OULU UNIVERSITY OF
APPLIED SCIENCES

Duy Anh Nguyen

UPDATING SARAKE SIGN INTERFACE

UPDATING SARAKE SIGN INTERFACE

Duy Anh Nguyen
Bachelor’s Thesis
Spring 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Nguyen Duy Anh
Title of the bachelor's thesis: Updating Sarake Sign interface
Supervisor: Veijo Väisänen
Term and year of completion: Spring 2020 Number of pages: 42 pages

---

This research was made to get knowledge about making a web application with VueJS and to support libraries based on making a new interface for Sarake Sign.

The thesis includes two main chapters. The first chapter refers to all knowledges which should be learn before developing a Sarake Sign product. The next chapter mentions as a working diary of an author in four months on developing the Sign product.

All information in the thesis is not a personal research but also summarized knowledge from documentation of Vue expert. The knowledge can be extended when the technology has to be changed and updated to a new version.

Keywords: JavaScript, VueJS, ES6

PREFACE

This thesis was written as a final project of studying Bachelor of Engineering, Information Technology. I would like to express my gratefulness for my company – Sarake Oy and Veijo Väisänen. He has supported me to discuss with Sarake Oy to have an agreement for making this project as a thesis. I would like to thank Kaija Posio for all the recommendation of writing and academic documentation.

Helsinki, 25.5.2020
Nguyen Duy Anh

CONTENTS

VOCABULARY

API	Application Program Interface
CEO	Chief Executive Officer
CSS	Cascading Style Sheet
CTO	Chief Technology Officer
ES2015 / ES 6	ECMAScript 2015 / ECMAScript 6
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LESS	Leaner Style Sheets
NPM	Node Package Manager
PDF	Portable Document Format
REST	Representational State Transfer
SCSS	Syntactically Awesome Style Sheets
SVN	Apache Subversion
UI	User Interface
UX	User Experience
VPN	Virtual Private Network
W3C	World Wide Web Consortium

1 INTRODUCTION

Sarake Oy was founded in 2016 to help modernize information management. From the beginning, the company’s mission has been to support business by providing a solution to make the management information easier to access.

Sarake Sign is a part of Sarake’s product ecosystem. With the growing of information technology, business can handle actions around the world without traveling. Sarake Sign is an electronic service that helps the user save time and money. There are several advantages when using Sarake Sign to replace a normal sign process:

- Sign speeds up and streamlines a signing process and provides up-to-date information on the status of signing process.
- An application has a friendly user interface with an unlimited number of signatures. Moreover, the application can be customized based on the customer’s requirements.
- Sign provides a secure and legally binding way to sign all the documents.
- With a convenience of web application, the user can sign documents anywhere and at any time.

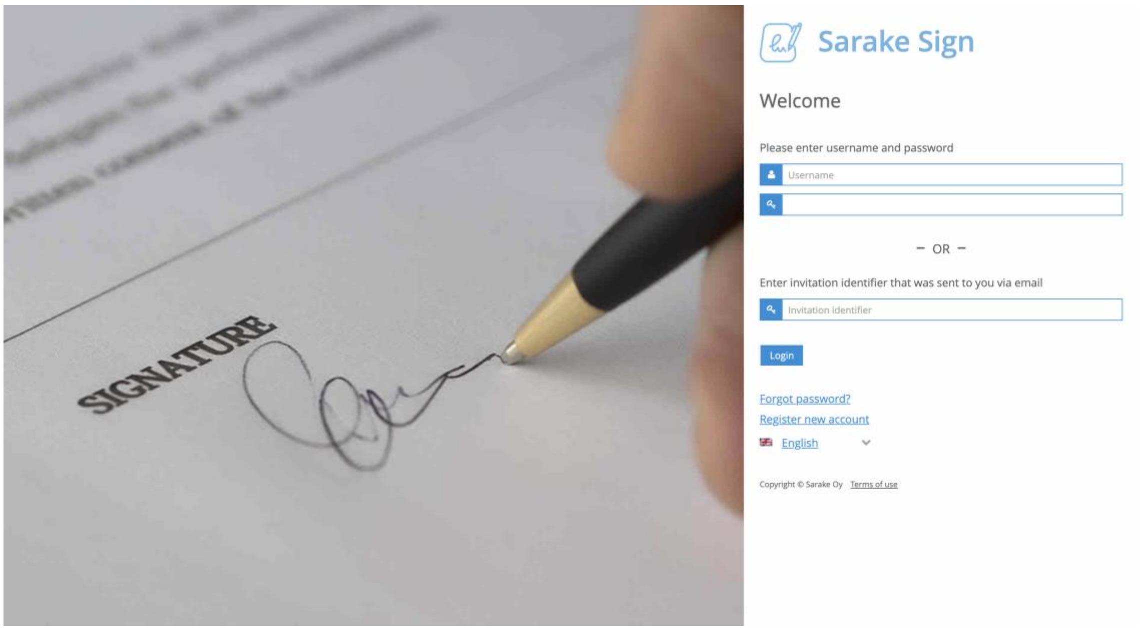


FIGURE 1. A login screen of Sarake Sign

Sarake Sign was published in 2017. A frontend site of an application was developed by Vaadin – a Java web framework. The Figure 1 shows a login page of Sign which is developed by Vaadin. With a growing of web technology, especially of web framework, such as VueJS and ReactJS, Sarake needs to improve a product to help the user have a better experience and help the developer the maintain product easier.

After investigating through all technology, the CTO of the company decided to choose VueJS to develop a frontend field and use GraphQL as an API design architecture replacement to REST. This project was made to develop a new version for Sarake Sign with the technologies.

A diary thesis provides a process to develop a web application using VueJS and other modules. The developing path was finished by the creator of the thesis in collaboration with Sarake’s developers.

2 BASIC TECHNOLOGY

2.1 HTML

2.1.1 Definition

HTML stands for the Hypertext Markup Language (4.). The markup helps a browser to show content, pictures and different types of multimedia on a website page. HTML is not a programming language, meaning that it cannot create dynamic functions. The function of HTML is similar as a text editor, used to layout and format web pages.

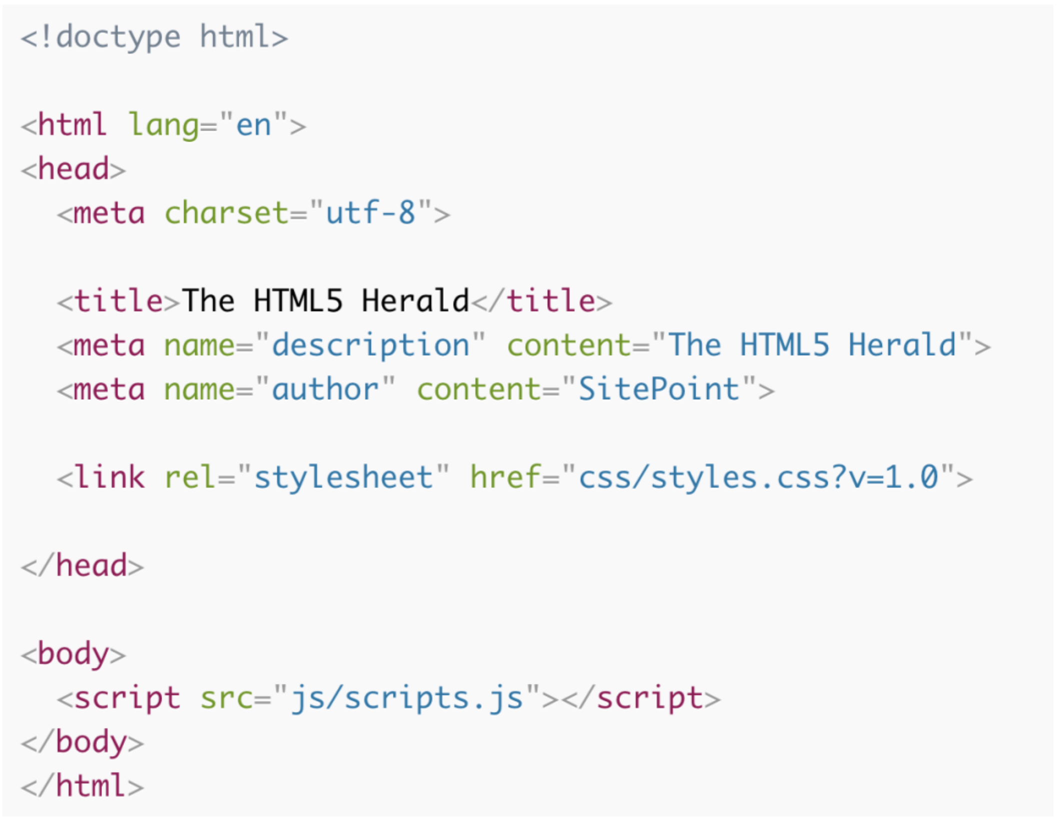
With a quickly gaining popularity, HTML is considered as the standard of a website. Constructions and structure of HTML are operated and developed by World Wide Web Consortium (W3C). HTML5 is the latest version of HTML (4.).

2.1.2 How HTML works

The HTML file can be recognized with an ending .html or .htm. The user sees the file by opening web browser, such as Google Chrome, Safari or Mozilla Firefox. The browser reads HTML files and publishes to the Internet.

A web page contains many HTML websites, such as homepage, about page, contact page. Each page needs separate HTML pages.

Each HTML page contains a set of tags, which can be seen as building blocks of a webpage. The tags form a tree structure including sections, paragraphs, headings and other content blocks. The Figure 2 demonstrates a structure of the basic HTML.



```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">

  <title>The HTML5 Herald</title>
  <meta name="description" content="The HTML5 Herald">
  <meta name="author" content="SitePoint">

  <link rel="stylesheet" href="css/styles.css?v=1.0">

</head>

<body>
  <script src="js/scripts.js"></script>
</body>
</html>
```

FIGURE 2. A basic template of HTML (1.)

2.2 JavaScript

JavaScript is known as a lightweight interpreted scripting language or programming language with object-oriented capabilities that enable to execute a complex task on site pages (2.). It is most commonly used as part of web pages whose implementation allows Client-Side scripts to interact with users and create dynamic web pages.

JavaScript was first known as Mocha, and later as LiveScript, but Netscape changed its name to JavaScript because of its popularity as a phenomenon of Java at the time (3.).

2.3 Node package manager

The Node package manager or NPM is a tool to create and manage JavaScript programming libraries for NodeJS (11.). NPM provides two main functions:

- An online repository for packages/ module.

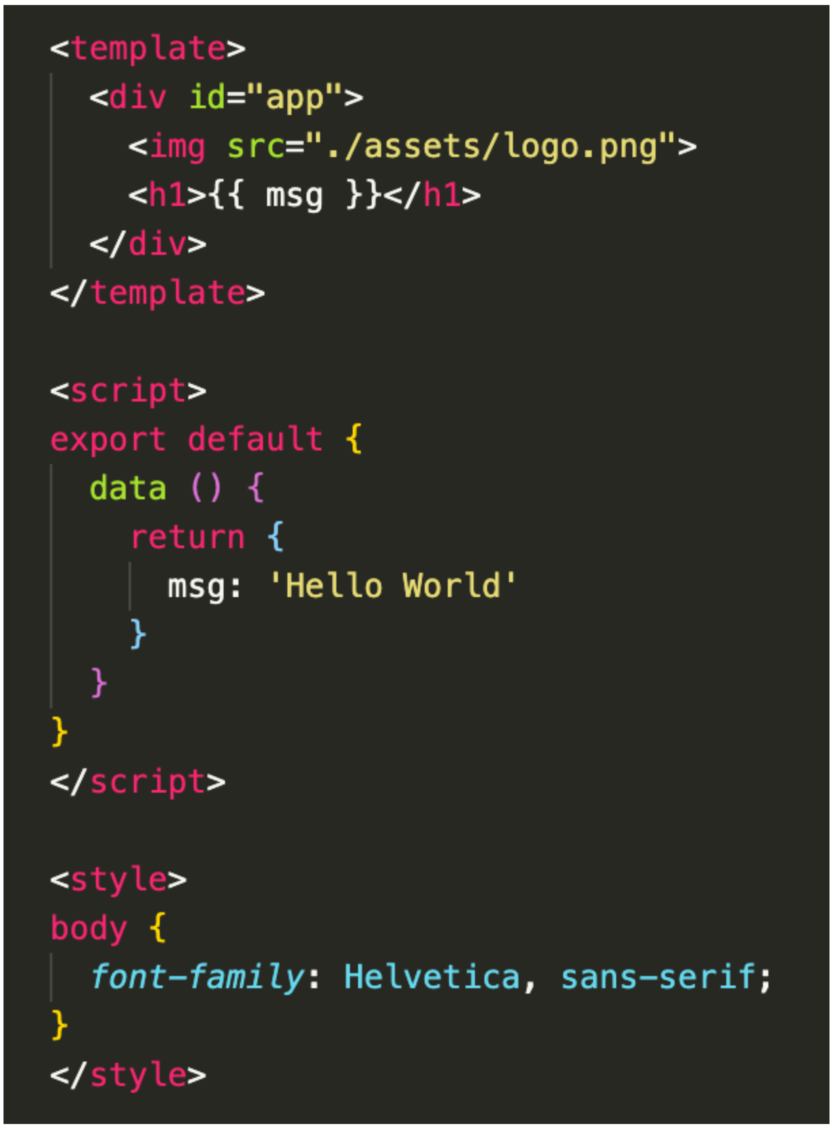
- Management of modules and versions in projects is simpler, easier and saving time.

2.4 ES6

ECMAScript 6 (also known as ES6, ES2015) is a popular version of the ECMAScript standard. ES6 is an advanced set of JavaScript techniques and it is a benchmark to follow (9.). For example, when a team has many members working on a project where each user uses a different code style, then the project will turn out. ES6 is also a standard for the framework from which to develop. In addition, the standard also helps programmers to code optimum and compact.

2.5 VueJS

Instead of the full-feature framework providing everything needed to build an app in a single framework, a progressive framework splits into different sub-components (8.). The programmer can choose a suitable component to use. VueJS is one of the progressive frameworks. The programmer can use the core of VueJS to create view and integrates with separate tools or libraries for different purposes. Figure 3 demonstrates a structure of Vue file. The Vue file has three main parts: template, script and style. The template part is a field to generate a structure of view. It is edited by HTML tags. The script part is used to handle all the actions of Vue. Moreover, the programmer can use the part to import component and declare data. The last part is a style tag which is used to decorate a style by CSS, SCSS or LESS based on the project requirement.



```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <h1>{{ msg }}</h1>
  </div>
</template>

<script>
export default {
  data () {
    return {
      msg: 'Hello World'
    }
  }
}
</script>

<style>
body {
  font-family: Helvetica, sans-serif;
}
</style>
```

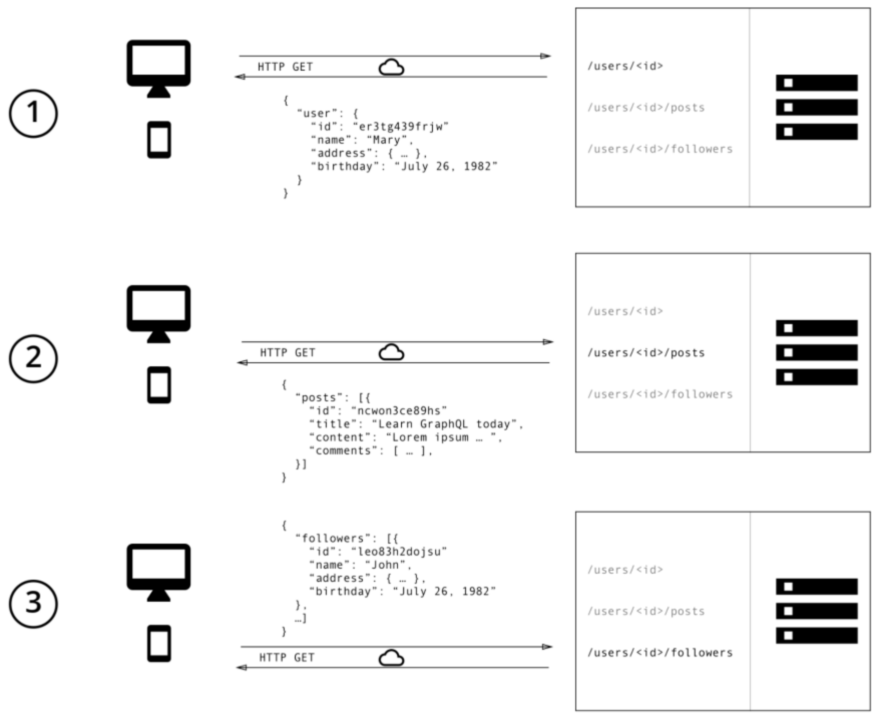
FIGURE 3. A basic template of Vue

2.6 GraphQL

GraphQL is a syntax that describes how to request data and is often used to load data from a server to the client (10.). One of the most common problems of REST is redundancy as well as data shortage. A method which helps the client get data by REST is call endpoints and server returns data with a fixed structure.

The redundancy occurs when the client has to fetch more data than it actually needs. For example, a screen needs to render a list of users' names. An application using REST API need access an endpoint */users* and get a JSON object of user’s data, such as: name, ID, gender, address. The object which is sent from a server contains unused data.

Another problem with REST is the lack of data when the server does not provide enough data needed in an endpoint. The client has to call additional endpoints to retrieve the missing data. For example, a screen needs to render user’s information including personal information, posts and followers. As the Figure 4 below shows, an application creates three GET requests to server to retrieve data: */users/<id>, /users/<id>/posts* and */users/<id>/followers*.



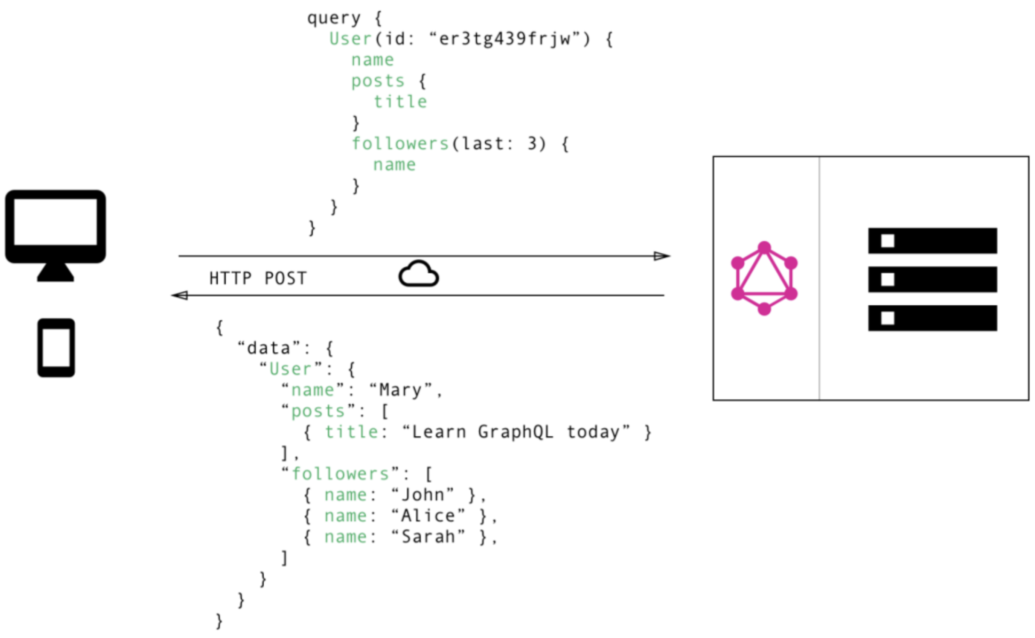
```
{
  "user": {
    "id": "er3tg439frjw"
    "name": "Mary",
    "address": { … },
    "birthday": "July 26, 1982"
  }
}
```

```
{
  "posts": [{
    "id": "ncwon3ce89hs"
    "title": "Learn GraphQL today",
    "content": "Lorem ipsum … ",
    "comments": [ … ],
  }]
}
```

```
{
  "followers": [{
    "id": "leo83h2dojsu"
    "name": "John",
    "address": { … },
    "birthday": "July 26, 1982"
  },
  …]
}
```

FIGURE 4. A REST API example (7.)

The Figure 5 is an example of usage GraphQL. With GraphQL, the client just needs to send a query to GraphQL server where necessary data is identified. Then server returns a JSON object containing only the data the client has requested.



```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

FIGURE 5. A GraphQL example (7.)

3 CREATE SIGN PROJECT

3.1 Requirement

At the first stage of development, two requirements of Sarake for a developer who takes part in Sign are similar to VueJS and create views based on design. A duration for the first stage is four months and a Sign application needs to finish an authentication login.

Login is a necessary requirement to access an application. A user opens Sarake Sign page to enter an email and a password. Then the server will authorize and let the user use an application.

After finished creating views and the login section, a number of components are created. The developer can build a Storybook to help others to understand about attributes, props and states of components which are used in the application. Storybook is a development environment for UI components (5.). The Figure 6 shows a user interface of Storybook. The Storybook refers two styles of button component, such as text and emoji. Based on styles of the component, the developer can have an overview about the appearance and the function of the components to consider using in a suitable case. The components are displayed at a specific port and not affected to the development project.

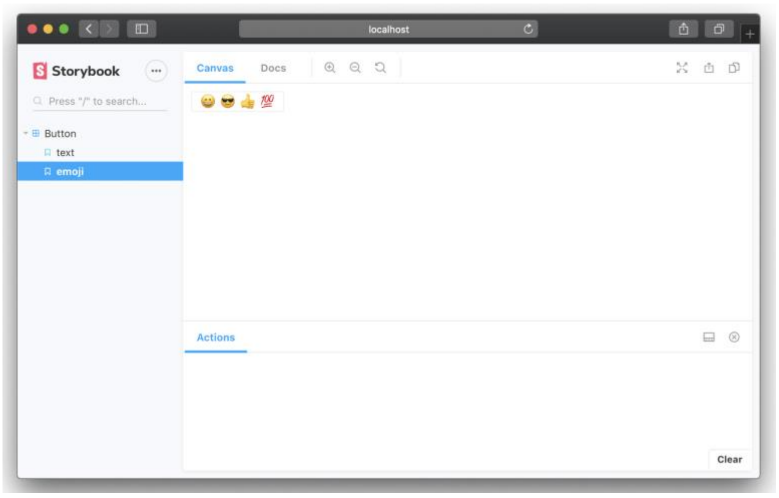


FIGURE 6. A Storybook example (5.)

3.2 Implementation

3.2.1 Working environment

Sarake provides a license for developer using IntelliJ IDEA – an IDE was created by JetBrains. The IDE has built necessary tools to help the developer code easier and faster. The Figure 7 shows a structure of Sign project which is opened by IntelliJ IDEA. Because of privacy term, VPN is set up for the developer to connect a Sarake network. Adobe XD was used to read UI design and VueJS was installed to serve a development part.

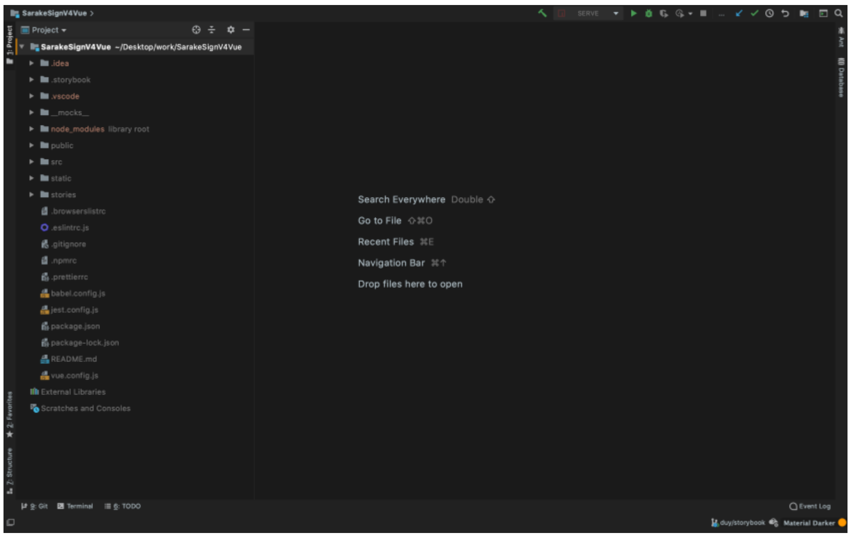
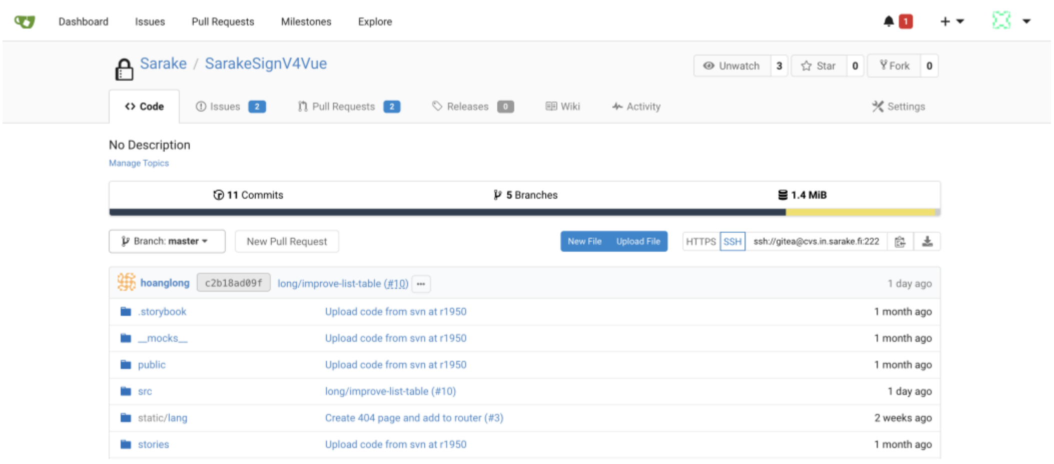


FIGURE 7. IntelliJ IDEA workspace

Gitea was used to replace GitHub in storing and managing a software version control. It was installed inside the Sarake server and can connect through VPN. The Figure 8 shows a repository of Sign which is stored in Gitea. The product has a friendly graphical user interface and gets the functions similar to Github.



No Description

Manage Topics

11 Commits | 5 Branches | 1.4 MiB

ssh://gitea@cvs.in.sarake.fi:222

hoanglong c2b18ad09f long/improve-list-table (#10) 1 day ago

.storybook	Upload code from svn at r1950	1 month ago
__mocks__	Upload code from svn at r1950	1 month ago
public	Upload code from svn at r1950	1 month ago
src	long/improve-list-table (#10)	1 day ago
static/lang	Create 404 page and add to router (#3)	2 weeks ago
stories	Upload code from svn at r1950	1 month ago

FIGURE 8. A Sarake Sign repository

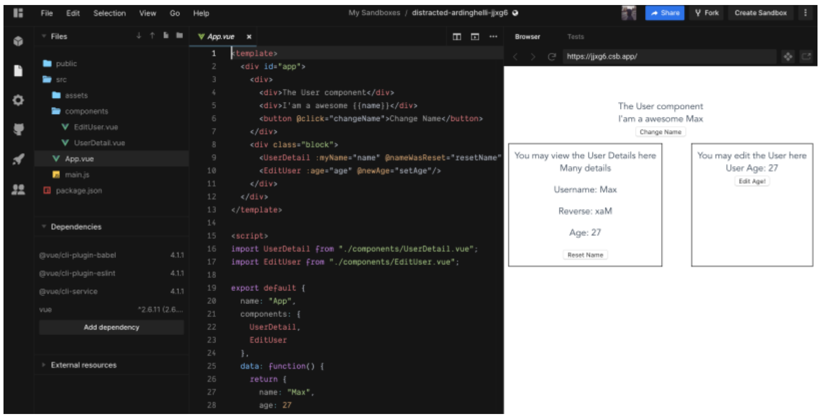
4 DIARY ENTRIES

4.1 Week 1

At the first week, the main task was to understand system engineering and software development at Sarake. Tanja Räikkönen – CEO of Sarake had a presentation to introduce the company and a target to achieve in 2020. Updating Sarake Sign is one of main targets of the company this year. After the presentation, a main task for the first week was to install all necessary tools and software to serve the development process. A setting includes setting an external monitor, configurating a personal shortcut for MacBook and installing tools which are needed for developing, such as IntelliJ IDEA, Adobe XD, Google Chrome, Vue.js devtools and ChromeiQL extensions. Moreover, Sarake provides a Microsoft account and an authorization account to connect the Sarake internal system. All the configurations were done at the end of the day.

On Tuesday, a task which needed to do first was trying to use the current version of Sarake Sign. The experience helped to understand which functions a product serves and to have an overview of system. Then researching use cases documentation helps to prepare for developing the new version of Sign.

Based on the requirement of Sign developing, learning VueJS could be seen as a primary because this framework was decided to be developed for a frontend side. Reading a document of VueJS gained a large of knowledge about how to install and handle components, actions in VueJS. Beside of reading documentation, developing some components with basic functions could help familiar and gain experience with the framework. The Figure 9 shows a code and view of App which was developed by VueJS. An App view contains two components, such as *UserDetail* and *EditUser*. The components were displayed inside the view to display and handle values.

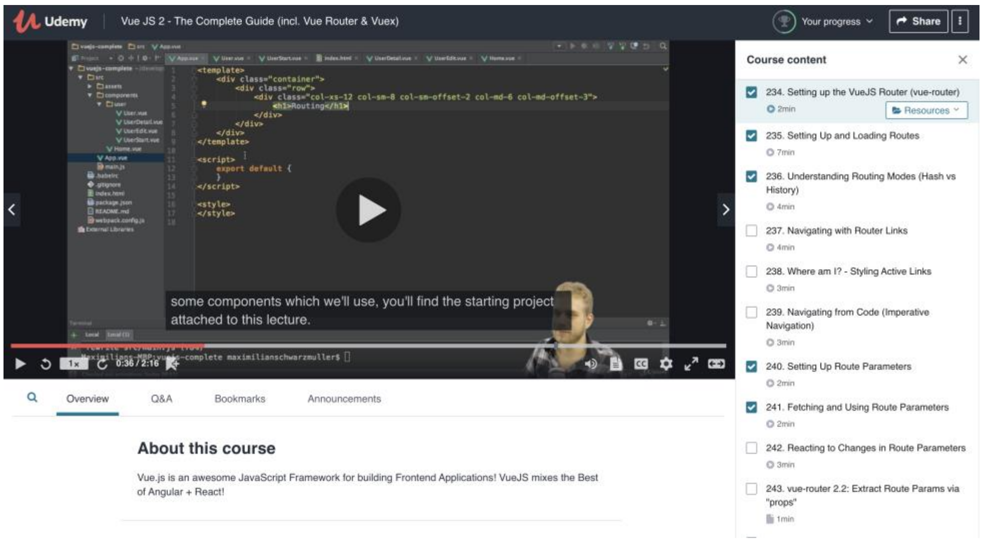


```
<template>
  <div id="app">
    <div>
      <div>The User component</div>
      <div>I'am a awesome {{name}}</div>
      <button @click="changeName">Change Name</button>
    </div>
    <div class="block">
      <UserDetail :myName="name" @nameWasReset="resetName"
      <EditUser :age="age" @newAge="setAge"/>
    </div>
  </div>
</template>

<script>
import UserDetail from "./components/UserDetail.vue";
import EditUser from "./components/EditUser.vue";

export default {
  name: "App",
  components: {
    UserDetail,
    EditUser
  },
  data: function() {
    return {
      name: "Max",
      age: 27
```

FIGURE 9. Practice with VueJS in codesandbox.io

4.2 Week 2

According to a developer's request, a company purchased a VueJS in udemy.com to help the developer to have a deeper knowledge with the framework. The Figure 10 is an overview of VueJS course at udemy.com. The website has two main contents: A tutorial video and a course's curriculum. Following a curriculum of a course, the developer should use a Vue router and a Vuex for managing the Vue application. A learning target for this week was have enough knowledge about the Vue router to handle routing between components inside the application.



```
<template>
    <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-8 col-sm-offset-2 col-md-6 col-md-offset-3">
                <h1>Routing</h1>
            </div>
        </div>
    </div>
</template>

<script>
    export default {
    }
</script>

<style>
</style>
```

some components which we'll use, you'll find the starting project attached to this lecture.

About this course

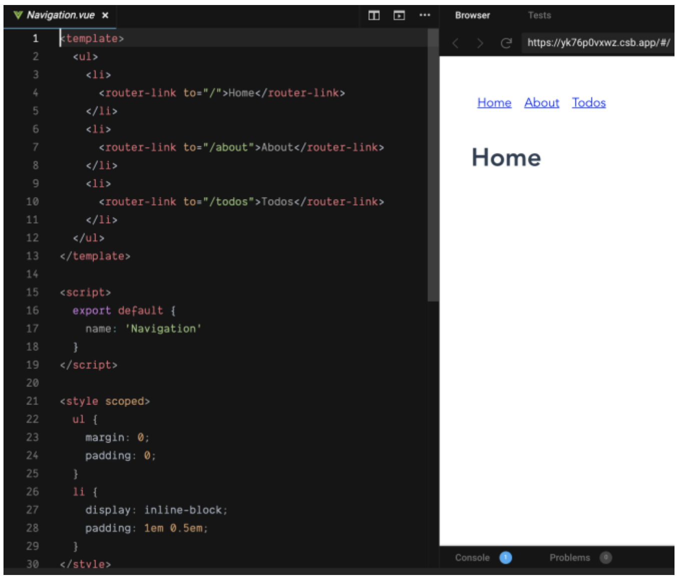
Vue.js is an awesome JavaScript Framework for building Frontend Applications! VueJS mixes the Best of Angular + React!

Course content

- [✓] 234. Setting up the VueJS Router (vue-router) — 2min
- [✓] 235. Setting Up and Loading Routes — 7min
- [✓] 236. Understanding Routing Modes (Hash vs History) — 4min
- [ ] 237. Navigating with Router Links — 4min
- [ ] 238. Where am I? - Styling Active Links — 3min
- [ ] 239. Navigating from Code (Imperative Navigation) — 3min
- [✓] 240. Setting Up Route Parameters — 2min
- [✓] 241. Fetching and Using Route Parameters — 2min
- [ ] 242. Reacting to Changes in Route Parameters — 3min
- [ ] 243. vue-router 2.2: Extract Route Params via "props" — 1min

FIGURE 10. Learn VueJS at Udemy's course

Learning and practicing with a mentor at Udemy’s course gained a knowledge of how the Vue router works and handles the package inside the Vue application to routing between components. The Figure 11 shows an example of using the Vue router to create a link to redirect to another component. A router-link tag was used to create a tag in HTML. When the user clicks to this link, an application will render a component that has a link at “to” keyword. For example, when the user clicks to a router link to */about*, an application will render a component *About.*



```
<template>
  <ul>
    <li>
      <router-link to="/">Home</router-link>
    </li>
    <li>
      <router-link to="/about">About</router-link>
    </li>
    <li>
      <router-link to="/todos">Todos</router-link>
    </li>
  </ul>
</template>

<script>
  export default {
    name: 'Navigation'
  }
</script>

<style scoped>
  ul {
    margin: 0;
    padding: 0;
  }
  li {
    display: inline-block;
    padding: 1em 0.5em;
  }
</style>
```

FIGURE 11. Practice Vue Router in codesandbox website

4.3 Week 3

In continuation of the learning process into the support packages for VueJS, week 3 started with a learning of Vuex. Vuex is a package for managing state at the Vue application, such as Redux for React. By using a global store for all components in the application, Vuex brings instant updates for data to components.

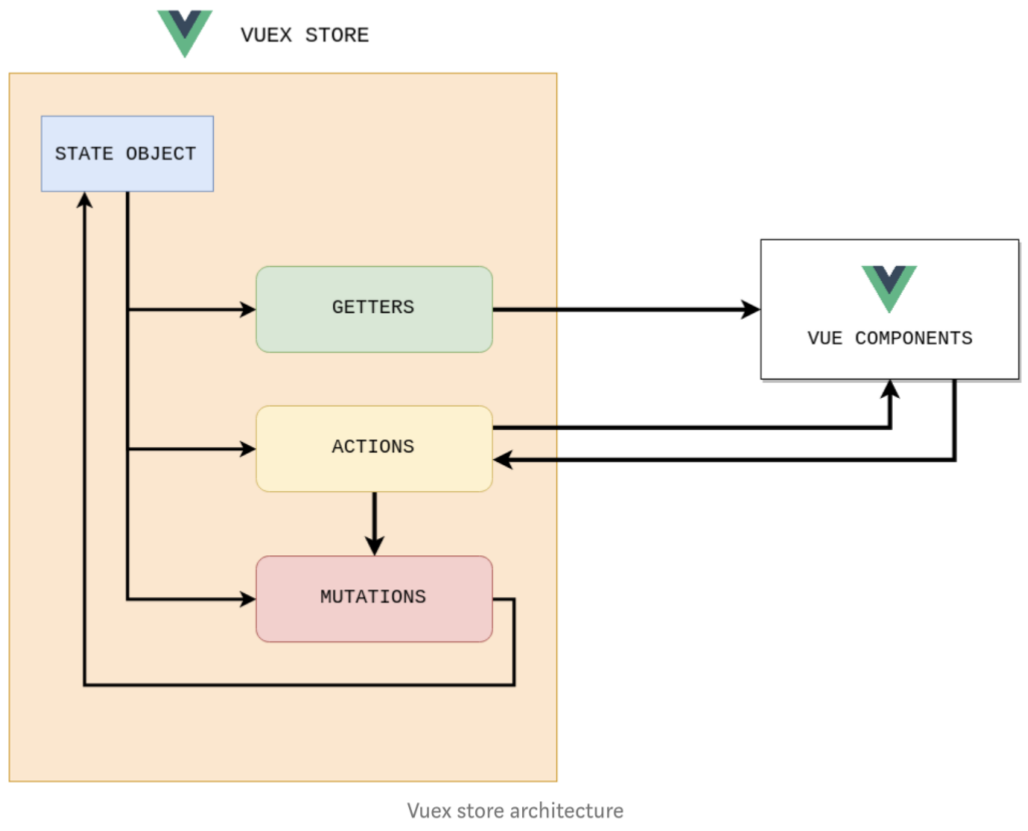
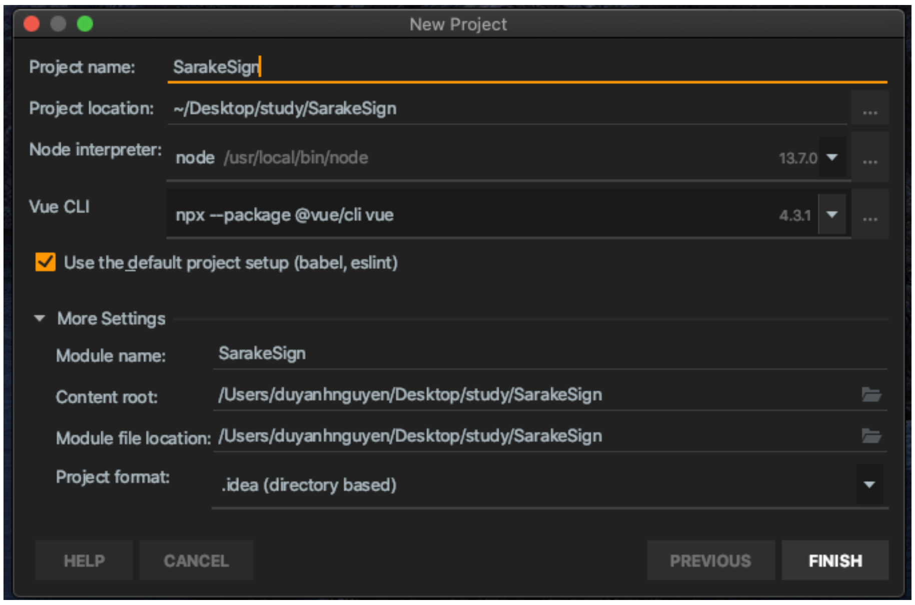


FIGURE 12. An architecture of Vuex (6.)

As the Figure 12 above shows, Vuex has four main parts: state, getters, actions and mutations. The state is used to store all data inside. Getters is used to filter data based on state. The function of mutations is a place to edit data of state by using a commit. Actions is used to execute mutation commit. When a Vue application has a complicated structure, a store of Vuex will become bigger. Vuex allows subdividing the store into smaller parts. Each module contains separate state, getters, mutations and actions. Moreover, Vuex allows modules to be nested. Choosing Vuex to manage data in the Sign product was the best solution at that time.

4.4 Week 4

After having enough knowledge to develop application with VueJS and supported packages, week 4 was a day to start installing and configuring the Sign project. First, the Sign project was installed through IntelliJ IDEA with a default configuration. The Figure 13 shows a window to create a Vue project by the IDE.


New Project

Project name: SarakeSign

Project location: ~/Desktop/study/SarakeSign

Node interpreter: node /usr/local/bin/node 13.7.0

Vue CLI: npx --package @vue/cli vue 4.3.1

[✓] Use the default project setup (babel, eslint)

More Settings

Module name: SarakeSign

Content root: /Users/duyanhnguyen/Desktop/study/SarakeSign

Module file location: /Users/duyanhnguyen/Desktop/study/SarakeSign

Project format: .idea (directory based)

FIGURE 13. Install VueJS project by IntelliJ IDEA

A Sign project is created still missing support packages which are needed to install manually through a command *npm install*. After installation, a *package.json* file will refer to all of packages that are used in Vue project.

A meeting was held this week to present a new design of Sarake Sign. Essi Jäppinen – a designer of company had a speech of UI rules for the Sarake product and started to show a screen for login page of Sign. There are four screens which are available to develop, such as a login, invitation, select identity and confirmation. During presentation, Essi often asked questions and gave advices for improving UI/UX. At the end of meeting, the developer team decided that the duration to develop these screens is two weeks.

First, all of colors that were used in Sign project were set at a *variables.scss file.* The colors were selected based on UI rules for the Sign project. A meeting was established by members of developer team to make a decision of the way to develop components. A result of the meeting was to develop first a small component which can be reused in views and to develop for view later. A list of components was created to sort out which should be done first.

4.5 Week 5

With the list of components which were created last week, a language tab and button were on the first priority. A *SignButton* component contains four kinds of colors, such as default, basic, disabled and success. The Figure 14 shows a style of the default and the basic button.

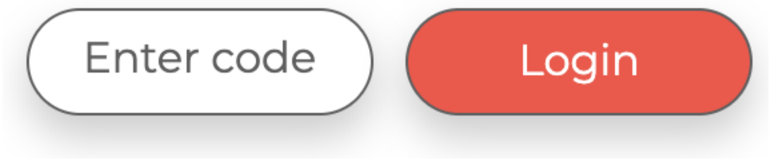


FIGURE 14. SignButton component

The next step was to develop a *LanguageTab.vue* component. The Figure 15 shows a user interface of the component. The component has a list of languages the application support, displayed as a dropdown and can be selected by a click action. When the user hovers, an item should be changing background and text color to highlight. To create a list in component, VueJS is using a loop with a *v-for* keywork to render a list of languages.

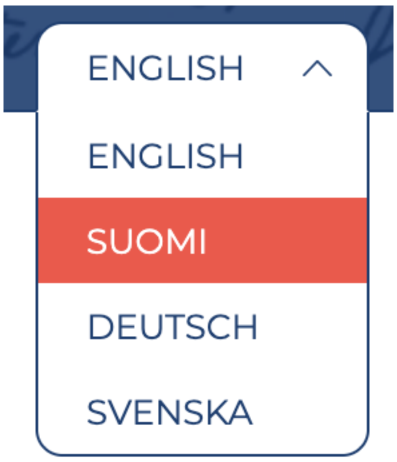
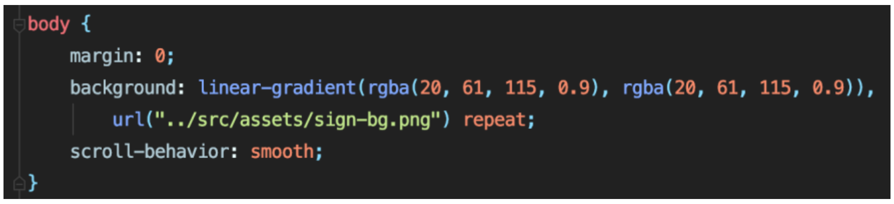


FIGURE 15. LanguageTab component when open and hover item

A new task had been created to continue a development. Before starting to develop views, a background for an application should be done first. A background is a combination between an image and a color. The Figure 16 shows SCSS code for styling the background. A *linear-gradient* function created a background color and *URL* was to generate the image.



```scss
body {
    margin: 0;
    background: linear-gradient(rgba(20, 61, 115, 0.9), rgba(20, 61, 115, 0.9)),
        url("../src/assets/sign-bg.png") repeat;
    scroll-behavior: smooth;
}
```

FIGURE 16. A style of an application background

A login view stared to do after finishing a configuration background's style. When looking at a design, there are five elements: images, text, input, link and button. An input should have a condition to add *error* class which had a red border when user has wrong value of username or password. The Figure 17 is a user interface of the login view. The view was finished on Friday.

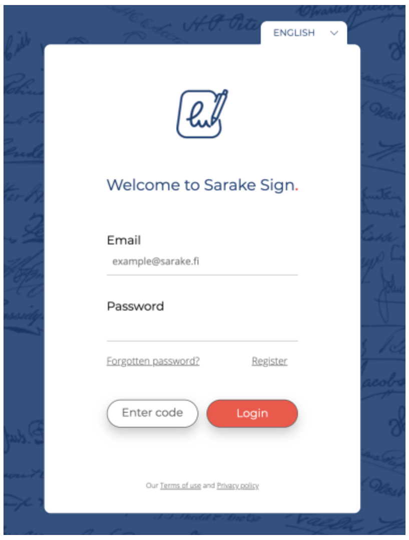
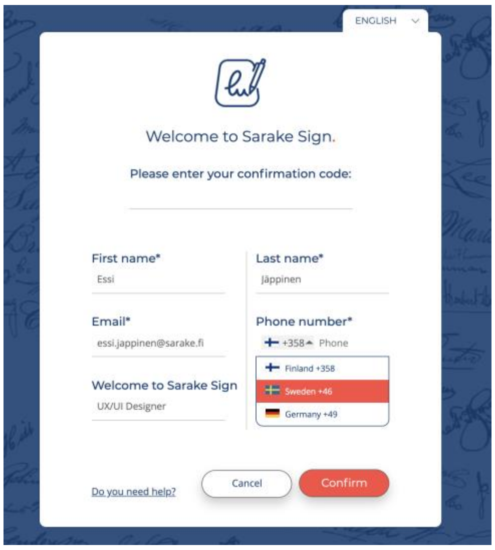


FIGURE 17. A login page

4.6 Week 6

A task for week 6 was to continue developing three views, such as invitation, select identify and confirmation. Each view was estimated to be finished in two days. The most difficult part of invitation part is a list of international phone inputs. An input has a list of countries code which an application supports. The list contains a country's flag, phone code and name. It was done by using a *vue-tel-input* package with a customization of style to be suitable with an application. The Figure 18 shows a user interface of the invitation page.


ENGLISH

Welcome to Sarake Sign.

Please enter your confirmation code:

First name*
Essi

Last name*
Jäppinen

Email*
essi.jappinen@sarake.fi

Phone number*
+358 Phone

Finland +358
Sweden +46
Germany +49

Welcome to Sarake Sign
UX/UI Designer

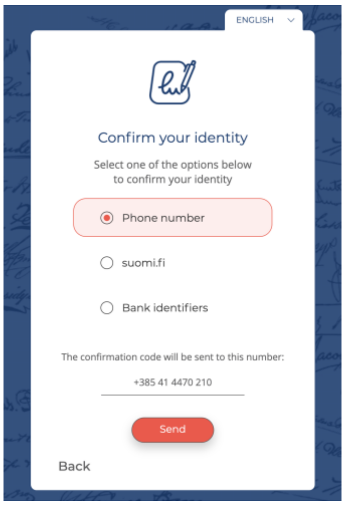
Do you need help?

Cancel

Confirm

FIGURE 18. An invitation page

A select identify view was created and developed shortly because a simple structure. The view allows the user to select an option to identify an account. There are three methods of identity in the application: phone, suomi.fi and identifying by bank account. Each option will give a separate action. For example, when the user click on the phone identify, a view displays an input to fill a phone number and change to a confirmation page. The Figure 19 shows the view when the phone number options was selected.


ENGLISH

Confirm your identity

Select one of the options below
to confirm your identity

[✓ Phone number]

suomi.fi

Bank identifiers

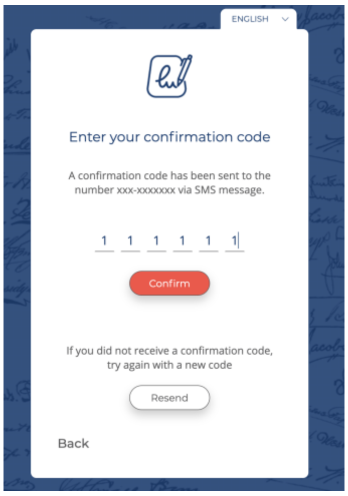
The confirmation code will be sent to this number:
+385 41 4470 210

Send

Back

FIGURE 19. A phone identify option

A confirmation was the last view which should be completed this week. The Figure 20 shows an overview of the confirmation page. After getting code from the mobile phone, the user need to fill it in confirmation view. An input was created by *vue-otp-input* package. This package helps to create an input value with the same style as a token input.



Enter your confirmation code

A confirmation code has been sent to the number xxx-xxxxxxx via SMS message.

1 1 1 1 1 1

If you did not receive a confirmation code, try again with a new code

FIGURE 20. A confirmation page

4.7 Week 7

On Monday, a meeting was held to analytic views which are created in two weeks. Essi felt that all views were developed with a same size of designs. After giving a feedback, Essi started to show a design for a new screen. As the Figure 21 below shows, an edit page has 4 main elements: basic information, add people, signature layout and send a request. The main task for this week was to develop this page. An author of the thesis took part in developing three components: basic information, add people and send a request. Long – a frontend developer had a task to finish the signature layout.

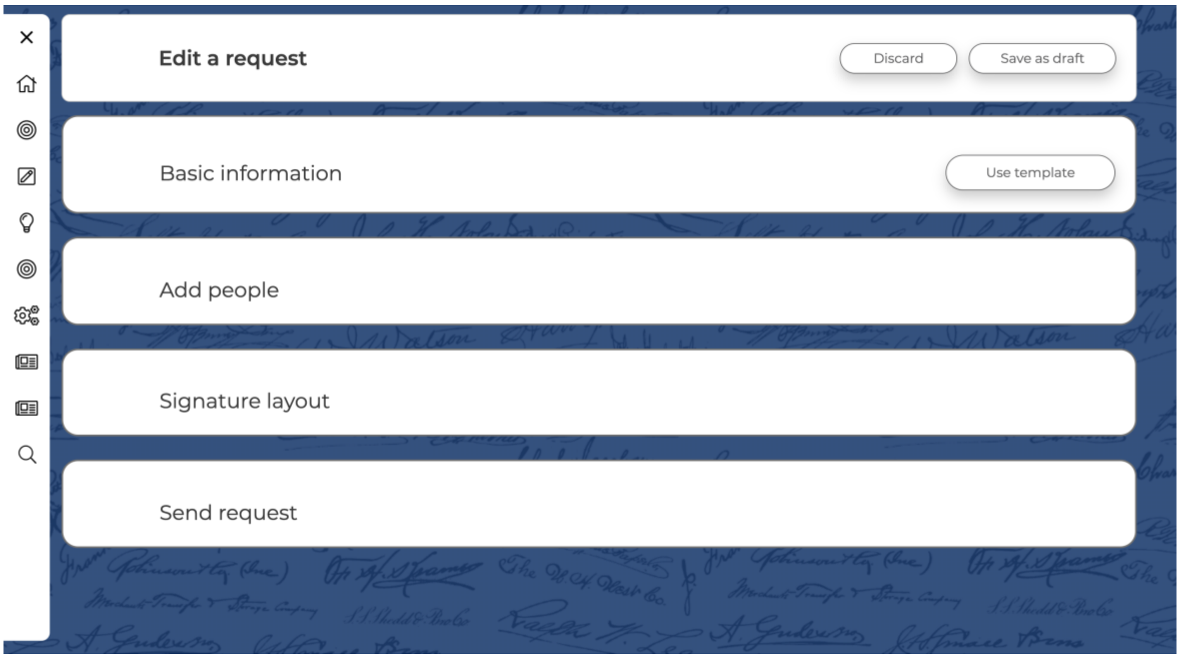
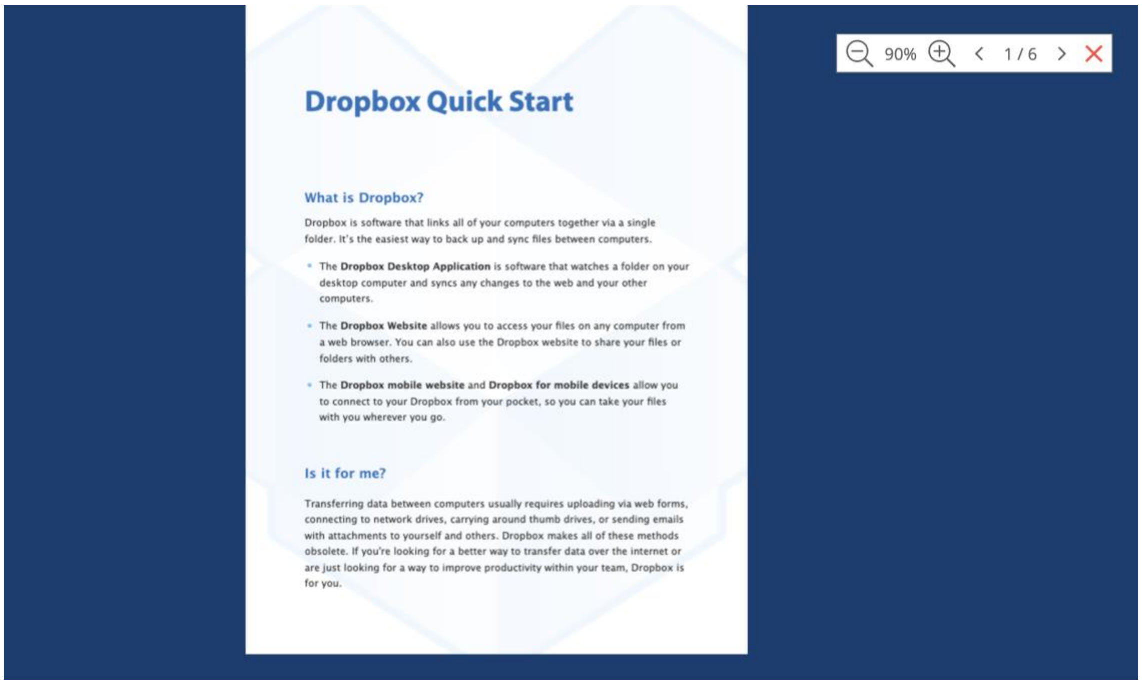


FIGURE 21. An overview of edit page

Basic information contains inputs to edit information and upload documents for sign request. Each document which is uploaded to the server can be previewed by clicking a preview icon. An order of documents can be arranged by dragging between documents. An option selected to render a document is *pdf.js* – a JavaScript library helped to render a PDF file using HTML5 canvas.

Because of a difficulty on approaching pdf.js, Tuuka – a team lead of Sarake had an advice to review code from another Sarake project which is using pdf.js to have a sight of how this library works and to handle it in the VueJS application. A *PDFViewer component* was created using the library to render a pdf file. As shown in Figure 22, the component has basic functions of preview, such as zooming, going to page or scrolling to navigating between pages. It had been tested and fixed to work fine when using a mouse or a trackpad.



Dropbox Quick Start

What is Dropbox?

Dropbox is software that links all of your computers together via a single folder. It's the easiest way to back up and sync files between computers.

- The **Dropbox Desktop Application** is software that watches a folder on your desktop computer and syncs any changes to the web and your other computers.
- The **Dropbox Website** allows you to access your files on any computer from a web browser. You can also use the Dropbox website to share your files or folders with others.
- The **Dropbox mobile website** and **Dropbox for mobile devices** allow you to connect to your Dropbox from your pocket, so you can take your files with you wherever you go.

Is it for me?

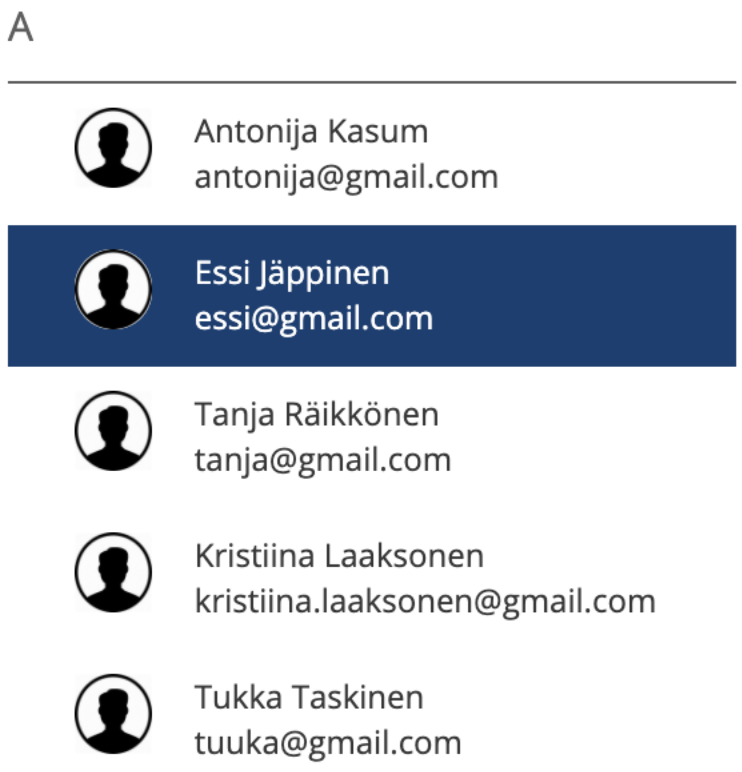
Transferring data between computers usually requires uploading via web forms, connecting to network drives, carrying around thumb drives, or sending emails with attachments to yourself and others. Dropbox makes all of these methods obsolete. If you're looking for a better way to transfer data over the internet or are just looking for a way to improve productivity within your team, Dropbox is for you.

FIGURE 22. A PDFViewer component

4.8 Week 8

The add people section has a group of inputs to add a person to sign request. There are four roles, such as signer, approver, manager and reviewer. An order to handle a sign request can be chosen with a dropdown option. A message of request can be written or selected from the template.

When developing the section, a *search* component was developed to serve a function add people to a specific role. As the Figure 23 below shows, the component has an input to take the value from the user and returns a list of people related with a search value. The user can select from the list to add people.


A

Antonija Kasum
antonija@gmail.com

Essi Jäppinen
essi@gmail.com

Tanja Räikkönen
tanja@gmail.com

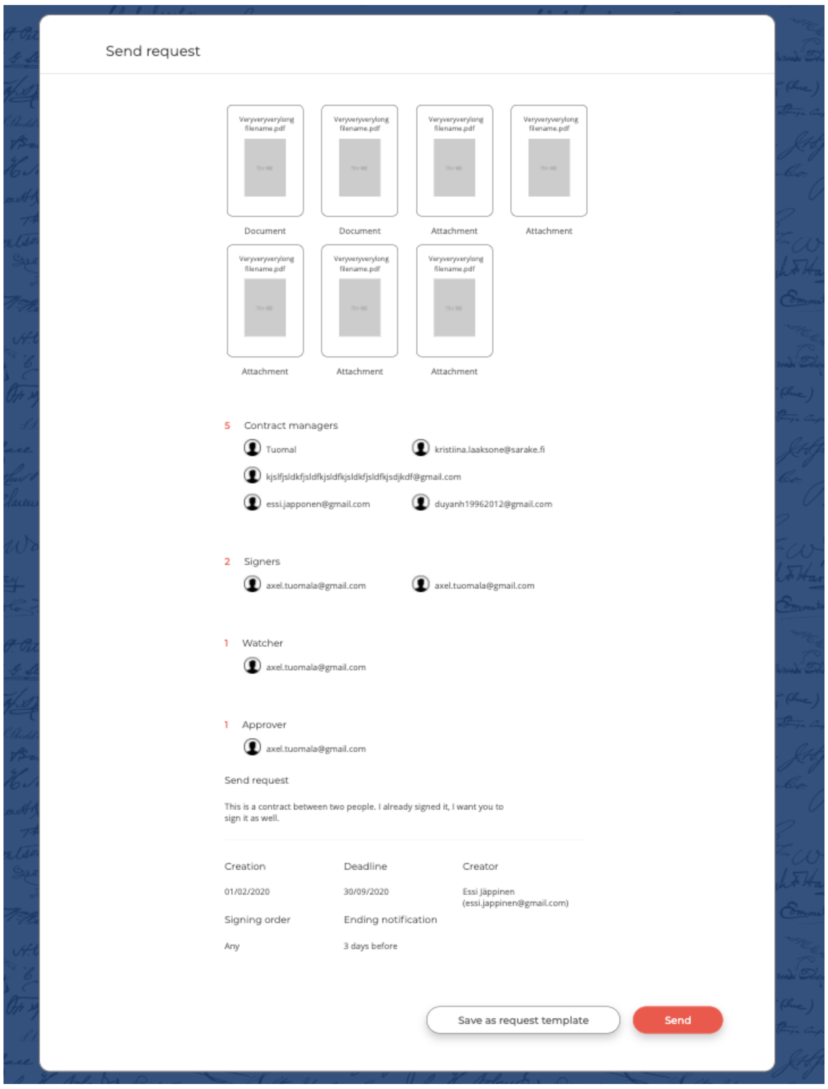
Kristiina Laaksonen
kristiina.laaksonen@gmail.com

Tukka Taskinen
tuuka@gmail.com

FIGURE 23. A search component

Signature layout is a space for edit content to a document with an online editor. User can drag and drop content to a preview document and customization a layout of sign document. A meeting was organized by Long to discuss with all member about how to create the editor. After a meeting, he got enough information to start a development.

A send request element is used to display all the information that edit in the previous parts. The Figure 24 shows an overview of the send request element.


Send request

Veryveryverylong filename.pdf
Document
Veryveryverylong filename.pdf
Document
Veryveryverylong filename.pdf
Attachment
Veryveryverylong filename.pdf
Attachment
Veryveryverylong filename.pdf
Attachment
Veryveryverylong filename.pdf
Attachment
Veryveryverylong filename.pdf
Attachment

5 Contract managers

Tuomal
kristiina.laaksone@sarake.fi
kjslfjsldkfjsldfkjsldfkjsldkfjsldfkjsdjkdf@gmail.com
essi.japponen@gmail.com
duyanh19962012@gmail.com

2 Signers

axel.tuomala@gmail.com
axel.tuomala@gmail.com

1 Watcher

axel.tuomala@gmail.com

1 Approver

axel.tuomala@gmail.com

Send request

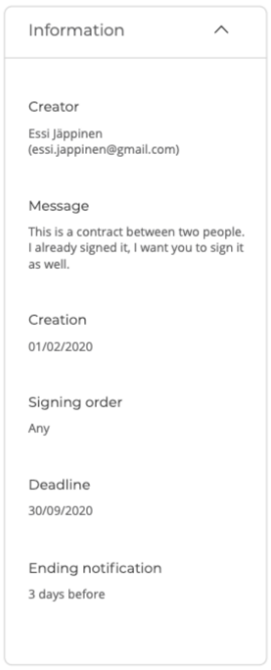
This is a contract between two people. I already signed it, I want you to sign it as well.

Creation	Deadline	Creator
01/02/2020	30/09/2020	Essi Jäppinen (essi.jappinen@gmail.com)
Signing order	Ending notification	
Any	3 days before	

FIGURE 24. A send request component

4.9 Week 9

On Monday, Essi had a presentation about new views of Sarake Sign. A accept-request page was assigned as a task for week 9 and week 10. After having an overview of the page, it should be started to develop from a smaller component and then grab into a view. A list of components was created for making managing easier. Components that should be done first were on the right side of page: information, people, and comments. The components can be toggled to show and hide. The Figure 25 shows a view of *information* component. The component provides a list of information related to sign a request, such as creator, message, creation, signing order, deadline, ending notification.


Information

Creator

Essi Jäppinen
(essi.jappinen@gmail.com)

Message

This is a contract between two people. I already signed it, I want you to sign it as well.

Creation

01/02/2020

Signing order

Any

Deadline

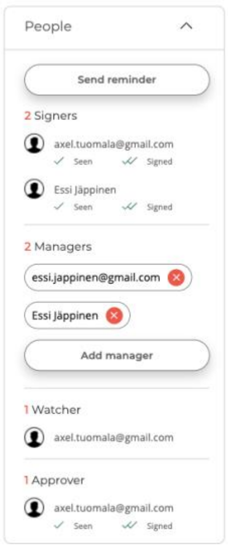
30/09/2020

Ending notification

3 days before

FIGURE 25. An information component

A *creation* component shows a list of roles with a tracking status based on people. If the user is an admin, the user can edit a manager role such as remove or add a new manager. A manager section has two versions. The first version is used for normal user who can just read manager information. The Figure 26 shows the second version which is used for the admin user to edit information of manager.


People

Send reminder

2 Signers

axel.tuomala@gmail.com
✓ Seen ✓✓ Signed

Essi Jäppinen
✓ Seen ✓✓ Signed

2 Managers

essi.jappinen@gmail.com

Essi Jäppinen

Add manager

1 Watcher

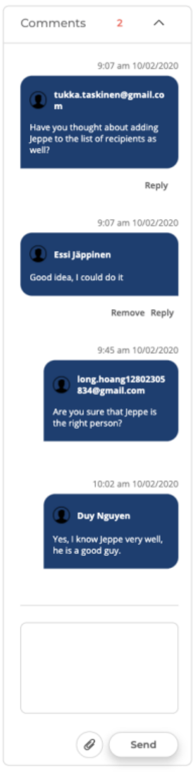
axel.tuomala@gmail.com

1 Approver

axel.tuomala@gmail.com
✓ Seen ✓✓ Signed

FIGURE 26. An admin view of people component

The most complicated part was to develop a *comment* component. The comment can be seen as a chat box. The user can make a comment or reply a comment. Moreover, the user can upload a file into as an attachment. A problem when developing the component is to separate a role of comment owner and reply to have a different style of message. The Figure 27 is an example of styles of comment.



9:07 am 10/02/2020
tukka.taskinen@gmail.com
Have you thought about adding Jeppe to the list of recipients as well?

9:07 am 10/02/2020
Essi Jäppinen
Good idea, I could do it

9:45 am 10/02/2020
long.hoang12802305834@gmail.com
Are you sure that Jeppe is the right person?

10:02 am 10/02/2020
Duy Nguyen
Yes, I know Jeppe very well, he is a good guy.

FIGURE 27. A comment component

4.10 Week 10

In continuation of the developing accept request page from week 9, a task for the week was to develop a section to preview a document. With an experience of using pdf.js on a previous page, a development of preview file is easier and helps the author to have time for styling a layout for a preview. As the Figure 28 below shows, a layout of preview section has three main parts: header, content and footer. A header contains a zoom factor and pagination to help the user easier to access to another page without scrolling. A content is a field for previewing the document. A footer has a download button for the user to download a document to the computer.

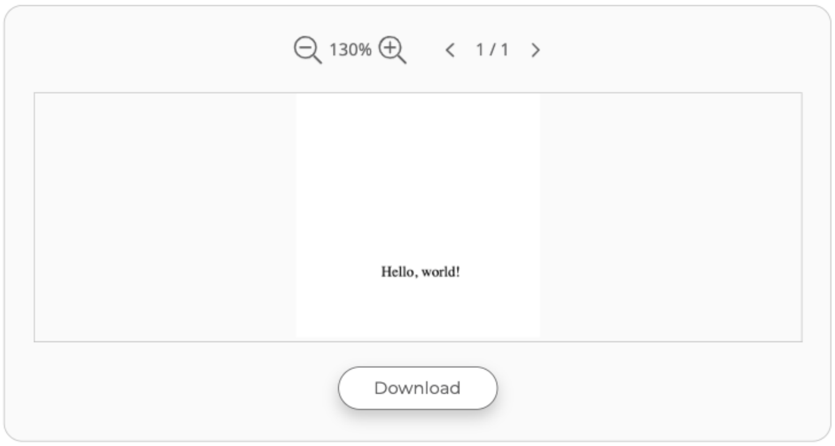


FIGURE 28. A PreviewFile component

Under a preview file section, there is a list of documents and attachments of sign request. The user can select a document and get a preview from a preview section. The user can download all the files by click to a button.

An accept request page has a group of user actions which always stick at the end of the page. The group of buttons handle a decision of user for a sign request, such as abort, delegate, sign and abort.

After finishing a small component, a final task was to grab all the components to a view and handle all the states between components to make sure that a view can work functionally (Figure 29.).

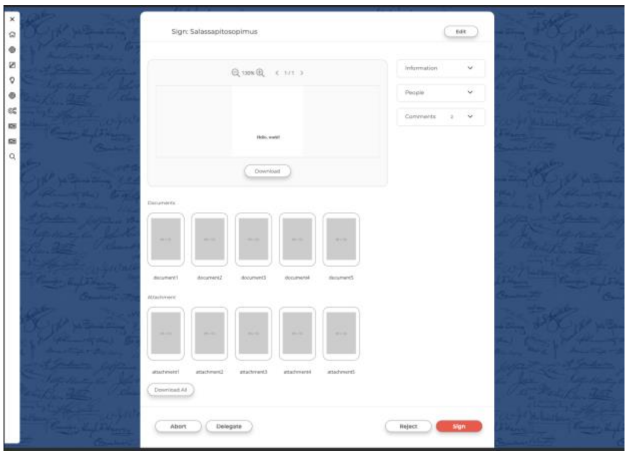


FIGURE 29. An accept request page

4.11 Week 11

Sarake used SVN as a version control system. After 10 weeks working with SVN, a developer team saw a drawback of using SVN with a review and managing code method when working as a team. A meeting was organized to find a solution to replace SVN. Git was the best option for this problem. Gitea was chosen as a solution that replaced Github for setting a self-hosted Git service.

After configuring Gitea to the Sarake server, a developer team had a mission to try to use a Gitea and made a meeting to unify a developing flow when using Git as a version control system.

Spending a week for using Gitea made a great opportunity to gain a knowledge of the product. The developer team had to write a documentation to guideline a developing flow and review a pull request with Gitea. All project of Sarake was changed to Gitea platform and can be continued to develop.

A replacement of using Git helps Sarake upgrade their workflow. It makes the management code and analysis quality of product more professional.

4.12 Week 12

Essi continued to provide designs for Sarake Sign's page at a meeting on Monday. Pages which need to be developed on this week are template and help center.

A template page is used to manage a list of documents and request templates of product (Figure 30.). A template can be shared for using with another user. A new template can be created on this page. A document and request template are divided by a tab title. A content will display based on the selected title.

The template page got a modal to handle share action. A modal content can reuse a *search* component which was created in previous week. A scrolling of selected users list is customization based on a design.

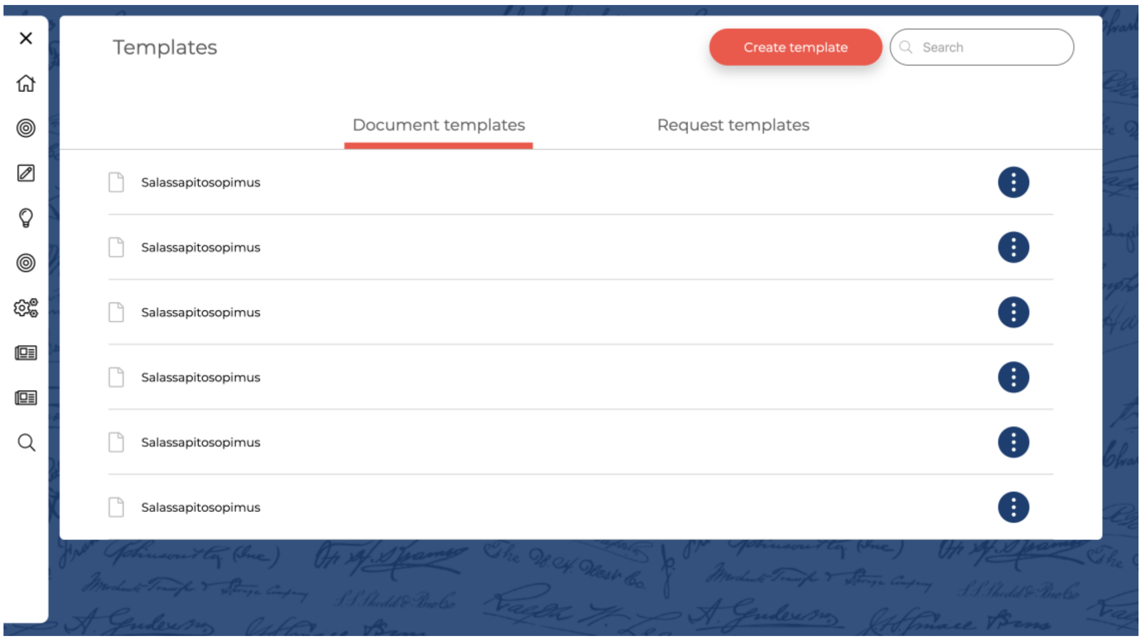
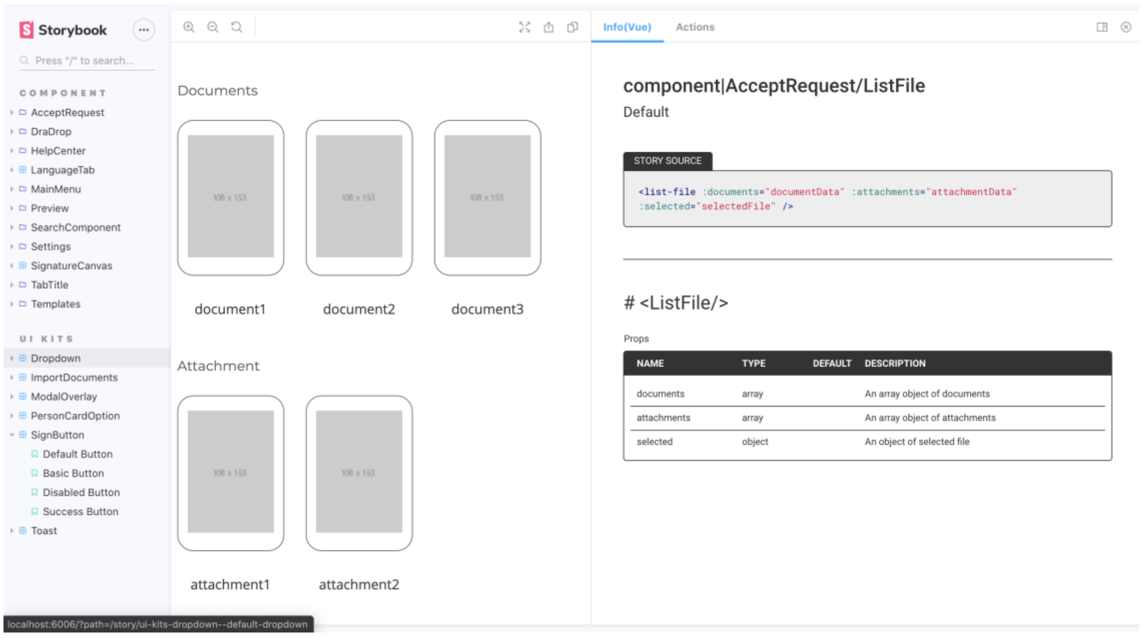


FIGURE 30. An overview of templates page

4.13 Week 13

There is a large of number components which was created in the Sign application. A developer though that the application might have a component documentation which helped another developer can see and avoid creating the redundant components. A storybook is a tool which can solve the problem.

A storybook is installed and configurated to run at a specific port. This tool is not affected to a product and can be used to see all the cases of a component that can occur. All components of the Sign application are imported to storybook (Figure 31.). A story will show a style of component, how to write a component in VueJS and props information. This information helps the developer easy to approach and reuse a component in the application.



component|AcceptRequest/ListFile

Default

STORY SOURCE

```
<list-file :documents="documentData" :attachments="attachmentData"
:selected="selectedFile" />
```

<ListFile/>

Props

NAME	TYPE	DEFAULT	DESCRIPTION
documents	array		An array object of documents
attachments	array		An array object of attachments
selected	object		An object of selected file

FIGURE 31. An overview of Storybook

4.14 Week 14

After finished on importing Sign components to storybook, Essi continued to give a material to create an error page. The user can see this page when entering a wrong address or going to a wrong direction component (Figure 32.). An application should display an error notification and a button to help the user to go back to the previous page or homepage based on the user history.

A difficult part when developing an error page is to find a way to detect the previous address of user. It can be cached in Vue router and saved into a local storage to prevent clearing data when refreshing the browser.

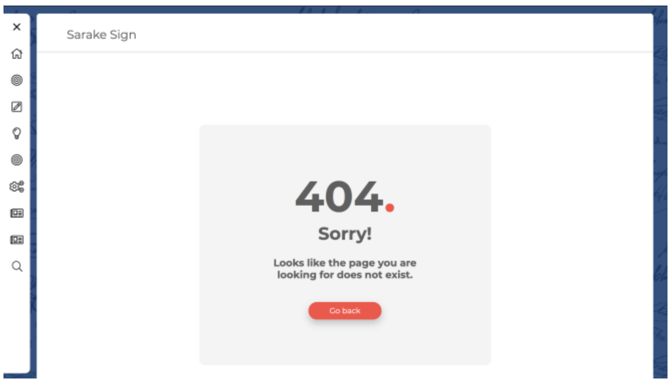
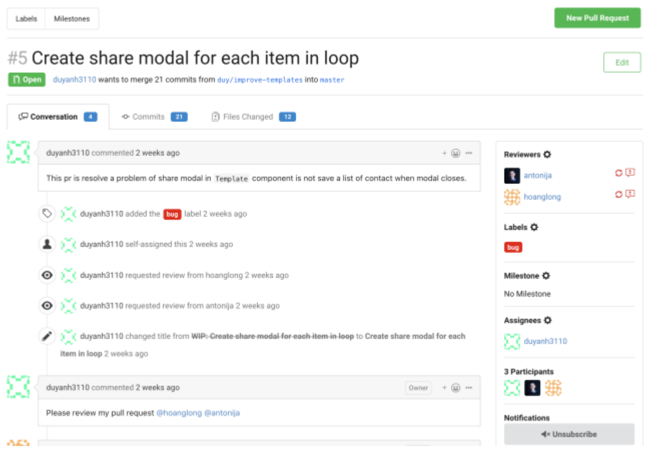


FIGURE 32. An error page

4.15 Week 15

With changing to Gitea, the developer can easier review a pull request and tracking issues of product. A task for week 15 was to get an improvement for Sign based on issues created in Sign repositories.

There is an issue with a *search* component, a list will be stretched out when there are too much results. A solution to resolve this issue is to make a list just display four items and the list can be scrolled to see other items. Moreover, the user can use both mouse and keyboard to handle a navigation between items. Fixing this issue and adding new method for combination of actions between a mouse and a keyboard spend much more time than expected because a search component and a child component are used in separate views with separate styles. An improvement needs to pass out all the tests in all views to make sure that a list works fine. A pull request was created to get reviews from other member in developer team (Figure 33.). After receiving an approval from reviewers, a pull request will be merged into a master branch.


Labels Milestones

New Pull Request

#5 Create share modal for each item in loop

Edit

Open duyanh3110 wants to merge 21 commits from `duy/improve-templates` into `master`

Conversation 4 Commits 21 Files Changed 12

duyanh3110 commented 2 weeks ago

This pr is resolve a problem of share modal in `Template` component is not save a list of contact when modal closes.

duyanh3110 added the bug label 2 weeks ago

duyanh3110 self-assigned this 2 weeks ago

duyanh3110 requested review from hoanglong 2 weeks ago

duyanh3110 requested review from antonija 2 weeks ago

duyanh3110 changed title from ~~WIP: Create share modal for each item in loop~~ to Create share modal for each item in loop 2 weeks ago

duyanh3110 commented 2 weeks ago Owner

Please review my pull request @hoanglong @antonija

Reviewers

antonija

hoanglong

Labels

bug

Milestone

No Milestone

Assignees

duyanh3110

3 Participants

Notifications

Unsubscribe

FIGURE 33. A pull request for improving search component

5 CONCLUSION

For four months, the author has participated in developing a new interface for Sarake Sign. All the targets from the beginning of the project have been passed by a contribution of all members. A development process helped the author on gaining knowledge about new technology and having working experience with team. The Sarake company and the author’s teammates always supported and gave feedback to help the author improve personal skills. Working as a frontend developer role at Sarake Sign gave the author have a great experience to start a career after graduation.

REFERENCES

1. Lazaris, L. 2020. *A Basic HTML5 Template For Any Project*. Date of retrieval May 20 2020 https://www.sitepoint.com/a-basic-html5-template/

2. *About JavaScript.* 2020. Date of retrieval May 21 2020 https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

3. Peyrott, S. 2017. *A Brief History of JavaScript.* Date of retrieval May 21 2020 https://auth0.com/blog/a-brief-history-of-javascript/

4. Rouse, M. 2020. *TheServerSide.* Date of retrieval May 21 2020 https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language

5. *Storybook documentation.* 2020. Date of retrieval May 19 2020 https://storybook.js.org/docs/basics/introduction/

6. Ahamad, M. 2019. *Architecting Vuex store for large scale Vue.js applications.* Date of retrieval May 20 2020 https://medium.com/locale-ai/architecting-vuex-store-for-large-scale-vue-js-applications-24c36137e251

7. *GraphQL is the better REST.* 2018. Date of retrieval May 19 2020 https://www.howtographql.com/basics/1-graphql-is-the-better-rest/

8. *Introduction.* 2020. Date of retrieval May 19 2020 https://vuejs.org/v2/guide/index.html

9. International, E. 2015. *ECMAScript® 2015 Language Specification.* Date of retrieval May 20 2020 http://www.ecma-international.org/ecma-262/6.0/index.html#sec-ecmascript-overview

10. *Introduction to GraphQL.* (n.d.). Date of retrieval May 20 2020 https://graphql.org/learn/

11. *About npm.* (n.d.). Date of retrieval May 20 2020 https://www.npmjs.com/about