Jarno Koppala

ERP Solution with ReactJS

Helsinki Metropolia University of Applied Sciences

Master's Degree

Information Technology

Master's Thesis

22 March 2018

Metropolia

PREFACE

This thesis includes a study with a modern application programming language from the perspective of business application needs. It took hundreds of hours of learning in an adequately level of programming to achieve the study objectives. I would like to thank my supervisor Kari Virtanen at work and my instructor Ville Jääskeläinen at the Metropolia University of Applied Sciences. This thesis gained me a new skill of programming and the school provided an updated perspective to information technology.

This thesis taught me how valuable time actually is after working with the master thesis and master thesis project for two years. This thesis would not be possible without the unbelievable family support throughout this journey. Special acknowledgement to my wife, who has worked towards my goal as many hours as I when taking care of our family on a daily basis.

Sincerely grateful,

Helsinki, 22.3.2018
Jarno Koppala

Metropolia

Author Title Number of Pages Date	Jarno Koppala ERP Solution with ReactJS 53 pages 22 March 2018
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Kari Virtanen, Technical Manager (Helsinki City Rescue Department) Ville Jääskeläinen, Head of Master's program (Metropolia)

Helsinki City Rescue Department performs twelve hundred periodic fire inspections and more than a thousand other supervisory tasks on a yearly basis. The Helsinki City Rescue Department uses a self-preparedness auditing model with fire inspections. These inspections and other tasks are monitored and reported with a self-build ERP solution. The current ERP solution has problems related to technology in use and this thesis research objective was to scrutinize a suitable replacement technology.

The master thesis project was a research and development project of the selected JavaScript technology called ReactJS. The thesis main objective was to develop an application with agile developing methods. The research phase of the study compared different state management technologies for the ReactJS and the selected Redux state technology fulfilled all needed features for the application.

The study conducted and presented best practices of a secure development methodology and related findings in the field. The theory part covered key aspects of a secure web application development and these best practices were utilized in the web application development-phase. Section 5 covers the development phases and an environment configuration. The web application development started with designing mock-ups and user flow diagrams. These design plans assisted to the agreed goals and helped the discussion with the organization. An agreed plan was a key for the successful project. The study project excluded a production ready application and features related to authentication and the EU general data protection regulation.

The selected JavaScript technology called the ReactJS was a success. The technology offered a good approach to the application development with its extensive capabilities. The technology is suitable for designing the production ERP application. The projects outcome was a success and all agreed features were implemented. The evaluation from the organization was positive and the solution developed with the ReactJS answered to the organization requirements.

Keywords	JavaScript, ReactJS, Redux, MySQL, OWASP

Contents

List of Figures

List of Abbreviations

ACID	Atomic, Consistent, Independent, and Durable
ATDD	Acceptance Test Driven Development
BDD	Behavior Driven Development
CORS	Cross-Origin Resource Sharing
DOM	Document Object Model
DRY	Do Not Repeat Yourself
ECMA	European Computer Manufacturers Association
ERP	Enterprise Resource Planning
HOC	High Order Component
JSON	JavaScript Object Notation
KEHMET	City of Helsinki Agile framework
OWASP	Open Web Application Security Project
REST	Representational State Transfer
SPA	Single Page Application
SSO	Single-Sign-On
TDD	Test-Driven Development
W3C	World Wide Web Consortium
QA	Quality Assurance

1 Introduction

Web developing methods and technologies evolve tremendously. During last few years the phenomenon in evolving web framework languages is outstanding. New enhanced web frameworks make web development closer to an application standard of dynamic views and responsiveness. In businesses, the amount of applications that needs an upgrade in terms of user experience and lifecycle point of view is growing.

1.1 Case Company and Technology Challenge

Helsinki City Rescue Department is one of the largest fire departments in Finland. More than a twelve hundred fire inspections are done yearly and to keep up-to-date status with the fire inspections is an essential function in the company business. In Rescue Act (379/2011), the monitoring and reporting responsivity is set to the Rescue Department. More than seven thousand actions related in fire inspection field were performed in 2015. When organizing one hundred employees working hours and monitoring the results in details an Enterprise Resource Planning (ERP) system is needed.

Helsinki City Rescue Department is using a self-build ERP solution for maintaining a record of person's duty hours and supervision score points. The resource efficiency is monitored to help managers to make decisions of workload in different processes. The self-build solution has come to its end when considering the used technology and the current solution does not perform in production as well as it should. The application has been built on Microsoft Access technologies and the next development step is to make the application work as a browser based application. In the beginning of 2017, the software database was moved from Access database to Oracle MySQL server but the frontend remained in Access database.

These are few problems related to Access usage. The software itself is not stable; the database can become corrupted when opening the system or a user can input empty data without necessary details. The current ERP system has also serious problems with latency when there are multiple concurrent users and a great number of data transactions in the system. The ERP system frequently corrupts the frontend Access databases when a user does not log off from the system and a computer goes in a sleep

mode. This is one of the challenges with Access technology. Access does not handle the user sessions correctly when e.g. a computer suddenly goes in a sleep mode; it instead corrupts the opened database.

1.2 Scope and Objectives

Main objective of this thesis is to develop a new dynamic web user interface for the ERP solution needs. The new developed web application interface should have a user search view for selecting the current user, a table view for handling records, an input form with validation and a user record report view. Other features are presented in section five. A validation of web application frontend form must have the at least the same business logic in calculations as database for displaying correct values to the user.

It was important to recognize the features that were excluded from this project. A list of excluded features helped the dialogue with the organization. In this master thesis project these features were listed as next phase features. These excluded features were an Active Directory authentication, a full application capability for EU General Data Protection Regulations, user management and supervisor report views with excel compatibility.

Other study objectives were to perform a comparison of ReactJS state containers for data streaming capabilities, a technology switch for current production application and a detailed documentation for the application functionalities and database structure. The project research section of the master thesis compares three different data stream models for ReactJS. Software development also requires a planning with realistic expectations when seeking best practices. Virtual development environment for the actual application development and testing is also needed and it is introduced in section 5.

1.3 Approach and Method

This thesis uses a qualitative case study method as an approach. In the study, the current ERP solution is first analyzed and application processes are defined. The actual project management part and coding was developed with agile methods and applying KEHMET

(Helsingin kaupunki, 2017) principles. This case study was conducted with KEHMET Alpha model framework. The approach to the challenges was to utilize Alpha model test in practice can an ERP application work as a web based application. The case study also produced detailed information about the limitations and answered the question whether the technology is capable enough. The research phase included a comparison between suitable ReactJS state management solutions. The case study also collected organization feedback and prepared organization to cost estimates for the future development. After the Alpha model phase, the organization is able to decide whether to continue to the Beta development, discontinue the project or start a new Alpha with a different type of a solution.

In the thesis, the first developed concept model plan is presented. The outcome of the thesis also includes a detailed documentation of the self-build ERP system. The solution design was carried out as a SPA (single page application) and therefore dynamic web framework technologies were needed to achieve that goal. The new ReactJS based web application was delivered in a relative short period of time and the project had some room for innovating throughout the solution. Later in the thesis the development steps are covered and the idea was first to define and deliver a minimum viable product (MVP). Development steps were done in eight small sprints and project always delivered a production ready solution. After the weekly delivery sprints, the product owner evaluated the developed application and then the iteration started again. In the weekly planning and review meeting of the project next week scope for desired feature set was planned, hour report for delivered features was presented and evaluation report was documented. Projects actual outcome was evaluated every week at sprint review with a live demo.

This report is written in six sections. First, there is an introduction and then the section 2 covers the background of the current system solution and the section presents the issues related to the system and usability. Section 3 introduces different technologies in web development framework for front and backend solutions. Section 3 also analyses quality measures related to software developing. Section 4 covers the software security implementation and vulnerability risks are dealt in more detail. Security aspects are extremely important when developing a new web application. Section 5 introduces the suggested implementation weekly plan and predesigned screens for the project. The application project work that was conducted is introduced in section 5. Section 6 reviews and analyses the outcome and sums up the conclusions.

2 Solution Background

In this section, the current enterprise resource planning reasons are described. The ERP application reporting capabilities are presented and the access control explained. The ERP systems key features and capabilities are discussed in the sub section 2.3. In the last part of this section the current solutions design is illustrated and problems are presented.

2.1 Enterprise Resource Planning (ERP)

The original justification for the ERP application in Helsinki City Rescue Department was when the Rescue Commander wanted more focus on risk management measures for ensuring operative conditions. The ERP solution within organization with scoring points makes easier to set overall score achievement based goals. The ERP system also conducts balance of the workload between workers and monitors the efficiency of duty hours.

The ERP system makes possibility to organize resources and see if the goals are achieved with current work force. The main feature of the ERP system is a possibility to set upcoming inspections and sort them with given dates and priority. The application value is at best when the end users inputs all of the information in a normalized form. In current state of the system, the alarms for upcoming reminders are not implemented. Earlier in 2014 the score points were valued with excel sheets but the solution was needed to improve. In the end of 2015, the next solution was carried out with Microsoft Access Database and Access Frontend user view for handling login, views and forms. It took approximately three months from Excel sheet based system to have the ERP running in the Microsoft Access. Originally some of the business logic calculation from excel sheets were implemented to the Access solution. Later the score point and point-weighted calculation were refined in Access database.

The ERP solution was designed to control users who work in field and accomplish score points related tasks or inspections. In the current system there are three usage roles; user, supervisor and administrator. In the user role, the number of workers are approx. one hundred users, six persons in supervisor role and one person in administrator role. The ERP system assists supervisors to organize their tasks in overseeing overall status

and report the results to the board of directors. The ERP system guides the team leaders who can see their own team status with score point's achievements.

Score points are collected from many different areas and tasks. Areas that can be measured in score points are safety management, accident risk management, document production related to safety, structural fire safety, safety technology and safety communications and competence. All tasks related to the monitored areas are reported and inputted to the ERP system and then the score points can be calculated.

The fire inspections are carried out with a self-prepared auditing model. Auditing model is originally based from the Rescue Act (379/2011) that was stated in 2011, but it was further developed in Helsinki City Fire and Rescue Department. (Finnish Goverment, 2011) Within the ERP system, the inspection task target auditing scores are visible to the user and scores are weighted based on the auditing model.

2.2 Reporting Capabilities

In the ERP systems, end users mark their inspection results on daily basis. The supervisors can alter all inputted points and correct possible faulty inputs. The supervisor group also maintain standard and planning phase information for the fire inspections. Only the supervisor can use the monitoring views in the system. The supervisors are typically senior fire inspectors who lead their own geographically divided team. The competence center also monitor score points in area of education and scheduled fire inspections. Based on the score point the competence center can evaluate the focus on education needs.

The ERP system produces few different kinds of an excel sheet reports for further usage. The reports to the board of directors are manually exported from the system and summarizations are followed quarterly and yearly basis. In management level, the reports are monitored at monthly basis. The monitor reports are exported to the Excel with the supervisor role.

2.3 User Interface Design and Functions

The current Access based ERP solutions user interface has several functions. At home screen, the end users upcoming and incomplete tasks are displayed. The upper part of the home views frame includes incomplete tasks and lower frame box users finished tasks as presented in the Figure 1. At home screen there are few navigation possibilities to make e.g. a new task input button and a link to user's private workforce management view. The user can also refresh the home screen's data view with a refresh button and click log out button when exiting the application.

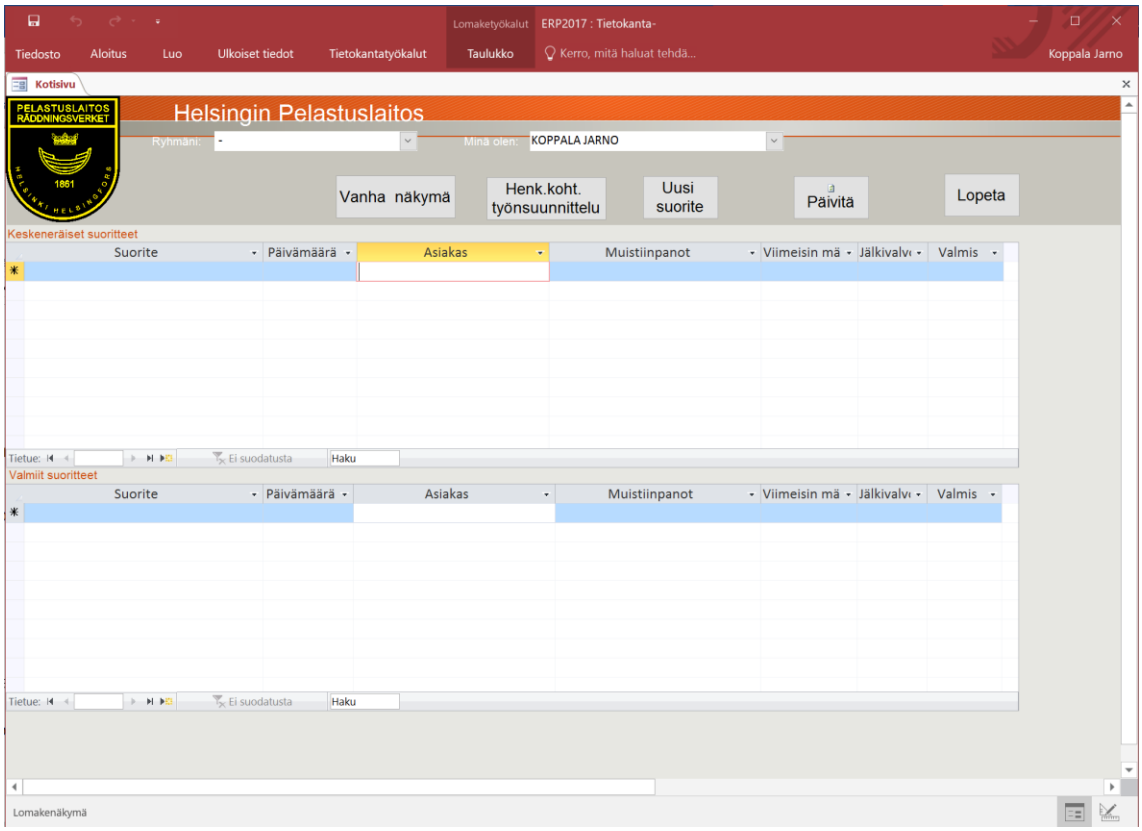


Figure 1. Home screen of the access based ERP system.

Interface follows the limitations of Microsoft Office Access solution being straightforward and similar to the original Excel-sheet. In the navigator bar, the user is selected from a list of all users and there are no user privilege limitations for controlling these changes.

2.4 Problems with Current Architecture and Design

The Microsoft Access causes a few problems related to usage. The views can be distorted quite easily with different machine setups, the database can corrupt frequently and the system comes unavailable from the end users after the error state has occurred. This error state has occurred several times in previous 12 months of usage. The error state is presented in the Figure 2. One systematic problem is when the user adds more personal tasks, and if during the inputting the form it is cancelled abruptly with the cancel-function it adds a data row in to the database with null information. The system administrator then needs to clean up the database from the extra rows and the maintaining has to be done regularly. The current Access ERP needs a MySQL Access plugin from the user workstation to work so it does not work just from any computer. The Access database plugin is installed per user profile and it causes installation effort with publically used computers. The Microsoft Access and the related MySQL database plugin has to be remembered to install when a user computer is changed.

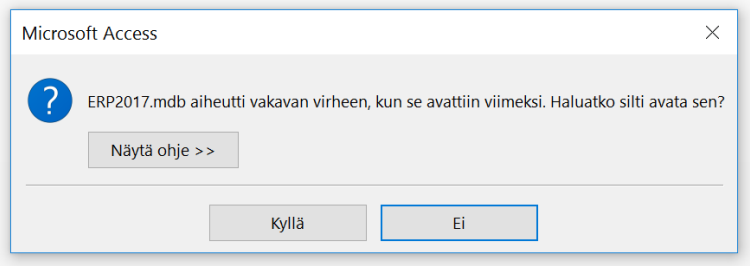


Figure 2. Screen capture of error state.

The new selected technologies will solve these Access based issues completely. The web-based user interface does not require any plugins from the company's end user computer environment. In the new applications form validation main objective is to obstruct faulty inputs.

3 Software Engineering Methods and Technologies

In this section an agile frameworks are presented related intensively to the master thesis project. The section 3 covers a quality and test driven development. The ReactJS technology is presented in detail. The qualitative research study consisted a state and data management analyze for suitable and selectable technology in this master thesis project. The section 3 presents different data flow technologies for JavaScript applications and especially suitable for ReactJS technology. The latter part of this section cover data access layer technologies.

3.1 Developing Methods

The application development requires a knowledge of quality development methods. There are several different developing methods available such as Agile, SCRUM, Waterfall, Kanban etc. This case study used City of Helsinki organization own KEHMET developing framework for principles and guidelines.

KEHMET

KEHMET has been developed by City of Helsinki and the framework has been awarded first price with Finland Ministry of Finance SuomiDigi award. The KEHMET framework includes agile methods from small to large projects in system architecture, information security, confidentiality and guidelines for procurements. These main topics are not separate projects and following with the KEHMET methods the main subjects are taken into account. The KEHMET model propose a possible solution in government environment for preparing and delivering the procurement. The KEHMET is designed for product owners and project leaders such as persons who are responsible for performing the change. The KEHMET method is especially focused on the customer needs and it scrutinize the change on all aspect. The method characterized the actual planning to hands-on level. (Helsingin kaupunki, 2017)

The KEHMET framework core concept for agile method is to have two main phases: agile planning and agile implementation. The agile planning is divided into Research and Alpha phase. Within Research phase, e.g. customer needs, service concept and goals are evaluated. The Research phase also carries out developing feature list and the scope

is set for the project. In the Alpha phase, the Research outcome is actually developed and tested. The Alpha tests can be redone as an individual efforts multiple times until the goals are achieved. After the Alpha test, the customer feedback is collected and the next conclusions can be done based on results. (Helsingin kaupunki, 2017)

The KEHMET frameworks agile implementation is divided into two phases; Beta and Live phase. The Beta phase includes the preparation for investment and actual procurement decisions. In the Beta phase, the implementation is developed further with chosen technology and methodology. The projects documentation and architectural decisions are detailed in the Beta phase. In the Live phase, developing urge should reduce after six months after the Live-phase publication and maintaining model should be reorganized. The service is monitored and measured with agreed methods and customer feedback guides smaller development needs. The Live-phase also includes guidelines and methods for replacing or ending the planned service. (Helsingin kaupunki, 2017)

3.2 Software Quality

Mastering the web development quality is a challenge if a developer does not understand or follow any of the key principles of the software quality development. A software quality directs also the security matters discussed in section 4. It is tempting to start coding with readymade snippets and see the results immediately without knowledge how the snippet affects other parts of the application structure and state. The software quality generally means implementing test approach tools or methods into the development process. In this section, JavaScript quality methods are detailed and other well-known principles for object-oriented design for C++ or Java programming are scoped out.

The software quality is not a one-aspect approach and it is not completed with mastering only few principles. Developers should take care of application memory state for keeping only needed data available for the components. Application styling, structure and naming should be persistent and logical. It is also important to handle software versioning for quick recovery and operability. Developers should follow DRY principles for creating modular code rather than repeating the same lines of code. An application should handle erroneous state and there should be handlers available for unexpected events. Writing tests before the actual function is a mandatory approach and principle with the Test-

Driven Development. Developers should optimize code with code reviews and pair programming methods. The documentation for application architecture and APIs is an advisable minimum. A detailed level documentation of components and commenting lines within code will provide base for maintaining the application quality. (Wilson G, 2014)

The Test-Driven Development limits the possibility of malfunctioning component or a function. The approach starts with writing the unit test of planned function and then the test fails immediately. After the test fails, it is possible to start the actual coding and goal is to have test pass with minimum amount of code. When refactoring the function or adding more code to the applications codebase, the automation test tool will run on background with the written tests and a faulty component is then prevented. Obstructing faulty component or function is not a guaranteed method of having satisfying quality software. When refactoring the code into smaller components and re-usable functions the application capability management should be taken into account. (Lawrence D. Spencer, 2015)

Managing the application behavior with test scenarios is called the Behavior-Driven Development. It is an evolution step from the TDD and the Acceptance Test Driven Development approach. The distinctive difference for other methods is that the BDD tests focus is on the business value and how the actual testing process will occur. (North, 2009) The JavaScript testing frameworks such as Mocha and Chai test language can utilize the BDD libraries as testing methodology.

```
1  var should = require('should');
2
3  var user = {
4      name: 'tj'
5    , pets: ['tobi', 'loki', 'jane', 'bandit']
6  };
7
8  user.should.have.property('name', 'tj');
9  user.should.have.property('pets').with.lengthOf(4);
```

Figure 3. Behavior Driven Development library called Should.js. (Bardadym, 2018)

A popular BDD driven library called Should.js syntax is demonstrated in the Figure 3. On line 1, the Should.js library is imported and on lines three to six an object is declared with named values `name` and `pets`. On the lines eight and nine `should` syntax is visible and test cases are presented. (Bardadym, 2018) Other popular library for testing frameworks

is Expect.js and it has similar approach for testing. These libraries expand the testing syntax and adds value to test cases.

3.3 Presentation Layer – Application Frontend Technologies

Web application technologies can be sub categorized to frontend and backend application technologies. These frontend technologies are visible and responsible for user actions in the web user interface. The backend technologies are covered in the section 3.4. The frontend technologies are typically HTML, CSS and JavaScript. The JavaScript technologies require a browser with a JavaScript engine and a runtime environment. One of the most popular open source JavaScript engine is the V8 developed by Google. The other active ECMAScript projects are e.g. Rhino (from Mozilla), SpiderMonkey (from Mozilla) and Chakra (from Microsoft). When using JavaScript libraries a runtime environment is needed for hosting the application. Most common runtime environment for JavaScript is Node.js technology. JavaScript frameworks usually do not require separate runtime environment but frameworks can be utilized with e.g. Node.js when needed for easier dependencies implementation. Node.js technology and dependency technologies are more detailed in section 3.4. Node.js is a technology for hosting frontend JavaScript libraries and utilizing package management system for the frontend and backend technologies.

The application architecture can be divided into three logical components. These components are Model, View and Controller. A JavaScript framework covers typically all of these areas such as AngularJS. There are also JavaScript technologies such as selected ReactJS technology that represents only View component of the MVC model. The Model component is responsible for handling the data between the View and the Controller components. The View is an output of any UI level component and the Controller component is responsible for a user interactions.

ReactJS and JSX Technology

The React JavaScript library has originally evolved from the MVC framework called BoltJS and FaxJS at the Facebook. In the beginning, the React was not an open source project. The BoltJS had same components and functional API features that originated the Reacts render createClass and refs. The refs feature was just a concept in the BoltJS

and when applications grew more complex, the Bolt frameworks codebase evolved to more intricate. The first solution was to develop a framework called the FaxJS. The FaxJS took same challenges that the BoltJS was trying to solve but it had different approach. The FaxJS have same fundamentals as the React today like props, state, server side rendering and concept of components. (Facebook, 2015)

The FBolt was introduced in 2012 by John Walke who worked at the Facebook. John Walke uploaded his functional programming style updated version of the Bolt in to the Facebooks codebase. Interoperability among the Bolt and the FBolt frameworks made possible to implement one component at a time and test it immediately. In the first adopts it was easy to see changes and manage debugging process. Throughout the framework projects, it was founded that functional APIs produced more scalable method when producing user interfaces. The name React came from the APIs way to react to any state or props change and how it rendered any nested and structured data form. (Facebook, 2015)

The Facebook necessitate expressing the acquainted tree structured syntax like XML in their UI components. Standard and properly defined syntax enabled contributors to be part of the open source project. The JSX extension was intended as a feature to the ECMAScript and XML style was chosen because of better readability. (Facebook Inc., 2014)

The React Team suggest six different templates for the ReactJS to begin with at their web pages and they recommend 19 other templates in specific use cases. In the Internet, there are hundreds of user-generated templates to choose from in exclusive needs. The Facebook React App command offers an easy approach to start developing with the React and there are no need for choosing any of the templates. The Node version above 6.x is required and few command lines as presented in the snippet Figure 4.

```
1  npm install -g create-react-app
2  create-react-app my-app
3
4  cd my-app
5  npm start
```

Figure 4. Creating a React my-app. (ReactJS, 2018)

The command `create-react-app` on line 2 will create a React App with the Facebook originated template and its features with build release process and basic linting for possible errors in code. The folder structure is also generated as illustrated in the Figure 5.

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   └── favicon.ico
│   └── index.html
│   └── manifest.json
└── src
    └── App.css
    └── App.js
    └── App.test.js
    └── index.css
    └── index.js
    └── logo.svg
    └── registerServiceWorker.js
```

Figure 5. React my-app default folder structure. (ReactJS, 2018)

The smallest and well-known example among all of the programming languages is the Hello World application. With the React, this can be done with six lines of code as illustrated in the Figure 6. The empty lines are excluded from the calculation. In order to run this application code is has to be implemented into the `App.js` file.

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  ReactDOM.render(
5    <h1>Hello, world!</h1>,
6    document.getElementById('root')
7  );
```

Figure 6. React Hello World application. (ReactJS, 2018)

The React Hello World application consist basic elements of React application. The `Import` command in the Figure 6 on lines one and two npm modules `react` and `react-dom` are loaded. On line four, the ReactDOM.render function injects the `Hello,`

`world!` to the root element of the `index.html` file. The React uses JSX in its render function and tags looks similar to well-known HTML tag as `<h1>`. The React syntax can be used with JSX elements for better readability but it also can used as a pure JavaScript markup. Before rendering the JSX element, the React DOM render function converts everything to a string and parses the lines to match actual written lines of code. This method eliminates the injection attack attempts. The Reacts JSX syntax has been developed for clean code purposes. It abbreviates and shortens the writable code providing clear methods e.g. how to call other component children. Few basic rules are set e.g. user generated components should typically start with a capital letter. The JSX has similarity from XML and HTML tags but it is not either one.

The React logic is to create snapshots called elements of a specific millisecond of time. These elements are disposable and cannot be altered. When user interface is needed to update a new `element` is created. The React DOM then compares the new and the old elements and alter only changed lines of code. This feature makes the React extremely fast when considering UI changes. (ReactJS, 2018)

When delivering a production grade code with the React all code should be written into pure components or classes. A simple and basic component structure is illustrated in the Figure 7. This method clarifies the written code and enhance further development. A components data flow is controlled with props and state. A component should not mutate its provided properties called props at any circumstances. When there is a need for changing the data a React concept called “state” comes in place. Within the React app, the state is the actual data storage for the application changes. The application stored data can be mutated through e.g. UI actions or data received from the API.

The React can handle multiple components and it is important to optimize application-reserved resources efficiently. The React has a method called the Lifecycle methods when loading and unloading components. The most common lifecycle methods are related to mounting called `componentDidMount` and `componentWillUnmount` for unmounting a component. There are several other methods for handling components newly received props or components way to handle the render method. (ReactJS, 2018)

```
function UserGreeting(props) {
  return <h1>Welcome back!</h1>;
}

function GuestGreeting(props) {
  return <h1>Please sign up.</h1>;
}

function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

ReactDOM.render(
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

Figure 7. React conditional rendering and component snippet. (ReactJS, 2018)

The React has similar conditional operators as JavaScript. React components can be rendered based on the normal JavaScript operators such as `if` statement as presented in the Figure 7.

The Higher-Order Component called HOC is a React feature that makes the codebase reusable efficiently. HOCs are common when controlled props are needed to a child component. HOCs can handle e.g. API requests and then pass the data to its child components. HOC commonly just passes application state props directly or through wrapped component. The `Connect` method is generally used when wrapping components to HOCs.

ReactJS with UI Framework

The ReactJS is compatible with dozens UI frameworks available. Different UI frameworks represent certain style and layout model. The decision between the best UI frameworks is not question about the features but rather desired view representation. The `react-bootstrap` UI framework was selected in this case study because the framework is written on pure ReactJS. The `react-bootstrap` is one of the most popular UI framework at the time and it has not been fixed for certain device environment. The `react-bootstrap` offers an excellent grid layout system and a set of general components such as navigation, tables, dropdowns and buttons.

During the development phase, other UI decisions should be made such as CSS and Font styling. In this research study, selections of suitable responsive UI or selected styling was selected based on popularity.

Application State and Data Management with React

The React allows the developer to choose from multiple dataflow technologies. Reacts most common application state and data management techniques are React state, Flux, Redux and Mobx. For the best practice, the React state should not handle application data. It should only store local state changes based on actions and pass values to other components as props. The Flux and the Redux technologies share similar features but are ideally suitable for different purposes after considering application complexity and size. In this research study, changing the database structure was intentionally scoped out and the focus was on choosing best state management system suitable for the application needs.

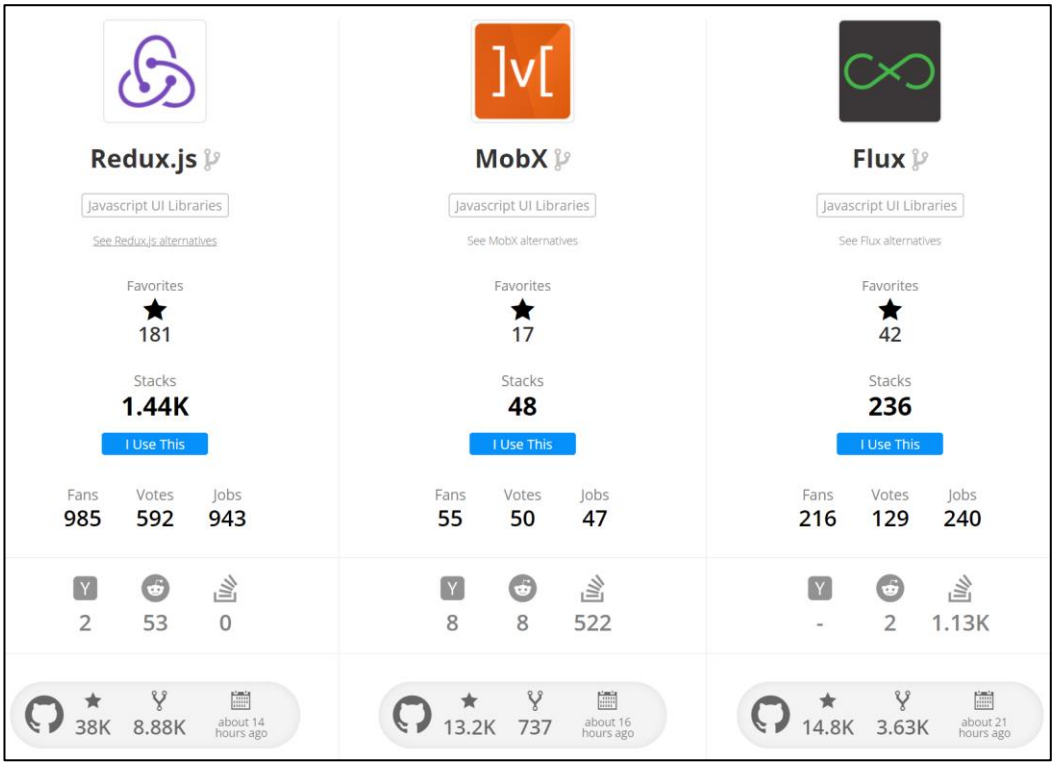


Figure 8. Application state management comparison. (Stackshare, 2018)

As illustrated in the Figure 8 the Redux.js is the most popular state management technology compared with Github shares and Stackshare.io votes. The Flux is an open

source software and it is maintained and developed at the Facebooks Github. The Flux is introduced in the Facebook React documentation as a starting point for handling application state. The Flux has short learning curve and easy to understand structure and unidirectional data flow as presented in the Figure 9. The Flux have certain disadvantages e.g. when developing connections to databases. With the Flux, handling several database connections is an overwhelming task to accomplish. The Facebook has answered to the needs of simplicity, they developed the Flux technology even further, and Relay Modern was released. The Relay Modern is suitable only if Facebooks backend GraphQL database structure is in use. (Facebook Open Source, 2018)

The Flux unidirectional dataflow differs from the MVC model idea of bidirectional flow. In Flux, data flow from the React Views such as Components it waits for an update from the Store and only updates the View if the Store is updated. React components can also query data from the Store if needed. In React Views at Figure 9 if a user based the Action is fired e.g. user conducts an action and then the Action Creator is triggered and forwarded to Dispatcher. The Dispatcher acts as a core service that manages all actions and then it updates the registered Stores. After a Store is being updated, it emits a change event that updates the React component on the webpage. (Flux, 2018) (Medium, 2018)

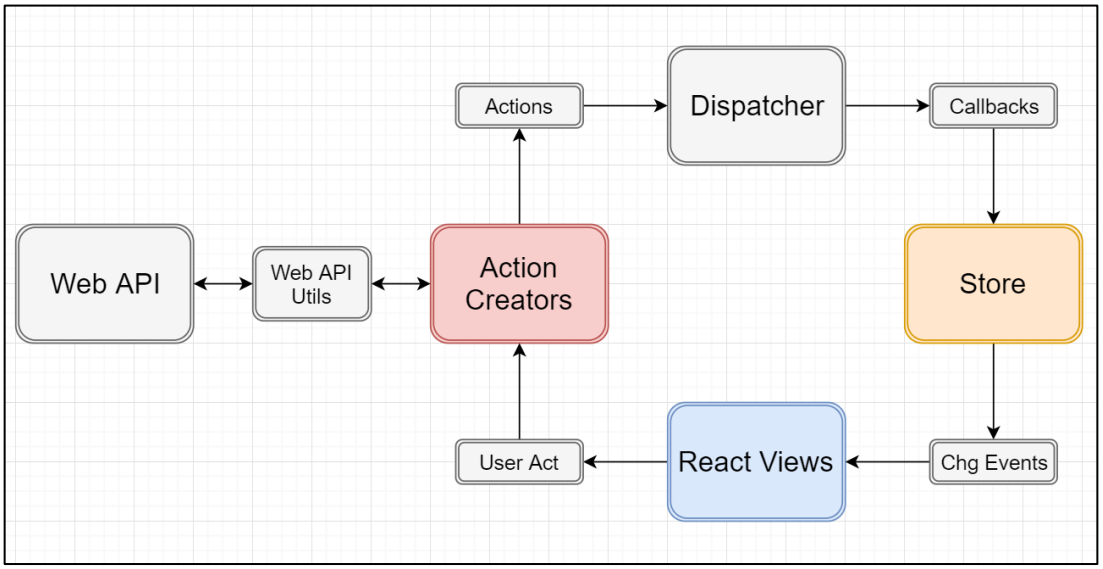


Figure 9. Flux unidirectional dataflow. (Medium, 2018)

The Flux serves only one Dispatcher and thus it makes the application data flow predictable and debugging easier. With the Flux, there can be multiple Stores for different

purposes. Stores can handle the application state and business logic. Different stores are registered with their callback to Dispatcher and are updated if `Switch` statement action is triggered. When the application complexity grows and multiple Stores are in use the Stores callback register method and maintaining the multiple callbacks becomes problematic and uneasy task to maintain. (Flux, 2018)

The Redux state management differs from The Flux such a way that there is only `One Store`. The entire application state is maintained within one read-only Store. All changes are dealt throughout Actions and for every change; a new Store object is created and old Store object is stored. This feature enables the Redux container to have time travelling feature for debugging state changes. In modern application development, the code complexity is common. The application has multiple ways of changing the application data and the data can be received throughout multiple APIs. With mutating model scheme where other model updates views and vice versa can lead to loss of application state control. (Abramov, 2017)

The Redux has predictable state flow as illustrated in the Figure 10. The Redux has three basic rules for controlling the state. The first rule is a single store where all of the application state is stored as an object. This rule facilitate easier implementation for traditionally complex code base for undo and redo features. The second rule is an application state that is read-only which can be mutated only through actions. These action creators can be easily motored and logged for debugging. The third rule is about controlling mutations throughout functions. These functions called reducers primary purpose is to take the old state and an initiated action and then return a new state of an object. In all parts of the flow, the state changes can be logged and it is an essential for debugging more advanced and complex applications. (Abramov, 2017)

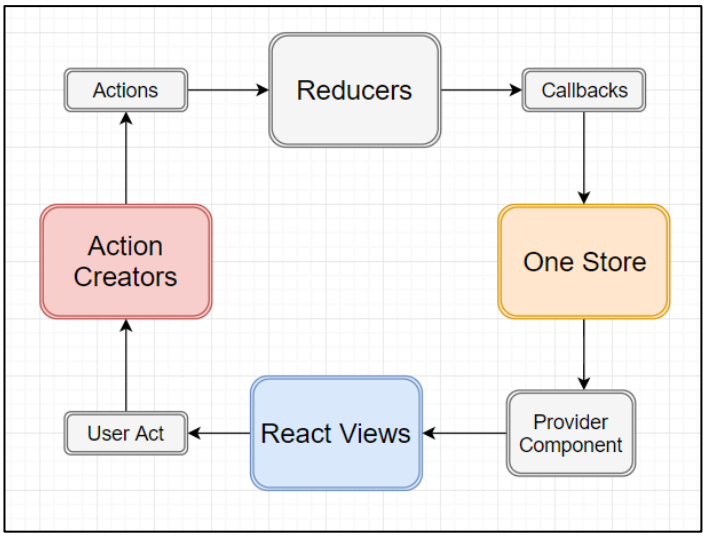


Figure 10. Redux predictable state flow. (Abramov, 2017)

The Redux dataflow is a unidirectional, which means the data is controlled in every step as presented in the Figure 10. This dataflow pattern keeps the code clearer from duplicated stored data and limits the variations. The Redux flow begins from call action from e.g. application or asynchronous calls with the `store.dispatch(action)` statement. The Action payload is combined with the Stores present object and then the Reducer function returns a new Store object. Typically, reducer functions are divided into separated reducer purposes e.g. listing or filtering objects but it is not mandatory. Multiple reducers are then joined with `combineReducers` function.

The Mobx is a useful solution for smaller projects where application size or amount of developers do not extend in size. The Mobx has shorter learning curve rather than the Redux but a huge amount of an actual code base is hidden from the developer. A developer needs to use the Mobx global annotation when accessing multiple stores. The Mobx is more object oriented rather than the Redux function programming approach. The Mobx has mutable data and minimal boilerplate for easier implementation. The Redux is definitely better solution for complex applications and team development projects for its testability and scalability. (Wieruch, 2018)

Asynchronous AJAX with Axios

When the application needs a method for accessing backend business logic or retrieving data from several APIs a client functionality is needed. The Axios is an asynchronous

HTTP client that has features for making XMLHttpRequests. It can handle multiple requests and alter the response to JSON data scheme. The Axios supports a promise requests and these type of requests are also known as chained asynchronous function calls. After a callback receives a response from the backend, the call can be then handled with `.then` callback function. Each of these functions can be chained with a promise for having control on e.g. combining the receivable data before next action or function. (Zabriskie, 2014)

3.4 Data Access Layer - Application Backend Technologies

The Database abstraction layer called backend architecture is needed when application has features that require e.g. database or retrieving data from the API. The backend is responsible for runtime environment and it handles data flow callbacks. Databases are also considered as backend technology. This section covers technologies used with this project. There are common runtime environments for web application such as Node.js, PHP and Ruby. In this project, the Node.js was selected because it is written in JavaScript and it has become a standard backend technology used with the React.js.

The Node.js is a new approach to traditional web application technology. It minimizes the latency between transactions by allowing and not blocking any of the requests made. Traditionally application sends a request and then waits for the response. In this kind of transaction, the wait time to the end users point of view is not occurring as responsive. The Node.js technology allows multiple simultaneous connections and it handles those requests as a callback functions. The callback functions are easier to organize to different CPU threads and it makes the Node.js architecture very adaptive and capable. The Node.js method is also highly efficient from resource perspective and it allows more callback handling with less resources. (Heller, 2017)

The Node.js architecture supports the ECMAScript 2017 standard features that are brought to the V8 engine. The Node.js Foundation maintains and updates the current Node.js version regularly. The Node.js performance is in the event loop process. It allows the Node.js handle non-blocking I/O operations. The event loop process is responsible for e.g. making asynchronous API calls and schedule timers for polling. The event loop process is divided into six phases; timers, I/O callbacks, idle, poll, check and close callbacks. There are also time constraints for any wait phases so that the event loop

process continues cycling until it does not have tasks to run and then it closes the process in a controlled manner. (Node.js, 2017)

One of the Node.js strengths is in its extensive library that has several modules that can be implemented to the Node.js settings codebase. The other benefit comes from the Node.js package ecosystem where coders or other open source code parties can share their code modules. The Node.js ecosystem has nearly half a million packages available and the packages are typically hosted from e.g. Github where hundreds of contributors can upgrade of suggest fixes to the packages. These Node.js packages usually contains several modules that are required for running the package. A package can be installed commonly through with a command line tool called `npm` or `Yarn`. These tools offers commands for maintaining the loaded modules and the tools for versioning. A Node.js offers modules for connecting to different databases and it has no fixed setup for specific backend. (Heller, 2017)

The web application commonly exchange information between several other services or servers. A favored web framework for handling API request for web application is called the Express framework. The Express framework is compatible with all current databases and it scales when application grows in complexity. It offers powerful routing capabilities with small amount of source code. The Express utilizes standards from representational state transfer (REST) API and it has built-in middleware functions e.g. `express.JSON` for handling JSON payloads.

Database

A database is needed when a collection of data needs to be organized. The database models can be put in a timeline of pre-relational, relational and the next generation databases. A relational database has a structure of tuples, relation, constraints, and operations. A relational databases represent a set of data to the user with un-normalized or in a normalized model. All changes are made through ACID transaction model: Atomic, Consistent, Independent and Durable transactions. The relational database with high performing query syntax is commonly known as SQL database with tables, views, stored procedures and functions. (Guy Harrison, Apress, 2015)

As the application development grew to object oriented model, the relational databases needed to evolve to store objects without normalization. This need was a starting point

to non-relational databases such as Hadoop and other Document databases. The current programming approach with the Asynchronous JavaScript and eXtensible Markup Language lead transition to JavaScript Object Notation (JSON) language. The JSON format has become a standard for storing objects into relational or document databases. Within relational databases, a JSON object is stored straight into columns. Document-based databases are popular among programmers because of less process work for organizing data to normalized form. The document databases uses self-explaining format and there is no need for database schema. Document database does not limit application-structuring changes. (Guy Harrison, Apress, 2015)

4 Software Security in Web Application Development

The section 4 covers aspects of secure web development methodology. Developing an up to date and secure web application a developer and the management level needs at least a fundamental level of knowledge about security issues related to the web technologies. In this section, these vulnerabilities and generally ideal solutions are introduced. The security section is divided into four logical parts, frontend, backend, access control and secure development and methods.

When considering security and implementing ideal solutions, many of the methods will include quality practices. These two subjects support each other in development process. In this thesis, the project study is about developing web application and therefore these security aspects are being covered in detail. It is recommended to learn a quality web development from the aspects of the security vulnerabilities. This type of application development approach can scrutinize logical errors and design flaws.

In the web application security field a non-profit community, based on an organization called the Open Web Application Security Project (OWASP) is usually mentioned and it has become a general standard with its Top 10 published vulnerabilities related to web applications. In the OWASP Top 10 list, all of the vulnerabilities are explained in detailed level and discussed how the vulnerability is exploited. The OWASP top ten presents a descriptive overview how to avoid making those issues. In addition to the Top 10 list, the OWASP provides several additional handbooks and suggestions in verifying applications. The application security approach can be divided into the following five sections; application security requirements, application security architecture, standard security controls, secure development lifecycle and application security educations. These separate manuals and OWASP standards will help a developer for developing a secure application from the beginning. The OWASP methodology for guiding secure development is illustrated in the Figure 11. (OWASP, 2017)

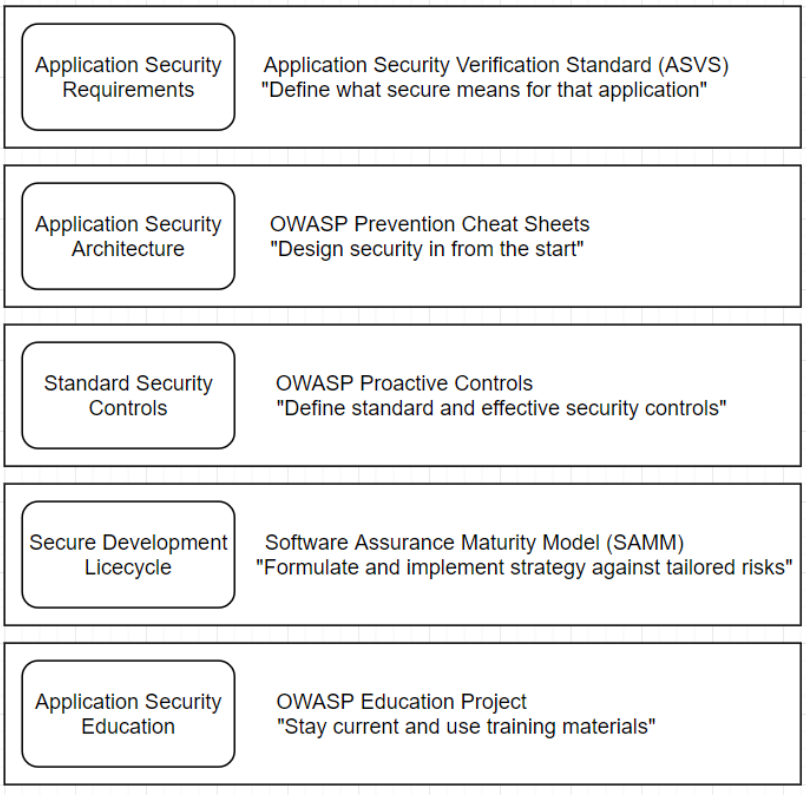


Figure 11. OWASP Developer guidebook. (OWASP, 2017)

4.1 Frontend Security Aspect

A web application uses browser technologies to work and a small part of the web application security issues are dealt with the browser security patches. The web browser developing companies have different response times to the security related issues and threats. In this frontend section, the focus is on programming aspects and exploits such as same-origin policy, cross-site scripting and request forgery, session management and error handling. In addition, the web developing technologies have vulnerabilities and the JavaScript and the Node.js security is important to discuss.

The Same-Origin Policy is a fundamental security principle for the web applications. It restrict attacker to run malicious code in the actual web page that has a legitimate source. The Same-Origin Policy is an agreed and standardized feature within browser developers. It defines and limits how scripting actually runs in client-side browser. There are some exceptions on how to run code from different source. One example in the HTML is using the Script-tag. It bypasses the Same-Origin Policy when “src” attribute is defined. The second way of bypassing the policy is when using Asynchronous JavaScript with

XML (Ajax) calls. There are secure and insecure ways of doing popular JSON-request with Ajax calls and when generating the JavaScript object or an array from the response JSON string. With Ajax calls cross-origin resource sharing (CORS) is recommended to allow specific domain to be trusted. The most insecure way of doing the JSON string transforming is use the “eval”-function. In eval-function the JSON string is handled as a code and that offers an opportunity to attacker use its exploit code in JSON. More secure way of doing the transformation is to use native JavaScript `JSON.parse` command. The `JSON.parse` only parses the provided string without executing it. (Bryan Sullivan, 2012)

One of the most common attack surface that web applications have is Cross-site scripting vulnerability. It means that the attacker can run its own code as JavaScript injection. The cross-site scripting is abbreviated commonly as a XSS vulnerability and it is generally targeted against web application users. Browsers have solutions such as SameOrigin Policy to prevent these attacks but it does not cover all possible exploit methods. A general issue is that the web application treats input data as pure HTML or JavaScript code and executes it. (Bryan Sullivan, 2012)

ECMA security

The international European Computer Manufacturers Association abbreviated ECMA is a standard for the language generally known as JavaScript. In 1997, the first ECMAScript standard was published and the standard has been evolving on yearly basis. The current version is ECMAScript 6 (ES6) and it has undoubtedly fixed the earlier security issues but it also have brought new features with new security issues with it.

The ES6 has new features that provide additional methods to execute the code. The new methods offers a way where functions can be executed without parenthesis where it was not allowed before and the ES6 introduced a new native objects with evaluation. In the ES6, function declarations can be made with flat arrows or shorthand functions without the word “function” being used. This is an important security issue and it should be carefully noted in input validating process. (Heiderich, 2014)

Error handling

One essential approach to secure and maintain satisfying quality of web application is to define how the application is handling errors. The error handling is a key aspect in overall

development cycle. The error handling structure is built into JavaScript as exceptions. Error listeners, detection and handling procedures should be written to the code base. These structures will also advice to debug and maintain the code. An uncaught errors or exceptions might leave developed application to unstable condition and expose it to potential vulnerabilities. In this study, the Node.js runtime is used for executing JavaScript code. In the event-driven Node.js, execution behaves very differently compared to other programming languages. The Node.js will break the application thread repeatedly when an uncaught exception occurs compared to other single request failures in computer programming languages such as PHP. The end user might not even notice these uncaught exception types of disruption in the web application usage. (Barnes, 2013)

The Node.js applications will generate four types of errors: standard JavaScript errors, system errors, user-specified errors and assertion errors. Standard JavaScript errors handles situations such as when the reply syntax is incorrect or undefined variables are thrown. Other erroneous behavior will occur if a uniform resource identifier (URI) handling function is violated or faulty types are passed as an argument. System errors have features inherited from standard error class and will be triggered e.g. when file does not exist or closed transmission session is called. All of the errors should be handled immediately with JavaScript throw handler. Throw handler will exit the Node.js process instantly if the exception is not managed with `try` or `catch` method. (Node.js, 2017)

4.2 Backend Security Vulnerabilities

The latest web applications urge with high demand of storing user data or API data into databases. The application storage needs are rapidly growing with traditional SQL and NoSQL databases. As databases, expand in size with sensitive photos or finance information and the actual data value increases. Valuable information will intrigue hackers to develop hacking methods with more complexity. One of the mostly violated backend vulnerability issue based on the OWASP Top 10 list is when hacker uses SQL injection technique against SQL based database. Commonly these attacks are meant to steal valuable information from the database or modify the current values without administrators noticing it. Delicate SQL injection can also be done with denial of service as an approach to the service itself or the attack can be directed to other service as well.

In order to be able to defend against these security vulnerabilities developer should have legitimate knowledge of the attacking methods. (Shema, 2012)

The SQL injection is about intercepting or alternating the command exchange between the application and the sending application. The database command transactions can vary from dynamic to static queries. With dynamic queries, the web application formulates the SQL command from any given parameter to the SQL statement and then it sends the query to the database. When the application normally gather these parameter values from the user into applications own variables, hacker exploits these inputs with malicious scripts or characters. One method for preventing SQL injections is input validation. The input validation capabilities differ with various programming languages but few basic rules can be followed regardless of selected language. Input data should be normalized with character set e.g. UTF-8 and the validation should always compare the input data to the expected data type and content. When the application is compromised with injected or faulty inputs, it should always reject the data entirely rather than trying to fix the input with excluded characters. (Shema, 2012)

The application logging is an essential part of recognizing anomalies within application events. One purpose of security logging is to detect intrusion attempts and general application behavior. Security logging can be done using third-party cloud services, which gives an easy approach to the developer. The OWASP organization suggest to record at least validation and authorization failures, authentication results, session management failures, system events, higher-risk functionalities, sequencing failure and excessive usage. (Barnes, 2013)

4.3 Access Control Methods

Understanding truly the difference between authentication and authorization is a mandatory. Authentications purpose is to prove the identity of a person with e.g. passwords or login credentials. The authorization in a web application scheme means for instance validation with certificate. Its purpose is to prove that the web site is what it claims to be and it is not a fraudulent site. Because of the HTTP is a stateless protocol a web site needs to provide session tokens for tracking user selections and movement. These session tokens can be saved into the web browsers local cache and when user authenticates with the credentials these session tokens can be mapped to user identity.

Session tokens are vulnerable for e.g. cross-site scripting, SQL injections and network sniffing. To improve web applications security a strong session tokens are needed for securing tokens against reverse engineering or brute force attacks. (Shema, 2012)

Securing applications with multifactor authentication is more common recently. Securing passwords with hash and salt mechanism could prevent hacker using rainbow tables with common hashes for decrypting passwords. Generating a rainbow table it demands a lot of processing power but afterwards the password hashes are checked against it and compared within seconds. Salted password hashes include random character set generated by a web application and with user given password it is calculated into a new hash with e.g. SHA-256 algorithm. If a hacker gets SHA-256 generated password with non-salted prefix it can easily reveal the passwords to readable format. (Shema, 2012)

Developing a new web application with single-sign-on authentication (abbreviated SSO), is necessity. When a web application authorization resolves whether user has an access to given resources or features within application the SSO is responsible for only the authenticating the user. The SSO purpose is to establish and share the encrypted user session token between domains. The concept is same with different SSO implementations; the central domain is responsible for authenticating the users and generating signed e.g. JSON Web Token. Regardless the session token is shared among other domain it cannot be altered. The authentication session token is commonly stored within browser cookie storage. When a new domain needs an authentication from the user it redirects the user to authentication domain and if succeeded the user is redirected back to requested domain with a valid authentication token. (Sebastian Peyrott, 2015)

The access control should imply the file system as well on the web server. After successful authorization to the resources, the user might exploit the file structure when using predictable directories e.g. `/admin`. The application code is stored on a public web folder and the user have normal access rights to the folder but when user changes the URI parameters or other expected values, the web application can be compromised with the not desired vulnerability e.g. exposing other user profile data or cookies. (Shema, 2012)

4.4 Secure Development and Methods

In secure web development many acronyms exists such as the object-oriented SOLID and the DRY principles. With JavaScript, the SOLID principles needs to be modified into more suitable manner, e.g. letter L stands for Liskov Substitution but it is not applicable for JavaScript. With JavaScript web application development, one of the security approach should follow a writing of a code that remains working and legitimate. This method is usually gained with a method such as the TDD that is described in section 3.2 with more detail. When the coding process begins a lot of coding needs to be done prior, the actual needed function or visible action desired. Using a testing framework eventually mean that the coding snippets will evolve into self-explained form and to a simpler functions. (Lawrence D. Spencer, 2015)

Security design is important when dealing with web applications. When user differentiates from the desired or planned application workflow, it can compromise the entire application security. These vulnerabilities are common in a generally badly designed application. These application logic errors can be seen e.g. when passing the normal verification steps or filling the forms in an irrational order. There are no one-solution method against these logic-based vulnerabilities but at least fundamental key factors are present. One of the methods against for badly designed application is documenting. When producing a great level of documentation it reduces the risk of logic errors significantly. Documenting the expected behavior of the feature with user responses could mitigate the security risk. (Shema, 2012)

Using a quality assurance (QA) process it can assist the development phase when developing a secure application. Writing a vital test case for the application it can reveal potential application logic vulnerabilities. Writing autonomous tests can offer a good approach for a quality code but not necessary in finding loopholes. Revealing method for a loophole is to write security related tests e.g. with actively invalidating a user session. Frauds and other anomalies can be analyzed from the application logs if collected properly. (Shema, 2012)

5 Developing Web User Interface with ReactJS

The application development project for replacing the old Access based ERP system with modern web technology was the case study for this project. One essential objective for this project was to test is the selected modern JavaScript technology suitable for the current backend environment. The application project utilized a test application using the KEHMET agile principles. The test purpose was to prove if the selected technology could replace the current production environment and solve the current technology issues.

This section describes how the project was planned with the organization and how the actual plan implemented from mock-ups to the actual Alpha model. After planning section, the next section describes project environment installation with selected technologies. The projects Alpha model is explained in detail and the projects scope is set. The section five covers the essential codebase snippets and application view relations to separate modules. The last part of this section illustrates mobile development approach for the ERP application.

5.1 Project Plan and Requirements

The target of the project was to develop a concept Alpha model of an application within eight-week period. The project plan was approved in a preliminary meeting before the projects coding work sprints. The project was divided into six overall feature sets. The first development step was to develop the homepage and a navigation bar as illustrated in the Figure 12. The second build should include the user selection field and a working API for retrieving data from the database. On the third development build, the application should display a table for users own inputted records from the existing test database. After these delivered packages, the application should have a form for saving user scores and fire inspection data. The next step in the application development should be editing existing record from the table selection. Lastly, in the sixth build, the application should display individual users and user group scores in a report view.

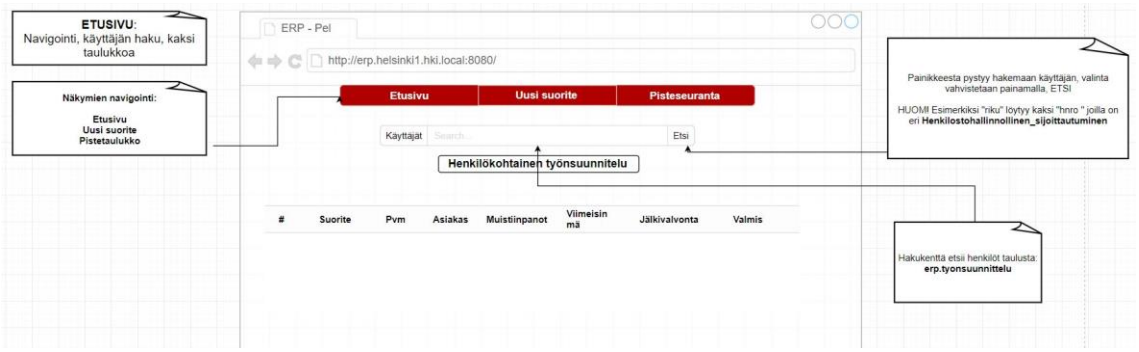


Figure 12. ERP Home design mock-up view.

All the application features were first designed as mock-up pictures to help the meeting process and a platform for information exchange between the coder and the organization. Mock-ups are easy to produce and can be adjusted during the meetings. Usually organizations ideas and thoughts are more cleared and focused when planned on the actual lookalike mock-ups.

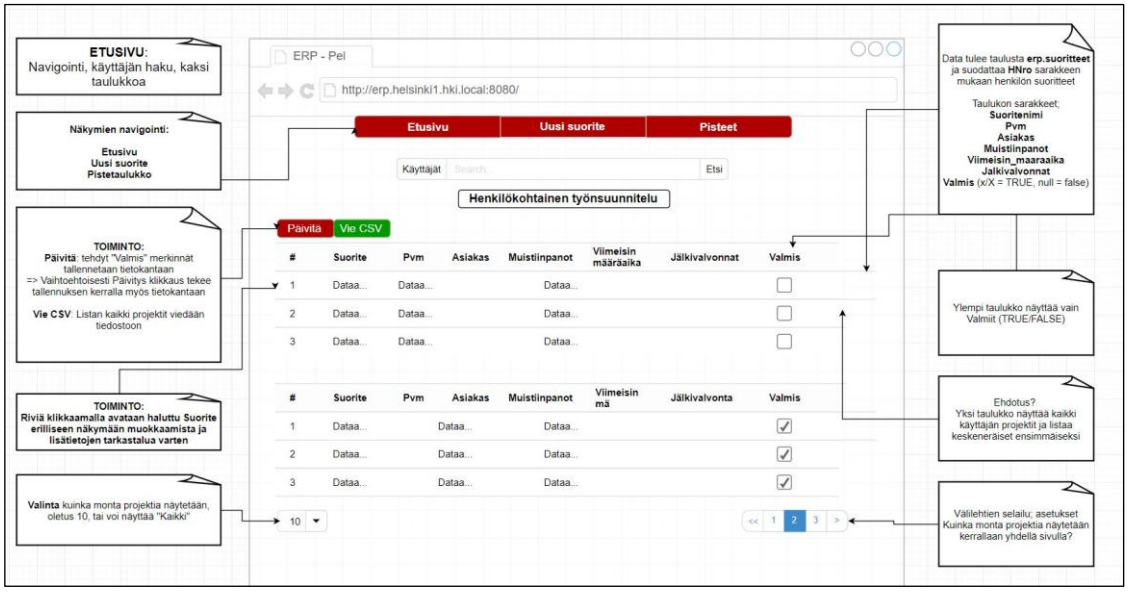


Figure 13. ERP Suoritteet design view.

As presented in the Figure 13 the main Suoritteet mock-up view is illustrated. It combined features from navigation bar for accessing the application navigation and opening a new task form. The Suoritteet design view illustrated how a user could mark tasks to done/undone status and other key functionalities e.g. pagination for listing several tasks. These mock-ups saved a lot of time from the actual coding process and the organization had immediate idea where the application development could lead. These mock-ups

were easier to alter to the desired result than the actual code. The mock-ups offered a solid foundation for the application feature list and these drawings were conducted to feature sets in an application development Kanban-table.

5.2 Minimum Viable Product for ERP Solution

In a help of organization discussion the ERP solution diagrams were planned before the application coding process started. A clear and good quality plan is essential key for successful projects. With a flow diagram, it is easier to see how user actions are going to work in desirable solution as presented in the Figure 14.

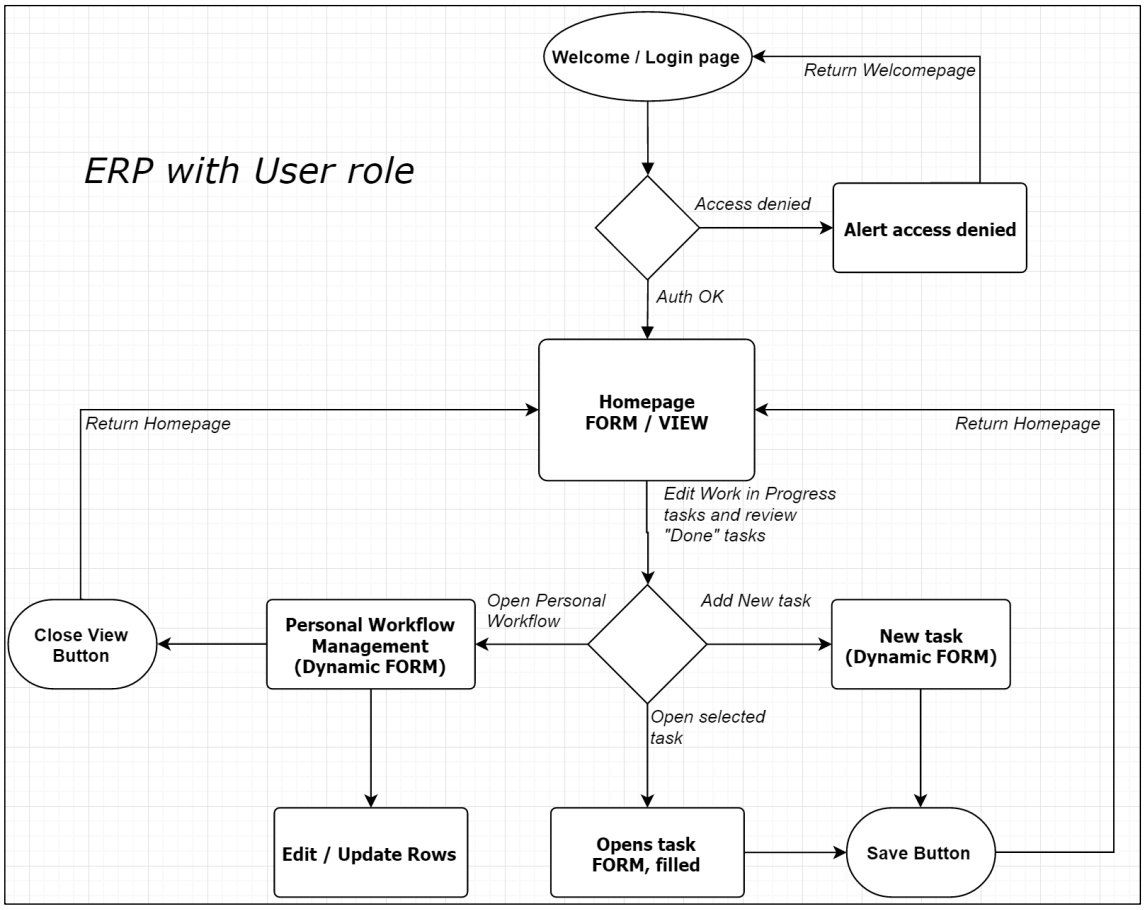


Figure 14. User action flow.

The application structure is visible in the diagram and it provides a good approach to design High Order Component (HOC) structure for the application.

5.3 Project Setup and Environment

Application development is easy to begin with ready-made environment container technologies available in the Internet. With this master thesis project the setup environment was conducted manually for research purposes and for evaluating best practices. The development environment benefitted the development process to have equally the same environment as production environment was based on.

Virtual environment

The development environment was running on the Oracle VM VirtualBox Manager that hosted the Red Hat Enterprise Linux. The VirtualBox manager is an open-source hypervisor for running various operating systems available. It has several features for optimizing performance on guest virtual machines and it is actively developed and maintained for supporting newest OS versions.

The Red Hat Enterprise Linux (RHEL) is a Linux distribution developed by Red Hat organization. The RHEL distro is targeted for commercial market with long-term support. The Red Hat distro was selected for subscription reasons for this project. The Red Hat, Inc. offers a developer subscription for easier access to their products. A developer subscription was also acquired for this project.

Tools

For source code, editing purposes and syntax checker for JavaScript a Sublime Text application was installed. The Sublime Text tool offers various features for helping the actual coding process. The Sublime Text is one of the most popular source code editors available based on various comparisons available in the Internet. It supports natively multiple programming languages and it has active package ecosystem actively maintained by the community.

Dependencies

In order to begin application development with node applications the Node.js and npm environment was installed. The Node.js as a runtime technology is more detailed covered in section 3.4 and the section covers its selection justification for this project.

During the development, Node.js and npm environments were updated regularly to latest update available. Several needed npm packages were installed and the application main packages were react, react-redux, redux-form, react-select, react-bootstrap, react-router and webpack components. There were quite many other dependencies used, but these main modules were developed to be pure react. The ReactJS does not limit using other solutions as well but the project scope was to develop with the React JavaScript library and these dependence modules were selected with that principle in mind.

Webpack

Webpack is a static module bundler and it was used for representing the modules used in this project. The Webpack offers a low latency loading times when there are dozens of dependencies needed. The Webpack utilizes installed dependencies into demand chunks and it has multiple control options for handling the application assets. It supports hot reloading feature for replacing modules and it saves developing time when the application state is not deleted during code changes.

Database environment

The existing collection of information was stored in MySQL database and for that reasons the MySQL database was installed into the development virtual environment. The MySQL is an open-source relational database management system. It offers for non-commercial usage scenarios the MySQL community version as free of charge. In this project, the MySQL Community Server (GPL) Version 5.7 was installed and it was updated on regular basis. The MySQL community server was installed with following commands:

```
# wget http://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm
# yum localinstall mysql57-community-release-el7-9.noarch.rpm
# yum install mysql-community-server
```

After installation, the `mysqld` service was started and allowed with commands:

```
# sudo service mysqld start
# sysctemctl enable mysqld
```

When the MySQL service has been started, the server is first initialized and an SSL certificate with keys are generated. A `root@superuser` account is a default account for the MySQL database. The superuser password needs to be changed after the installation because the temporary password is stored into mysqld logs. For easier database management a visual tool named MySQL Workbench was installed. It provided administrative tools for server and database schema configuration and a clear user rights management. The MySQL Workbench offered a straightforward method for importing and exporting queries and database tables. The actual production database collection of information was regularly updated during the development process. The MySQL Workbench has a visual performance dashboard with many tools for analyzing and optimizing SQL statements performance.

A firewall rules should be considered for allowing the database connections especially when dealing database and application server on separate servers. A database connection was allowed with a new `INPUT` rule in `/etc/sysconfig/iptables` file. The following line was implemented:

```
-A INPUT -p tcp -m multiport --dports 3306 -j ACCEPT
```

Depending on the production environment, more specific ruleset needs to be applied. After inserting a new line to the configuration file the `iptables` service was restarted.

5.4 Projects Codebase Structure

With the ReactJS there are generally available some ideal practices for folder structure. Depending on the selected MVC model components different folder structure is applied. Facebooks own `create-react-app` follows one type of folder structure and other templates can follow other. The applications project folder structure is free from templates and developers usually generates their own. This project followed React Redux ideal recommendations for folder structure.

The main app folder root consist of static `index.html` file and node.js `package.json`. This project used the Webpack for running application and its configuration file `webpack.config.js` was located in root folder. The root folder included several sub folders such as `backend` for middleware code, `style` folder for CSS-styling and `src`

folder for actual codebase. The Node.js applications can have individual `node_modules` folder for an application specific saved `npm` modules. The npm module versions are easier to handle within application with separate folder rather than global package installations.

As illustrated in the Figure 15 the source folder called `src` the main application components and codebase are structured. A common logic is to save applications general components available all other components into the `common` folder. High order components for example main views can have their own folder structure to include sub components related only to the one view.

The React Redux related folder structure also included `actions`, `reducers` and `selectors` folders. The `actions` folder included all applications actions. For satisfactory readability, it should have `index.js` file for actions and high order component based file structure. In this project, the `suorite_lomake` has multiple actions and functions and those functions are separated in individual component files. These implemented practices are visible in the Figure 15.

The DRY coding means “Do Not Repeat Yourself” and the React component structure model is recommend because e.g. short functions, length of usually less than one hundred rows are easier to notice from mistakes. When code file extends to huge amount of code lines with all functions included in the same file, typos and other mistakes are harder to notice. Separating functions to own code files it makes it easier to implement and reuse code when needed.

Figure 15. App folder structure for the project.

5.5 Alpha Model of the New ERP

A one of the most important and difficult decision is made in the initial phase of a coding project. There are more than a couple of hundred different templates available when starting to develop a new application with the ReactJS and these boiler templates can vary tremendously. Templates can be fully integrated with extra plugins with developing, testing or reporting related. Templates can also be production oriented with certain abilities to generate builds etc.

In this project after iterating several React templates with desired data model with the Redux state management capabilities, the `redux-simple-starter` template from the author Stephen Grider was chosen. It is available on the public Github code sharing service and the template is ideal for writing manually every line of code by a coder. It will truly enable a coder to understand what the application is capable of and how it is working. The author Stephen Grider has written clear instructions to new coding students how to generate this template from the beginning. Some of the advanced level templates are harder to begin with and fully understand used codebase why and how the application is receiving various information. With these restraints some automated validation, testing and build publishing were excluded from this project.

Before the actual project, the development environment was configured and several dependencies were installed during the development phase. These installed and required dependencies are illustrated in the Figure 16.

Figure 16. Development and package dependencies.

As a starting point for the application development was to develop template structure and the Redux store integration where all features were coded manually from an empty codebase. After implementing the application template called the `redux-simple-starter`, the first actual feature was to produce a user search field with the React. The guidance from the mock-up planning the `react-select` component was chosen to fulfil needs. In the initial phase, certain design templates were selected and pure React `react-bootstrap` component was chosen for application layout-design purposes. The React is not selective about the layout and numerous layout components could be chosen. In this situation, `react-bootstrap` included actual react native code and therefore selected.

The application dataflow management was implemented with `react-redux` and the application state management was stored in the Redux store with several collections. Between the application and the database, an API middleware was implemented with the Express RESP API commands. The development database was exact copy of the current production database in use, and during development, the database scheme was updated weekly.

```
// HAETAAN KÄYTTÄJÄT (UserHakua) VARTEN

    router.get("/users",function(req,res){
        var query = "SELECT ??,??,?? FROM ??";
        var table = ["HNro", "Nimi", "Henkilostohallinnollinen_sijoittautuminen", "tyonsuunnittelu"];
        query = mysql.format(query,table);
        connection.query(query,function(err,rows){
            if(err) {
                res.json({"Error" : true, "Message" : "Error executing MySQL query"});
            } else {
                // res.json({"Error" : false, "Message" : "Success", "Users" : rows});
                res.json(rows);
            }
        });
    });
```

Figure 17. Rest API middleware snippet for `UserHaku`.

During the development phase, the optimizing of the application was in an essential aspect. The application retrieved all users' data on the background with one query from the database and the query received 759 user details. The not optimized query took 170ms roundtrip time on average. Within application, all inputted rows with the test database included 5500 test rows and several information fields, as database columns were 53 per row. Handling this amount of data at time the query needs a powerful and optimized code structure to work with and JavaScript application code should be develop to be efficient. A basic optimizing was done with every query e.g. from User tables with 15 columns total and reducing retrievable columns only to three which was actually used and needed. The query optimizing reduced the retrieving roundtrip time from 170ms to 40ms on average. Implementing query with `*` (`star`) option could compromise performance and security when querying and handling large database tables. In the Figure 17, the optimized query is presented.

```
96          return (
97              <div className="section">
98
99                  <h3 className="section-heading">{this.props.label}</h3>
100
101                 <i>
102                     Valintakenttään voi kirjoittaa hakusanoja etunimen, sukunimen tai henkilönumeron
103                     mukaan sekä esimerkiksi: " asema 10 ... "
104                 </i>
105                 <p />
106
107                 <Select ref="stateSelect"
108                         autoFocus
109                         simpleValue
110                         placeholder="Valitse henkilö..."
111                         options={options}
112                         clearable={this.state.clearable}
113                         name="selected-state"
114                         disabled={this.state.disabled}
115                         value={this.state.selectValue}
116                         onChange={this.updateValue}
117                         searchable={this.state.searchable} />
118
119                 <div style={{ marginTop: 10 }}>
120
121                     <button disabled={!this.state.selectValue}
122                             className="haku btn btn-primary"
123                             type="button"
124                             onClick={this.onClickSelectValue}
125                     >Hae suoritteet</button>
126
127                     <button disabled={!this.state.selectValue}
128                             className="haku btn btn-success"
129                             type="button"
130                             onClick={this.onClickSelectValueUusiSuorite}
131                     >Uusi suorite</button>
132
133                 </div>
134
135             </div>
```

Figure 18. React-select component snippet and reactive buttons.

The ERP applications front page called `Etusivu` included search input field that retrieved user information from the database. The search input field used `react-select` component as presented in the Figure 18. On the lines between 107 and 117, the `react-select` component is configured. The menu bar also had other features eg. `Kaikki Suoritteet` page that loaded all possible task related information from the database. This page was ideal for performance testing. The `Henkilo` page displayed user's personal tasks and scores. The `Uusi suorite` link directed the user to a new task inputting form. The `Admin` link functionalities were only for illustration purposes.

Figure 19. The ERP Home page on week 2 with react-bootstrap.

The user selection retrieved selected users information with `this.onClickSelectValue` function and the button was disabled if no user is selected. The user selection-dropdown menu is presented in the figure 19. The application was a single page app (SPA) that displayed the current component as selected. For user navigation and routing between components `react-router` component was used. The navigation menu is visible for example in the figure 19.

The ReactJS is depended on information it should illustrate. In the case of missing information, the ReactJS application crashes before it has actual data to display. The React JavaScript render function codebase loops rapidly in milliseconds for adapting possible changes. The ReactJS application might typically crash in asynchronous data retrieving cases when the data is not present for the application and the waiting loop process has not been noted. During development, all considerable erroneous scenarios should be implemented especially when retrieving asynchronous data from database.

```
55      render(){
56
57          // ODOTETAAN ETTÄ SAADAAN KÄYTTÄJÄN TARKAT TIEDOT
58
59          if (!this.props.valittu) {
60              return <div>Loading...</div>
61          }
62
63          return (
64
65              <div>
66
67                  <SuoritteetHenkiloNavBar valittu={this.props.valittu} />
68
69                  <SuoritteetHenkiloKaikki {...this.props} {...this.state}/>
70
71              </div>
72
73
74
75          );
76      }
77  }
```

Figure 20. Snippet of a render function for asynchronous waiting.

In this master thesis project a typical wait time for the user data retrieving Ajax calls took around 100ms to load. The React loop cycle is approximate 25-35ms and therefore simple `if` statement were needed as illustrated in the Figure 20 on the lines 59 to 61. The React render function can return `Loading…` text for couple of cycles and after retrieving the actual data from database, it can then pass the data as props to the sub components as illustrated in the Figure 20.

On the User personal view, `Omat suoritteet` contains a table where users all inputted tasks are listed. User can toggle between done and undone tasks. Manual and instruction panel is default closed and it can be easily slide open by clicking the `Ohje` panel. The instruction panel is illustrated in the Figure 21.

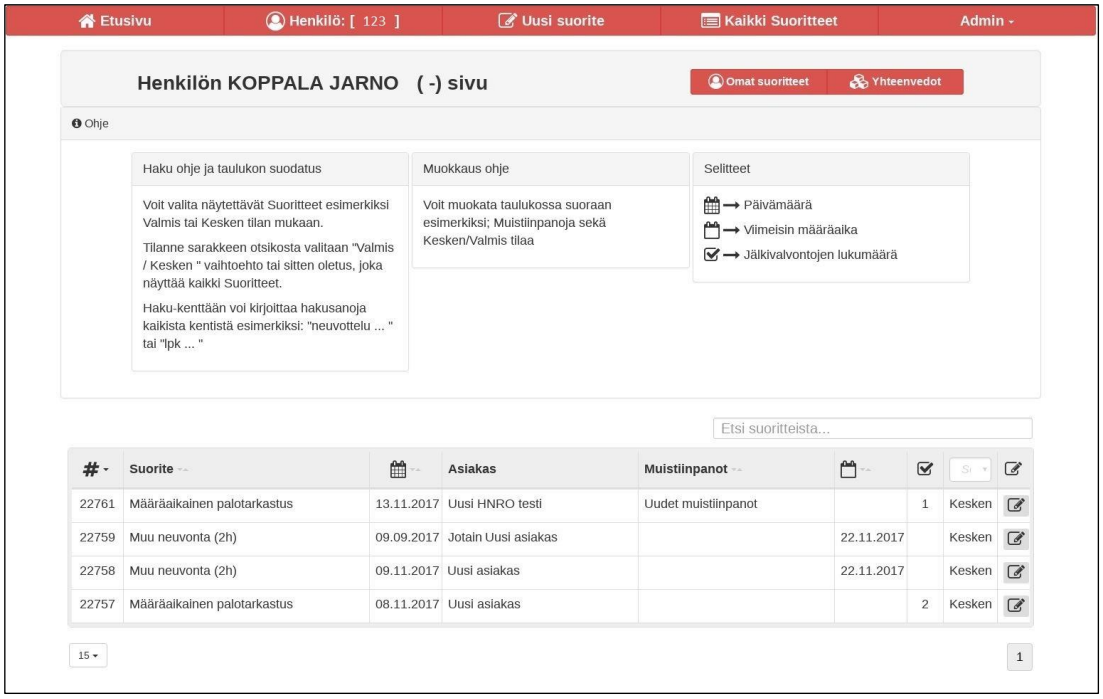


Figure 21. User personal page and tasks.

In the personal user-view menu header there is a small navigation bar where the user can toggle between two views: `Omat suoritteet` and `Yhteenvedot`. The user task table uses `BootstrapTable` component for displaying data as illustrated in the Figure 21. When clicking editing link on the task row the application calls the `this.props.history.push` directive to open `MuokkaaSuoriteSivu`, which is a high order component. The `MuokkaaSuoriteSivu` component uses the React lifecycle method component called `componentDidMount`. Inside that method it calls `this.props.fetchSuorite(id)` function for retrieving selected task data details from the API with selected `id` parameter. Within the `MuokkaaSuoriteSivu` components render function it calls and loads `SuoriteLomakeMuokkaa` component that handles the actual editing form from `common-components` folder structure.

The `BootstrapTable` component has multiple features for handling the data and it has watchers to handle event based user actions. These actions are general dummy

functions that can be coded to handle any kind of events. In this project the objective was to edit data directly on the table view and one click edits the selected row data e.g. `Asiakas` field. Before saving action the `BootstrapTable` calls `onBeforeSaveCellSHK` function that was coded to throw an html `confirm` handler box for the user action. If a user selection was true, the `onAfterSaveCellSHK` function then alerts user: “Saving succeeded on row `${row.SuorTunnus}` with `${cellName}` and `${cellValue}`”.

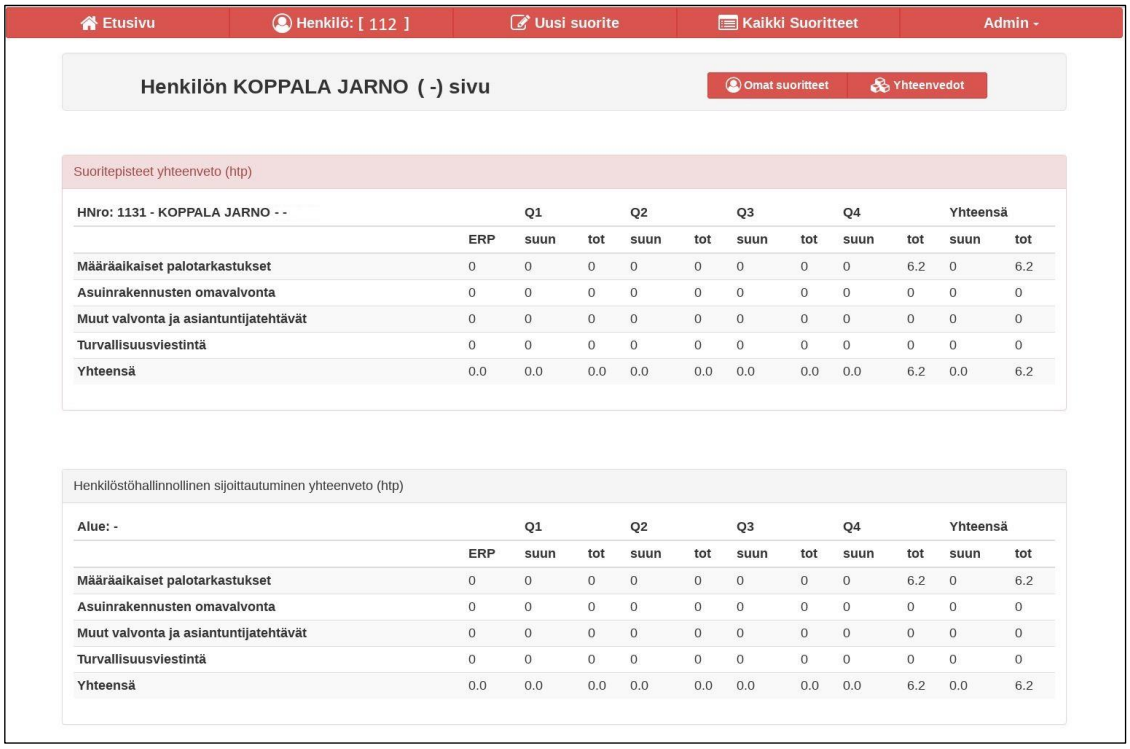


Figure 22. Personal score charts.

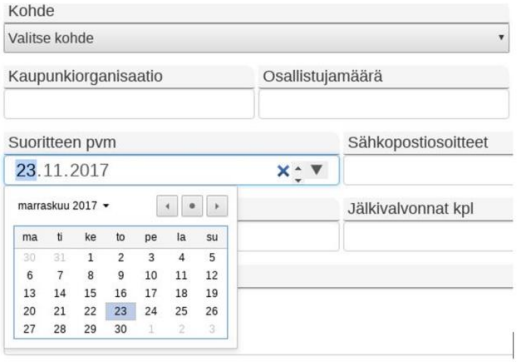
Personal score tables were coded with npm UI library called `react-bootstrap` and it included a general table component that could be used with fixed data. The personal score table is presented in the Figure 22. The personal view `SuoritteetHenkiloYhteenveto` high order component is responsible for connecting to the Redux state data and mapping the user personal score data to application state. The components `SuoritepisteetHHsijTaulukko` and `SuoritepisteetHnroTaulukko` handled the data from the application state and returned it structured as a table inside the parent `SuoritteetHenkiloYhteenveto` render function.

The ERP application has an interface to input and edit data through this `redux-form` component. The form component is dynamic and versatile for multiple use case scenarios. The form can be initialized with database data if user selects edit a task from at the task list table as illustrated in the Figure 23.

The ERP applications value is in inputted and a valid normalized data. The input form has multiple features that were coded to assist the user's daily work. The input validation and dynamically changing form based on user selection is in essential role when considering user experience. More than a half of this projects work hours were spent on the user-form component validation and its features.

Figure 23. Input form – new task step one.

When creating a new task the form begins with only one selectable field and `Tallenna` button is disabled by default as presented in the Figure 23. When advancing at the form, more fields are displayed and after the minimum validation passes, the `Tallenna` button is then enabled. The user disabled and read-only viewable fields were greyed out e.g. average calculation fields as illustrated in the Figure 25 of the editing task form example. The dynamic form will guide the user for advancing in desirable logic at the form. This type of an approach offers better user experience. Depending on the type of a task, the form will revealed only needed fields for inputting.




```
260  //  DATE-TYYPIN LASKENTA
261  //  LASKETAAN LODASH DATETIME JA laitetaan momentilla input "datetime"
262    if (values.Pvm) {
263      const lodashDate = (values.Pvm);
264      const momentDate = moment(lodashDate).format("YYYY-MM-DD").toString();
265      values.Pvm = momentDate;
266    };
```

Figure 24. Datetime picker and moment function snippet.

Users input for date were performed with date component field as illustrated in the Figure 24. The `datetime` picker was initialized with the current date if not changed. For displaying and saving the `datetime` correctly, the data from the database was parsed with `moment` function to alter the format to correct form on line 264. Users benefitted with easier access for inputting the date and faulty dates were prohibited.

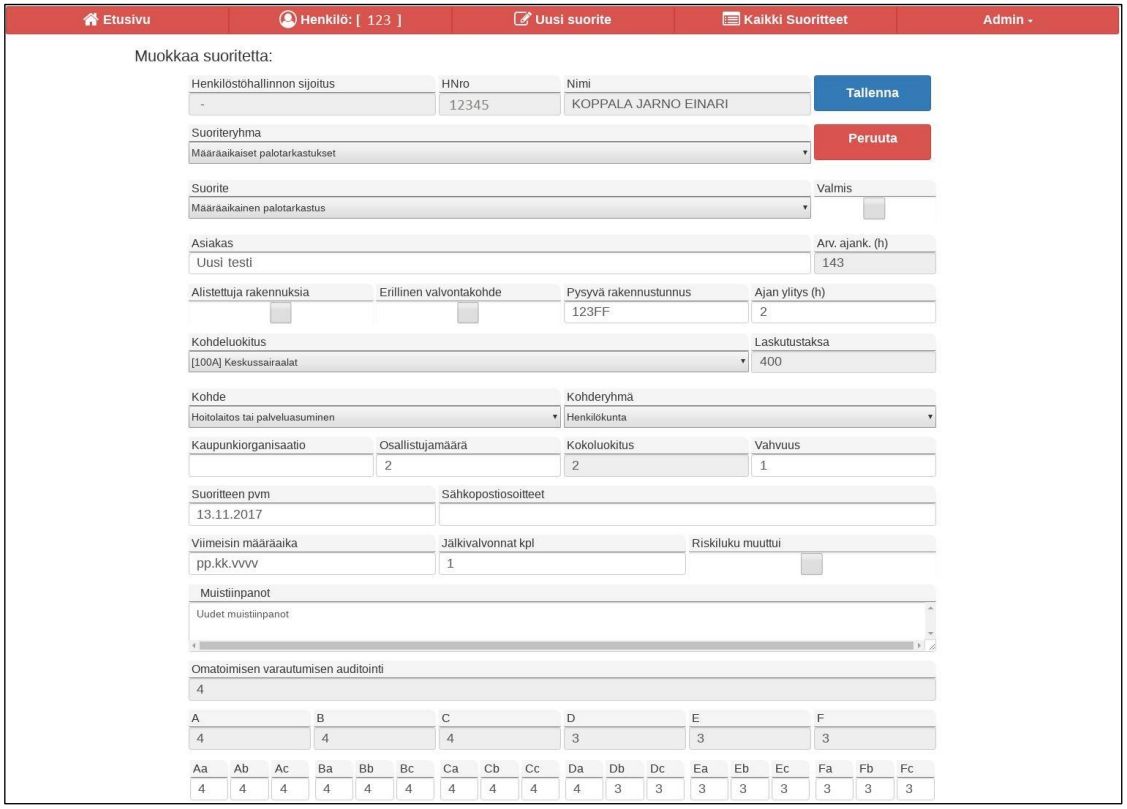


Figure 25. Edit task input form.

The user form had 52 data fields and from generic text type field all to gather 13 form functions were done. These field types consisted three basic input fields, four number fields, textarea field, two checkbox fields, date field, select menu field and an email field. Different needs were typically related e.g. in number field to different allowed values and handling disabled or enabled status. The developed input field are presented in the Figure 25. The categorization of the fields guided the form more intuitive in its behavior.

All of the field functions were separated to common components for the DRY method reasons. The user form validation was an extensive function with several coded features. The editing form components validate function included 22 `if` statements for handling validations and creating a new-task form validation included 14 `if` statements. These validation purposes are presented in the Figure 26.

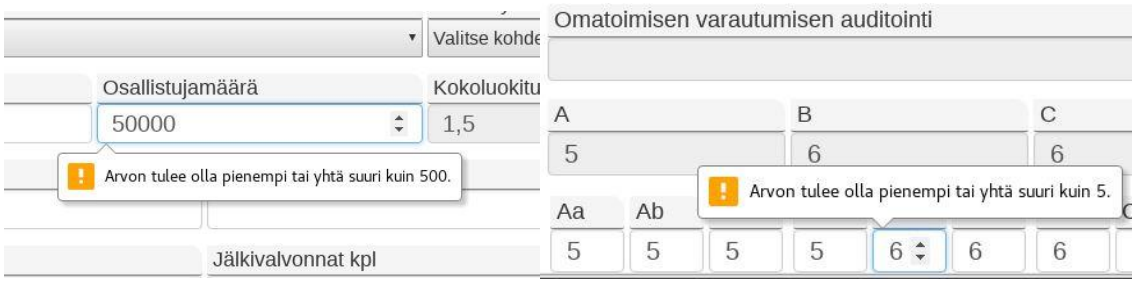


Figure 26. Form validation examples.

The fire inspections audition calculation values were set from one to five and a user received notification if the number was not valid. The calculation on averages could result faulty without form value validation.

```
62  // ERROR - Jos kenttä on tyhjä, laitetaan merkki * (tämä määritelty kentässä, missä virhe näytetään) /
63  if (!values.Asiakas) {
64      errors.Asiakas = "*";
65  };
```

Figure 27. Input form validation snippet - error example.

The validation was for not only checking users valid input as in the Figure 27 but the input form also conducted business calculations based on rules as illustrated in the Figure 28. The generic number fields and e.g. email field uses its own html based input validation and external validation were not needed in such scenarios. The basic HTML component itself checks its validation based on World Wide Web Consortium (W3C) rules and the error tip notification is displayed on top the input field. Other features such as `Clear` user input and information of the users saving details were hidden by default.

```
214         // JÄLKIVALVONNAN TUOMAT PISTEET
215
216         if (values.Jalkivalvonnat >= 0 || values.Jalkivalvonnat == null) {
217           const JalkivalvonnatHelper = (values.Jalkivalvonnat * 0.25) + values.SuorPisteet;
218           values.SuorPisteet = JalkivalvonnatHelper;
219           values.Arvioitu_ajankaytto = (values.SuorPisteet * 4);
220         };
```

Figure 28. Input form validation snippet - calculation example.

The most challenging part in the form was a data retrieving dropdown select menus to accomplish. The select menu options were needed to retrieve a new set of information from the database based on the user selection and dynamically change the displayed form fields. Typically, the illustrated data was not the `key` or `id` for the sub-selection

menu `id` for displaying the correct data. An example of developed select menu is presented in the Figure 29.

```
193 {!!this.props.SuorRyhmaArvo && (
194 <Field
195     name="SuorNro"
196     component="select"
197     className="select-box">
198         <option value="">Valitse suorite...</option>
199         {this.props.valintavalikko.perussuoritteet.map(perussuorite => (
200         <option value={perussuorite.STunnus} key={perussuorite.STunnus}>
201             {perussuorite.Suoritenimi}
202         </option>
203         ))}
204 </Field>
```

Figure 29. Select-box snippet.

As demonstrated in the Figure 30 the business-logic calculation validation was also related to the select menu selections. The selected values had to filter information needed `lodash` sort function called `_.find` on line 43 and then the correct value was passed as a multiplier or sum value to the other input field.

```
37  if (values.Kohdeluokitus == null) {
38      values.Laskutustaksa = 0;
39      values.KokoLuokitus = 1;
40      values.Vahvuus = 1;
41    } else {
42
43      const KohdeluokitusHelper = _.find(props.valintavalikko.kohdeluokitukset, function(o,props) {
44                                      return o.Kohdetyyppi == values.Kohdeluokitus; });
45
46      values.Laskutustaksa = KohdeluokitusHelper.Laskutustaksa;
47      values.KokoLuokitus = KohdeluokitusHelper.Kokoluokitus;
48
49    // VALINTA - Alistettuja rakennuksia TRUE / FALSE. Asetetaan uusi KokoLuokitus sekä Laskutustaksa
50      if (values.AlarakennuksiaB) {
51
52          if (values.SuorPisteet > 0) {
53            values.Laskutustaksa = (values.Laskutustaksa + 100);
54            values.KokoLuokitus = (values.KokoLuokitus + 0.5);
55          };
56
57      };
58  };
```

Figure 30. User selected value and validation calculation example.

Throughout the form-developing phase, some of the database field types also needed to change. Some findings were actually important to alter for future developing aspects and improvements e.g. static building number field from `INT`-type of field to `VARCHAR`-type of field. This static building number will include characters in few years from now as nationwide normalization is done. These type field changes were committed to the actual production environment as well.

The applications layout was developed as the application development process advanced. The main goal was not to adjust layout and design to its perfect but demonstration purposes some improvements were made. One improvement was `react-bootstrap glyphicon` module implementation with clickable icons and menu navigation. With table views, the column header space with short number data is actual layout dilemma and with glyphicons, it can be solved apparently. All of the implemented application glyphs were explained in detail in the info panel. This information panel method improved the application usability and the user experience.

5.6 Mobile Development

During the development phase, the mobile usage scenario for tablets and smartphones was taken into consideration and all application views were tested with small resolution screens in the web browsers developer mode and on an actual device.

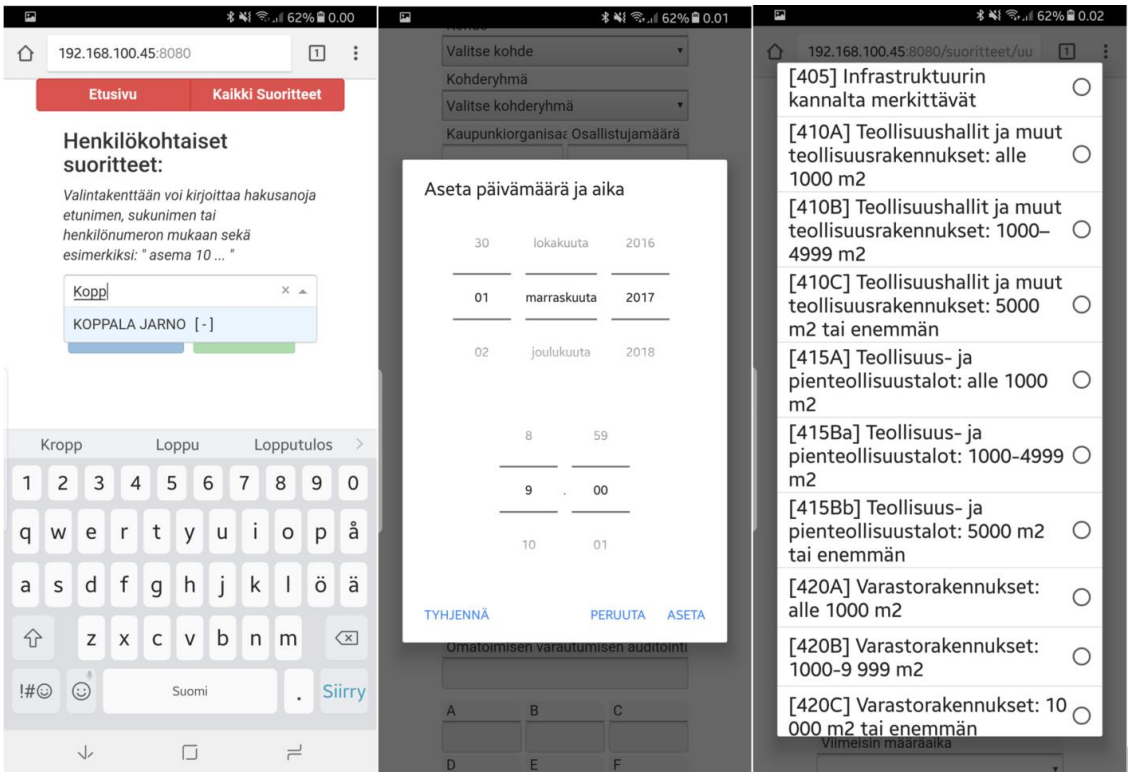


Figure 31. Design view for Android, Date picker and Select menu.

One of the use scenario would be that the fire inspector could fill the basic form information on the site and fulfil the details afterwards at the office if needed. These

mobile device views offered the organization a better understanding how the actual application could work in a small factor device. The response from the organization was positive and the solution could offer a simple approach to storing notes during the inspection. As presented in the Figure 31 an application selection menu e.g. date and selection menu, were operating differently with various mobile browsers. The selection menu values were not visible in full details with Samsung's integrated browser compared to the Chrome browser. The web application could be offered securely for the end users with organizations current VPN platform for web applications.

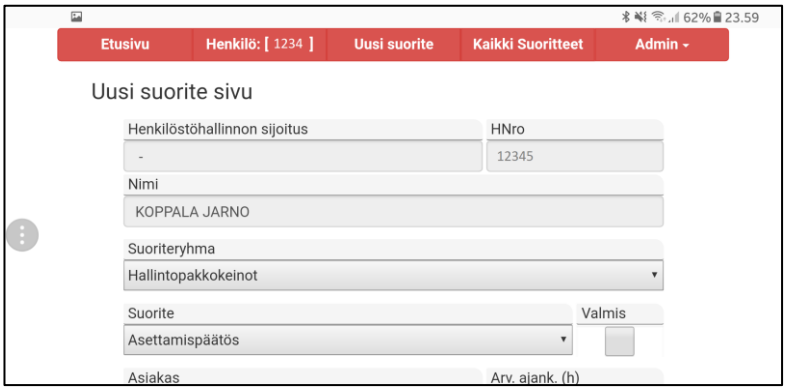


Figure 32. New task page on Android device.

The input form has been designed for a mobile, a tablet and a large device as illustrated in the Figure 32. With different screen sizes, the input field order has to be formulated and coded correctly in order to obtain the form layout to work as expected.

5.7 Component and Manual Testing

The application testing is mandatory. Typically, some type of automated script testing models are integrated with the React template to receive the developer environment up and running in a relative short period. Within the development Alpha project, only manual testing were included and automated testing was intentionally scoped out. The selected template did not include automated testing environment nor build publishing or versioning structure. The application was developed in the Webpack hot reload environment and all errors were caught immediately. Faulty components could be excluded easily when components were not working and the applications other modules and components still operated normally. When the amount of components expand, the manual testing approach is not possible to maintain.

6 Summary and Conclusions

The master thesis project included a research and a development of an Alpha test of the selected JavaScript technology named the ReactJS. The projects main objective was to develop a functional application utilizing agile developing methods. The research phase of the study compared different state container technologies for the ReactJS. The development phase included an eight-week period Alpha-model developing project utilizing the KEHMET method principles.

In order to achieve a good quality project with a desired outcome a documentation is in an essential role. The project plan with mock-ups, an exclusion list, and a project work-hour list assisted the development and guided the projects weekly plan. The installation and environment documentation were extremely necessary after few months from the end of the project. After the project, the initial setup with all configurations done could not be clearly remembered without a detailed documentation. The detailed documentation is especially needed when the project is handed to the maintaining process or the development continues. A pure developing container technology was not an option for this project because the test application needed to work in actual production environment as it is.

This master thesis scope and objective guided the actual project and in the beginning these excluded features where discussed with the organization. The discussion helped to navigate through distractive expectations to the actual agreed goal. When developing with agile methods it is relatively easy to change the application features and selected components during the development. These changes could however burden the developer especially within a fast phased eight-week project with limited resources and tight time constraints. During the development, all of the components were wrapped into the ReactJS HOC component structure for codebase readability and reusability purposes. A fast phased and agile development projects enable organizations to implement a new application solution with a low cost and a controlled risk. It is recommended to use readymade and manual templates with a new JavaScript projects when the skill level of the programmer is at junior grade.

The challenges of the project were in the initial phase of the project when the new components were implemented. One of the challenges was to understand complex data models. It is essential to understand from the theory to practice how the data flows

through the application. All of the components of the project that were implemented and used from the npm library were time-consuming tasks to understand to truly master the component capabilities. In addition, e.g. the REST structured database API was difficult to implement from the code snippets from the internet.

When developing an application, testing sets a foundation for producing a good quality application. The Chrome developer tools are useful for debugging a problem, but not for a consistent testing. Testing with a live browser is usually slow and unit tests with Mocha or Chai are needed. When Mocha testing runs on a background the function testing does not require web browser at all. The Webpack hot reload feature is necessary only for a browser testing. When using the hot reload feature there is no need for a refresh page to reload the DOM function. The logical next step would be implementing a unit-test environment template for the application because the application includes multiple components and functions developed.

The current production environment also benefitted from this project, as several discussions were done throughout the project and the developer of the current production environment had not even considered some of the solutions beforehand. The development project was a successful project to the organization from the point of view of project objective, scope and timeline. The developed application met its timeline every week and all agreed features were developed. The project conducted additional features to the test application. After the master thesis project, the organization has a newly skilled developer and a solid perspective how to continue development with the production applications in the future.

References

Abramov, D., 2017. *Redux.* [Online]
Available at: http://redux.js.org/
[Accessed 01 06 2017].

Bardadym, D., 2018. *Should.js.* [Online]
Available at: https://github.com/shouldjs/should.js
[Accessed 02 2018].

Barnes, D., 2013. *Node Security.* Birmingham, UK: Packt Publishing Ltd.

Bryan Sullivan, V. L., 2012. *Web Application Security.* United States of America: McGraw-Hill.

Facebook Inc., 2014. *XML-like syntax extension to ECMAScript.* [Online]
Available at: https://facebook.github.io/jsx/
[Accessed 03 2017].

Facebook Open Source, 2018. *Introduction to Relay.* [Online]
Available at: https://facebook.github.io/relay/docs/en/introduction-to-relay.html
[Accessed 02 2018].

Facebook, 2015. *A Javascript library for building user interfaces.* [Online]
Available at: https://facebook.github.io/react/
[Accessed 17 01 2017].

Finnish Goverment, 2011. *Laws.* [Online]
Available at: http://www.finlex.fi/fi/laki/kaannokset/2011/en20110379.pdf
[Accessed 01 2017].

Flux, F., 2018. *Flux in Depth.* [Online]
Available at: https://facebook.github.io/flux/docs/in-depth-overview.html#content
[Accessed 02 2018].

Guy Harrison, Apress, 2015. *Next Generation Databases.* New York: Apress.

Heiderich, M., 2014. *ECMAScript 6 for Penetration Testers.* [Online]
Available at: https://cure53.de/es6-for-penetration-testers.pdf

Heller, M., 2017. *InfoWorld - Node.js JavaScript runtime.* [Online]
Available at: https://www.infoworld.com/article/3210589/node-js/what-is-nodejs-javascript-runtime-explained.html
[Accessed 02 2018].

Helsingin kaupunki, 2017. *KEHMET.* [Online]
Available at: https://digi.hel.fi/kehmet/

Lawrence D. Spencer, S. H., 2015. *Reliable JavaScript.* ISBN: 978-1-119-02872-7 toim. Indianapolis, Indiana: John Wiley & Sons, Inc.

Medium, 2018. *Understanding MVC architecture with React.* [Online]
Available at: https://medium.com/of-all-things-tech-progress/understanding-mvc-architecture-with-react-6cd38e91fefd
[Accessed 02 2018].

Node.js, 2017. *Node.js v7.7.4 Documentation.* [Online]
Available at: https://nodejs.org/api/errors.html

North, D., 2009. *Introducing Behaviour Driven Development.* [Online]
Available at: https://dannorth.net/introducing-bdd/
[Accessed 02 2018].

OWASP, 2017. *The OWASP Foundation.* [Online]
Available at:
www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project

ReactJS, 2018. *React documentation.* [Online]
Available at: https://reactjs.org/docs/
[Accessed 02 2018].

Sebastian Peyrott, 2015. [Online]
Available at: https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/
[Accessed 03 2018].

Shema, M., 2012. *Hacking Web Apps.* ISBN: 978-1-59749-951-4 ed. Waltham, MA: Syngress.

Stackshare, 2018. *State Management Comparison.* [Online]
Available at: https://stackshare.io/stackups/flux-vs-mobx-vs-reduxjs
[Accessed 02 2018].

Wieruch, R., 2018. *Redux and Mobx confusion.* [Online]
Available at: https://www.robinwieruch.de/redux-mobx-confusion/
[Accessed 02 2018].

Wilson G, A. D. B. C. C. H. N. D. M. G. R., 2014. *Best Practices for Scientific Computing, PLoS Biol 12(1): e1001745..* [Online]
Available at: https://doi.org/10.1371/journal.pbio.1001745
[Accessed 02 2018].

Zabriskie, M., 2014. *Axios.* [Online]
Available at: https://github.com/axios/axios
[Accessed 02 2018].