Game of Frameworks

Process to rewrite a site to a new Framework

Thesis Front18
by Matthias Nyman, 2020-05-27

Abstract

The Stockholm based software company Basedo has a web-based offer to their customers that they have been very successful with. The company has an outdated version of a frontend framework - Angular - which in the long run will cause problems with both service development and the maintenance of the product.

The choice they had to make was to either update to the latest and most recent version of Angular or to make a re-encoding of a new framework. They chose to take the opportunity to change the framework. The work effort was equal but changing offered new opportunities.

The assignment given to me by the company was to:

- Update existing code from Angular.js to React

- Provide exactly the same design and functionality as the previous framework

- Document the work and create the conditions for further development

Despite some work logistical problems caused by the ongoing pandemic, I managed to code and create two views, ie tabs. With the requirement of design and functionality fulfilled.

The conditions now exist for the company to take the next step for them to become more efficient. It is now possible when it is easier - and technically feasible - to implement new features and update their code base.

I am really proud of the work I have done and the experiences that I bring with me from this project are mainly:

- How important it is to have understanding and knowledge in different frameworks. The problems and opportunities that arose during the course of the work provided me with a deeper knowledge and understanding of how the tools work...

- The practical experience from this project to really understand and be able to work with different types of coding.

TABLE OF CONTENT

1 Introduction

1.1 Background

If you look at the software development lifecycle it is crucial that you use the right tools, the right frontend frameworks to secure future services. There are a lot of challenges for today's frontend developers as new frameworks are constantly coming up and you have to choose the right one for today, and for tomorrow.

Frameworks are very powerful tools for the development of modern web solutions. If the product has been on the market for a few years, it may be that the technology development has been so fast that it moderation might be a good idea to update to newer technologies. To continue to be competitive in the market and also to find and attract people with programming skills who can continue to develop the product.

Besedo is a service company in Stockholm that offers web-based services in three areas: customer service, quality assurance and fraud prevention.The business idea is to automate a large part of the work through artificial intelligence and machine learning. AI and human powered content tools and services for online platforms.

Besedo was founded in 2002 and has 6 offices across the world. The Swedish office has some 18 employees and a turnover of 40 MSEK. The company informs that they review and moderate more than 570 million pieces of content, respond to over 1 million user support inquiries and block almost 40 million fraud and scam attempts every year.[1]

The company is currently implementing a conversion from AngularJS to React, and I am part of this team.

1.2 Objectives and criteria

The purpose of this paper is to investigate the advantages and disadvantages of changing framework compared to updating an existing framework to the latest version.
- Update the existing code to React
- Describe the benefits of upgrading to a new framework
- Estimate resources / man-hours / required for the work

[1] **https://besedo.com/**

1.3 Problem description and problem analysis

- What are the threats?
- What are the possibilities: Will the product be better and in what way
  -user experience
  -stability
  -speed

1.4 Purpose

Many companies need to move to a new framework. A company can have an outdated version or is looking for some features that another framework can provide. The investment should of course provide benefits as the product becomes better and easier to continue to develop.

1.5 Limitations

This essay will be delimited to specific components of the selected platform.
The work also intends to be limited to converting AngularJS to React. There are different ways to implement the new code but i have only worked with one.

Angular has some old packages that won't work on react. They need to be replaced to one that works with react but still has the same design and functionality.

2 Method

Besedo works according to an agile working method where they have 2 weeks sprinting. Before each sprint they go through and analyze how the previous sprint went. Were there any problems and what should be done for the next sprint?
Every morning they have a daily standup where you go through what you have done, what to do and if you have encountered problems.

During the first sprint, the focus was on getting into the system. It meant getting into all the systems and understanding how to use/work with them. My plan was to go straight into production mode and read/ask for help when any problems arose.

The task given to me was to build a component that was in itself a view. Here I chose to divide the component into parts.

Step 1: Creating what I call “the skeleton”. In the Angular project you will find this part as HTML with some differences as they have some own attributes with some specific functionality. This I rewrote in React as JSX and I also transferred all css to a new file where it might require some editing to get the same result.

Step 2 adds functionality. You then take all functions from Angular and rewrite them with Typescript and make them work with the new skeleton.

Step 3: Some functionality required here requires securing the previously implemented functions. These are usually features that show if an event has happened and its results.

Step 4 is to test these features. To this end, I created test functions that implement an event and then see if the result is what I expected

Step 5 is to hand over the code to the company so that another programmer can comment on the outfall. This is to check if any problems are found where the component does not act as it should or follow good coding practices. If I do receive any comments I will adjust the problem and send it back for further review.

3 Theory

The project needs the understanding of some tools that have to be used in the process.

3.1 Angular.js

Angular.js is built with JavaScript and is the first version of angular that later got renamed to Angular. Angular is mainly updated by Google and is used in all their products. The later version switched to typescript and continues to be updated. Angular is built with the MVC-concept. [2]

3.2 React

React is described as a ui-library to write reusable components. It has some features like JSX and hooks. JSX is React syntax that is mainly used when writing a React component. Hooks are the new technique to write components in React. Before hooks the best practice was class-based components. Facebook, the creator of React, noticed that numerous

[2] https://docs.angularjs.org/api
https://www.w3schools.com/whatis/whatis_angularjs.asp

people thought that class-based components were too hard to learn and therefore created hooks. [3]

3.3 Angular to React

There are several ways to transition from Angular to React. One way is to start a new project and rewrite the site and the switch from the old. That means that the whole project needs to be done before you can deploy the new code. Besedo wanted an ongoing way of moving the code over as the developers could continue to work on the older framework. This means two frameworks deployd and working simultaneously. The base of the project is Angular and when a module is written in React Angular will route to it. [4]

3.4 TypeScript

TypeScript is Open Source and called a“superset” of Javascript that makes the production safer from bugs and simpler code to reed. It is strongly typed and transcompiles to javascrip. [5]

3.5 Enzyme

To test your React code you can use Enzyme, a JavaScriptbased testing utility. This allows you to simulate runtime and manipulate output to test your user interface. [6]

3.6 NPM-package

NPM is a package manager for Node.js packages. It is the default package manager for the JavaScript runtime environment Node.js. It’s an online database of public and paid-for private packages, called the NPM registry.[7]

[3] https://reactjs.org/docs/hooks-intro.html
https://www.w3schools.com/whatis/whatis_react.asp

[4] https://buttercms.com/blog/migrating-from-angularjs-to-react
https://www.toptal.com/react/why-did-i-switch-from-angularjs-to-react

[5] https://www.typescriptlang.org/
https://github.com/microsoft/TypeScript/wiki/Roadmap

[6] https://enzymejs.github.io/enzyme/

[7] https://www.npmjs.com/

3.7 Visual Studio Code

Code editor made by microsoft. Lightweight but powerful editor that works with mac or windows. It has built in support or Javascript, typeScript and Node.js[8]

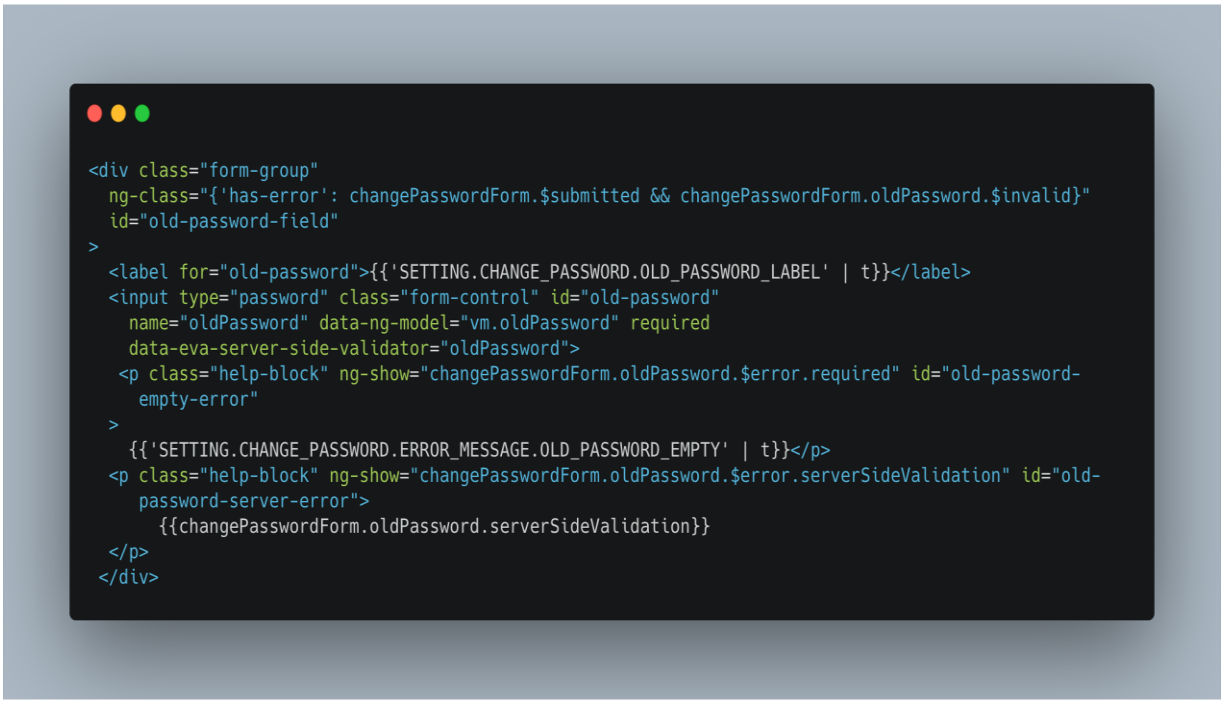
3.8 GitLab

A platform with a complete DevOps solutions, ci/cd tools, git-repositories and continuous features[9]

4 Result

I developed a working method with the aim of focusing on defined parts of the component, which I then could build on in the next step.

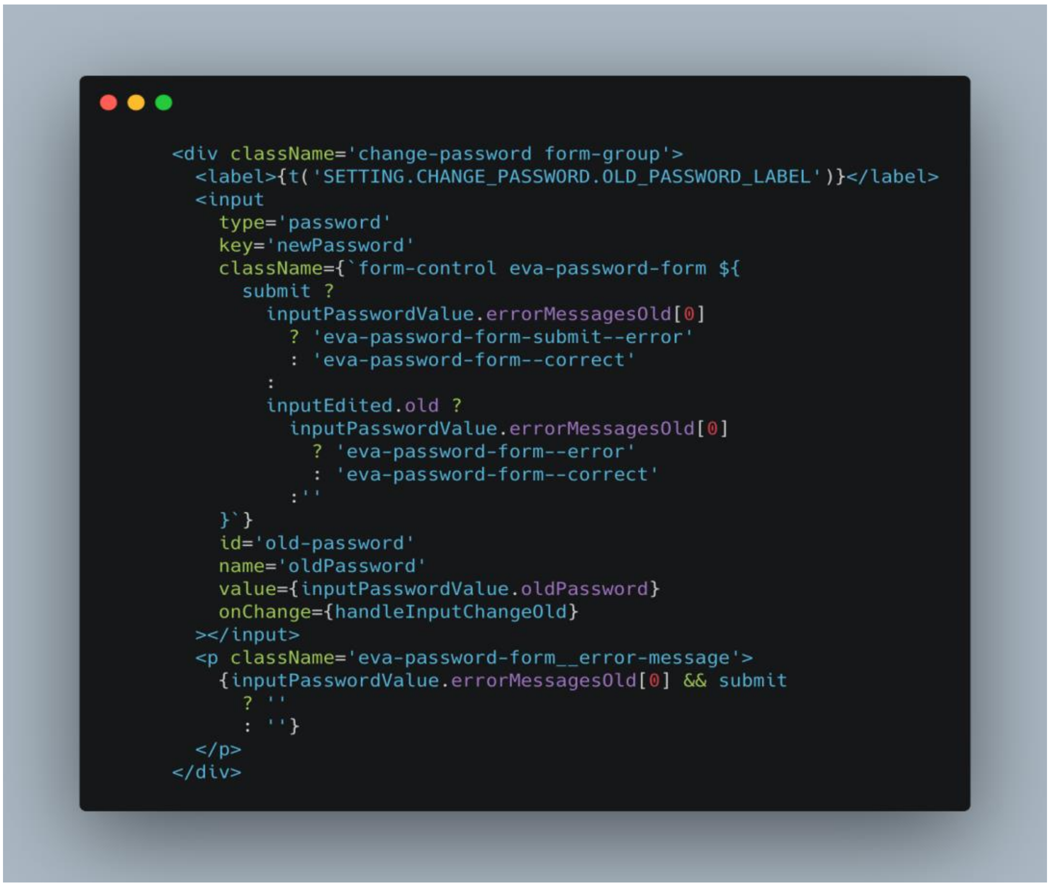
In step 1 I built up a skeleton for the component. From the HTML in Image 1 I rewrote it to jsx as seen in Imagee 2. This requires reformatting of Angular's attributes and syntax to Reacts.
I also created a new css file for the React components where minimal adjustment was required to get the same look.



```
<div class="form-group"
  ng-class="{'has-error': changePasswordForm.$submitted && changePasswordForm.oldPassword.$invalid}"
  id="old-password-field"
>
  <label for="old-password">{{'SETTING.CHANGE_PASSWORD.OLD_PASSWORD_LABEL' | t}}</label>
  <input type="password" class="form-control" id="old-password"
    name="oldPassword" data-ng-model="vm.oldPassword" required
    data-eva-server-side-validator="oldPassword">
   <p class="help-block" ng-show="changePasswordForm.oldPassword.$error.required" id="old-password-
     empty-error"
  >
    {{'SETTING.CHANGE_PASSWORD.ERROR_MESSAGE.OLD_PASSWORD_EMPTY' | t}}</p>
  <p class="help-block" ng-show="changePasswordForm.oldPassword.$error.serverSideValidation" id="old-
     password-server-error">
      {{changePasswordForm.oldPassword.serverSideValidation}}
  </p>
 </div>
```

[8] https://code.visualstudio.com/
[9] https://about.gitlab.com/

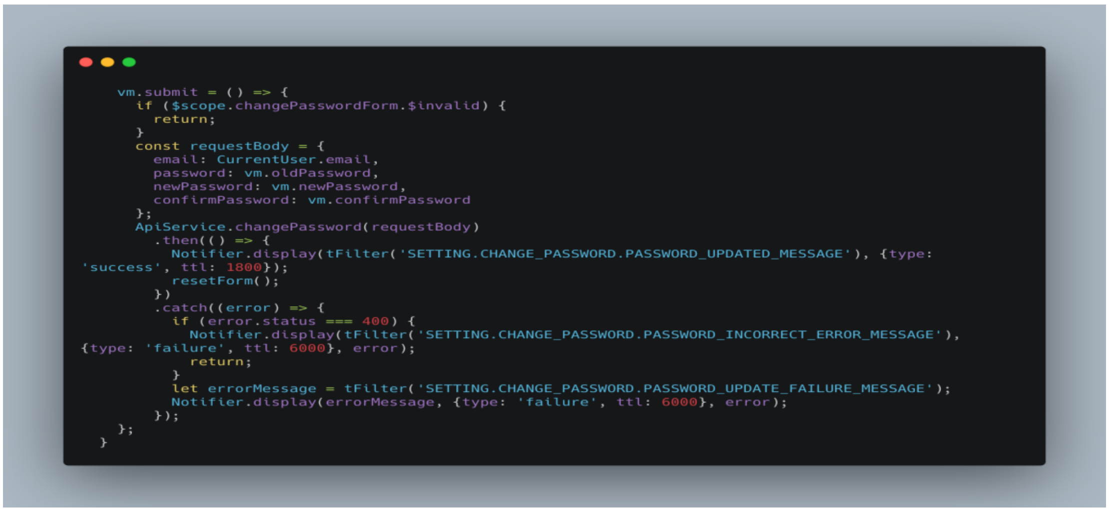
Image 1: The old html for a input



```
<div className='change-password form-group'>
  <label>{t('SETTING.CHANGE_PASSWORD.OLD_PASSWORD_LABEL')}</label>
  <input
    type='password'
    key='newPassword'
    className={`form-control eva-password-form ${
      submit ?
        inputPasswordValue.errorMessagesOld[0]
          ? 'eva-password-form-submit--error'
          : 'eva-password-form--correct'
        :
        inputEdited.old ?
          inputPasswordValue.errorMessagesOld[0]
            ? 'eva-password-form--error'
            : 'eva-password-form--correct'
          :''
    }`}
    id='old-password'
    name='oldPassword'
    value={inputPasswordValue.oldPassword}
    onChange={handleInputChangeOld}
  ></input>
  <p className='eva-password-form__error-message'>
    {inputPasswordValue.errorMessagesOld[0] && submit
      ? ''
      : ''}
  </p>
</div>
```

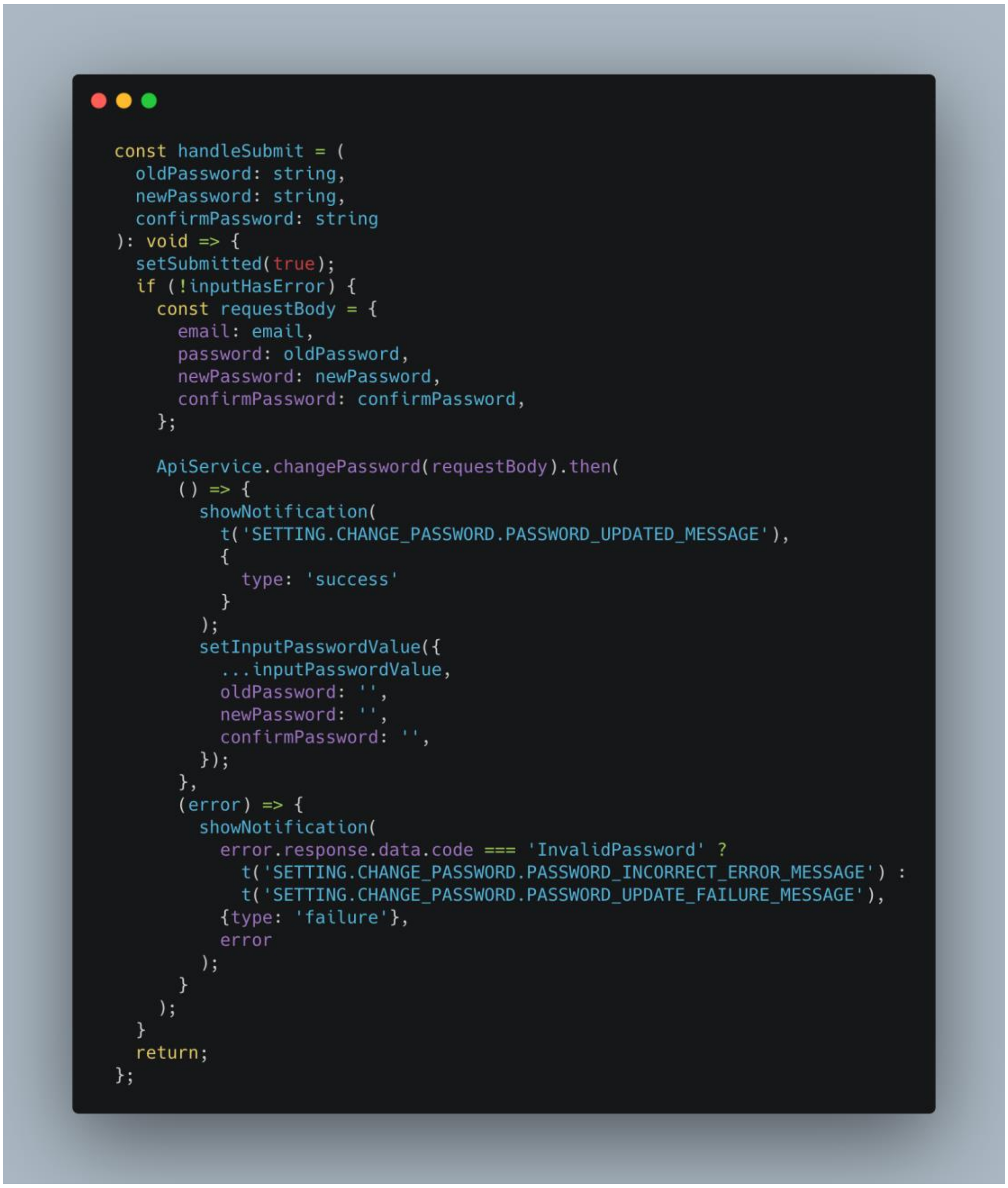
Image 2: The new code in jsx for a input

During step 2, I created the functionality. The code varied more here than in the step before when Angular had controllers (as can be seen in Image 3) and that I wrote functions in the React part with TypeScript (as can be seen in Image 4)



```
    vm.submit = () => {
      if ($scope.changePasswordForm.$invalid) {
        return;
      }
      const requestBody = {
        email: CurrentUser.email,
        password: vm.oldPassword,
        newPassword: vm.newPassword,
        confirmPassword: vm.confirmPassword
      };
      ApiService.changePassword(requestBody)
        .then(() => {
          Notifier.display(tFilter('SETTING.CHANGE_PASSWORD.PASSWORD_UPDATED_MESSAGE'), {type:
'success', ttl: 1800});
          resetForm();
        })
        .catch((error) => {
          if (error.status === 400) {
            Notifier.display(tFilter('SETTING.CHANGE_PASSWORD.PASSWORD_INCORRECT_ERROR_MESSAGE'),
{type: 'failure', ttl: 6000}, error);
            return;
          }
          let errorMessage = tFilter('SETTING.CHANGE_PASSWORD.PASSWORD_UPDATE_FAILURE_MESSAGE');
          Notifier.display(errorMessage, {type: 'failure', ttl: 6000}, error);
        });
    };
  }
```

Image 3: Function for submit form Angular

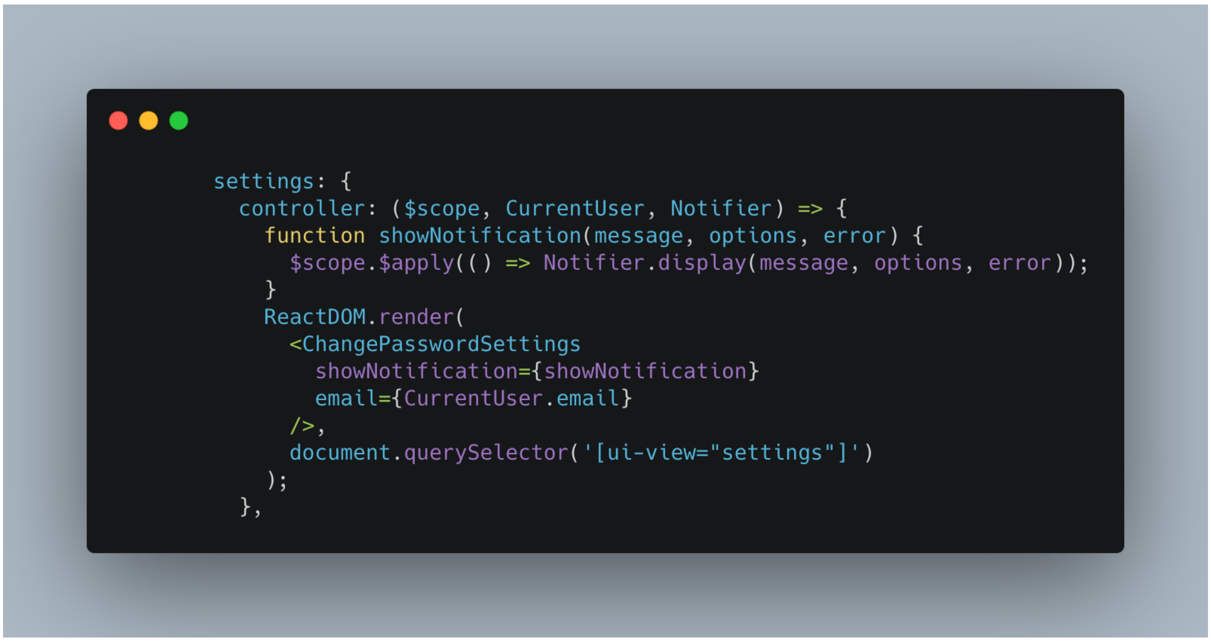


```
const handleSubmit = (
  oldPassword: string,
  newPassword: string,
  confirmPassword: string
): void => {
  setSubmitted(true);
  if (!inputHasError) {
    const requestBody = {
      email: email,
      password: oldPassword,
      newPassword: newPassword,
      confirmPassword: confirmPassword,
    };

    ApiService.changePassword(requestBody).then(
      () => {
        showNotification(
          t('SETTING.CHANGE_PASSWORD.PASSWORD_UPDATED_MESSAGE'),
          {
            type: 'success'
          }
        );
        setInputPasswordValue({
          ...inputPasswordValue,
          oldPassword: '',
          newPassword: '',
          confirmPassword: '',
        });
      },
      (error) => {
        showNotification(
          error.response.data.code === 'InvalidPassword' ?
            t('SETTING.CHANGE_PASSWORD.PASSWORD_INCORRECT_ERROR_MESSAGE') :
            t('SETTING.CHANGE_PASSWORD.PASSWORD_UPDATE_FAILURE_MESSAGE'),
          {type: 'failure'},
          error
        );
      }
    );
  }
  return;
};
```

Image 4: Function for submit form React.

Step 3 in my working process was to add event functionality, that is the functionality to show the user what is happening. The Angular part used npm for this part and the same package could be used in React. Therefore I could send them as props when I route to the component, see Image 5.



```
settings: {
  controller: ($scope, CurrentUser, Notifier) => {
    function showNotification(message, options, error) {
      $scope.$apply(() => Notifier.display(message, options, error));
    }
    ReactDOM.render(
      <ChangePasswordSettings
        showNotification={showNotification}
        email={CurrentUser.email}
      />,
      document.querySelector('[ui-view="settings"]')
    );
  },
```

Image 5: Sending module as props.

The next step was to write the tests to validate the functionalities that previously were created. Most often you test a process where you simulate everything to go well to see if the result is as expected. There is also a process where you test the implications if something goes wrong.

Then I handed over the code to gitlab so the company can check if the code works similar to the previous version and that the code is well done. If they find any problems they would comment on what they have found and what they expect to be fixed. From there I fix the issue and send the code back for a further review.

I used visual studio code as a code editor throughout the project.

5 Discussion

Basedo has not built up any major Frontend in-house operations as their products and offerings are basically based on Backend development. After an analysis, the company chose to migrate to React as it is the largest and one of the most easily worked frameworks. There were other options and they could easily have chosen to update Angular to the most modern version.

Rewriting a page in React so that it gets the same design as in Angular is relatively simple but getting the same functionality requires a little more effort and knowledge. Getting started and operating in a large project, finding everything and understanding the structure also takes time. It also would have been an advantage if there had been a Frontend-developer at the office that had knowledge and routine from the existing project.

The tool used by the company earlier was Angular.js, a very old and outdated version. Today, there are modern versions of Angular that offer the same benefits and opportunities as React. And in comparison the work effort would have been roughly equivalent to implementing another new framework.

But the comparison between these given benefits to React because that it is more popular, used by more programmers and therefore has the advantage of being constantly updated via Facebook. More skilled people to employ and risk minimization for future operations and updates.

There are also two different products to compare here. React, which is a library and not really a framework, is more flexible and easier to use. About Angular can be said that there is greater built-in functionality - but that is not always needed or priceworthy if there is no added value.

Thus: Angular is a modular and rigid system compared to React which is more flexible and adaptable to the needs of the user

Tho the component needed to be cleared from a developer and then cleared from the lead developer i had no component up and running when submitting this report.

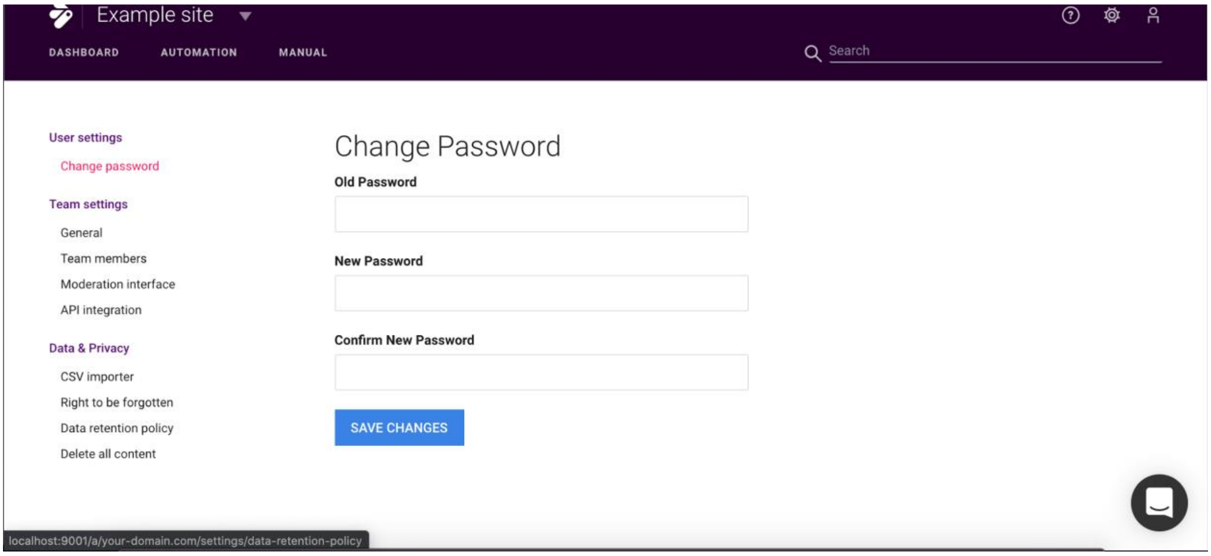
There is a plan to add redux to the project and use it as a global state for both of the frameworks. That would help the transition so you don’t need to focus on the state when creating a component.

6 Conclusions

Working "component by component" as a working method was a doable strategy, although it proved difficult to work with so many files. Also, implementing TypeScript and hocks proved to be difficult as there were so few examples to learn from. Thus, it resulted in a slow start up as there was a lot to learn and read on. The fact that everyone worked from home due to the ongoing pandemic made it difficult and cumbersome to ask for help around the platform.

The troubles I had in the beginning of the course was to get an overall understanding of the platform to be able to start working. That is something if you don’t need to worry about if you already are working on the project.

As an example, the Code Review process took a lot of time. However, I managed to update and rewrite two views. See Image 6 and Image 7.



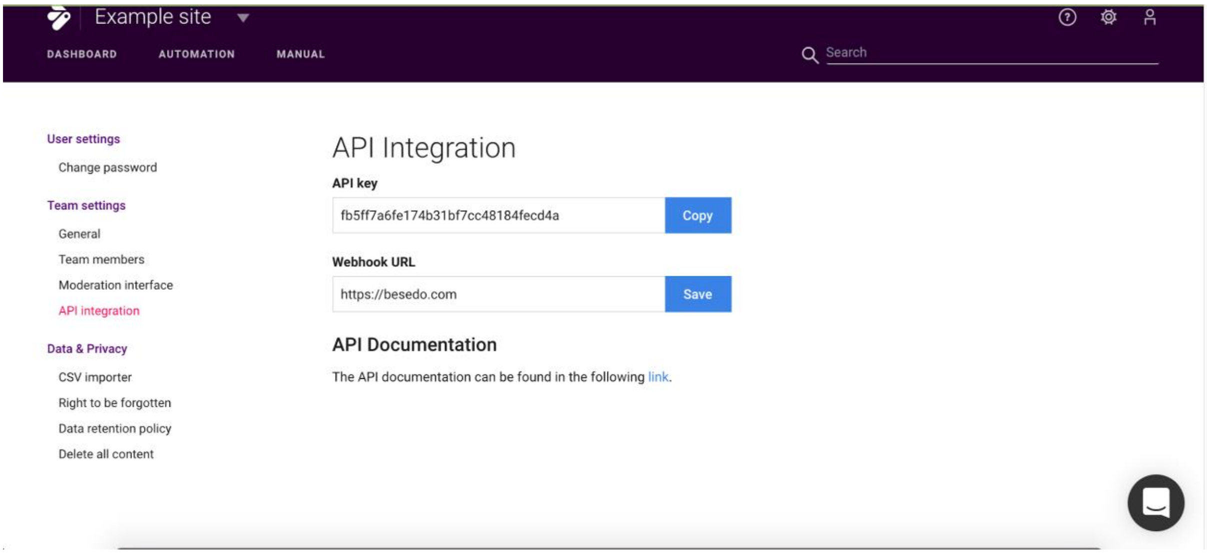
Change Password

Old Password

New Password

Confirm New Password

Image 6: Change password view



API Integration

API key: fb5ff7a6fe174b31bf7cc48184fecd4a

Webhook URL: https://besedo.com

API Documentation

The API documentation can be found in the following link.

Image 7: API Integration View

The company made the decision on this investment to secure its product for the future in the form of user-friendliness, stability etc..

And what is the outcome then?

Since only part of the project code is rewritten, it is difficult to make an objective measurement of outcome and effect. At this stage you cannot see any differences in speed and in terms of user experience it is the same as for the predecessor - which in itself was also the directive.

But now when the code is updated and written in TypeScript, so it is strongly typed and no old bugs will be in the background to interfere, which means that stability has increased. And further on increasing and developing user experience will be easier. Another important advantage of the work done is that the code becomes easier to maintain and develop now when the foundation is set.

References

Besedos website: https://besedo.com/ (2020-06-04)
Besedos vision: https://slides.com/binpur/deck (2020-06-04)
AngularJS documentation: https://docs.angularjs.org/api (2020-06-04)
AngularJS Info: https://www.w3schools.com/whatis/whatis_angularjs.asp (2020-06-04)
React documentation: https://reactjs.org/docs/hooks-intro.html (2020-06-04)
React Info: https://www.w3schools.com/whatis/whatis_react.asp (2020-06-04)
TypeScript documentation: https://www.typescriptlang.org/ (2020-06-04)
TypeScript Roadmap: https://github.com/microsoft/TypeScript/wiki/Roadmap (2020-06-04)
Blog about React, Hooks and typescript: https://fettblog.eu/typescript-react/ (2020-06-04)
Enzyme documentation: https://enzymejs.github.io/enzyme/docs/api/ (2020-06-04)
Article about switching from AngularJs to React: https://www.toptal.com/react/why-did-i-switch-from-angularjs-to-react (2020-06-04)