Wilska Aki

DEVELOPING SHAREPOINT WEB PARTS

Bachelor's thesis
Information technology

2019

Kaakkois-Suomen
ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Aki Wilska	Insinööri (AMK)	Syyskuu 2019
Opinnäytetyön nimi		33 sivua
Web Partien kehitys SharePoint-ympäristöön		
Toimeksiantaja		
T-Base Oy		
Ohjaaja		
Lehtori Marko Oras		

Tiivistelmä

Tämän opinnäytetyön aiheena oli tuottaa asiakkaalle henkilöiden hakutyökalu käyttäen Microsoftin SharePoint-ympäristön Web Partia. Web Part sisältää lukuisia eri hakuparametreja, joiden avulla se hakee henkilöt tietokannasta käyttäen olemassa olevaa rajapintaa. Haetuista henkilöistä voidaan tämän jälkeen valita henkilö, jonka tiedot siirtyvät ostoskoriin. Ostoskorista voidaan ladata pakattu tiedosto sisältäen henkilöiden tietoja.

Ennen kehitystyötä, piti käydä läpi Web Partin määrittelyt ja vaatimukset. Vaatimuksiin kuului mm. käyttöliittymän yhtenäisyys sekä SharePoint-ympäristön että muiden Web Partien kanssa, joita asiakkaalle on kehitetty. Tämä onnistui käyttämällä kehityksessä UI Fabric-kirjastoa.

Kehitys alkoi asentamalla virtuaalikone sekä tarvittavat ohjelmistot ja työkalut. Web Partin pohja asennettiin käyttämällä Yeoman-työkalua. Tämän jälkeen piti suunnitella Web Partin struktuuria eli miten eri komponentit olisivat toisiinsa liitoksissa. Web Partia päädyttiin rakentamaan kuudesta eri komponentista. Jokaisella komponentilla on omat tehtävänsä. Tärkein näistä on pääkomponentti, joka sisältää kaikki muut komponentit ja jonka tilaa käytetään ohjaamaan Web Partia. Hakuparametriarvojen vastausvaihtoehdot haetaan rajapinnalta, jotka asetetaan pääkomponentin tilaan.

Kehitystyö sujui onnistuneesti. Kirjoittaja oppi React-työkalun käytön nopeasti, joka edesauttoi kehityksessä. Kehitetty Web Part toimi vaatimusten mukaisesti, ja se tullaan viemään asiakkaan testaukseen mahdollisimman nopeasti. Web Partin pohjalta voitaisiin myöhemmin kehittää asiakkaalle lähes vastaava ratkaisu, joka toimisi myös eräänlaisena hakukoneena.

Asiasanat

SharePoint, Web Part

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Aki Wilska	Bachelor of Engineering	September 2019

Thesis title

Developing SharePoint web parts

33 pages

Commissioned by

T-Base Oy

Supervisor

Marko Oras

Abstract

The purpose of this thesis was to develop a personnel search tool for a client using the Microsoft SharePoint platform's web part. The web part would contain numerous different search parameters that would be used to search the correct personnel from the database. The user could then pick the people they wanted to be put to a shopping cart, where they could download the personnel information in a compressed zip-archive.

Before starting development, it was necessary to go through the specifications and requirements for the web part. One of these requirements included same look and feel as the other web parts developed for the same client. This was achieved using the UI Fabric library.

The development began by installing a virtual machine and the required software. The web part was first scaffolded using the Yeoman tool. The structure of the web part was designed next. It was decided that the web part would consist of six components. Each of these components would have its own purpose. The main component was the most important. It holds all the other components. Its state holds the possible search parameters fetched from the API, and the parameters that the user would input. The search parameters will then be used to fetch the people from the API.

Successful development was achieved. The author quickly learned the use of the React, which helped with the development. The web part functioned as required and would be deployed for testing in the client's environment. Based on the developed web part, a similarly functioning search tool could be developed in the future.

Keywords

SharePoint, web parts, development, React

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

Acceptance testing	A form of testing where the client determines the quality of the product.
Ad-hoc testing	A form of testing that is done without documentation or planning. Its aim is to break the system being tested.
API	Application Programming Interface
Boolean	Variable type that holds simple true or false information.
CLI	Command Line Interface
CSS	Cascading Style Sheets
ES6	ECMAScript 6
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JSX	XML-like syntax extension for JavaScript
RAM	Random Access Memory
Tenant	An object that represents an organization in Azure Active Directory
UI	User Interface
URL	Uniform Resource Locator, also known as a web address.
UX	User Experience
Virtual Machine	Virtualized computer running its own operating system and software.
XML	Extensible Markup Language
Zip	Archive file format

1 INTRODUCTION

SharePoint as a platform is becoming increasingly more popular among companies around the world. While many of the basic features such as lists, charts, and embedded videos are easily available, more specialized web parts require a developer.

While the author of this thesis was working on their internship, this project was just a normal assignment. The author and the commissioner later discussed the subject of the thesis and it was decided that this subject would be suitable.

1.1 Objective

The objective of this thesis was to develop a SharePoint web part for a client. The web part would function as a personnel search tool. Given search parameters would be checked against an already existing database, and a list of people would be shown to the user. The user could then select the people they wanted and download their information in a zip archive.

This thesis focuses on the development process of a SharePoint Online web part using tools listed in the next chapter. The process involves the creation of a virtual machine, the installation of required tools and the development of the web part.

Source material used in this thesis is mostly the official documentation of each tool or technology. There are some sources with contradicting information which are both reviewed where they are used.

1.2 Commissioner

The commissioner of this thesis is T-Base, a small Microsoft certified IT-Consulting company based in Kotka, Finland. T-Base was founded in 1999 and has had hundreds of successful projects since then. Better known clients include companies such as ABB and Andritz. (T-Base s.a.)

2 TOOLS AND TECHNOLOGIES

Websites have evolved a lot in the last twenty years, and they are constantly evolving. They have become more beautiful and more widespread, but the complexity has also increased dramatically. Modern websites and their development use many different tools and technologies.

While web parts are not full websites, their design require the same type of complex design with many different technologies. Web parts are building blocks of a SharePoint page, and a single SharePoint page can contain many web parts.

This chapter lists the many tools required to develop the web part. Each section also includes a small explanation of what they are used for. More exact usage of each tool or software is explained in the next chapters.

2.1 Google Chrome

The world's most popular web browser, available for both mobile and desktop. According to different sources the usage of the desktop client ranges from 55.4% (W3Counter 2019), to 71.0% (statcounter 2019).

In this case, the client uses Google Chrome, which makes the choice of a browser easy. Google Chrome is also used for its extension support and easy-to-use developer tools such as a debugger and network traffic activity monitor.

2.2 Gulp

Gulp is a streaming JavaScript toolkit for automating, organizing and running tasks rapidly. It uses small instruction files that are easy to understand. These files can also be chained together to form many complex pipelines. (Maynard 2017.)

In this project gulp will be used to build the web app and serve it running Node.js server runtime.

2.3 Hyper-V

Hyper-V is a hypervisor-based hardware virtualization tool developed by Microsoft. It allows the creation, use and management of many simultaneously running virtual machines. (Microsoft 2016a.)

Hyper-V is used in this project to host a virtual machine. The entire development process is made within that virtual machine.

2.4 JavaScript

As described in MDN web docs (2005): “JavaScript is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions”. Most modern websites use JavaScript, as it allows the more dynamic design.

2.5 Node.js

Node.js is a free open source JavaScript runtime designed for scalable network applications (Node.js). Gulp uses it for its server runtime for hosting the web part locally.

Included in the Node.js installation is the npm CLI. npm is the world’s largest software registry that is accessed with the npm CLI. The registry is used to share open source software. Anyone can use the software freely to share and use JavaScript code. (npm s.a.)

npm is used in this project to download and install the required packages and dependencies.

2.6 React

React is an open source JavaScript library that is made for user interface design. It is a powerful tool that can be used to make websites simple or complex. There is also a mobile framework known as React Native that allows developer to easily create mobile applications using JavaScript and React. It is developed and maintained by Facebook. (React s.a.)

React comprises many concepts to make developing easier. A few of the key aspects are described below. The first them is JSX (example below, figure 1), an XML-like syntax extension for JavaScript. It combines the well-known JavaScript syntax with the HTML syntax. JSX produces React elements, which are the smallest building blocks of React. (React s.a.)

```
let example = <b>An example of JSX</b>;
```

Figure 1. JSX Syntax example

The second key aspect is React's component system. Components are in their simplest form a reusable, independent class. They can take in different properties known as props and have their own states. Props are arbitrary inputs for components. Example props could be anything from variables to objects or functions. A component cannot change its own props. State houses the component's internal variables and can be changed asynchronously at any time using the `setState` function. Every time the state is changed, the component and its children are updated. (React s.a.)

```tsx
export interface IExampleProps{
    txt: string; // Component requires the txt prop
}

export interface IExampleState{
    num: number // Component's internal state variable
}

class Example extends React.Component<IExampleProps,IExampleState>{
    constructor(props: IExampleProps){
        super(props);
        this.state = {
            num: 4 // Set the state variable 'num' to 4 in the constructor
        };
    }

    public render(): React.ReactElement<IExampleProps>{
        // Calling the render function returns the React element below
        return(
            <div>
                <span>Text value is: {this.props.txt}</span>
                <span>The number is: {this.state.num}</span>
            </div>
        );
    }
}
```

Figure 2. React Component with props and state

The “`Example`” class pictured above in figure 2 could simply be called like regular HTML or JSX. Figure 3 presents an example of this, where the “`txt`” prop value is inserted as shown.

```tsx
<Example txt="This text will be displayed on the Example component"/>
```

Figure 3. Calling the "`Example`" component

React was chosen for this project for its ease of use and because it is used in other web parts developed for the same client.

2.7 SharePoint

SharePoint is Microsoft’s web-based cloud platform for organizations of all sizes, it is used by over 200 000 organizations around the world. Collaboration is SharePoint’s most important feature. SharePoint is used to create websites,

share important files, and for team management. Communication and intranets are also top features of SharePoint. SharePoint is also available for mobile devices. (Microsoft 2019.)

SharePoint uses Microsoft's cloud services, although organizations with Office 365 Enterprise subscription can choose to host their own SharePoint Server on-premises. SharePoint hosted on Microsoft's cloud services is called SharePoint Online. (Microsoft 2019.)

The SharePoint Framework is a page and web part model providing support for client-side development and integration with SharePoint data. It is responsive and can handle any JavaScript framework. Users operate SharePoint Framework client-side solutions such as web parts or extensions. (Microsoft 2016b.)

2.8 TypeScript

TypeScript is a JavaScript superset that allows typed variables. It includes many of the features known to other programming languages that are not included in JavaScript such as interfaces and enums. It compiles into JavaScript, so that every browser supports it. TypeScript is open source and being developed by Microsoft. (Cherny 2019.)

2.9 UI Fabric

Microsoft (s.a.) explains UI Fabric as follows: "UI Fabric is a collection of UX frameworks that seamlessly fit into a broad range of Microsoft products". UI Fabric helps keep the UI consistent, and easily recognizable. It includes small, easy-to-use React components that are used in this project.

UI Fabric is used in this project for its 12-column responsive grid system, it's icons and it's easy to use components that all make it easy to seamlessly integrate it with SharePoint.

2.10 Visual Studio Code

Visual Studio Code is open source code editor with built in capabilities for extensions, debugging and version control (Visual Studio s.a.).

The web part is developed almost entirely within the Visual Studio Code editor. It has been the authors editor of choice for numerous other projects, making its use easy.

2.11 Yeoman

Yeoman is a scaffolding tool for web applications. Yeoman website also provides generators for many modern applications. Users can also create their own generators to share with others. (Yeoman 2014.)

Yeoman is used in this project to scaffold the SharePoint web part development. The SharePoint generator sets nearly everything ready for development.

3 SPECIFICATIONS AND REQUIREMENTS

Every software needs specifications before development. Without specifications the software will most likely not function as intended or how the client had desired. The specifications and requirements for the web part are explained in this chapter.

3.1 Web part specifications

The requirements and specifications were discussed and detailed between the author’s co-workers and the client. Afterwards they were handed out to the author.

The web part would be used by very few people in a desktop environment making mobile development redundant. The web part UI had to keep the look consistent with other web parts developed for the same client. It had to have a high number of search parameters given by the client. Given parameters

would then be compared against an already existing database using the API, and a list of personnel would then be displayed. The user could then pick the people they wanted to a “shopping cart”. The shopping cart would be a list of people picked by the user. The user could then download the people’s information to a zip-archive.

The client wished the web part to have three levels of search parameters. The first level would be only a few parameters that are used most often to be visible on default. The second and third levels would require the options to be opened with a button and the third level of options would require the second level to be open.

3.2 Web part requirements

The requirements for the project were the already existing database for storing the personnel information, the search parameters, and the also already existing API to connect to those. Another existing API was to be used for supplying the user with the zip-file. The APIs were secured using Azure Active Directory, connecting to them requires an API client, which was also an already existing part used in other projects.

The development of the APIs or the database is not a part of this thesis as they were made before for many other web parts developed for the same client.

4 DEVELOPMENT

The development process is detailed in this chapter. Starting from setting up a virtual machine, scaffolding the project and connecting the APIs to the project. The workbench is then extended, web part structure planned and finally, the web part components are developed.

4.1 Setting up a virtual machine

The virtual machine would be used to develop the web part. The virtual machine would need to run a high number of different pieces of software, so a high-performance virtual machine would be needed.

Hyper-V is easily installed to a Windows 10. Going through the Windows settings, a menu titled “Windows Features” or “Turn Windows features on or off” would then need to be opened. After selecting the Hyper-V option and restarting the computer Hyper-V would be ready to use. Hyper-V Manager is used to manage existing VM installations and create new ones. Hyper-V Manager interface is shown in figure 4.

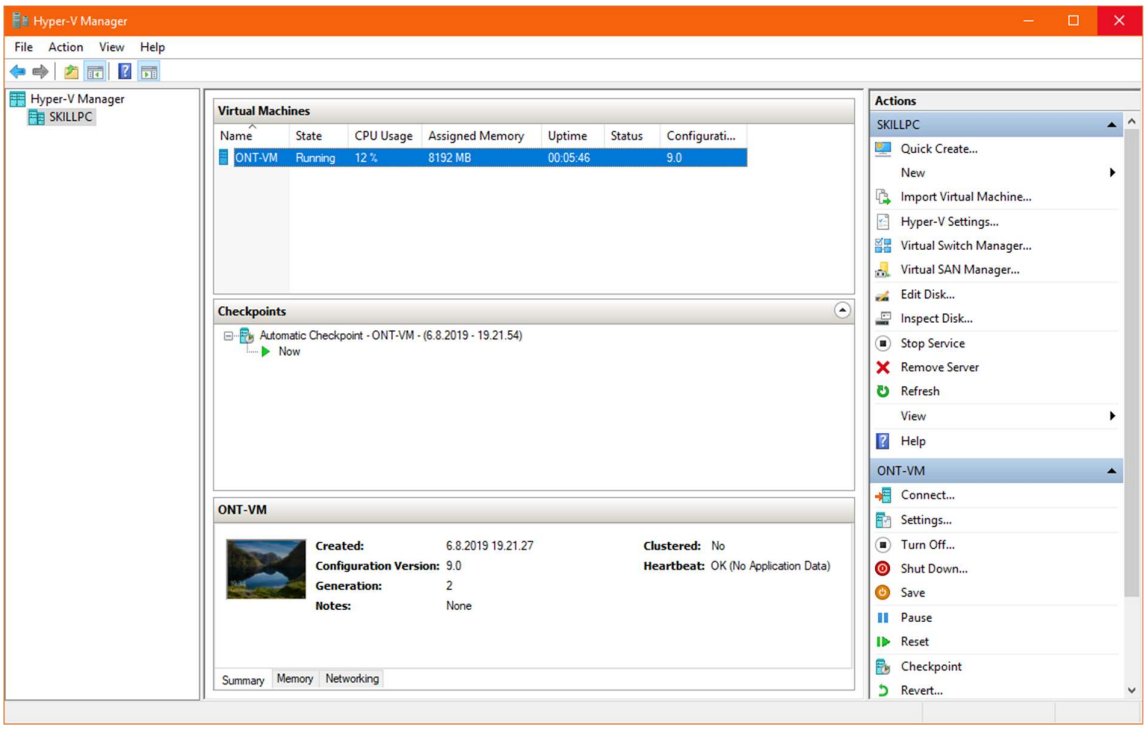

Figure 4. Hyper-V Manager User Interface

Creating a new virtual machine in Hyper-V Manager is rather straightforward. The “New Virtual Machine Wizard” instructs users from start to finish, asking questions such as the name for the virtual machine and where to store it. In this case, a virtual machine with Generation 2 option and 8192 MB of RAM is created and a Windows 10 bootable image file is installed. After the standard Windows 10 install process, the virtual machine is ready to use.

Required software is then downloaded and installed, starting with Google Chrome, Node.js and Visual Studio Code. With Node.js installed, access to npm is gained and Yeoman is installed by running the command: `npm install yo -g` in Windows command prompt. The "`-g`" flags the install to be global. Yeoman then needs a generator. Running the following command will install Microsoft's SharePoint generator: `npm install @microsoft/generator-sharepoint -g`. Last thing needed is gulp, it can be installed by running these two commands: `npm install gulp-cli -g` and `npm install gulp -D`. The virtual machine is then ready for development.

4.2 Scaffolding the project

The project is then scaffolded with Yeoman. This is done by running the command `yo` and navigating the menu to "Run a generator" and selecting the option: "`@microsoft/sharepoint`".

The generator will then ask several questions as seen in figure 2. Each question will have either a yes or no answer or multiple-choice options. After answering the last question "Which framework would you like to use", the generator will run and create the solution using given parameters.

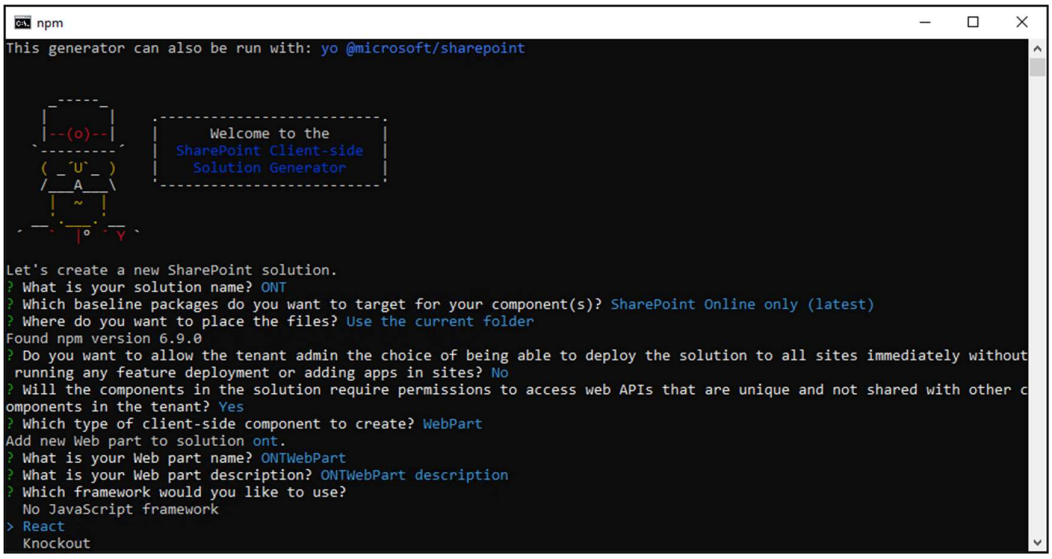

Figure 5. Yeoman running Microsoft's SharePoint generator

After scaffolding the web part, it can be tested by running the command `gulp serve`. SharePoint offline workbench will run, and the web part can be added to the site as shown in figure 3.

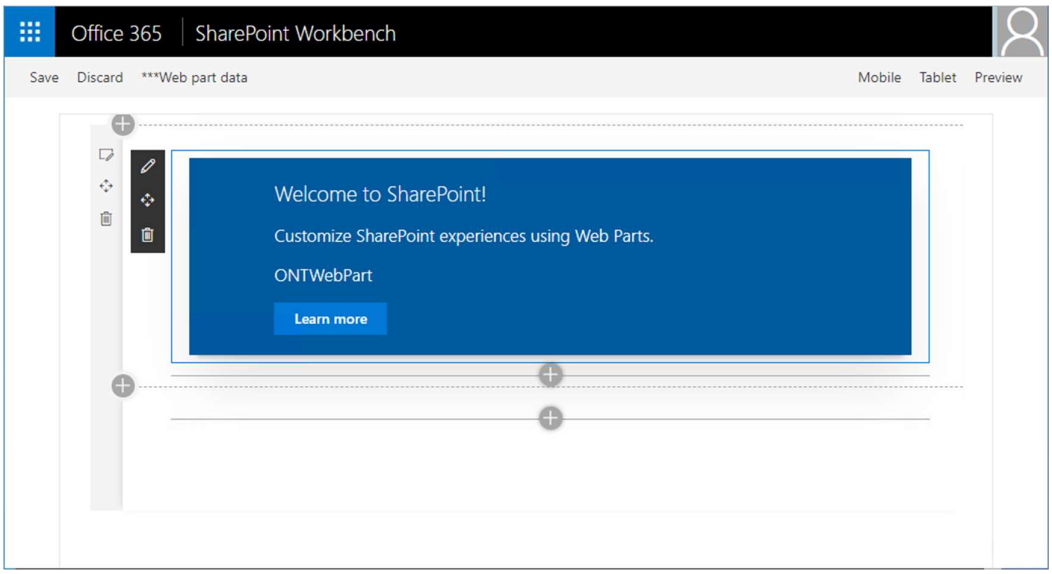

Figure 6. Default scaffolded web part

4.3 Connecting the web part with the APIs

The API client would connect to the already existing and configured APIs that are used in other web parts developed for the same client. The main API will supply the web part with multiple-choice question options configured in the database and get the personnel list when supplied with correct JSON syntax. The second one functions as the file API. When it is supplied with correct JSON syntax it archives and supplies the web part the compressed zip-archive containing the personnel files.

The APIs are secured by Azure Active Directory. Connecting to them requires signing into Office 365, that requires the use of the tenant's online workbench for the development purposes. The APIs use user impersonation to sign into required services using the user's credentials.

Programmatically the process is quite simple with the use of `AadHttpClient` class. After configuring the class, the class handles the authorization process automatically when signed into the online workbench or SharePoint Online.

Configuration of the class is not a part of this project, as it was part of the previously developed web parts made for the same client and was pre-configured for use in other web parts.

4.4 Extending the online workbench view

The online workbench used has the CSS attribute "`max-width`" set to 924 pixels. However, the web part will need more space than that allows. When the web part is deployed there is no such limit. What is needed is something that allows the workbench to be wider than the limit. For this, the choice was a Google Chrome extension Tampermonkey.

Tampermonkey enables the use of user scripts, small pieces of mostly JavaScript that can automatically edit the pages visited (Greasy Fork. s.a.). A small user script was developed (see Appendix 1). It searches through every readable stylesheet in the specified URL (the @match section, the URL has been censored in the image) searching for every CSS rule that sets the max-width value to "924px" and changes them to "". This nullifies the max-width rule and allows a wider online workbench.

4.5 Planning the structure

The web part structure is shown in figure 7. It consists of the main web part itself that has the main component `PersonnelSearchTool`, which includes the `SearchParameters`, `PersonnelList` and `ShoppingCart` react components. The `SearchParameters` component includes the `ParameterList` component inside, and that component could have as many `ParameterListItems` inside as the user wanted.

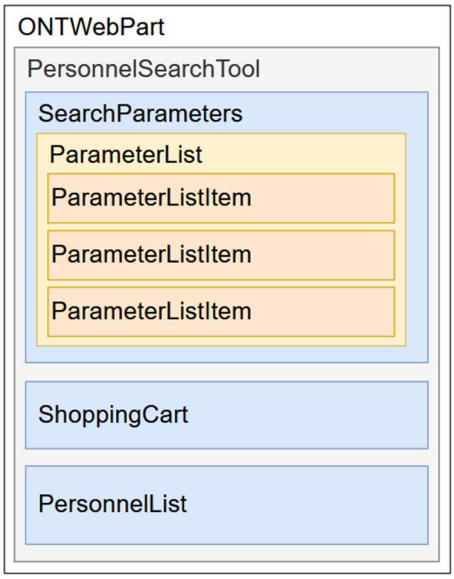

Figure 7. The web part's react component structure

To keep the web part UI clean, a few different options were tried at it was decided that four parameter options per row is a good choice. It was also decided that to help development, every class and component would have its own interface that would contain properties that had to be implemented in the component or class. Component's state and prop interfaces would be implemented in the same file above the components. Model files such as those for the search parameters are made according to the database information. These model files would all have their own folder.

A model interface would have to be structured as in the database. For example, the `ICurrency` interface would contain the following values

- `code: string`
- `id: number`
- `name: string`

By creating an interface for `ICurrency`, its value is easily set by the API. Interfaces can also contain other interfaces. For example, the `ISearchParameterChoices` interface contain every single interface for every search parameter.

4.6 Developing the components

Every component's default values are set in the constructor along with "`super(props)`". Without calling that, `this.props` is undefined in the constructor. The keyword "`this`" in JavaScript refers to different objects, which JavaScript is very strict about. The keyword is then bound to the component's functions, so they have access to it. An example of the binding is shown in figure 8. (React.)

```
this._deleteFromList = this._deleteFromList.bind(this);
```

Figure 8. Binding functions

Every text value in the components should be set in the locale files, then imported as a module, making future localization easy. By default, the locale files are in the folder called "loc" that contains "mystrings.d.ts" file and English localization file "en-us.js". To add a text to be used in the components, it is first defined in the interface located in the "mystrings.d.ts" file, and then added to the desired language localization file. To add a different language, the localization file could be copied and renamed to a different locale according to ISO standards and then the text inside could be changed. Example files are shown in figure 9.

Ad-hoc testing would be the primary testing method during development. After developing a component or a part of one, the web part would be run and quickly tested. Problem solving would be handled with the included debugger in Google Chrome DevTools.

mystrings.d.ts

src > webparts > ontWebPart > loc > mystrings.d.ts > IOntWebPartWebPartStrings

```ts
declare interface IOntWebPartWebPartStrings {
  ExampleString: string;
}

declare module 'OntWebPartWebPartStrings' {
  const strings: IOntWebPartWebPartStrings;
  export = strings;
}

```

en-us.js

src > webparts > ontWebPart > loc > en-us.js > ...

```js
define([], function() {
  return {
    "ExampleString": "This is an example.",
  }
});
```

fi-fi.js

src > webparts > ontWebPart > loc > fi-fi.js > ...

```js
define([], function() {
  return {
    "ExampleString": "Tämä on esimerkki.",
  }
});
```

Figure 9. Localization example

The text values could then be called as shown in figure 10.

```
<span>{strings.ExampleString}</span>
```

Figure 10. Localized string variable being called

4.6.1 UI Fabric components and styles

UI Fabric contains many of the common components used in this project. Most of them are simple and easy to use. Listed below are all the components used in this project, and what they are. Their uses are explained in the component sections where they are used.

- `IconButton`, an icon that is used as a button.
- `DefaultButton`, button with borders. Can be with or without an icon.
- `TextField`, an input field.
- `ComboBox`, choice options picker, can be set to allow multiple choices.
- `DatePicker`, a date picker.
- `Icon`, an icon.
- `Label`, a label.
- `List`, contains a list of JSX elements.
- `DetailsList`, same as list, but in preset columns.
- `Spinner`, a spinning icon presenting the that the component is processing.
- `Panel`, an openable side panel.

The 12-column responsive grid is the backbone of the UI in this project. Every search option is configured to be specific width of the using these columns. For example, if a component was set to be three columns wide, it would have anything from a single nine column wide to multiple one column wide components next to it and be on the same row. If the width of a row exceeded 12 of these column widths, the reactive grid adapts and sets the available components to the next row.

4.6.2 Main Component

The main component contains the all the other components, so it was developed first. Its state contains the following variables:

- `errorText`, any errors in operating the web part would be passed to `errorText`, which would be shown to the user.
- `searching`, contains the status of whether the web part is in the process of fetching the personnel list from the API.
- `searchingParameters`, contains the status of whether the web part is in the process of getting the search parameters from the API.
- `cartOpen`, contains the status of whether the shopping cart interface is open.
- `searchParameters`, contain’s the data of search parameters.
- `searchParameterChoices`, contain’s the data of the search parameter choices available.
- `items`, contains an array of the people fetched by the API
- `cartItems`, contains an array of people selected from the `PersonnelList` component.

When the `componentDidMount` function is called, the state boolean value `searchingParameters` is set to `true`, and the component starts fetching

the parameter options from the API and setting the values to the state. When the values have been fetched, the value is set back to `false`. This boolean value is used to display the user a different view while fetching. This is done by checking the value in an if-statement in the render function shown in figure 11.

```
if(this.state.searchingParameters){
  return(
    <div>
      <span>{strings.Searching}</span>
    </div>
  );
}
```

Figure 11. Displaying a different view depending on the value of the state `searchingParameters` parameter

The main component contains the `SearchParameters` and the `ShoppingCart` component by default. Conditional checking is used to define whether to display the `PersonnelList` component. If the API client is currently fetching the list of people, the message “Searching, please wait…” is displayed. If the API returned an empty array, and no results were found, the message “The list is empty” is displayed. Only when the items array has more than one item in it, the `ShoppingCart` component is displayed.

Since the search parameters live in the state of the main component, most of the functions in the component are for handling them. Every time a property is changed, it must be updated to the state. Besides handling the search parameters, the component also has functions for various other features that are controlled by other components such as opening and closing of the shopping cart, adding or removing items from the shopping cart and sorting the personnel list.

These functions are called from different actions or buttons that are in other components, but their control is set in the state of the main component. Since changing the parent components state in React is not possible, the child must be given a prop that calls the parent function. This is done by writing a function in the parent that changes the state, and then set that function as a prop

to the child component. Every time the parents state must be changed, this is how it is done.

4.6.3 SearchParameters component

This component houses every input the user can use to search personnel. The props for the component are:

- `searchParameters`, to display the value set in each input.
- `searchParameterChoices`, to display the available options in each drop-down menu.
- `cartHasItems`: boolean, to display whether the shopping cart icon as empty or not empty.
- `onChange`, this function is called whenever any of the inputs is changed. It updates the parents `searchParameters` state to hold the new inputs given by the user.
- `openCart`, this function is called whenever the user clicks on the open basket button. It updates the parents state to toggle the boolean variable `cartOpen`, which in turn opens the panel that holds the cart.
- `searchPeople`, this function is called when the user clicks the search-button. It causes the API to start fetching people using the `searchParameters` and update the list to parents' items state.
- `removeFromList`, this function removes the selected item from the `SearchParameters' ParameterList`.
- clear, this function clears all `SearchParameters`.
- `clearList`, this function clears all `ParameterListItems` from the `ParameterList`.

The state holds two boolean variables to whether to display more inputs to the user. These are controlled with UI Fabric `IconButtons`. By default, the user is only displayed five options. Should the user need more options they can click the button to show more options. This is done similarly to showing completely different contents in the component such as in figure 11. In this case, the if-statement is in the return function checking the variable and return an empty `<div/>` element if the variable is false.

Many of the `TextField` inputs have validation function set to the `onChange` function. The `onChange` function takes a parameter to define if the input must be a number. If this is the case, the new input is not updated and the `TextField` shows no change.

The option choices must be mapped to each `ComboBox` components, so they can be shown to the user and updated to the parents' state. Figure 12 presents one of the `ComboBox` components used. It controls the variable "currency" in the `searchParameters`. The `onChanged` prop in the selected "key" that is then updated in the parent to `searchParameters`. The mapping part in the options prop uses the id and code values located in the database to view correct choices in the `ComboBox` component. The final look of the component is shown in figure 13.

```
<ComboBox
    multiSelect={false}
    useComboBoxAsMenuWidth={true}
    onChanged={(o) => this.props.onChange("currency", o.key)}
    selectedKey={searchParameters.currency}
    options={searchParameterChoices.currencies.map((v,i) => {return {key: v.id, text:v.code};})} />
```

Figure 12. ComboBox mapping

Figure 13. The 'Currency' ComboBox component

4.6.4 SearchParameterList and SearchParameterListItems components

The `SearchParameterList` component has no internal state variables, and its props are only the array of items to be on the list and a function that can remove selected item.

It holds the UI Fabric component `List` inside. By default, the `List` component renders only the items that are shown to the user. For example, when the user is scrolling the page, the list is constantly updating what is shown using the `onRenderCell` prop function. The function is called when trying to render each single item from the array. The function is set to return a `SearchParameterListItem` component.

The `SearchParameterListItem` component takes the same prop function as its parent, used to remove the specific item from the list. It also takes its own index in the list and the item values to show to the user.

When mutating objects in the list, the list is not automatically updated. The items array must be recreated to re-render the list using the mutated items. To do this an open source JavaScript library *object-path-immutable* was downloaded from npm. It allows the modification of an object without modifying the original (object-path-immutable). Using it in the function `deleteFromList` becomes as shown in figure 14.

```
private _deleteFromList(index:number){
  const items = immutable.del(this.state.searchParameters,`items.${index}`);
  this.setState({searchParameters:items});
}
```

Figure 14. The function that deletes items from the list

4.6.5 PersonnelList component

The purpose of `PersonnelList` components was to hold the list of people fetched from the API. Its props were:

- `items`, the array of items retrieved from the API.
- `cartItems`, the array of items that are currently in the cart, used to display people already in the cart.
- `toggleCart`: when a cart icon is pressed in each item, it will add/remove it from the cart using this function.
- `sort`: function that sorts the items array using the desired parameter.

Instead of using the List component from UI Fabric, `DetailsList` was used. `DetailsList` component allows the use of columns in the list. Using columns was important to make the UI look and feel simple to use. Columns must be defined using a variety of different parameters. Example of the "`lastname`" column definition is shown in figure 15.

```
{
  key: 'lastname',
  name: strings.Lastname,
  fieldName: 'lastname',
  minWidth: 75,
  maxWidth: 150,
  isRowHeader: true,
  isResizable: true,
  isPadded:true,
  onColumnClick: () => {
    this.props.sort('lastname');
  }
},
```

Figure 15. lastname column definition

The columns defined for the list were the persons first and last name, email address and a shopping cart icon. The icon displayed would be conditional on whether the person is the shopping cart or not. This is done with a local function that quickly checks for each item if they are in the shopping cart. A white cart icon would be displayed for people not in the cart, and a black one for those in the cart. Example of the list is shown in figure 16 with a person in the cart.

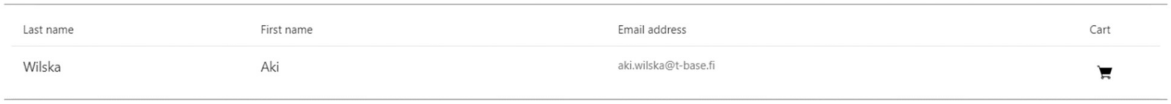

Figure 16. Shopping cart with a person in the cart

4.6.6 ShoppingCart component

The final component in the web part is the `ShoppingCart` component, which holds the shopping cart. The shopping cart is simply a list of people added from the `PersonnelList` component. Its state holds `creatingFile`, a single boolean variable which checks if the component is in the process of waiting for the API. While the `creatingFile` variable is set to true, a small spinning icon is displayed along with a message: “Creating file, please wait”. This is done similarly to the if-statement displayed in figure 11. The components props are:

- `items`, the array of people added to the cart.
- `showPanel`, boolean value used to determine whether to display the cart or not.

- `removeFromCart`, function that removes selected item from the cart.
- `clearCart`, function that removes all items from the cart.
- `hidePanel`, function that hides the shopping cart panel.

The list is in a UI Fabric component `Panel`. It is opened from the button in the `SearchParameters` component. `Panel` is a UI overlay that blocks input from outside the panel. It can be anchored to left or right side of the screen, or it can be the width of the entire page. Its height is always the same as the visible page. In this case it is anchored to the right side of the screen.

The component uses the same `List` component from UI Fabric as the `SearchParameterList` component. The lists `onRender` function simply shows the person's first and last names as well as a button, to remove the person from the list. Component is shown in figure 17 with the same person as selected in previous section.

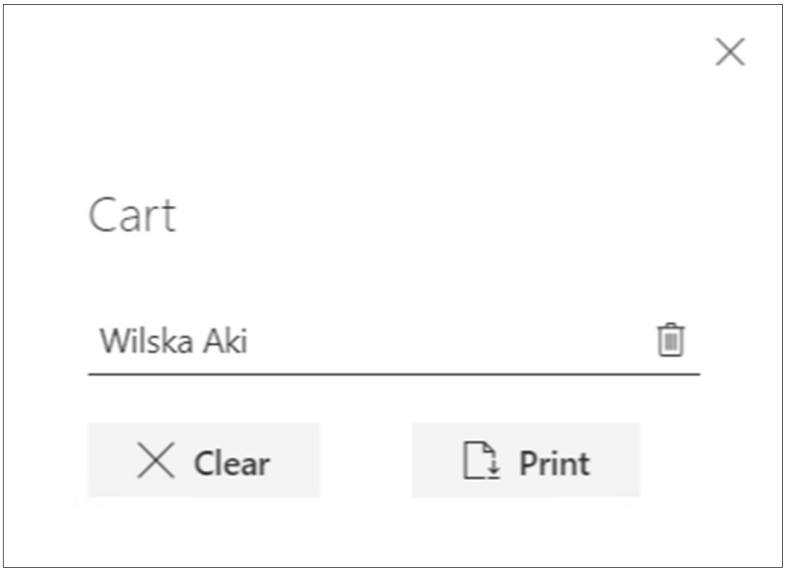

Figure 17. The `ShoppingCart` component

Downloading the personnel information is done by pressing the "Print" button. It calls the `print` function, which sets the state variable `creatingFile` to true and parses the items array into JSON format. The JSON is then used to fetch the file containing the information from the file API. After downloading the file, the `creatingFile` variable is set to false, and the list is cleared.

5 CONCLUSION

The finished web part was put through ad-hoc testing. When all the major bugs were cleared, the web part could be deployed to the client's environment to be put through acceptance testing. The web part could then later be localized for other languages.

The development of this web part was the authors first React project, so there was a learning curve. For the author React was adopt, the Reacts website includes a small tutorial that was helpful. After completing the tutorial and developing the first component, the rest were quite easy.

The documentation of most of the tools was clear and simple. However, at the time of the development the UI Fabrics List components documentation was not quite on the same level, and the author had some time-consuming difficulties with trial and error. The documentation has since been updated and is easier to follow. The finished web part is shown in figure 18. It should be noted that most parameter names are not shown.

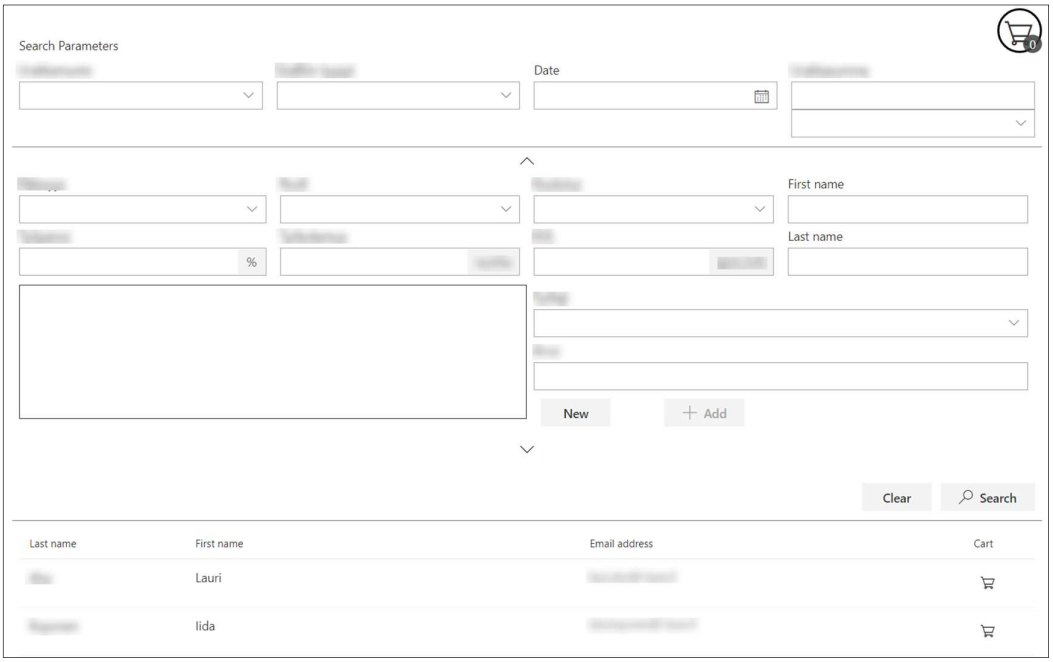

Figure 18. Finished web part

REFERENCES

Cherny, B. 2019. Programming TypeScript. E-Book. California: O'Reilly Media. Available at: https://learning.oreilly.com [Accessed 13 Aug 2019].

Greasy Fork. s.a. What are user scripts? WWW-document. Available at: https://greasyfork.org/en [Accessed 13 Aug 2019].

MDN web docs. 2005. JavaScript. WWW-Document. Updated 25 Jul 2019. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript [Accessed 25 Jul 2019].

Maynard, T. 2017. Getting Started with Gulp – Second Edition. E-book. Birmingham: Packt Publishing. Available at: https://learning.oreilly.com [Accessed 13 Aug 2019].

Microsoft. 2016a. Hyper-V Technology Overview. WWW-Document. Updated 4 Nov 2018. Available at https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview [Accessed 8 Aug 2019].

Microsoft. 2016b. Overview of the SharePoint Framework. WWW-Document. Updated 8 Jan 2018. Available at: https://docs.microsoft.com/en-us/sharepoint/dev/spfx/sharepoint-framework-overview [Accessed 20 Aug 2019].

Microsoft. 2019. SharePoint, Team Collaboration Software Tools. WWW-document. Available at: https://products.office.com/en-us/sharepoint/collaboration [Accessed 8 Aug 2019].

Microsoft. s.a. UI Fabric – Get Started. WWW-document. Available at: https://developer.microsoft.com/en-us/fabric#/get-started [Accessed 8 Aug 2019].

npm. s.a. About npm. WWW-Document. Available at: https://docs.npmjs.com/about-npm/ [Accessed 1 Aug 2019].

object-path-immutable. 2015. GitHub. WWW-Document. Updated 21 Jun 2018. Available at: https://github.com/mariocasciaro/object-path-immutable [Accessed 22 Aug 2019].

React. s.a. A JavaScript library for building user interfaces. WWW-document. Available at: https://reactjs.org/ [Accessed 25 Jul 2019].

statcounter. 2019. Desktop Browser Market Share Worldwide. WWW-document. Available at: http://gs.statcounter.com/browser-market-share/desktop/worldwide [Accessed 6 Aug 2019].

T-Base. s.a. Microsoft-sertifioitunut IT-asiantuntijatalo. WWW-document. Available at: https://www.t-base.fi/ [Accessed 24 Jul 2019].

Visual Studio. s.a. Visual Studio Code. WWW-document. Available at: https://code.visualstudio.com/ [Accessed 8 Aug 2019].

W3Counter. 2019. Trends. WWW-document. Available at: https://www.w3counter.com/trends [Accessed 6 Aug 2019].

Yeoman. 2014. The web's scaffolding tool for modern webapps. WWW-document. Updated 25 Apr 2019. Available at: https://yeoman.io/ [Accessed 1 Aug 2019].

LIST OF FIGURES

Appendix 1

```
// ==UserScript==
// @name         Sharepoint Online Workbench Width Extender
// @namespace    http://tampermonkey.net/
// @version      0.1
// @description  try to take over the world!
// @author       You
// @match        [redacted]
// @grant        none
// ==/UserScript==

(function() {
    'use strict';

    let unread = [];

    let ss = document.styleSheets;
    let searchStr = "max-width: 924px; ";

    for(let i = 0; i < ss.length; i++){
        let sheet = ss[i];

        if(sheet.href == null)
        {
            unread.push(i);
        }
    }


    unread.forEach((i) =>
    {
        let sheet = ss[i];
        for(let j = 0; j < sheet.cssRules.length;j++){
            let rule = sheet.cssRules[j].cssText;
            if(rule.indexOf(searchStr) > -1){
                sheet.cssRules[j].style.maxWidth = "";
            }
        }
    });
})();
```

Figure 1. SharePoint Online workbench width extender userscript