our design process with the evaluation step of the already implemented SabisApp GUI parts and quickly built a small mockup (See Figure 2.2 and the footnote link) [7] with already implemented features using "pencil" - a simple desktop application for building low-fidelity mockups [8].

We also did a small heuristic evaluation of the mockup according to the list of the 10 guidelines by Nielsen and Mollich (See Figure 2.3).
After checking the mockup against the guidelines, we found out that the design lacks good graphic design, screen layout and use of colors, consistency, feedback how a user progress, good error messages and introductory information for the learning and quiz parts.

With the help of the online questionnaires [9] (See Figure 2.5) and user interaction observations with talking out loud while using the mockup (e.g. to find their current score or to read the theory in micro-learning and do the quiz and so on.), we gathered a lot of information how the users interacted with the mockup, the problems and expectations they had, what they liked and did not like.
The online questionnaire design was made of a mixture of several famous usability experience questionnaires:
The Questionnaire for User Interface Satisfaction (QUIS) [10], System Usability Scale

[6] **Heuristic evaluation** is a method of formally assessing the usability of a user interface. In the described method by Jakob Nielsen et al., a small group of usability experts ("evaluators"; n = 5) tries to find as many potential usability problems as possible that later real users could have, using a list of heuristics e.g. a list of 10 guidelines for the user interface design by Nielsen and Mollich.

[7] `https://shulikaga.github.io/Bachelor_project_mockups_part/`

[8] `https://pencil.evolus.vn/`

[9] `https://github.com/shulikaga/Bachelor_project_ux_part/blob/master/survey.pdf`

[10] `https://garyperlman.com/quest/quest.cgi?form=QUIS`

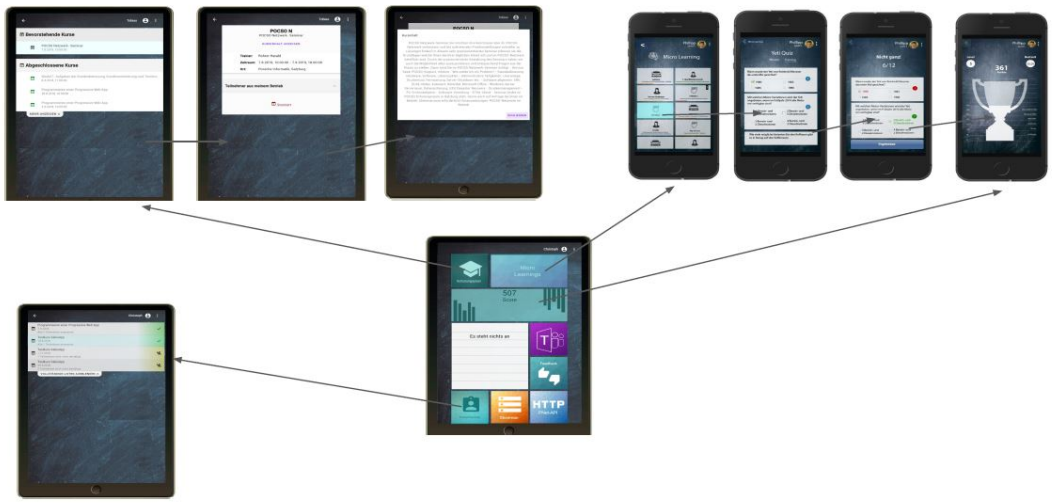

Figure 2.2: First Mock-up

**The 10 Original Nielsen and Molich Usablity Inspection Heuristics**

- Simple and natural dialogue
  - Good graphic design and use of color
  - Screen layout by gestalt rules of human perception
  - Less is more; avoid extraneous information
- Speak the users' language
  - User-centered terminology, not system or technology centered
  - Use words with standard meanings
  - Vocabulary and meaning from work domain
  - Use mappings and metaphors to support learning
- Minimize user memory load
  - Clear labeling
- Consistency
  - Help avoid errors, especially by novices
- Feedback
  - Make it clear when an error has occurred
  - Show user progress
- Clearly marked exits
  - Provide escape from all dialogue boxes
- Shortcuts
  - Help expert users without penalizing novices
- Good error messages
  - Clear language, not obscure codes
  - Be precise rather than vague or general
  - Be constructive to help solve problem
  - Be polite and not intimidating
- Prevent errors
  - Many potential error situations can be avoided in design
  - Select from lists, where possible, instead of requiring user to type in
  - Avoid modes
- Help and documentation
  - When users want to read the manual, they are usually desperate
  - Be specific with online help

Figure 2.3: Original Nielsen and Mollich Heuristics

(SUS) [11], Computer System Usability Questionnaire (CSUQ) [12], Software Usability Measurement Inventory (SUMI) [13], After Scenario Questionnaire (ASQ) [14], Post-Study System Usability Questionnaire (PSSUQ) [15], Website Analysis and Measurement Inventory (WAMMI) [16], The Lavie and Tractinsky questionnaire [17], The Usefulness, Satisfaction, and Ease of Use (USE) questionnaire [18], Measuring the Usability of Multi-Media Systems (MUMMS) [19].
The users gave us valuable feedback [20] and useful specialists information about the context, structure and processes of SABIS and SabisWeb systems and the future app.

The next step was to sum up and analyse all the gathered feedback information. We implemented the results into the second mockup to gather more information about all the user groups. In the ideation and prototyping steps we designed all other features (micro-services) to be developed in SabisApp: "Chat", "Events Calendar", "Feedbacks" and "Logs". We built the new mockup [21] (See Figure 2.5 and the footnote link) with the help of Justinmind [22] prototyping tool which allows to build large high-fidelity mock-ups and prototypes with a lot of animations and database objects.

After testing the second mockup with users and receiving the new feedback and new ideas from the online questionnaire (with the same questions) we got more detailed information about the user groups: trainees (a trainee can be also a trainer but in other course, a trainee can be very good/not so good/bad at learning and participating in the course quizzes), trainers (a trainer can be an internal or external) and training responsible persons. In the design step we took into consideration Material design, which is standard to use for the mobile GUI in the company. During this step it was also possible to design personas (the archetypes of the user groups and types) and their needs and work which was summed up in the flow chart and personas documents [23].

[11] https://blog.seibert-media.net/blog/2011/04/11/usablility-analysen-system-usability-scale-sus/
[12] https://garyperlman.com/quest/quest.cgi
[13] http://sumi.uxp.ie/en/
[14] https://garyperlman.com/quest/quest.cgi?form=ASQ
[15] https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/
[16] http://www.wammi.com/samples/index.html
[17] http://visawi.uid.com/
[18] https://garyperlman.com/quest/quest.cgi?form=USE
[19] https://www.sciencedirect.com/topics/computer-science/usability-questionnaire
[20] https://github.com/shulikaga/Bachelor_project_ux_part/blob/master/SurveyResults.pdf
[21] https://github.com/shulikaga/Bachelor_project_mockups_part/tree/master/second_mockup
[22] Download and install Justinmind 30 days trial software https://www.justinmind.com/, download and add the files from https://github.com/shulikaga/Bachelor_project_mockups_part/tree/master/second_mockup to Justinmind, click "Start Animation" button to see all the features and accounts of the prototype
[23] https://github.com/shulikaga/Bachelor_project_ux_part/blob/master/SabisApp_flow_model.pdf. https://github.com/shulikaga/Bachelor_project_ux_part/blob/master/Sabisapp_Personas_Tabelle%20-%20Tabellenblatt1.pdf

| | | | |
|---|---|---|---|
| Roles | Are you a trainer or a trainee? | Multiple choice | Before Mockup Testing |
| Gadgets | What gadgets will you use for SabisApp? | Multiple choice | Before Mockup Testing |
| Tasks | What tasks will you complete for SabisApp? | Long free text | Before Mockup Testing |
| UserNeeds | How important will be the following aspects for you using SabisApp? The Scala: 5 means "very important". | Array (5 point choice) | Before Mockup Testing |
| homePage | Home page for differen display resolutions: - Open the mockup (with the password "sabis1") in the new window and move it to the second display. - Click on the gadgets icons (bottom) to change the view. - Rotate the views. - Change the size to have the whole view. - Try to click on all buttons. - Write your impression of the page: what you liked, problems, expectations, errors, ideas what to improve in the design. | Multiple short text | While Mockup Testing |
| Schulungsplan | "Schulungsplan" page for different screen resolutions: - In home page click on "Schulungsplan" button. - Click the gadgets icons to change the view. - Rotate the views. - Change the size to have the whole view. - Try to click on all buttons. - Write your impression of the page: what you liked, problems, expectations, errors, ideas what to improve in the design. | Multiple short text | While Mockup Testing |
| Microlearnings | "Microlearnings" for different screen resolutions: - In home page click on the "Microlearnings" button. - Click the gadgets icons to change the view. - Rotate the views. - Change the size to have the whole view. - Try to click on all buttons. - Write your impression of the page: what you liked, problems, expectations, errors, ideas what to improve in the design. | Multiple short text | While Mockup Testing |
| Quiz | "Quiz" for different screen resolutions: - In "Microlearnings" page click on the "Kursname Quiz" button. - Click the gadgets icons to change the view. - Rotate the views. - Change the size to have the whole view. - Try to click on all buttons. - Do the Quiz (Wissen, Training) and go to the "Score" page. - Write your imression of the page: what you liked, problems, expectations, errors, ideas what to improve in the design. | Multiple short text | While Mockup Testing |
| DasStehtAn | "Das steht an" page with different screen resolutions: - From home page open the page with all of the upcoming events. - Click the gadgets icons to change the view. - Rotate the views. - Change the size to have the whole view. - Try to click on all buttons. - Write your impression of the page: what you liked, problems, expectations, errors, ideas what to improve in the design. | Multiple short text | While Mockup Testing |
| Teilnehmerliste | "Teilnehmerliste" with different screen resolutions: (for trainers only) - In home page click on the "Teilnehmerliste" button. - Click the gadgets icons to change the view. - Rotate the views. - Change the size to have the whole view. - Try to scroll the page in the device. - Try to click on all buttons. - Write your impression of the page: what you liked, problems, expectations, errors, ideas what to improve in the design. | Multiple short text | While Mockup Testing |

| Code | Question | Question type | Group | Mandatory | Other |
|---|---|---|---|---|---|
| Learnability | Learnability The scala: 5 means "totally agree". | Array (5 point choice) | After Mockup Testing | * | ⊘ |
| Efficiency | Efficiency The scala: 5 means "totally agree". | Array (5 point choice) | After Mockup Testing | * | ⊘ |
| UserSatisfaction | Aesthetics and User Satisfaction The scala: 5 means "totally agree". | Array (5 point choice) | After Mockup Testing | * | ⊘ |
| SystemFeedback | System Feedback The scala: 5 means "totally agree". | Array (5 point choice) | After Mockup Testing | * | ⊘ |
| OverallImpression | Overall impression: The scala: 5 means "Excellent". | Array (5 point choice) | After Mockup Testing | * | ⊘ |
| Comments | Final comments, suggestions, problems (optional). | Long free text | After Mockup Testing | | ⊘ |

Figure 2.4: Survey Design

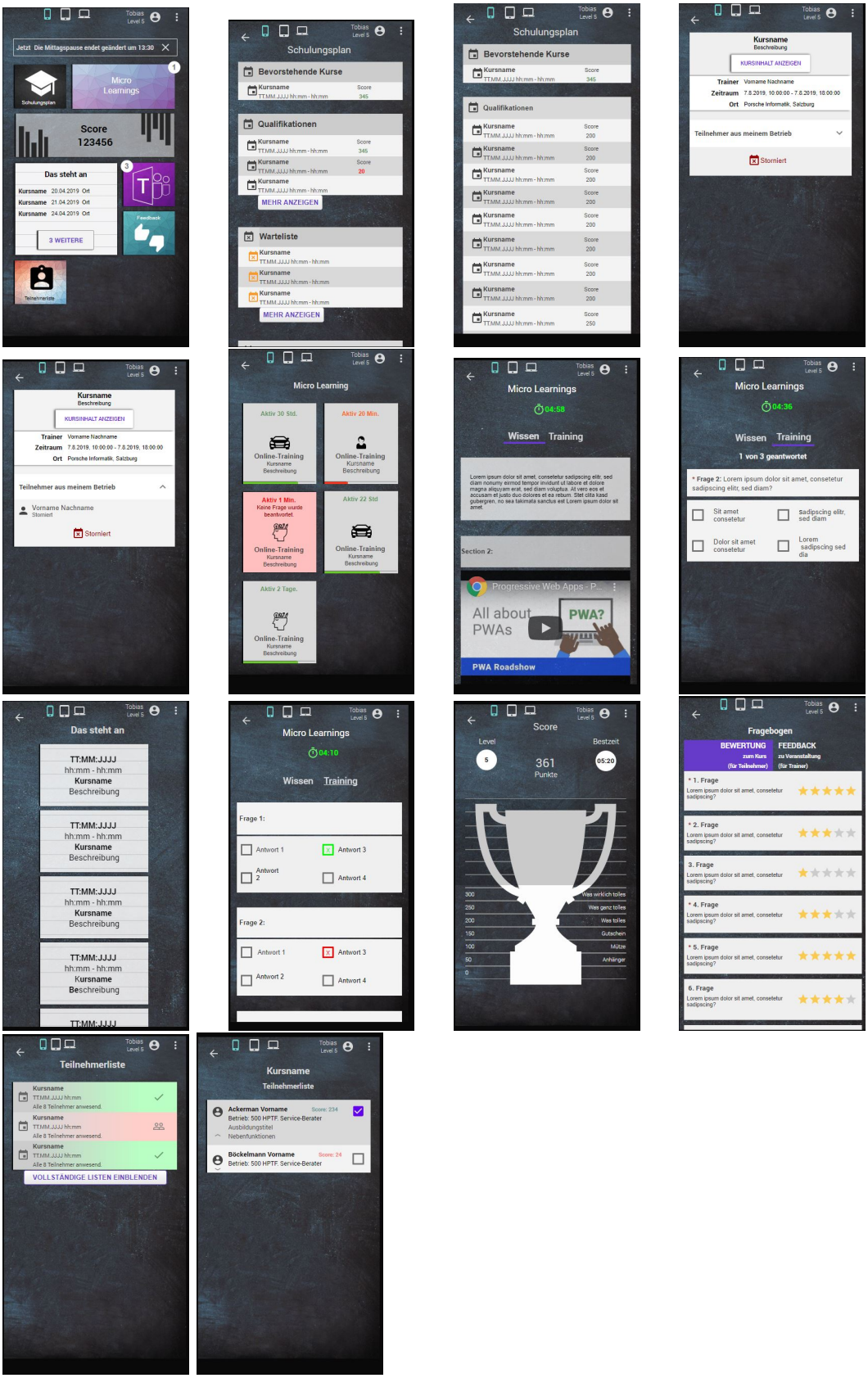


Figure 2.5: Second Mockup

For designing the final prototype we also used Justinmind tool which also allows designing GUI with Material design[24] elements and use a small database to save the designed personas accounts with speaking names (Max Traner, Clint Lehrer, Dexter Prof, Brian Fleißig, Paul Lehrling, Norman Pupils and Kathrin Responsible). The final prototype (See Figure 2.6) consists of different views for every persona[25].

For the final prototype evaluation we asked several users to test and tell what they liked and what needs improvements, but this time there were little critique and the users were satisfied with the last version.

## 2.4 The Final Architecture and Resources Plan

After the final GUI design was presented and accepted we started to brainstorm of an architecture, requirements and to plan the resources needed (time and budget). Unfortunately, one of our key team members, who was assigned to this project, left and we had to plan the whole architecture almost from scratch, except the core back-end infrastructure (See Figure 2.7). The final requirements plan was then presented in the "SABIS-App-API_Projektauftragv1.pdf" [26] document.

The scope of the bachelor thesis covers only the GUI part of the SabisApp project, that is why we present and describe here only the micro-services part (Micro Learning, Planer, Score, Trainer/Trainee Feedback, Participants List, Events Overview and User Logs) which will be implemented with Angular and Vue.js frameworks in order to compare and choose the right one for SabisApp project.

The we decided to use TypeScript for both frameworks for a better comparison and to be able to code type safe. All the GUI elements will be developed with the help of Material design frameworks for both frameworks.

The longer the development time, the higher is the cost, the less attractive it is for the investors, the less money it is possible to earn. But it is always important to consider if the project needs to be done in a short time, then the speed of development is very important. The speed of the software development is inversely proportional to the quality of the software [14]. It implies, that only after understanding the size of the project, the timing of the development process and the technological capabilities of the frameworks (Angular and Vue.js) and after choosing the web framework that fits better, it is possible

[24] https://material.io/design/introduction/

[25] Download and install Justinmind 30 days trial software https://www.justinmind.com/, download and add the files from https://github.com/shulikaga/Bachelor_project_mockups_part/tree/master/latest_mockup to Justinmind, click "Start Animation" button to see all the features and accounts of the prototype

[26] https://github.com/shulikaga/Bachelor_project_ux_part/blob/master/20190704_SABIS-App-API_Projektauftragv1_obep_kue_beh.pdf

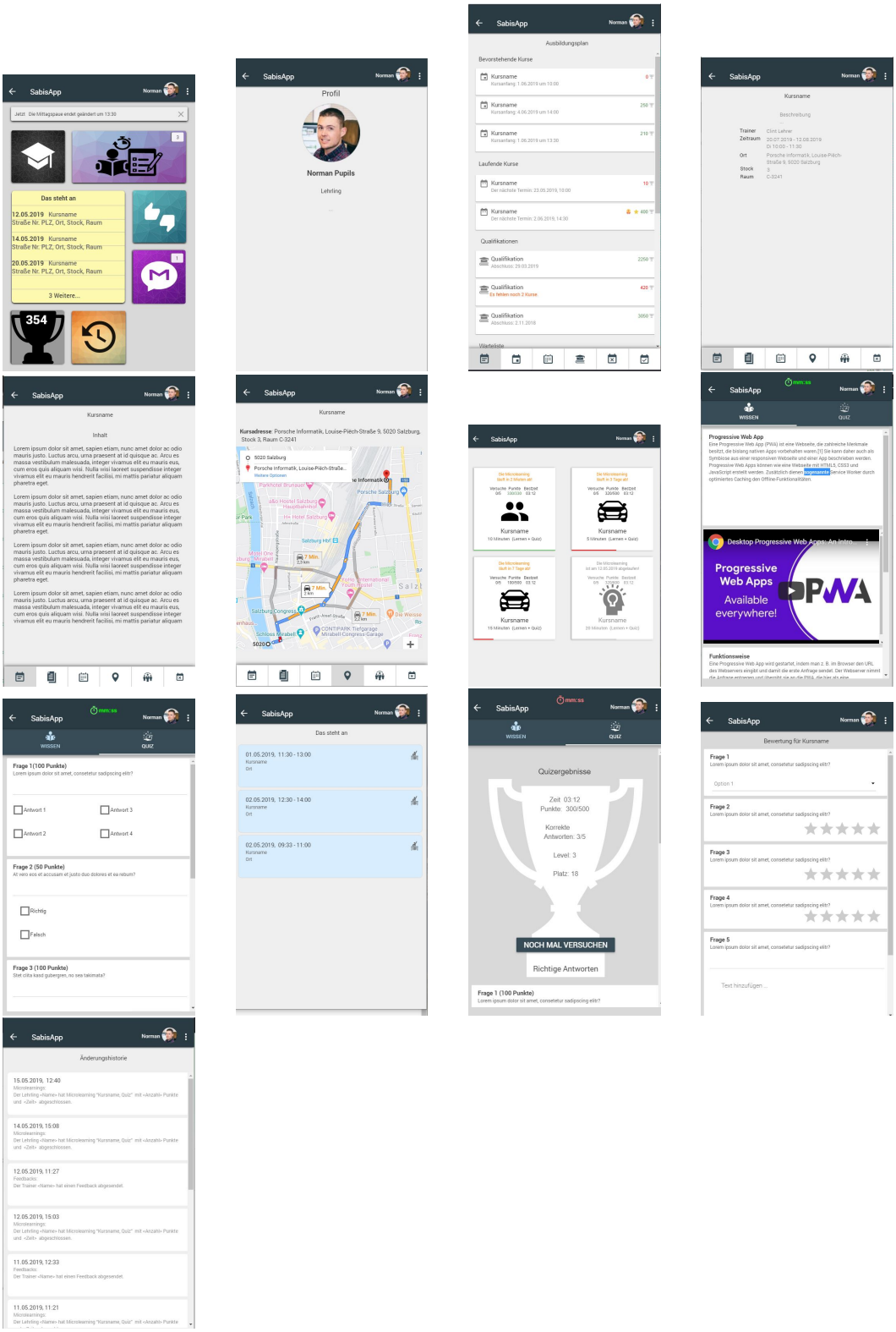


Figure 2.6: Final Prototype with Material Design

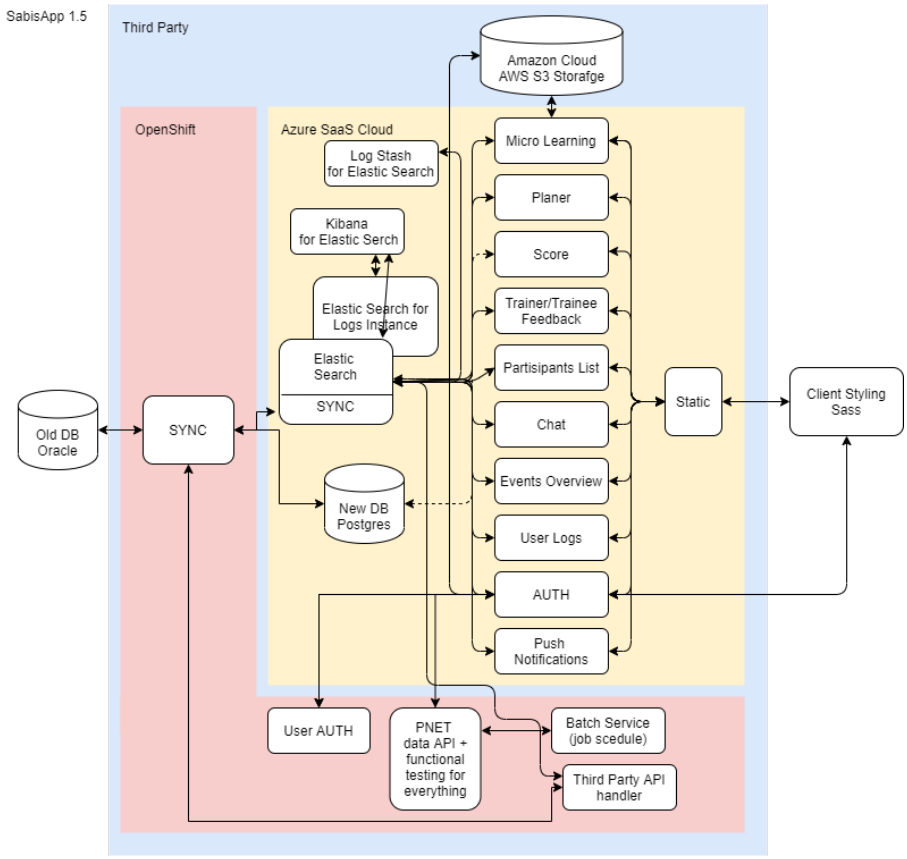


Figure 2.7: SabisApp Architecture 1.5

to make a preciser plan for the SabisApp project.

The right choice of framework depends not only on the technology, but also on the development team due to the different approaches of the frameworks[14] [13].

In the next chapter we compare the technical capabilities and other aspects of Angular and Vue.js frameworks to have an overview of their advantages and disadvantages.

# Chapter 3

# Angular vs. Vue.js

This chapter presents different comparison criteria of Angular and Vue.js frameworks. The frameworks will be compared against history background, framework architecture concepts and other important criteria such as: security, modularity, performance, testability, maintainability and future security, platform independence and compatibility, popularity and available Know-how, simplicity and ease of use, suitability for the implementation of lean and fat SPAs, size of code and cyclomatic complexity. Next the framework choice survey, based on the bachelor thesis questionnaire by Frederik Schlemmer, will be summarised in a table and the choice result, relevant for the SabisApp project will be made.

## 3.1 Background Comparison

Angular is an open-source framework for developing web applications, completely rewritten from scratch in TypeScript, was initially the 2.0 version of Angular.js. Angular.js was developed by a Google team and a community of developers from various companies. The drastic changes in this version caused controversy among the developers. It had a different architecture and was not backward compatible with the first version, that is why, to prevent confusion, it was decided to develop it as a separate framework, the version numbering of which starts with 2. For clarity, it was also recommended to use different names to distinguish "Angular" from "Angular.js" and for the differences in syntax, language and new concepts. Angular was released on September 14, 2016. But Angular does not describe itself as a framework, but as a platform for creating web applications. The term platform comes from the basic idea that Angular applications can operate different platforms such as smartphones, desktops or televisions[4].

In 2013, a Google employee, Evan You, working on one of the projects, came to the conclusion that there were no ready-made solutions for rapid prototyping of complex user interfaces of web applications: React was then at an early stage of development, the main tools were only the frameworks like Angular.js or the MVC architecture-oriented Backbone.js, which were not simple to learn and use and were focused on developing large applications. To overcome this gap, Evan You began developing Vue.js (in various

literature sources it is called Vue or Vue.js, but throughout this document we will use Vue.js), which, while maintaining simplicity, proved to be suitable not only for prototyping, but also for full-fledged development. In October 2015, version 1.0 of the library was released, in September 2016 - version 2.0. Vue.js version 3, currently at the prototype stage, plans to switch to TypeScript. Vue.js is an open-source, client-side JavaScript framework for creating single-page, reactive-style web-applications based on the MVVM pattern. Interestingly, all three platforms use the MIT license, which provides restrictions on reuse, even in proprietary software [5].
The famous products, written in Angular are: Youtube, Netflix, PayPal, Gmail, Microsoft Office Home, Xbox, Forbes, Udacity and many others[1]. The famous products, written in Vue.js are: Netflix, Radiohead Public Library, Adobe Portfolio, Xiaomi Products, Alibaba, GitLab, Official Nintendo Website, Facebook's Newsfeed [2]. It is interesting to observe that Netflix was built with Angular and some parts with Vue.js together. Facebook is known to be built with React framework, but also uses NewsFeed which was built with Vue.js. It implies that Angular and Vue.js together build a big intersection of common features but also bring other features that are not available in the other frameworks.

## 3.2 Main Architecture Concepts Comparison

An **Angular** application is a collection of many individual modules. Every module represents a feature area in an application. e.g. there can be a "user" module that is related to the application's users, and an "admin" module, that is related to the application's administrators. An angular application has at least one module which is the root module and by convention this is called the app module. Each module in turn is made up of components and services. A component controls a portion of the view on the browser e.g. there can be a component for navigation, one for the sidebar and one for the main content. But again an angular application will have at least one component which would be the root component of the application and this again is called the app component by convention. All other components will be nested inside this root component. Each component will have the HTML-template to represent the view in the browser and a class that controls the logic of the particular view module.

A module has also got one or more services which is basically a class that contains the business logic of the application. In addition to components and services, a module can contain a few more pieces of the Angular application like "package.json" file that include all needed configurations and dependencies for the application to work etc. Modules interact and ultimately render the view in the browser.

With this understanding of the high-level architecture concepts, let's take a look at some of the files and folders and how the execution flows. We will look at the initial structure of the project after it was created. The project folder contains a lot of files. But for this

[1] `https://www.madewithangular.com/categories/angular/`
[2] `https://madewithvuejs.com/`

stage we will look at the very important files which represent the explained architecture above.

The packages are installed when the developer runs the command *ng new projectName*. All packages are then installed inside the "node_modules" folder.
The "src" folder contains the files which represent modules, components and html-templates which we have already mentioned above. In the folder there is a "main.ts" file which is the entry point to the Angular application.

There is also the "app" folder which contains the "app.module.ts" which is the root module and also "app.component.ts" which is the root component.

When we run the command *ng serve* to start the application, the execution comes to the "main.ts" file. Over here we bootstrap the root module. From the app.module.ts the app component will be bootstrapped. This "app.component.ts" file has two things: the HTML template and the class to control the view logic.

The created project already contains all possible library imports, dependencies and a hierarchical file structure for components, to quickly develop complex and powerful web application with different features and to be able to test and maintain it.

**Vue.js** is a progressive JavaScript Framework which means if there is already an existing server side application, it is possible to plug Vue.js into one part of the server-side application to make a richer and more interactive user experience. Or if there is a need to build more business logic into the front-end part, Vue.js has the core libraries (e.g. Vuex, vue-router) and the ecosystem to scale the project. Vue.js framework was designed, unlike other monolithic frameworks, to be incrementally adoptable. The core library is focused on the view layer only, and can be integrated with other libraries or existing projects (mixins like Vuex, Redux, Vue-router etc).

Like in Angular, with Vue.js it is possible to split the web page into reusable components (i.e. header and navigation, sidebar), each having its HTML, CSS and JS files. But it can be done only manually by creating folders, putting files with the components source code. Angular does all this automatically.

During the project creation (cmd or with the help of GUI) there are several configurations to be chosen: default or manually select features: TypeScript support, Vuex, Router etc. After the project was created (e.g. the default configuration), the project folder contains of very few files: index.html (1 part with html elements) and main.js (JavaScript part) [17]. In the main.js file there is an instance of the app:

```
1  const app = new Vue({
2  el:'#app'
3  })
```

It is possible to create components, but the project initially contains only one class instance and index.html files.The template file consists also the style part by default, so there are less files to generate in comparison to Angular.
If it comes to larger applications, there is a need to split the file into several components and files. Vue.js has a command line interface: the command *vue init webpack newProject* (file/folder system: build, config, node_modules, public, dist, src, static, test folders).
Let us look at the important files for bootstrapping of the project: package.json with the preinstalled dependencies, src/main.js, App.vue, src/components/, dist/index.html. Index.html contains the html-code to be shown in the browser, in the body there are scripts that run the code from app.js. App.js has vue-template of the html components to be shown in the index.html, a script that defines the components of the app.vue and the styles. main.js imports app.vue and creates a new instance and renders app.vue to the browser.

### 3.2.1 Components

[9] In **Angular**, a component plays the part of the controller/viewmodel, and the template represents the view. To create a new component in Angular there is a cli command *ng add generate component componentName*. After the execution of this command, a test folder in the app folder with 4 files will be generated: e.g. test.component.html, test.component.spec.ts, test.component.ts, test.component.css and the file app.module.ts will be updated (the new component is imported and added to the ngModule, declarations array). The word "component" in the files names is the naming convention for components in angular-cli.

In **Vue**, to create a component, the folloving steps are needed: firstly, the command *vue init webpack*, then to create a new folder and manually and put the component source file inside, then to register it in parent component (App.vue). This method takes more time as generating the component in Angular, because a developer has to write the whole structure manually. The cource code for the component is located in one file (html, JavaScript and styles, but it is also possible to put these parts into separate files if the html code grows).

### 3.2.2 Routing

Routing allows users to navigate between different components based on actions taken by the user. It can also be directly done by using the route URL in the browsers. Angular's and Vue's routes are configured as part of initialization of the app; it happens before any of the components are rendered. Angular provides its own routing module; so does Vue:JS, though it requires an additional installation [17].

The **Angular** Router is an optional service that presents a particular component view for a given URL. It is not part of the Angular core. It is in its own library package, @angular/router. There is a separate module (app-routing.module.ts) within the app to configure the routing.

For a **Vue.js** app, the routing configuration is specified within main.js file where the app is initialized. It is recommended to use the officially-supported vue-router library [3] [17].

### 3.2.3 Syntax Comparison

As already described above, Vue.js was written by the former Google/Angular.js employee, and meant to be the framework that uses the benefits of Angular.js and improves or solves its weak points and problems, this implies, that the syntax of Vue.js may not be very different from that of Angular.js/Angular. As we can see from the Syntax Comparison Overview Table (See Tables 3.1), it is indeed the fakt, that the syntax of both frameworks look the same or almost the same: filters, interpolations, binding. The most differences are e.g. in naming and formatting with [], {}: "v-if", "v-model" vs. "*ngIf", "*ngModel" and so on. If it comes to the Typing, there is a big difference of Angular and Vue.js. Firstly, as it was already stated. Angular is written in TypeScript and Vue.js in JavaScript, which means, that Angular fitts better for the developer teams with experience of OOP (object oriented programming). Though Vue.js also supports TypeScript and the documentation has a detailed description how to use Typescript with Vue.js, it is still impossible to directly create generic and optional types (See Tables 3.1).

[3] https://github.com/vuejs/vue-router

Components

**Angular**

Modules:

```
@NgModule({
    declarations: ...,
    imports: ...,
    exports: ...,
    providers: ...,
    bootstrap: ...})
class MyModule {}
```

Components:

```
import { Component } from '
    @angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1>{{title}}</h1>
    <h2>My favorite hero is: {{
        myHero}}</h2>`
})

export class AppComponent {
  title = 'Tour of Heroes';
  myHero = 'Windstorm';
}
```

Services:

```
import { Injectable } from '
    @angular/core';

@Injectable({
  providedIn: 'root'
})

export class MyserviceService {
    constructor() { }
}
```

**Vue.js**

Components:

```
 Vue.extend({
    data: function(){ return
        {...}}.
    created: function(){...},
    ready: function(){...},
    components: {...},
    methods: {...},
    watch: {...}
    //other props excluded
});
```

```
Vue.component('button-counter',
    {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button
      v-on:click="count++">You
      clicked me {{ count }}
      times.</button>'
})
```

Directives

**Angular**

```
<section *ngIf="showSection">
<li *ngFor="let item of list">
<div [ngSwitch]="conditionExpression">
<ng-template [ngSwitchCase]="case1Exp">...</ng-template>
<ng-template ngSwitchCase="case2LiteralString">...</ng-template>
<ng-template ngSwitchDefault>...</ng-template>
</div>
<div [ngClass]="{'active': isActive, 'disabled': isDisabled}">
<div [ngStyle]="{'property': 'value'}">
<div [ngStyle]="dynamicStyles()">
```

**Vue.js**

Conditional Rendering:

```
<h1 v-if="awesome">Vue is awesome!</h1>
<h1 v-else>Oh no/h1>
<li v-for="let item of list">
```

Directives:

```
<div v-text="message"></div>
<div v-on="click : clickHandler"></div>
</div>
<my-component v-ref="some-string-id"></my-component>
<div v-show="showMsg" v-transition="{{dynamicTransitionId}}"></div>
<div v-pre>
  <!-- markup in here will not be compiled -->
</div>
```

```
// Custom directives: to register a global custom directive called `v-focus`
Vue.directive('focus', {
  // When the bound element is inserted into the DOM...
  inserted: function (el) {
    // Focus the element
    el.focus()
  }
})
```

```
//To register locally
directives: {
  focus: {
    // directive definition
    inserted: function (el) {
      el.focus()
    }
  }
}
```

Filters

| Angular | Vue.js |
|---|---|
| Filters (Pipes in Angular)<br>`@Component({`<br>`  selector: 'app-my-birthday',`<br>`  template: ` My birthday is {{ birthday \| date }} `<br>`})`<br>`export class MyBirthdayComponent {`<br>`  birthday = new Date(1988, 3, 15); // April 15, 1988`<br>`}`<br>also custom pipes | Filters:<br>`{{ message \| capitalize }}`<br>`{{ message \| filterA \| filterB }}`<br>` `<br>`Vue.filter('reverse', function (value){`<br>`  return function(value){...}`<br>`});` |

Interpolation

| Angular | Vue.js |
|---|---|
| `{{myVariable}}` | `{{myVariable}}` |

Model Binding

| Angular | Vue.js |
|---|---|
| Property Binding<br>`<input type="email" [value]="user.email">`<br>Event binding<br>`<button (click)="cookBacon()"></button>`<br>Two-way Data Binding<br>`[(ngModel)] = "[property of your component]"` | `<input type="text" v-model="myVar">`<br>`<p v-model="myVar"> </p>`<br>`<div v-on="click: myMethod($event)"></div>` |

Types

TypeScript

```
let decimal: number = 6;
let done: boolean = false;
let color: string = "blue";
let list: number[] = [1, 2, 3];
let list: Array<number> = [1,
    2, 3];
let fun: Function = () =>
    console.log("Hello");
\end {lstlisting}
\begin{lstlisting}[
    style=htmlcssjs]
enum Direction {
    Up,
    Down,
    Left,
    Right
}
let go: Direction;
go = Direction.Up;
```

```
class Person {};
let person: Person;
let people: Person[];
let notsure: any = 1;
function returnNothing(): void
    {
  console.log("Moo");
}
```

Generics, optional Types:

```
interface Type<T> extends
    Function {
  new (...args: any[]): T
}
```

JavaScript:

```
var length = 10;  // Number
var lastName = "Peterson";//
    String
var x = {firstName:"Tom",
    lastName:"Simpson"};//
    Object
```

With TypeScript Support:

```
const Component = Vue.extend({
  // type inference enabled
})
```

```
export default class
    MyComponent extends Vue {
  message: string = 'Hello!'
  onClick (): void {
    window.alert(this.message)
  }
}
```

```
import Vue, { VNode } from 'vue
    '
const Component = Vue.extend({
  data () {
    return {
      msg: 'Hello'
    }
  },
  methods: {
    greet (): string {
      return this.msg + ' world
          '
    }
  };
  render (createElement): VNode
      {
    return createElement('div',
        this.greeting)
  }
})
```

No generic types support

Figure 3.1: Angular vs. Vue.js Syntax

## 3.3 Other Comparison Criteria

### 3.3.1 Security

As a result of the increasing functionality of web applications, the number of attacks on the Internet has also increased. Confidential information such as user data and payment information is stored in a database by the applications. It is therefore necessary to provide protection against general problems. Angular documentation security page describes all the built-in protections against common web-application vulnerabilities and attacks e.g. cross-site scripting attacks. But It doesn't cover application-level security, (authentication and authorization). But refers further to OWASP Guide Project [4]. Angular teams regularly update the Angular libraries, and these updates may fix security defects discovered in previous versions. To systematically block XSS (cross-site scripting) [5] bugs, Angular treats all values as untrusted by default. When a value is inserted into the DOM [6] (document object model) from a template, via property, attribute, style, class binding, or interpolation, Angular sanitizes and escapes untrusted values. Angular has also built-in support to help prevent two common HTTP vulnerabilities, cross-site request forgery (CSRF or XSRF) and cross-site script inclusion (XSSI). Both of these must be mitigated primarily on the server side, but Angular provides helpers to make integration on the client side easier. Also found new vulnerabilities can be reported to the Angular team per email[4].
Vue.js has also built in security measures. Whether using templates or render functions, content will be automatically escaped. That means in this template: <h1> userProvidedString </h1> if userProvidedString contained: '<script>alert("hi")</script>' then it would be escaped to the following HTML: lt;scriptgt;alert(quot;hiquot;)lt;/scriptgt; preventing the script injection. Vue escapes html content by using native browser APIs, like textContent, so a vulnerability can only exist if the browser itself is vulnerable. Dynamic attribute bindings are also automatically escaped. But there are still potential security issues for HTML, URLs ans styles because user-provided HTML can never be considered 100% safe. In the Vue.js documentation about the framework security several best practices are presented to avoid such issues e.g. for using SSR (server-side rendering) [17, 16].

### 3.3.2 Modularity

An increased range of functions results in a more complex application, whereby the modularity reflects an important aspect for modern web applications. A division into smaller isolated components should be made possible, so that they can be reused in later applications [9]. Modularity enables the entire application to be divided into smaller isolated components, which means that the components can be used outside of the original application. Due to the strict adherence to a design pattern, the maintenance

[4] https://wiki.owasp.org/index.php/OWASP_Guide_Project
[5] https://de.wikipedia.org/wiki/Cross-Site-Scripting
[6] https://de.wikipedia.org/wiki/Document_Object_Model

of an application is favored and an exchange or further development of components is easily possible [16, 7, 6, 12, 11, 12].

### 3.3.3 Performance

The performance of a SPA application is crucial for whether a user has a positive user experience. With a longer loading time, users tend to leave an application before completion. In this area, seconds, sometimes milliseconds, can be decisive for the user experience. Furthermore, caching options can not only make the application more performant, but also reduce the transfer between server and client[16]. Due to its MVC pattern structure, Angular splits tasks into logical chunks, thus reducing the initial loading of a web application.The MVC model also presents the view part on the client side, drastically reducing queries in the background. Also, communication is done in an asynchronous mode, meaning that less calls to the server are performed. Angular shows changes made to the model instantly into the views[4]. But due to many features of this framework used in a large application may cause slower performance compared to Vue.js. With its small size and the ability to incrementally adopt parts of its technology, Vue.js application occur to be very performant[4].

### 3.3.4 Testability

Test-based development is a common practice and very important these days. An application must be adequately tested for accuracy by automatic tests. The ability to check the program code for correctness using various tests is an important feature for high-quality applications. With the help of test options such as unit, integration and function tests, it can be guaranteed that changes made do not interfere with the functionality of the application [16].
Angular supports automated testing. Angular documentation offers tips and techniques for unit and integration testing Angular applications. The framework's CLI uses Jasmine and Karma testing tools. It is also possible to use other testing libraries and test runners. The CLI can also run unit tests and create code coverage reports. Code coverage reports show any parts of the code base that may not be properly tested by the unit tests[4].
Vue CLI has built-in options for unit, integration testing with Jest or Mocha libraries. There are also official Vue.js Test Utils documentation which provide more detailed guidance for custom setups[17].

### 3.3.5 Maintainability

An application evolves over time after the first release. The maintainability plays an important role for the further development, so that further developers join the team and new functionalities can arise on the project. The future viability of the application should be supported by a simple expandability, as well as a constant further development of the framework used. A high maintainability of a project is crucial for the future. In addition, the framework used should be continuously developed[16].
Talking about maintainability of code it is also important to know how experienced

framework developers describe this aspect. In Angular components are easily decoupled from each other and can be replaced with better implementations so it is easy to code with large teams and still have a maintainable code.
As Vue.js is a flexible framework where developers can experiment with application architectures adding different libraries and extend the framework. This flexibility may lead to a not maintainable code, which is not understandable for new team members. The solution of this drawback are web tutorials, best practices and blogs about writing clean and maintainable code [1, 7, 10, 6, 15, 12, 11].

### 3.3.6 Platform independence and Compatibility

Today's variety of devices for displaying web applications requires increased compatibility. The application must produce the same result on all common browsers and all mobile devices. Web frameworks offer the possibility to use applications on the desktop or download on mobile devices[16]. Angular and Vue.js provide a good compatibility with all popular web browsers and are made for implementation SPA and PWA web apps. The only advice of the both frameworks is to update the framework, libraries and dependencies after a new release for everything to work [5, 4].
Different projects require compatibility with different tools, libraries frameworks and etc. Our SabisApp project (its front-end part) requires to be compatible with Material design library because of the company's GUI design guidelines. Material design enables web applications to look professionally, well designed and provide with a lot of GUI elements.

To integrate Material Design to the Angular project there is Angular Material UI component library [7]. It can be added with the help of the command *ng add @angular/material*. This command adds and installs automatically all the elements, animations and dependencies needed. After this step it will be possible to import and add Material elements in the app.module.ts file:

```
1  import { MatSliderModule } from '@angular/material/slider';
2  @NgModule ({....
3    imports: [...,
4    MatSliderModule,
5    ]
6  })
```

and add the <mat-slider> tag to the app.component.html like this:

```
1  <mat-slider min="1" max="100" step="1" value="1"></mat-slider>
```

For Vue.js there are several very popular Material design libraries: Vue Material [8], Vuetify.js [9], Material Components Vue [10], developed by one of our former team members.In our implementation we will try to integrate two of them (Material Components

[7] https://material.angular.io/
[8] https://vuematerial.io/
[9] https://vuetifyjs.com/
[10] https://github.com/matsp/material-components-vue


Figure 3.2: Google Trends: Vue vs. Angular
11

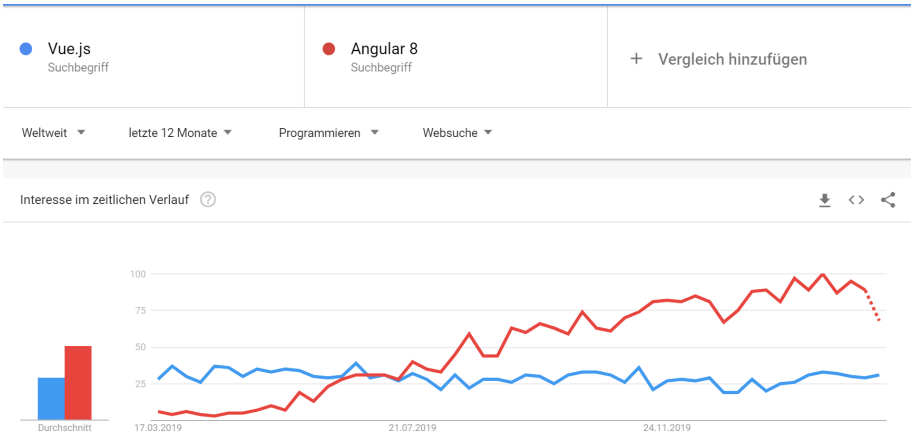


Figure 3.3: Google Trends: Vue vs. Angular 8
12

Vue and Vue Material), because our team members and other colleagues from the company have experience using them and it could help much faster and productive GUI developments.

### 3.3.7 Popularity and Available Know-how

The popularity of a framework has an impact on the size of the associated community. Increased popularity offers a larger community. This means that a wide range of know-how is available through forums or blogs. A user can benefit from a large community by providing experienced developers on platforms such as StackOverflow. The framework also benefits from faster troubleshooting on GitHub[16].

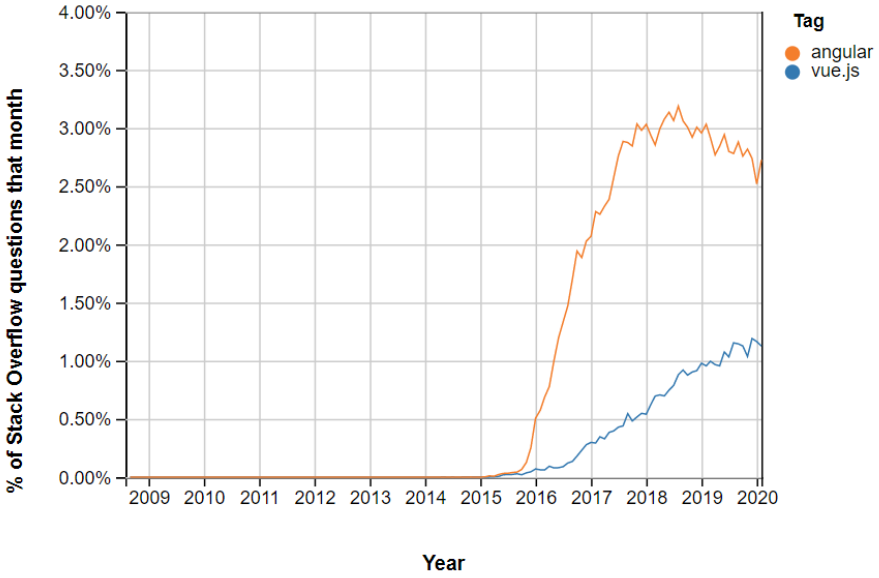


Figure 3.4: StackOverflow Trends
13

There is no exact open data on the breadth of framework use, however, an indirect analysis of Google Trends, npm trends, GitHub trends, StackOverflow websites gives a rather interesting distributions (See Figures 3.2, 3.3, 3.4 and 3.5).
Google Trends service allows to analyze user queries by frequency, relative to any region of the world.The data is not a priority because requests can occur for various reasons. For example, a developer who is looking for answers to questions regarding the framework in use at the moment. However, such a comparative characteristic may show interesting data. Although Vue.js came out later than Angular, it is already bypassing it and it is trying to reach the pace of Angular [3]. The results for "Vue vs.Angular 8" over the last twelve months show a more exacter than the results for "Vue vs. Angular". At the beginning of the last yeas Vue.js was more popular than Angular, but then Angular got its popularity back, but last months loses it a bit. Vue.js does not grow, as it can be seen in Figure 3.3.
In StackOverflow Trends the results for "Vue vs. Angular" over the last 6 years show that Angular has also more search popularity then Vue.js. Unfortunately, these results can also be not exact, because the searches might be dome also for the Angular.js framework[3]
.
npm trends results show, that Vue.js is also gradually gaining momentum, but Angular grows with the same speed. [3] [12] [15].

When choosing a specific architecture, one cannot ignore the support of the community of developers. Indeed, even simple functions can cause difficulties. And the more complex

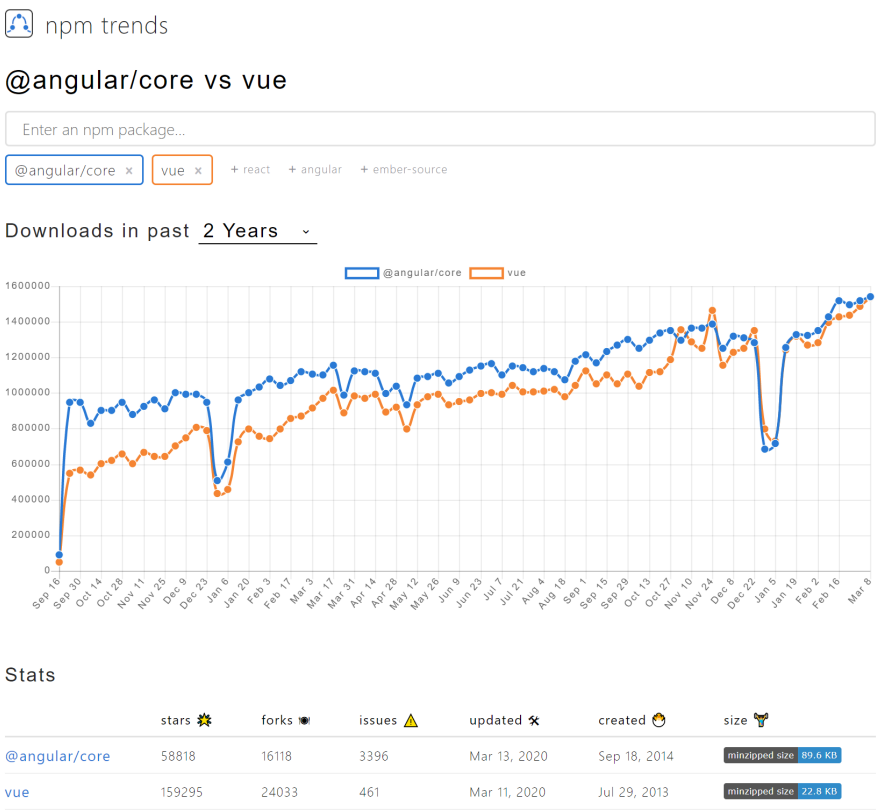


Figure 3.5: npm trends
14

the project, the more often developers turn to the technical documentation of the selected environment, and for the advice to the community. Therefore, it is worth comparing the "support" of each web framework.

Detailed documentation for Vue.js is a "visiting card" . However, supporters of the environment do not mention that most of the "Manuals" do not have a good translation into English or other languages except of Chinese[12]. The opinion of the experienced developers in our company, in the team an from the developers blogs also helped to understand the communities and documentation status of both frameworks. According to them, the user community, despite the great popularity of Vue.js on Habré or Github, is rather modest, especially compared to Angular or React. As a rule, the answer to the question about the implementation of the task is expected to take several days. And not the fact that someone will be able to answer in comparison to Angular. It will somewhat facilitate the use of the chat environment on the official website of the project.

Unlike Vue.js Angular has not only detailed documentation which was written for the framework, but also a lot of guidelines were released. The number of developers using the framework is hundreds of thousands. A developer can ask a question from the Howto category at any time. Developers do not have have to look for the official site of the framework. They can ask for help on Stackoverflow and on other platforms

[1, 7, 15, 11, 12].

### 3.3.8 Simplicity and Usability

Web frameworks should support the development process of the application. Standardized processes should also be available. It is believed that Angular framework requires a lot of learning of a great number of concepts and features such as TypeScript and Model View Controller (MVC). Although it takes some time to learn Angular, it does have a clear idea of how the interface works. The developers describe Vue.js as easier to learn than Angular because the API of Vue.js was kept simple. In order to be able to use Vue.js, only knowledge of JavaScript and HTML is necessary, the exemplary documentation at vuejs.org is sufficient for use and complete learning. Vue.js is intuitive, because its development will be quit fast. However, when working with it, a lack of code is acceptable, therefore it requires constant debugging and additional testing [1, 7, 6, 15, 12, 11].

### 3.3.9 Suitability for the Implementation of Lean and Fat SPAs

A SPA (single page application) can be implemented for a very small as well as for a very large project. Web frameworks can cover both cases, but a lightweight framework is ideal for a small project. In the opposite case, it is worth using a heavyweight framework. A flexible adaptation to the implementation of Lean or Fat SPAs should be made possible. As a result, smaller applications are not unnecessarily overloaded by a heavyweight framework [16].
The documentation of Vue.js describes the framework as progressive and incrementally adaptable compared to other web frameworks. This allows the developer to use the structure of the application according to their own needs. Vue.js easily integrates into different web-projects using other JavaScript libraries [5].. But sometimes developers speak of the Vue.js as an overflexible framework and it is its drawback. The only rebuke that the author sees in Vue.js is that there are too few development requirements and no explicit style guide. Over the past three years, 3 projects have been met on Vue.js, and each was so unique that it is difficult to say what unites them. However, the simplicity and flexibility of Vue.js is a double-edged sword - it allows bad code, making debugging and testing difficult [1, 7, 6, 15, 12, 11, 2].

### 3.3.10 Updates and Future of the Frameworks

Although it does not require a lot of changes to switch from one version to another, it's important to keep track of these points, as updates can sometimes be more significant and you need to pay attention to compatibility [16].
Typically, Angular updates come out every six months. There is a six-month pause before any major APIs are released, giving developers two release cycles - about one year to make the necessary changes. With regard to migration, Vue.js assumes that 90% of the APIs remain the same if the switch will be done from 1.x to 2. In addition, there is an auxiliary migration tool that works from the console. But it is to remember that if Evan

You retires, it might complicate Vue.js support a bit and Angular is being developed by thousands of people.
The company behind Angular is none other than Google. This means that there is a great deal of financial scope and a correspondingly large team.
In addition to a huge developer base from Google itself, Angular is maintained and expanded by a large open source community. According to the angular.io about page, the Google team comprises around 40 members as well as selected open source collaborators and Google Development Experts.
In the past, important developers such as Victor Savkin or Jeff Cross leave the Angular Team to take on new tasks. The remarkable thing is that even the departure of central developers due to the high bus factor (key figure for estimating project risks) has no negative effects.
Vue.js was created by an individual (Evan You), and large parts of the framework were primarily maintained by him for a long time. That's why there were always articles or opinions that this would be a pure one man show and that the framework would have no future when Evan You retired. In an interview, You says that he is currently working on decoupling the dependency on himself in order to enable the future of the framework without him.
"I personally don't want to be responsible for everything. I want the project to be able to evolve, even if some day I stop working on it. So part of my work right now is to make that happen "- Evan You
The team around Evan You has grown enormously and has found some prominent supporters of well-known companies: Sarah Drasner (Netlify), Natalia Tepluhina (GitLab) or Pine Wu (Microsoft). In addition to the actual core team, there is a huge developer community that supports the framework. The Vue.js team receives financial support from funding platforms such as Open Collective or Patreon[1, 7, 15, 12, 11].

### 3.3.11 Size of Code

The size of the libraries is an important characteristic that should be considered, since sometimes the execution time depends on the size of the files. At the moment of writing this document Angular source code has about 500 KB and Vue.js - 80 KB. Although the differences between these files are significant in terms of size, they are still small compared to the average web page size, which usually exceeds 2 MB [15].

## 3.4 Survey for Choosing the Framework

After comparing and making overview of different aspects of Angular and Vue.js frameworks, we also tested the Questionnaire by Frederik Schlemmer The source code of the questionnaire to run locally[16].This survey computes the choice of the framework for a specific project needs according to the aspects we analysed above. The aspects that

[15] https://www.codeinwp.com/blog/angular-vs-vue-vs-react/
[16] https://github.com/FrederikSchlemmer/FrameworkBuddy

were relevant for our project were: security, modularity, performance, testability, maintainability, compatibility, popularity available know-how, simplicity and ease of use. The questions that were not crucial for the choice of the framework were: performance tests, support of stub and mock objects in tests, code style, popularity by well-known companies, ranking in the HotFrameworks. And the result of the questionnaire was Angular (See FrameworkBuddy Survey Questionnaire tables).

**FrameworkBuddy Survey Questionnaire**

**Security**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is a security policy for dealing with security vulnerabilities? | relevant | Angular |
| How relevant is a support for troubleshooting? | relevant | Angular |
| How relevant is support for 'static typing'? | relevant | Angular |
| How relevant are measures against injections? | relevant | Angular |
| How relevant are options for implementing authentication? | relevant | Angular |
| How relevant are standard security-relevant configurations? | relevant | Angular |
| How relevant are measures against cross site scripting (XSS)? | relevant | Angular |
| How relevant are measures against Cross Site Request Forgery (CSRF)? | relevant | Angular |
| How relevant is the avoidance of components with known vulnerabilities? | relevant | Angular or Vue.js |

**Modularity**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is a design pattern for separating layers and components? | relevant | Angular or Vue.js |
| How relevant is a component-based approach to development? | relevant | Angular or Vue.js |
| How relevant is the flexibility of the framework? | relevant | Vue.js |
| How relevant is the separation of JavaScript / CSS / HTML? | relevant | Angular or Vue.js |

**Performance**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant are test results with Chrome Lighthouse Audit? | not crusial | Angular or Vue.js |
| How relevant is the size of the file to be transferred between server and client? | not crusial | Vue.js |
| How relevant are automated data transfer reductions? | relevant | Vue.js |
| How relevant are caching options by the developer? | relevant | Angular or Vue.js |
| How relevant is performance for actions during runtime? | relevant | Angular |
| How relevant is the performance in Lighthouse Mobile Simulation? | not crusial | Vue.js |
| How relevant is lazy loading support? | relevant | Angular |
| How relevant is the use of an ahead-of-time compiler? | relevant | Angular or Vue.js |

**Testability**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant are test options on the part of the framework? | relevant | Angular or Vue.js |
| How relevant is the possibility for unit tests? | relevant | Angular or Vue.js |
| How relevant is the possibility for functional tests? | relevant | Angular or Vue.js |
| How relevant is the possibility for performance tests? | not crucial | Angular or Vue.js |
| How relevant is the support of stub / mock objects in tests? | not crucial | Angular or Vue.js |
| How relevant is the support of HTTP requests in tests? | relevant | Angular or Vue.js |

**Maintainability and Future Security**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is a code style on the part of the framework? | not crucial | Angular or Vue.js |
| How relevant is the distribution of the framework among developers? | relevant | Angular |
| How relevant is securing the further development of the framework? | relevant | Angular |
| How relevant is the use of the framework by well-known companies? | not crucial | Angular or Vue.js |
| How relevant is the amount of code required for applications? | relevant | Vue.js |

**Platform Independence and Compatibility**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is the browser functionality test for each release of the framework? | relevant | Angular |
| How relevant is the support of current browser versions? | relevant | Angular or Vue.js |
| How relevant is the support of previous browser versions? | relevant | Angular or Vue.js |
| How relevant is the version of JavaScript in which the framework is developed? | relevant | Angular or Vue.js |
| How relevant is function on mobile devices? | relevant | Angular or Vue.js |
| How relevant is the possibility to extend the framework? | relevant | Angular |
| How relevant is the possibility of native development? | relevant | Angular or Vue.js |

**Popularity and Available Know-how**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is the level of awareness of the framework based on Google Trends? | relevant | Angular |
| How relevant is the ranking of the frameworks at HotFrameworks? | not crucial | Angular |
| How relevant are the current statistics on GitHub? | relevant | Angular |
| How relevant are the statistics on StackOverflow? | relevant | Angular |
| How relevant is the number of job offers? | relevant | Angular |
| How relevant is a broad community of users for support, documentation and code? | relevant | Angular or Vue.js |
| How relevant is the amount of libraries and applications that are developed with the framework? | relevant | Angular or Vue.js |

**Simplicity and Ease of Use**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is the provided documentation? | relevant | Angular or Vue.js |
| How relevant is support for automatic code generation? | relevant | Angular |
| How relevant is the support of the framework of common IDEs? | relevant | Angular |
| How relevant is the training period of an outside developer? | relevant | Vue.js |
| How relevant are guidelines / best practices for typical use cases? | relevant | Angular |
| How relevant is support for dynamic or static typing? | relevant | Angular or Vue.js |
| How relevant is the support of different development environments? | relevant | Angular or Vue.js |

**Suitability for the implementation of lean and fat SPAs**

| Question | Relevance for the Project | Framework Choice |
|---|---|---|
| How relevant is the lightness of the framework? | not crucial | Vue.js |
| How relevant is the performance of the framework for large applications? | not crucial | Angular |
| How relevant is the increase in complexity due to the necessary components? | not crucial | Angular |
| How relevant is the integration of necessary components in the framework? | relevant | Angular |
| How relevant are finished build tools? | relevant | Angular or Vue.js |

**Survey Results**

| Framework | Summed Points |
|---|---|
| Angular | 42 |
| Vue.js | 25 |
| **Choice** | **Angular** |

## 3.5 Summary

Of course, comparing frameworks is objectively quite difficult. But the analysis will allow a novice developer to choose a platform better.

To develop a large project it is worth considering Angular as the basis. It provides maximum flexibility and rendering speed. The vast experience of other developers will help to resolve issues that are sure to arise when working on the application. Vue.js does not yet have a large number of guide lines.

If other web developers will be involved in development in the future, then Vue.js will

be the best choice. After all, this framework is not only easy to learn, but also allows to change the application without destroying its architecture.
If the project provides for multi-stage updating and expansion of functionality in the future, then it is worth using Vue.js due to excellent backward compatibility.
Indeed, the time spent studying the specifics of the framework postpones the release of the application for commercial use for an indefinite period. At the same time, technical support of the implemented project may turn out to be an unbearable burden that will "disperse" full-time programmers.
Angular is a good choice for large corporations, as it has a complete set of tools and quality support. However, its learning curve is steep, so for beginners this framework will be very complex. Vue.js has been around for several years, but in fact is new to the market. Nevertheless, most large Asian companies have switched to its use, which indicates its quality. Especially suitable for those who need fast growth. For example, a start-up with a young team should pay attention to this particular development environment. Each of the frameworks is good in its own way, it has its own strengths and weaknesses. Next we sum up all the benefits and drawbacks of Angular and Vue.js.

Angular Benefits: full-featured and powerful, elegant programming style and patterns, detailed documentation, backed by a big team/company, well-establisched option, has a rich package ecosystem, has powerful developer tooling, robust, less error-prone code, its structure and architecture is specially designed for high scalability of the project, written in TypeScript.

Angular disadvantages: bloated and complex, has hard learning curve (the variety of different structures as injectables, components, pipes, modules, etc. complicates the study compared to Vue.js, which only have "Component"), relatively slow performance, given various indicators.

Advantages of Vue.js: simple and lightweight HTML reinforced. This means that, has many characteristics similar to Angular, and this, through the use of various components, helps optimize HTML blocks, detailed documentation, easy to learn and saves a lot of time developing the application using only basic knowledge of HTML and JavaScript, adaptable, has a good integration potential, supports TypeScript.

Disadvantages of Vue.js: Lack of resources and small but growing community, has a risk of excessive flexibility, sometimes Vue.js may have problems integrating into huge projects, and there is still no experience with possible solutions.

After making a literature, documentation review, summing up experienced developers opinions and the survey shoved us advantages and disadvantages of Angular and Vue.js frameworks it is already possible to make a decision what framework to use for out SabisApp project. But trying to implement GUI with both of the frameworks and make experience of the aspects, explained above is more crucial for the choice. In the next chapter we present the implementation results with Angular and Vue.js projects to make the final decision what framework is better to choose for SabisApp.

# Chapter 4

# Implementation Aspects

This chapter describes the development process and the results of the comparison of the Angular and Vue.js projects. The final choice of the framework will be done.

## 4.1 Setup of the Angular and Vue.js Projects

We installed and created two projects using Angular and Vue.js frameworks.
For the Angular project there were 3 steps to make: install NodeJS [1], TypeScript [2] and Angular-CLI, to run the project in the browser on `http://localhost:4200`. The files and all default configurations were installed in node_modules and the file structure with already a lot of files and folders was automatically created.
Then we installed. created and run the project with Vue.js. These steps were made to create a project: install NodeJS, Vue-CLI (in cmd npm install -g @vue/cli), create a project (cmd vue create projectName or use the Web-GUI), run on localhost.

## 4.2 Implementation Experience

The default structure of the default projects Angular and Vue.js was already described in the previous chapter. We found it very useful, that the Angular default project consisted already of routing, e2e testing functionality and the components, each had four files tor different purposes (styles, html, and TypeSctipt files) were organised in the folders. It was very easy to add new components by writing a command *ng generate component myComponent*. After Vue.js project was created, the file structure looked smaller as of the Angular's project and the files consisted of less code. We had also to make TypeScript support and create a webpack [3] to have a clear structure with components, test support like in Angular project. For the Vue.js project, we had extra to install and integrate a vue-router (create a new file index.ts and register all the components) to be able to navigate to different pages.
index.ts file:

[1] `https://nodejs.org/en/download/`
[2] `https://www.npmjs.com/package/typescript`
[3] [https://cli.vuejs.org/guide/webpack.htmlinspecting-the-project-s-webpack-config

```
import Vue from "vue";
import VueRouter from "vue-router";
import Home from "../views/Home.vue";

Vue.use(VueRouter);

const routes = [
  {
    path: "/",
    name: "home",
    component: Home
  },
  {
    path: "/about",
    name: "about",
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () =>
      import(/* webpackChunkName: "about" */ "../views/About.vue")
  }
];

const router = new VueRouter({
  mode: "history",
  base: process.env.BASE_URL,
  routes
});

export default router;
```

In Angular we had only to import and register the components in the app-routing.module.ts file:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomeComponent} from './home/home.component';
import { AccountComponent} from './account/account.component';
import { SettingsComponent} from './settings/settings.component';
import { LoginComponent} from './microservices/login/
    login.component';

import { PlanerComponent} from './microservices/planer/
    planer.component';
import { ScoreComponent} from './microservices/score/
    score.component';
import { MicrolearningsComponent} from './microservices/
    microlearnings/microlearnings.component';
import { EventsComponent} from './microservices/events/
    events.component';
import { CourseComponent} from './microservices/course/
    course.component';
import { FeedbacksComponent} from './microservices/feedbacks/
```

```
    feedbacks.component';
15  import { FeedbackComponent} from './microservices/feedbacks/
        feedback/feedback.component';
16  import { EvaluationComponent} from './microservices/feedbacks/
        evaluation/evaluation.component';
17  import { ChatComponent} from './microservices/chat/chat.component'
        ;
18  import { ParticipantsListComponent} from './microservices/
        participants-list/participants-list.component';
19  import { ChecklistComponent} from './microservices/
        participants-list/checklist/checklist.component';
20
21  import { LogsComponent} from './microservices/logs/logs.component'
        ;
22
23  const routes: Routes = [
24    {path: '', component: HomeComponent},
25    {path: 'account', component: AccountComponent},
26    {path: 'settings', component: SettingsComponent},
27    {path: 'login', component: LoginComponent},
28
29    //micro services routes
30    {path: 'planer', component: PlanerComponent},
31    {path: 'score', component: ScoreComponent},
32    {path: 'microlearnings', component: MicrolearningsComponent},
33    {path: 'events', component: EventsComponent},
34    {path: 'course', component: CourseComponent},
35    {path: 'feedbacks', component: FeedbacksComponent},
36    {path: 'feedbacks/feedback', component: FeedbackComponent},
37    {path: 'feedbacks/evaluation', component: EvaluationComponent},
38    {path: 'chat', component: ChatComponent},
39    {path: 'participants-list', component: ParticipantsListComponent
          },
40    {path: 'participants-list/checklist', component:
          ChecklistComponent},
41    {path: 'logs', component: LogsComponent}
42  ];
43
44  @NgModule({
45    imports: [RouterModule.forRoot(routes)],
46    exports: [RouterModule]
47  })
48  export class AppRoutingModule { }
```

Creating components in Angular was much easier with the command *ng generate component myComponent*. To create a component in Vue.js several steps are required: creating a .vue-file and writing the course code e.g. Home.vue:

```
<template>
  <div class="home">
    <img alt="Vue logo" src="../assets/logo.png" />
    <HelloWorld msg="Welcome to Your Vue.js App" />
  </div>
</template>

<script>
// @ is an alias to /src
import HelloWorld from "@/components/HelloWorld.vue";

export default {
  name: "home",
  components: {
    HelloWorld
  }
};
</script>

</style>
//TODO
</style>
```

All .vue-files consist of 3 parts: a template, a script and a style parts. A template is actually an HTLM code wrapped up in a vue-template. A script is a TypeScript part with functions, components registering, may include instanciation of Vue.js class. A style part consists all of the CSS code for the page's lookfeel. In a small application all-in-one-file approach might look compact and readable, but if the functionality fot one component grows it could be difficult to find the parts of code and it would be easy to loose the whole overview. It is also possible to separate all the parts in different files, but still every step needs to be done manually, which is already preset in Angular project at the start. Vue.js seems very flexible in comparison to Angular and allows developers to make its own structure bu it has potential risks of making the large project not maintainable, bad structured and difficult to test.
The next step was to integrate Material design libraries into two projects. Angular material was easy to integrate and register the needed UI elements to the pages. After the installation of Angular Material Library a material.ts file was created and we registered all the needed global elements:
material.ts

```
import {
  MatButtonModule,
  MatCheckboxModule,
  MatMenuModule,
 ...
  MatInputModule
 }
 from '@angular/material';
```

```
 9   import { HttpClientModule } from "@angular/common/http";
10
11  import {NgModule} from '@angular/core';
12
13  @NgModule({
14    imports: [
15      HttpClientModule,
16      MatButtonModule,
17       ...
18        MatInputModule
19      ],
20    exports: [
21      HttpClientModule,
22      MatButtonModule,
23      MatCheckboxModule,
24       ...
25       MatInputModule
26      ],
27  })
28
29  export class MaterialModule{ }
```

After registering all Material GUI elements to the Angular project we added them in every component's .html-file like simple HTML tags bit wit "mat-" prefix, defined and explained in node_modules folders e.g. in planer.component.html. These GUI elements have already a well-designed look$feel and look almost exactly like in the final mockup which was done with Material design (See Figure 4.1. vs. Figure 2.6).
planer.component.html

```
1  <h2>Ausbildungsplan</h2>
2
3  <div class="grid-container">
4      <div class="grid-item item1">
5          <mat-card>
6              <button mat-menu-item>
7                  <mat-icon class="gray_icon">event_note</mat-icon>
8                  <span>Überblick</span></button>
9                  ...
10             <button mat-menu-item>
11                 <mat-icon class="green_icon">event_available</
                       mat-icon>
12                 <span>Abgeschlossene Kurse</span></button>
13         </mat-card>
14     </div>
15     <div class="grid-item item2">
16         <h5>Bevorstehende Kurse</h5>
17         <mat-list>
18             <button routerLink="/course" mat-list-item class="
                   raised_list_button md-btn">
19                 <mat-icon class="green_icon" mat-list-icon>
                       event_available</mat-icon>
20                 <h4 mat-line><b>Kursname</b></h4>
21                 <p mat-line>Beschreibung</p>
22             </button>
```

```
23          </mat-list>
24      </div>
25  </div>
26
27  <mat-tab-group mat-stretch-tabs headerPosition="below">
28      <mat-tab>
29          <ng-template mat-tab-label>
30              <mat-icon class="gray_icon">event_note</mat-icon>
31          </ng-template>
32      </mat-tab>
33      ...
34  </mat-tab-group>
```

Adding new components and GUI elements was easy and the results looked exactly as it was planned to be developed. It took less than a day to add, register all the needed GUI elements of almost all the micro services, which were planned to program for the whole project. The next step would be to connect all the elements with the database. But for the bachelor project and for the visualising the results we describe only the GUI prototyping part [4].

[4] See the whole fungtionality on the GitHub pages address, resize the browser with a minimum width to see the mobile view: `https://shulikaga.github.io/Bachelor_project_angular_part/`

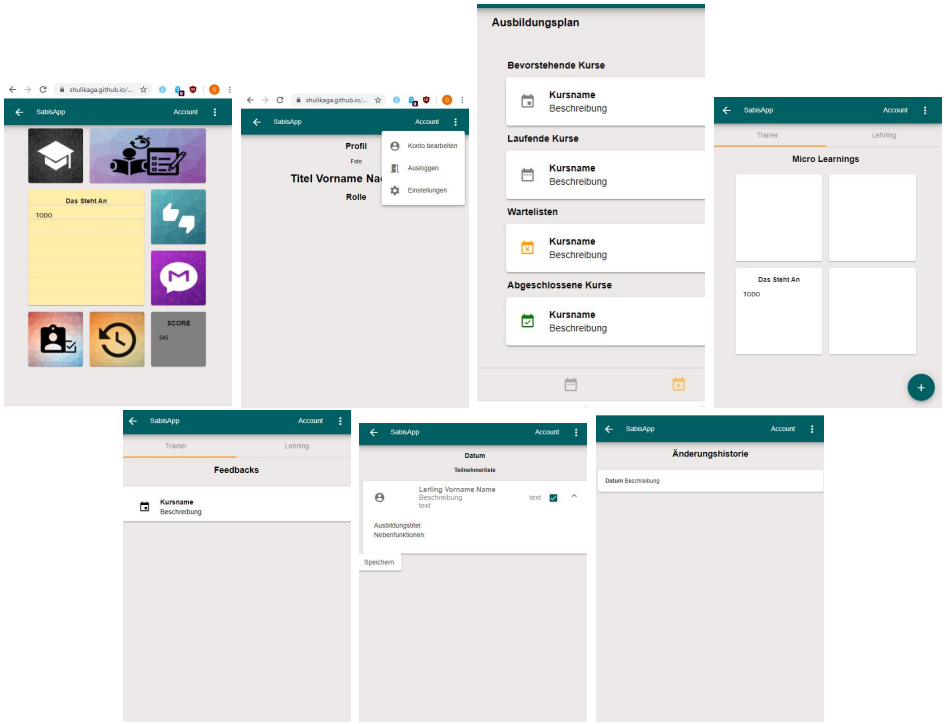


Figure 4.1: Angular Project for SabisApp

The integration of the Material design libraries into the the Vue.js was not easy. Firstly we tried to install and register the described above Material Components Vue library to the project, but it occured not compatible with the latest versions of JavaScript, Vue.js and other dependencies. We tried to improve this library to be compatible with our Vue.js project, but after trying and spending a lot of time we searched for other libraries that has the same functionality, have updates for the latest versions of Vue.js and have good documentation and community. The choice was to use Vue Material library. It appeared to be well integrated in the project. We registered all the needed elements and added them in the template parts of the components.The GUI elements int the template part have "md-" prefix, similar with Angular Material library.
Home.vue file:

```
1     <div class="md-layout-item md-medium-size-15 md-small-size-20
          md-xsmall-size-30"  @click="$router.push('/planer')">
2       <md-card md-with-hover >
3       <md-card-media-cover>
4         <md-card-media>
5           <img src="./../assets/home/planer.png" alt="Planer">
6         </md-card-media>
7       </md-card-media-cover>
8      </md-card>
9      </div>
10
11     <div class="md-layout-item  md-medium-size-40 md-small-size-50
```

```
          md-xsmall-size-60" @click="$router.push('/microlearnings
         ')">
12         <md-card md-with-hover >
13       <md-card-media-cover>
14         <md-card-media>
15           <img src="./../assets/home/microlearnings.png" alt="
                Microlearnings">
16         </md-card-media>
17       </md-card-media-cover>
18     </md-card>
19     </div>
20
21      <div class="md-layout-item md-medium-size-15 md-small-size-20
           md-xsmall-size-30" @click="$router.push('/events')">
22         <md-card md-with-hover >
23       <md-card-media-cover>
24         <md-card-media>
25           <img src="./../assets/home/next_courses.svg" alt="Events
                ">
26         </md-card-media>
27       </md-card-media-cover>
28     </md-card>
29     </div>
30
31
32     <div class="md-layout-item  md-medium-size-15 md-small-size-20
           md-xsmall-size-30" @click="$router.push('/feedbacks')">
33         <md-card md-with-hover >
34       <md-card-media-cover>
35         <md-card-media>
36           <img src="./../assets/home/feedbacks.png" alt="Feedbacs"
                >
37         </md-card-media>
38       </md-card-media-cover>
39     </md-card>
40     </div>
41
42     <div class="md-layout-item  md-medium-size-15 md-small-size-20
           md-xsmall-size-30" @click="$router.push('/messenger')">
43         <md-card md-with-hover >
44       <md-card-media-cover>
45         <md-card-media>
46           <img src="./../assets/home/messenger.png" alt="Messenger
                ">
47         </md-card-media>
48       </md-card-media-cover>
49     </md-card>
50     </div>
51
52       <div class="md-layout-item  md-medium-size-15
           md-small-size-20 md-xsmall-size-30" @click="$router.push
           ('/participants')">
53          <md-card md-with-hover >
54       <md-card-media-cover>
55         <md-card-media>
```

```
56            <img src="./../assets/home/participants_list.png" alt="
                  Participants">
57          </md-card-media>
58        </md-card-media-cover>
59      </md-card>
60        </div>
61
62      <div class="md-layout-item  md-medium-size-15 md-small-size-20
            md-xsmall-size-30" @click="$router.push('/logs')">
63        <md-card md-with-hover @click="onClick">
64        <md-card-media-cover>
65          <md-card-media>
66            <img src="./../assets/home/logs.png" alt="Logs">
67          </md-card-media>
68        </md-card-media-cover>
69      </md-card>
70      </div>
71    </div>
72
73    </div>
74  </template>
75
76  <script>
77  // @ is an alias to /src
78  import Notification from "@/components/Notification.vue";
79
80  import Vue from "vue";
81  import { MdCard } from 'vue-material/dist/components'
82  Vue.use(MdCard)
83
84  export default {
85    name: "home",
86    components: {
87      Notification
88    }
89  };
90  </script>
91
92  <style lang="scss" scoped>
93  .md-layout{
94    margin-left:auto;
95    margin-right:auto;
96  }
97  .md-layout-item {
98      height: 10px;
99      margin-bottom: 160px;
100     margin-left: 8px;
101     margin-right: 8px;
102   }
103 </style>
104 The .vue file wit a lot of HTML-/Material elements grow a lot and
        the other script and style parts were difficult to find.
105 After adding all the needed elements to the Home page with a
        dashboard elements and running the code to compare look&feel
        we found out that it did not look so exact as it was in the
```

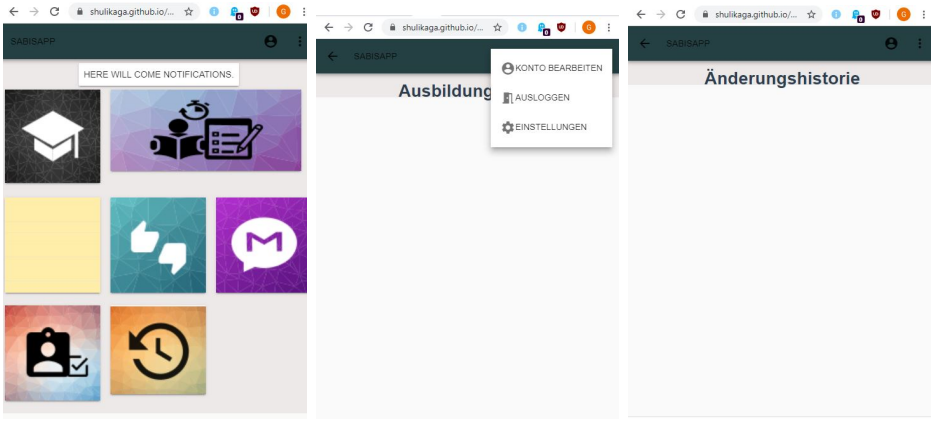


Figure 4.2: Vue.js Project for SabisApp

```
mockup. Vue Material uses different styles by default and it
was difficult to make the GUI look the same as in the mockup (
See Figure 4.2 vs. Figure 2.6).
```

After implementing these two projects with Angular and Vue.js we found out that Material design integration and well-structured project, quickly installed libraries and lookfeel were crucial to make the final choice of the front-end framework. With Angular it was easy and quick to install, register and all all the features needed and get even more functionality if the project grows, and it would definitely grow if the SabisApp project will be started. Vue.js project occured very flexible to add only needed features incrementally, but it took more time to add them manually. On the contrary it is also possible to exclude unnecessary features, files, components in the Angular project and it will be much easier as manually add every needed library. It is also interesting to observe, that for the GUI elements prototyping with Material design Angular appeared to be is a better choice. That is why we choose Angular for our future project.

# Chapter 5

# Results

This chapter presents the research results for the comparison of the front-end web-frameworks for the graphical user experience design and graphical user interface implementation. All of the presented goals in the introduction chapter were accomplished. We sum up here all the results of these objectives.

- 1.We gathered a lot of information about users, their work, needs, frustrations and expectations by using such research methods of UX Engineering as: user tests, contextual enquiry, online questionnaires. After analysing the results we built flow diagrams and described user groups and personas.

- 2. Next, we designed mockups, tested them with the users, got feedback and improved them to make the final prototype for the development of SabisApp GUI.

- 3. After the high-fidelity prototype with all needed features and animations was ready, we developed the architecture for the future app. All the requirements were summed up in a project requirements order document.

- 4. The project time planing was still not clear because of the discussion what web-framework to choose for the front-end part (Angular or Vue.js) and this topic needed more research. That is why we compared the most relevant technical aspects of both frameworks and summed up their advantages and disadvantages to choose the framework which fits best for the SabisApp project.

- 5. We also compared non-technical aspects of Angular and Vue.js and summed up the experiences of Vue.js and Angular developers.

- 6. After the literature research it was still not 100% clear what framework is better to choose because of the project's specific requirements (TypeScript preference, Material design integration, SPA). That is why the practical part was inevitable to

make a correct decision. We implement two projects with Angular 8 and Vue.js and compared the results.

- 7. After implementing several micro-services and basic GUI elements with Angular and Vue.js it was clear that Angular took less time to build a very well structured, clear, type safe source code with easy Material design elements integration and professional look&feel. Vue.js occured to be flexible to support TypeScript and Material design integration. But it was too flexible for building GUI elements and lookfeel took more time to make it look good.

Since we would like our application to be sustainable in the future, we preferred the technology with a larger community. Additionally, from the previous experiences of company's colleagues and web developers' blogs we know that a lot of time is wasted on choosing the right library/framework to use, often discovering its pitfalls half-way in implementation. This prolongs the development process and frustrates the programmer. Although Vue.js and Angular are very capable, reasons stated above made us select Angular. We chose Angular, mostly because of the type safe and well structured architecture, Material design elements support and a big number of Angular professionals community in the company and the web portals as www.stackoverflow.com and www.reddit.com/.

# List of Figures

# Acronyms

- GUI - Graphical User Interface
- PWA - Progressive Web Application
- SPA - Single Page Application
- UX - User Experience

# Bibliography


[1] Angular vs. React vs. Vue. Which is the best choice for 2019? https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847, Last accessed on 2019-10-30.

[2] Front-end frameworks - overview. https://2018.stateofjs.com/front-end-frameworks/overview/, Last accessed on 2019-10-30.

[3] Game of frameworks: JavaScript trends of 2019. https://hackernoon.com/game-of-frameworks-javascript-trends-of-2019-1a303fa3aaa7, Last accessed on 2019-10-30.

[4] Introduction to the Angular docs. https://angular.io/docs, Last accessed on 2020-03-30.

[5] Michał Sajnóg et al. Evan You. Vue.js documentation. https://vuejs.org/v2/guide/, Last accessed on 2019-10-30.

[6] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. Comparative evaluation of JavaScript frameworks. In Proceedings of the 21st International Conference on World Wide Web, pages 513–514. ACM, 2012.

[7] Abrahamsson Graziotin. Making sense out of a jungle of javascript frameworks – towards a practitioner-friendly comparative analysis. 2013.

[8] Rex Hartson and Pardha S Pyla. The UX Book: Process and guidelines for ensuring a quality user experience. Elsevier, 2012.

[9] DM Hutton. Clean code: A handbook of agile software craftsmanship. Kybernetes, 2009.

[10] Dhyanendra Jain, Ashu Jain, and Amit Kumar Pandey. Quantification of dynamic metrics for software maintainability prediction. 2018.

[11] Mukhammadjon Jalolov. Analyzing JavaScript frameworks and Dart for front-end development in building automation. 2018.

[12] Marin Kaluža, Krešimir Troskot, and Bernard Vukelić. Comparison of front-end frameworks for web applications development. Zbornik Veleučilišta u Rijeci, 6(1):261–282, 2018.

[13] Abrahamsson Pano, Graziotin. What leads developers towards the choice of a JavaScript framework? Researchgate, 2016.

[14] Paul Roberts. Guide to project management: getting it right and achieving lasting benefit. John Wiley & Sons, 2013.

[15] Brandon Satrom. Choosing the right JavaScript framework for your next web application. Vitbok RITM0012054. Progress Software Corporation, 2018.

[16] Frederik Schlemmer. Angular, React oder Vue.js? Eine Entscheidungshilfe. adesso. 2019-07-19 https://www.adesso.de/de/news/blog/angular-react-oder-vue-js-eine-entscheidungshilfe.js.

[17] Evan You. Vue. js introduction. https://vuejs.org/v2/guide/index.html, Last accessed on 2020-03-04.