TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Valentin Popov 142421IAPB

TIME TRACKING WEB APPLICATION

Bachelor's thesis

Supervisor: Deniss Kumlander
PhD

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Valentin Popov 142421IAPB

TÖÖAJA JÄLGIMISE VEEBIRAKENDUS

bakalaureusetöö

Juhendaja: Deniss Kumlander
PhD

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Valentin Popov

19.05.2018

Abstract

The aim of this thesis is to investigate new solution including technology stack selection for Unit4 Eesti OÜ, to migrate from Excel sharing document to a web platform. And then use it for working time tracking and other features like dashboard, statistics and reports.

As the result of the work will be full stack web application including front-end and back-end working code. It allows users to keep their working hours on server, edit them, see statistics and stay calm for data safety. Application provides great opportunities for further expansion of the functionality. The application is made for any screen size and very easy to use.

During project development researched many JS dashboard libraries and further implementation with Vue.js.

This thesis is written in English and is 42 pages long, including 5 chapters, 29 figures and 3 tables.

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on uurida uut lahendust, sealhulgas Unit4 Eesti OÜ tehnoloogiakogumi valikut, et minna Exceli dokumentide vahetamisest veebipõhiseks platvormiks. Ja siis kasutada seda, et jälgida tööaega ja teisi funktsioone, nagu tööriistapaneel, statistika ja aruanded.

Töö tulemusena saab kogu veebirakendus, sealhulgas kasutajaliidese ja serveri töökood. See võimaldab kasutajatel salvestada oma tööaega serveril, muuta neid, vaadata statistikat ja olla rahulik andmete turvalisuse tagamise küsimuses. Rakendus pakub suuri võimalusi funktsionaalsuse edasiseks laiendamiseks. Rakendus on tehtud ükskõik mis suurusega ekraani jaoks ja on väga lihtne kasutamises.

Projekti väljatöötamisel uuriti paljude raamatukogude JS-juhtpaneelid ja edasine realiseerimine Vue.js-ga.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 42 leheküljel, 5 peatükki, 29 joonist, 3 tabelit.

List of abbreviations and terms

JSON	*JavaScript Object Notation*
TUT	*Tallinn University of Technology*
HTML	*HyperText Markup Language*
API	*Application Programming Interface*
AJAX	*Asynchronous Javascript and XML*
REST	*Representational State Transfer*
JS	*JavaScript*
CSS	*Cascading Style Sheet*
DataTable	*Plugin for the jQuery Javascript library*
jQuery	*JavaScript library*
SharePoint	*SharePoint is a web-based, collaborative platform that integrates with Microsoft Office*
NPM	*Package manager for the JavaScript programming language*
Gmail	*Free, advertising-supported email service developed by Google*
Blowfish	*Symmetric-key block cipher*
Brute force	*Cyberattack consists of an attacker trying many passwords or passphrases*
SPA	*Single Page Application*
UX	*User Experience*
UI	*User Interface*
JSX	*Extension to the JavaScript language syntax*
ES6	*ECMAScript 6*
HTTP	*The Hypertext Transfer Protocol*
CLI	*Command line interface*
Gmail	*Google Mail*

Table of contents

List of figures

List of tables

1 Introduction

Nowadays web development goes very quickly due to increasing number of different frameworks and tools that allow to develop front end and back end more faster and more efficiently. This fact allows us to change all view logic to client side and use our server much less. As the result, applications can be more scalable, flexible to many devices display size, faster and so on. Selecting the correct technology stack can make or break a project. Sometimes, the dream to use the most fresh and new technology leads to make bad decisions in this selection. Through this thesis, there will be explained and showed which technologies were picked and why.

1.1 Background and Problem

In different field of human's work, companies count hours, which workers spend and keep by this way their progress and whole way of business development. For example Unit4 Eesti OÜ has to record and keep data of every employee, to track how many hours do they spend on different fields of their work, which takes place in Microsoft Excel documents, which always shared between workers. This method has a lot of disadvantages:

- Small amount of the devices where it can be opened
- Hard to keep and manipulate data
- Currently access from SharePoint not very reliable, one user blocks another
- Excel starts fail due large number of format
- Old technology

In this thesis will be discovered and apply new solution for this task by creating full-stack Single Web Application (SPA).

1.2 Ready solutions

- Google Docs – is a free Web-based application in which documents and spreadsheets can be created, edited, and stored online. Files can be accessed from any computer with an Internet connection and a full-featured Web browser. But using gmail account is a must, none of scalability, keeping private data, and custom features.

- Rent third-party software – is one of the fastest way to solve a problem, but in this case the client has to pay annually or monthly, overpriced unnecessary functions as well, as Unit4 do not want to share their data with third-party companies.

2 Requirements

Potential users of this application are R&D team members.

2.1 Functional

- Authentication should be done with email and password
- User has to have ability to create a new record in table
- User has to have ability to edit any record in table
- User has to have ability to remove any record in table
- User has to have ability to show different number of records per page in table
- Table should be filterable by cells, user might have mixing filterable queries
- User should see basic statistics of hours spend annually by months and field of work
- User has to have ability to change password and email inside the application

2.2 Non-functional

- Application has to have an API
- Application has to be scalable for future development
- User data has to be secured
- Application has to use JSON Web Token for server communication
- Application has to use one of the following JS framework: Angular.js , Vue,js or React
- Application has to be responsive to any display size
- Application has to store data in database

- Data has to be showed with DataTables

These requirements are full, practical, unique and necessary.

3 Used technologies

3.1 SPA

Single-page application is a web application or website that works by dynamically rewriting code on front-end without downloading entire new page from server. This way user loses interruptions from loading data and application behaves like desktop one. Interaction with the single page app often has dynamic communication with the server behind the scenes – SPA.

3.2 SPA Architecture

First of all, we will examine different web application architectures in order to fully understand difference between various approaches and eventually comprehend the nature of Single Page Applications.

3.2.1 MVC

One of the most popular architecture gives us ability to divide code of application by three parts: Model, View, Controller. The first time it was described in 1979 year. Partitioning allows you to simplify a large code in terms of volume. If the code is written with one long script, it becomes difficult to understand it, and it's hard to make changes without making an error. MVC is not tied to any particular programming language.

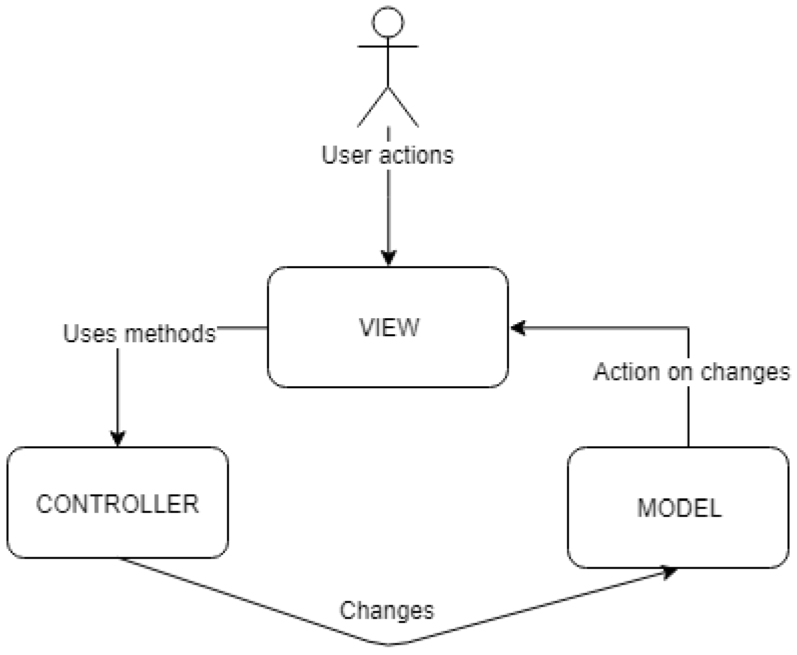


Figure 1: MVC

Partitioning here does not mean that there should be exactly 3 files in the code (or 3 file folders, or 3 classes) with the names model, view and controller. MVC does not tell us anything about how to organize code files. In practice, the model often occupies the bulk of the application, and is represented in the form of large number of different types of classes - entities, services, database classes, and separate folders for each class.

3.2.2 MVVM

MVVM was introduced in 2005 as modification of Presentation Model. It is used to separate the model and its representation, which is necessary to change them separately from each other. For example, the developer specifies the logic of working with data, and the designer accordingly works with the user interface.

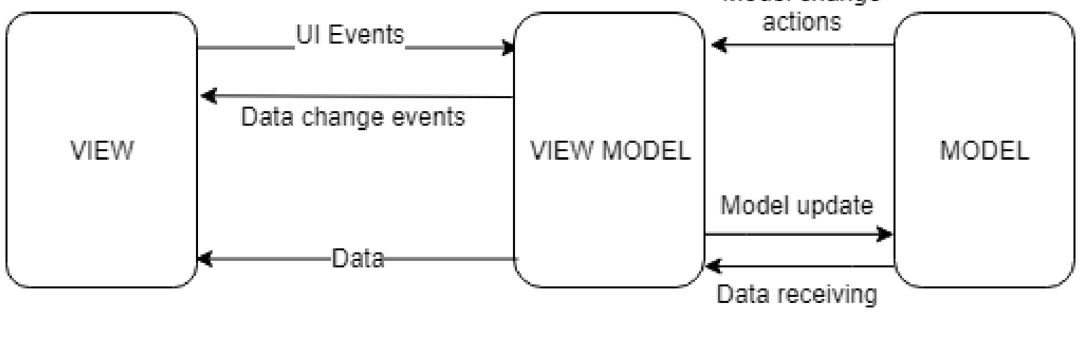


Figure 2: MVVM

Characteristics of MVVM

- Model - just as in the classic MVC, the Model is the logic of working with data and a description of the fundamental data needed to run the application.
- View - this is a graphical interface, that is, a window, buttons, etc. The view is a subscriber to the event of changing property values or commands provided by the View Model. On the event that a property has changed in the ViewModel, it notifies all subscribers about it, and the View in turn requests an updated property value from the ViewModel. On the event that the user is working on an element of the interface, the View invokes the appropriate command provided by the View Model.
- View - is, on the one hand, the abstraction of Representation, and on the other hand, provides a wrapper of data from the Model that is to be bound. That is, it contains a Model that is converted to a View which also contains the commands that the View can use to influence the Model.

3.3 Front end as Vue.js

Figure 3: Vue.js logo

This is a framework for creating user interfaces. Unlike monolithic frameworks, Vue is designed for gradual implementation. Its core primarily solves the tasks of the presentation level (view), which simplifies integration with other libraries and existing projects. On the other hand, Vue is completely suitable for creating complex single-page applications (SPA, Single-Page Applications), if used in conjunction with modern tools and additional libraries. The purpose of creating Vue.js is to provide an easy-to-learn, universal, powerful, easily supported and tested JavaScript framework.

Vue.js also aims to be progressive; this means that if you already have a ready project, then Vue support can easily be added to this project, thereby expanding the functionality and interactivity of the existing application.

Vue uses MVVM pattern.

Latest version of Vue is Vue.js 2.0, which was released in 2016 year. Now Vue has virtual DOM, server-side rendering, ability to use JSX and other useful things.

What is advantage of Vue.js?

- Small Size of framework – Vue ecosystem small and fast, it weight under 20 kilobytes after gzipping. By the way user can separate the template-to-virtual-DOM compiler and run time.
- Simple integration – Vue is very easy for developing SPA even including it in a existing application.
- Flexibility - Vue can easy handle applications with JSX, EX6, routing and handling
- Full documentation in Russian.

3.3.1 Additional Vue.js libraries

These additional libraries are used in application.

- Vue-router – is an official Vue.js supported library for easy routing configuration of the SPA
- Vuex – is a state management pattern and library. It serves as a centralized store for all the components in an application.
- Vue2-datatable-component – is a popular open-source DataTable library for Vue.js.
- Vue-js-modal – is an open-source model component for Vue.js.

3.4 Back end as Node.js

Figure 4: Node.js logo

Node.js is a server platform for working with JavaScript through the V8 engine. JavaScript executes the action on the client side, and the Node on the server. With Node, you can write full-fledged applications. Node can work with external libraries, invoke commands from JavaScript code, and act as a web server

What is the advantage of Node.js?

- JSON API - Non-blocking input/output and JavaScript make Node an excellent option for writing a wrapper around a database or a web service that communicates with the client in JSON format.

- NPM - Being an open-source technology, node.js has a shared repository of pre-made tools and modules. The number of modules in the *Node Package Manager (NPM)* increases everyday

- Real-time web applications - The *event-driven architecture of node.js* is appropriate for real-time applications

- Speed of work - Node.js *uses JavaScript in the backend*, and that's enough to understand how fast the codes works and compile. Also, it runs on the Google's V8 Engine, which compiles the JavaScript directly into machine code making it faster than most.

3.4.1 Additional Node.js libraries

These additional libraries are used in application.

- Express.js – is a popular and big web application framework for Node.js, implemented as open-source, which serves to create Web application and API's.

- Body-parser - is an open-source library, which helps to parse incoming request bodies in a middleware before your handlers.

- Mongoose - This is a JavaScript library, often used in the Node.js application with the MongoDB database. Simply, it is a object modeling tool designed to work in an asynchronous environment.

3.5 Database as MongoDB

Figure 5: mongoDB logo

MongoDB implements a new approach to building databases, where there are no tables, schemas, SQL queries, foreign keys, and many other things that are inherent in object-relational databases.

- MongoDB is classified as a NoSQL database.

- MongoDB uses JSON-like documents with schemas.

- MongoDB supports field, range queries, regular expression searches.

Using this database with combination of server-side Node.js and client-side Vue.js gives a lot of features, because there are a lot of additional modules in NPM.

4 Application Development

The current version of project source code is accessible on GitHub: https://github.com/Firrero/vue-dashboard

In order to access to all features, strongly recommended to use latest version of any browsers (at least ES5).

4.1 Application overall architecture

A general description of the architecture shows the main component that the system owns. Also, these components are described more deeply in this thesis.

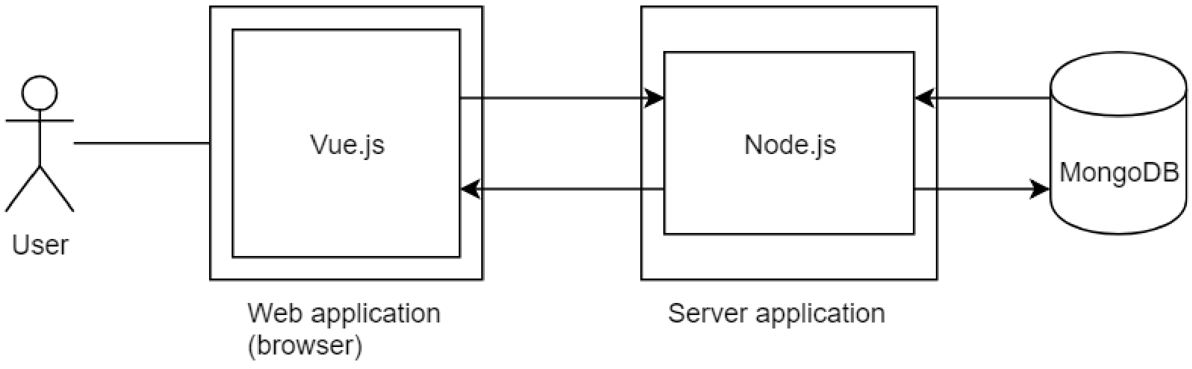


Figure 6: Application overall architecture

4.2 Use case models

The following use case diagram describes use cases that are related to data managing:

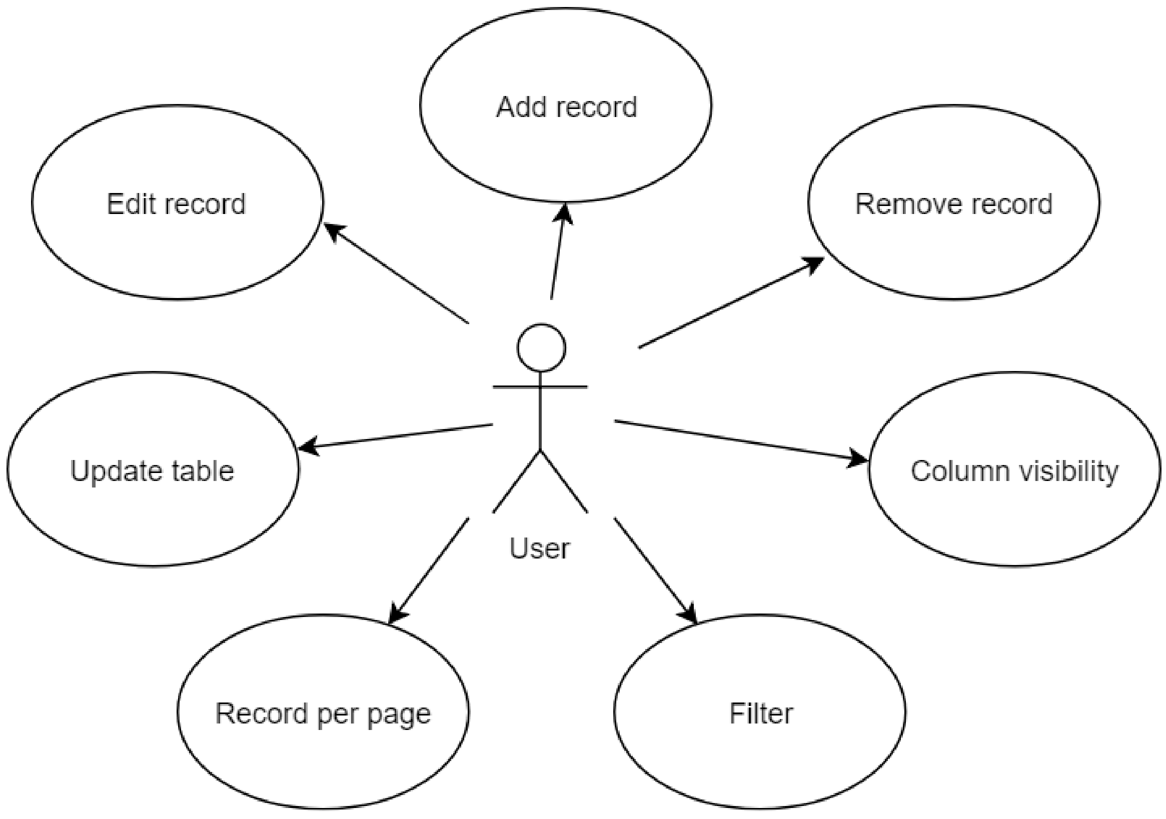


Figure 7: Data managing use case diagram

In this table 'Record' means an entity which DataTable consists of.

Use case	**Description**
Add record	Authenticated user can add record to DataTable.
Remove record	Authenticated user can remove record from DataTable.
Column visibility	Authenticated user can change the visibility of cells in DataTable.
Filter	Authenticated user can filter cells ascending and descending.

Record per page	Authenticated user can change the number of record to see per one page.
Update table	Authenticated user can manually update DataTable
Edit record	Authenticated user can choose record and edit any filed of it.

Table 1: Data managing table

The following use case diagram describes use cases that are related to account managing:

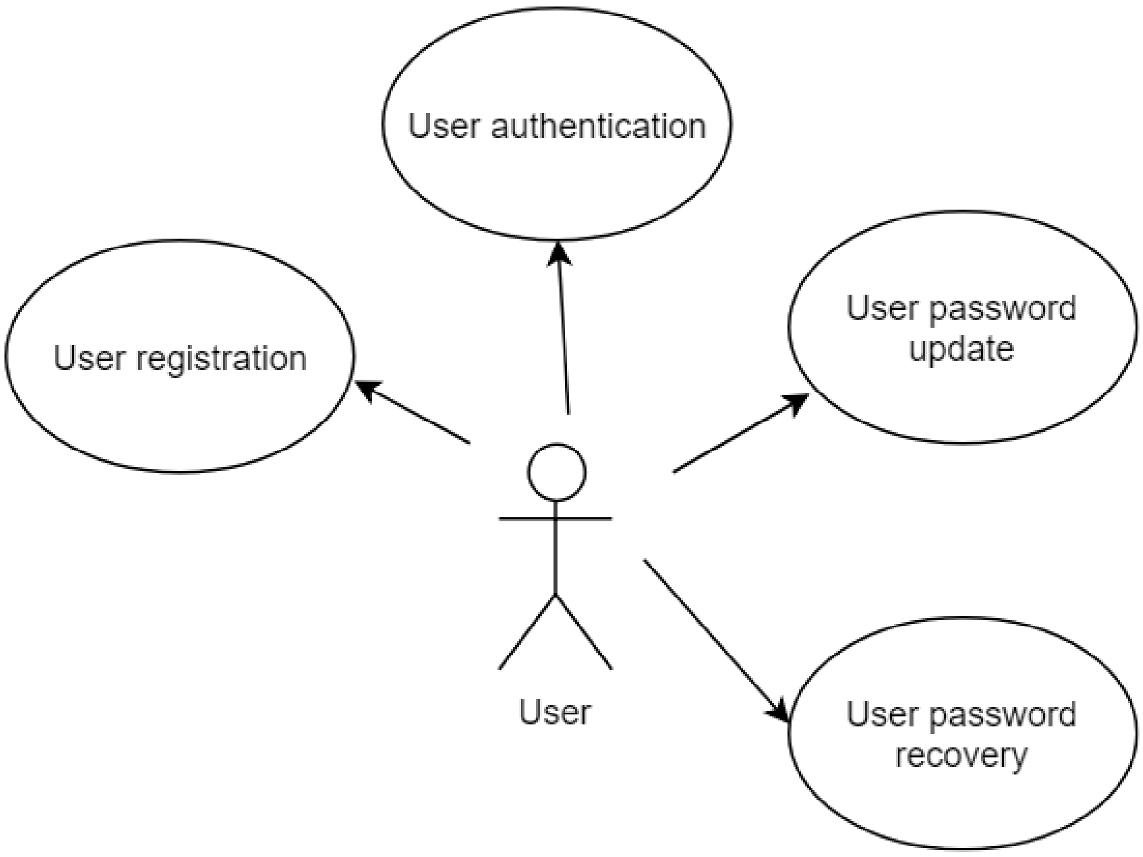


Figure 8: Account managing use case diagram

User case	**Description**
User registration	Anybody can register a new user
User authentication	In order to log in, User must provide valid credential data. Email and password
User password update	Authenticated user can change his password in user options menu
User email update	Authenticated user can change his email in user options menu

Table 2: Account managing table

4.3 Application security

The security of web applications is one the most important part of an application. Indeed, modern business processes and everyday life – more and more depends on the use of web applications in a variety of aspects: from complex infrastructural systems to IOT devices. Nevertheless, specialized tools for protecting Web applications are rather small, for the most part this task is assigned to developers. This and the use of various frameworks means of sanitation, data cleaning, normalization and much more. Nevertheless, even with the use of these tools, the web did not become safer, moreover, all the vulnerabilities of the 'classic web' practically migrated to mobile development almost unchanged. In this thesis will be described main topics and basics methods of data security of a developed application.

4.3.1 Storing password using Bcrypt

Bcrypt is an adaptive cryptographic key change function used for securely storing passwords and developed by Niels Provos and David Mazier. The function is based on the Blofwish cipher. To protect against attacks using rainbow tables, bcrypt uses salt. In addition, the function is adaptive, its operation time is easy to configure and it can be slowed down to complicate the attack by brute force

More over using and hashing bcrypt with Node.js is very easy. Npm has a lot of modules to use.

4.3.2 Data transferring using JWT

JSON Web Token (JWT) – is a JSON object that is defined in the open standard RFC 7519. It is considered one of the safe ways of transferring information between two participants. To create it, you need to define a header with general information on the token, payload, such as the user id, its role, etc. and a signature.

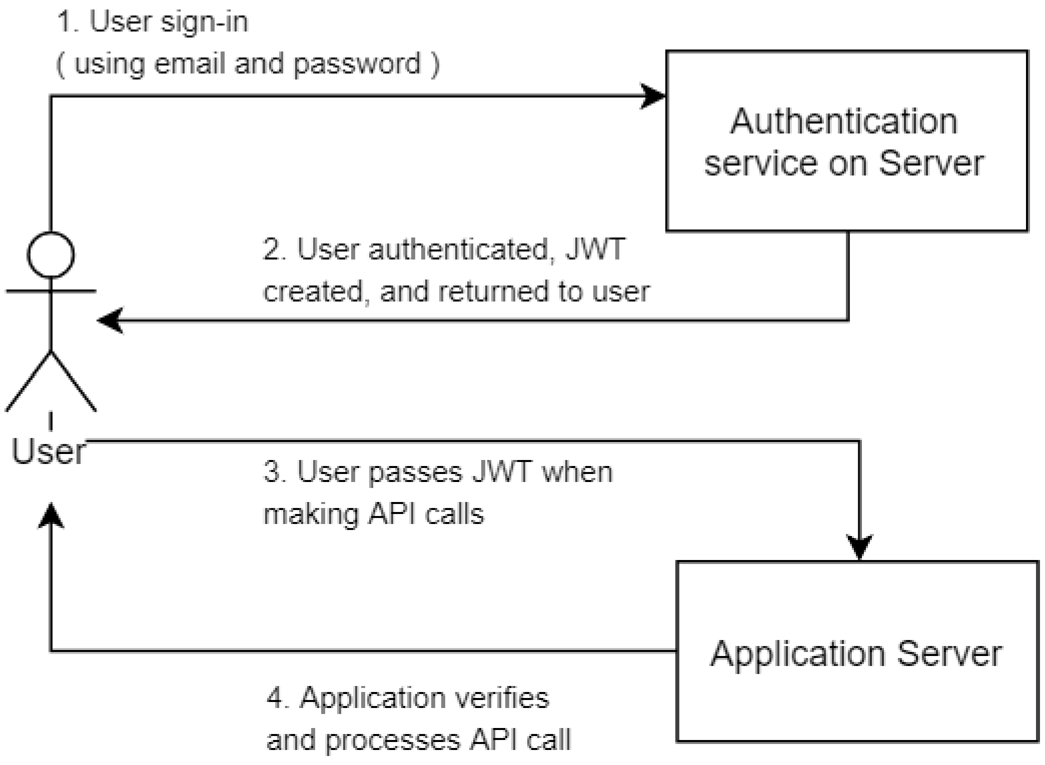

1. User sign-in
(using email and password)
Authentication service on Server
2. User authenticated, JWT created, and returned to user
User
3. User passes JWT when making API calls
Application Server
4. Application verifies and processes API call

Figure 9: JSON Web Token diagram

The application uses JWT to verify user authentication in this way:

- First, the user enters the authentication server using an authentication key (email, password pair).
- The authentication server then creates the JWT and sends it to the user.
- When a user makes a request to the application API, he add the JWT previously received to it.

- When a user makes an API request, the application can check whether the user submitted the JWT with request is the valid user or not. In this scheme, the application server is configured so that it can check whether the incoming JWT is exactly what was created by the authentication services.

Structure of JWT consists of three parts: header, payload, signature. The module which is used in application already has a header options (algorithm - HS256) and ready out of the box to make signature by itself. In this way it needs only payload data, which can be email, username, password.

4.4 Server-side implementation

The server-side of the application is implemented by using Node.js based server, which uses Express.js framework for HTTP request processing and API creation. MongoDB is used as a database of an application.

4.4.1 Server-side architecture

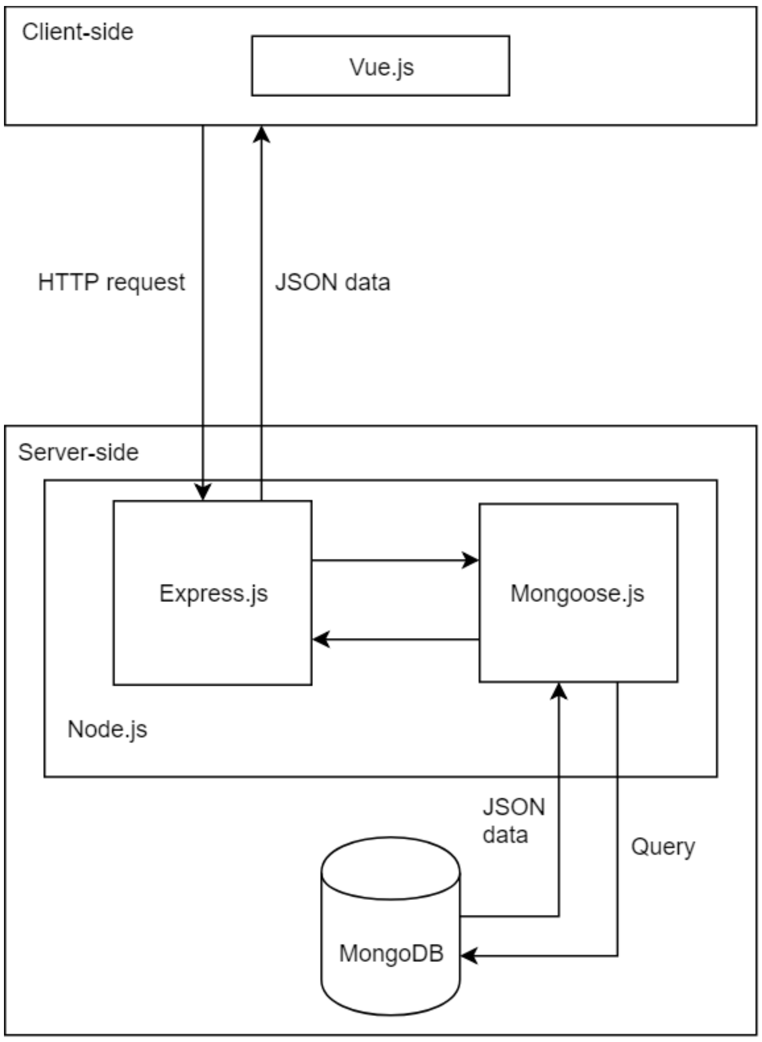
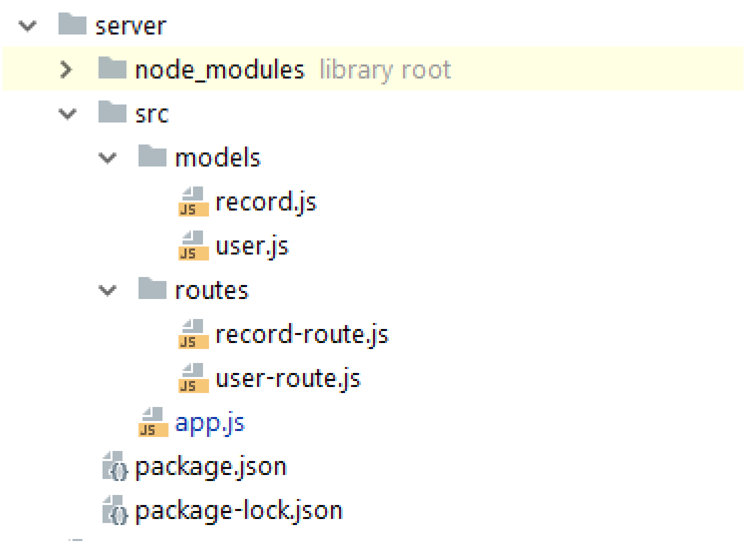


Figure 10: Server-side architecture

4.4.2 Server-side project files structure



```
server
  node_modules library root
  src
    models
      record.js
      user.js
    routes
      record-route.js
      user-route.js
    app.js
  package.json
  package-lock.json
```

Figure 11: Server-side project files structure

App.js – is a core file on the server, all server configuration like Express.js settings, API and so on locate there.

Models folder contains two files: record.js and user.js. These files describe MongoDB data model schema.

```
const user = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    email: {type: String, required: true},
    password: {type: String, required: true},
    userId: {type: String, required: true}
});
```

Figure 12: User mongoDB Schema

Routes folder contains two files as well: record-route.js and user.route.js. These files are responsible for API request processing routes.

```
router.post('/removeRecord', function (req, res) {
 var uid = req.body.uid;
 jwt.verify(req.body.token, 'secret', function (err, decoded) {
  if (err) {
   res.status(500).json({
    error: err
   });
  } else {
   if (uid) {
    Record.remove({$or: [{"uid": uid}]}, function (err, removed) {
     res.status(200).json({
      success: 'Record removed'
     });
    });
   }
  }
 })
});
```

Figure 13: API remove record example

4.4.3 Entity relationship model

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge.

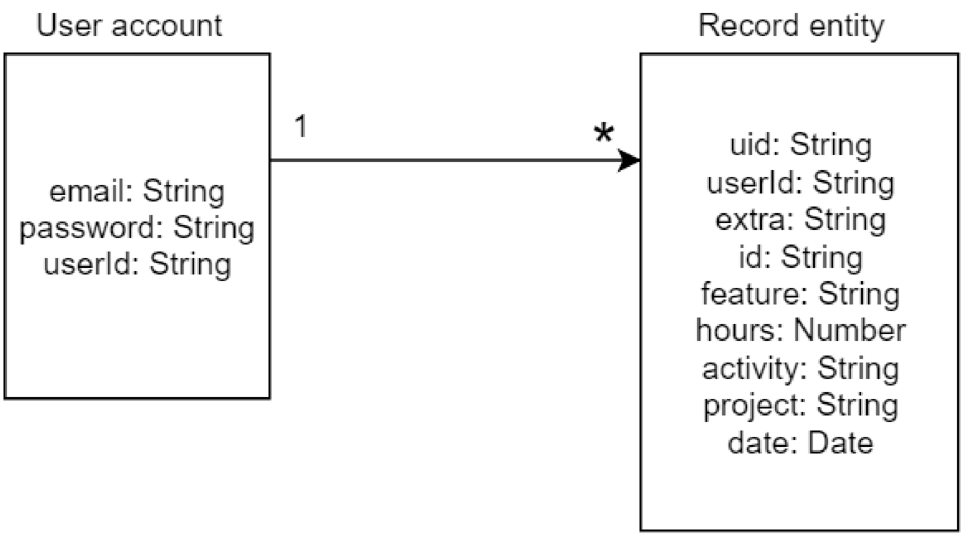


Figure 14: Entity relationship model

4.4.4 Application REST endpoints

URL	HTTP Verb	POST Body	Use case
/api/record/create	POST	JSON Object	Create record or Edit record
/api/record/removeRecord	POST	JSON Object	Remove record
/api/record/getRecord	POST	JSON Object	Update DataTable
/api/record/getStatistics	POST	JSON Object	
/api/user/signup	POST	JSON Object	User registration
/api/user/signin	POST	JSON Object	User authentication
/api/user/editUser	POST	JSON Object	User email update or User password update

Table 3: Application REST endpoints

4.5 Client-side implementation

The client-side of the application is implemented by using Vue.js framework.

4.5.1 Application design

Design is one of the most important part of any product. An application design that has a stunning look and feel is a result of efficient UX and UI work.

While creating this application, there were exact requirements:

- Fully responsive – an application must work on any devices with a different display size

- Simple and easy access – the design must be very easy to understand, and user dot not need too much explanation

- Elegant visual appearance – a well-designed interface including colors, and fonts

Taking all these factors, it was decided to use the dashboard style template. There are a lot open-source dashboard/control panel templates, but in this application is used one theme called AdminLTE.

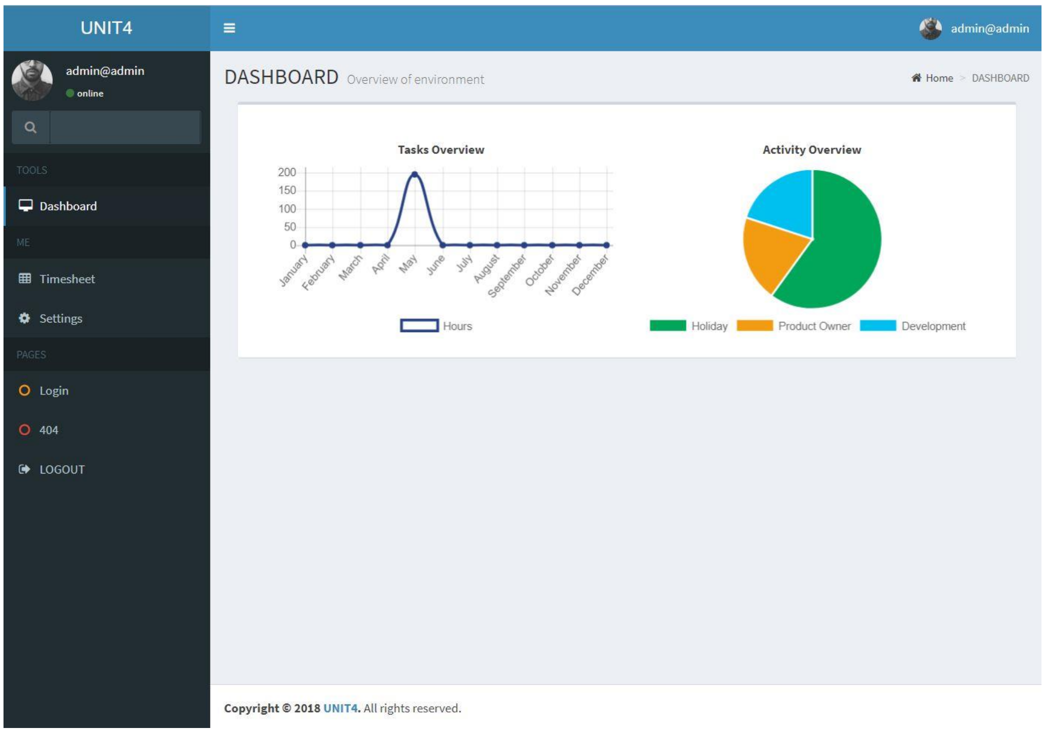


Figure 15: Application desktop view

AdminLTE is an open-source admin dashboard theme. Built with Bootstrap 3, jQuery, Font-Awesome libraries. It is fully customizable and easy to use.

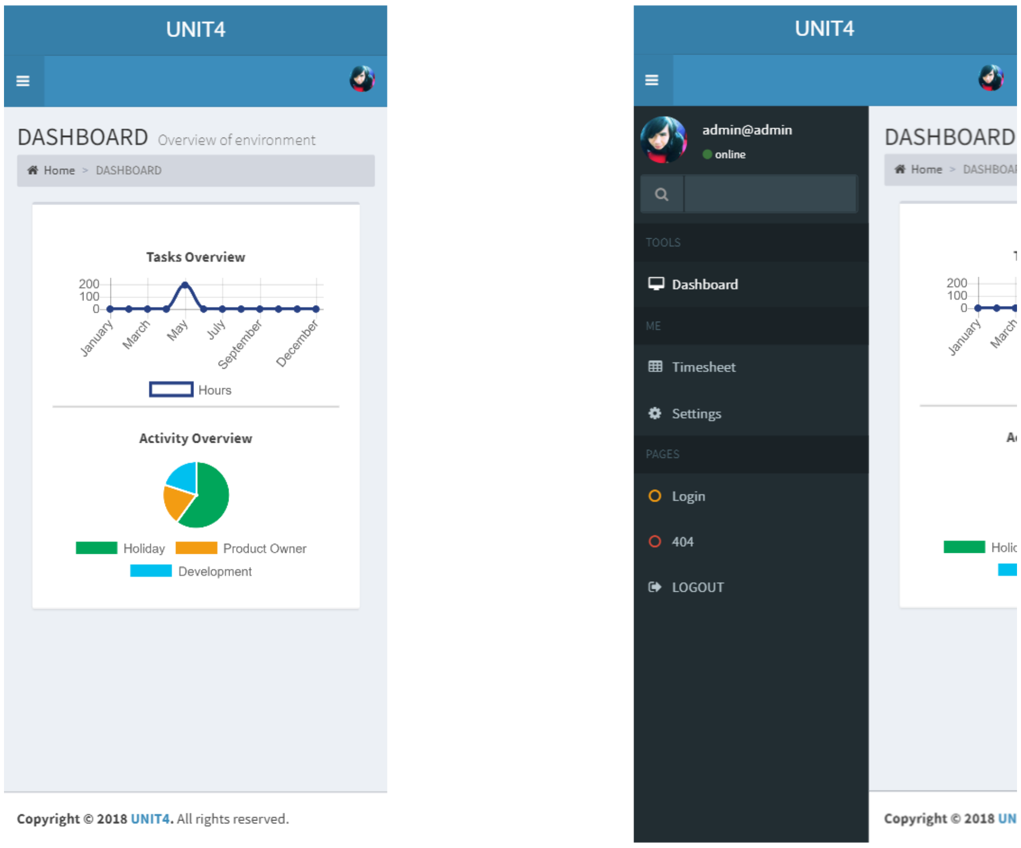


Figure 16: Application mobile view

4.5.2 Vue.js concept

Vue.js is focused on the ViewModel layer of the MVVM pattern, which was mentioned in previous chapters.

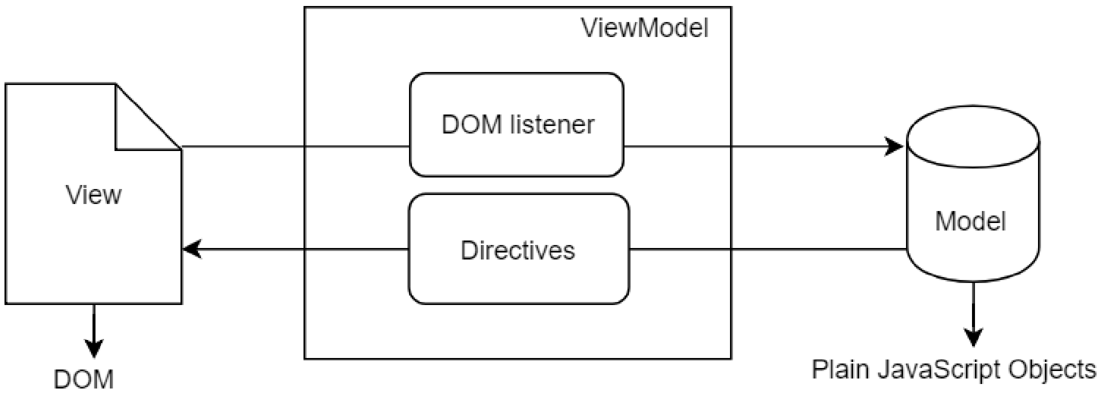


Figure 17: Vue.js concept

Vue connects the View with two-way data bindings. Dom manipulations and output formatting are abstracted away into Directives.

ViewModel is an object that syncs the Model and the View. In Vue.js, every Vue instance is a ViewModel.

View is the actual DOM that is managed by Vue instances.

Model is a modified plain JS object.

Directives is a prefixed HTML attributes that tell Vue.js how to manage DOM element.

4.5.3 Client-side overall architecture

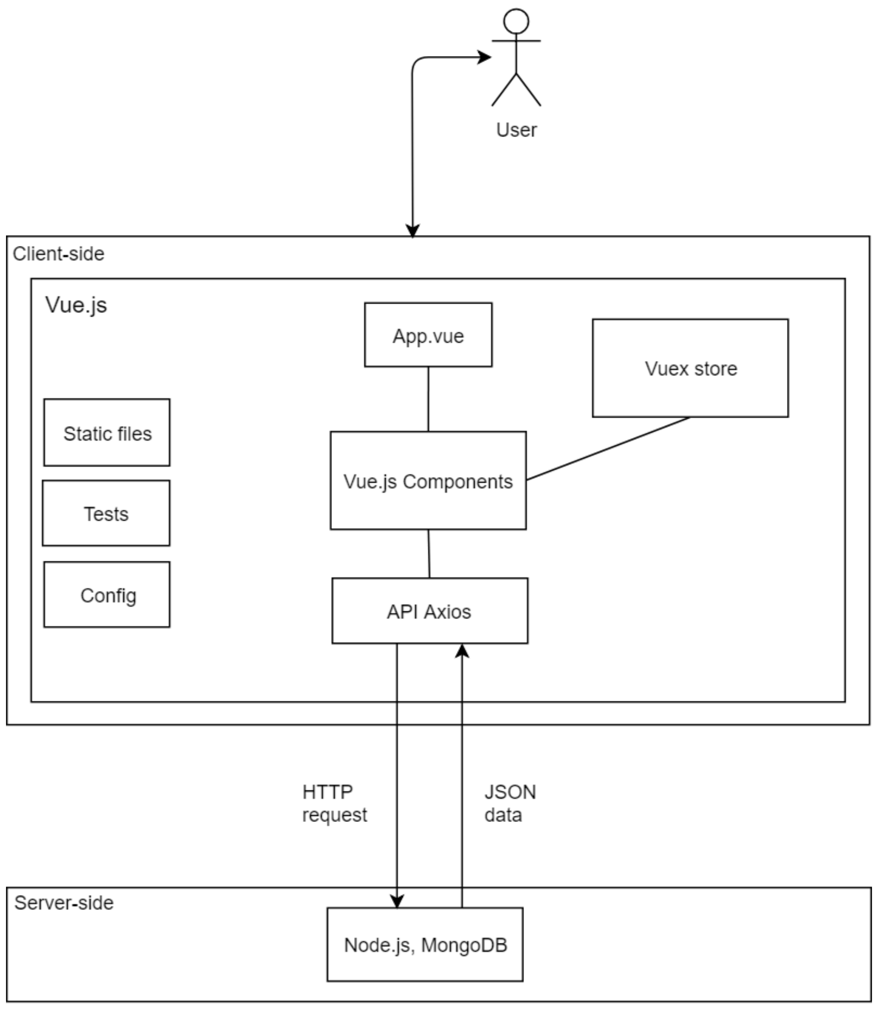


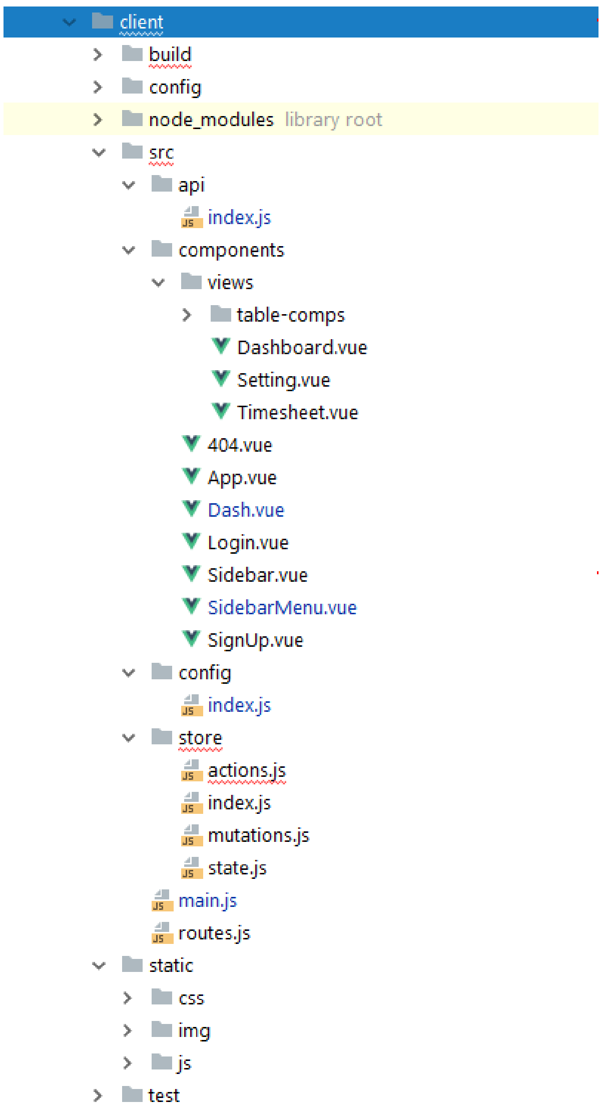
Figure 18: Client-side overall architecture

All from this architecture was described in previous chapters instead of two things:

- Vuex – is a state management pattern with library for Vue.js. It works as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable form.

- Axios – is a promise-based HTTP client for browser and Node.js.

4.5.4 Client-side project files structure



```
client
  build
  config
  node_modules  library root
  src
    api
      index.js
    components
      views
        table-comps
        Dashboard.vue
        Setting.vue
        Timesheet.vue
      404.vue
      App.vue
      Dash.vue
      Login.vue
      Sidebar.vue
      SidebarMenu.vue
      SignUp.vue
    config
      index.js
    store
      actions.js
      index.js
      mutations.js
      state.js
    main.js
    routes.js
  static
    css
    img
    js
  test
```

Figure 19: Client-side project files structure

Application project is created using Vue.js official CLI. Build, config, test folders are created automatically. More detailed information can be found by this url: *https://github.com/vuejs/vue-cli/blob/dev/docs/README.md*

Api folder contains Axios plugin configuration JS file.

```
export default {
  request (method, uri, data = null) {
    if (!method) {
      console.error('API function call requires method argument')
      return
    }

    if (!uri) {
      console.error('API function call requires uri argument')
      return
    }

    var url = config.serverURI + uri
    return axios({ method, url, data })
  }
}
```

Figure 20: API methods export

Components folder contains all Vue.js components and views.

Config folder contains server URI information.

```
export default {
  serverURI: 'http://localhost:8081/api',
  fixedLayout: false,
}
```

Figure 21: API Server URI example

Store folder contains four Vuex files with all actions, mutations, states. Vuex is described in previous chapters.

Main.js file is the main file on the application, it contains root element creation, Vue-router registration and project files importation.

Router.js contains all routes and dependencies for vue-router.

4.5.5 Components

Vue.js component is every Vue instance. Components form tree like hierarchy that shows your application interface.

4.5.6 Application components hierarchy

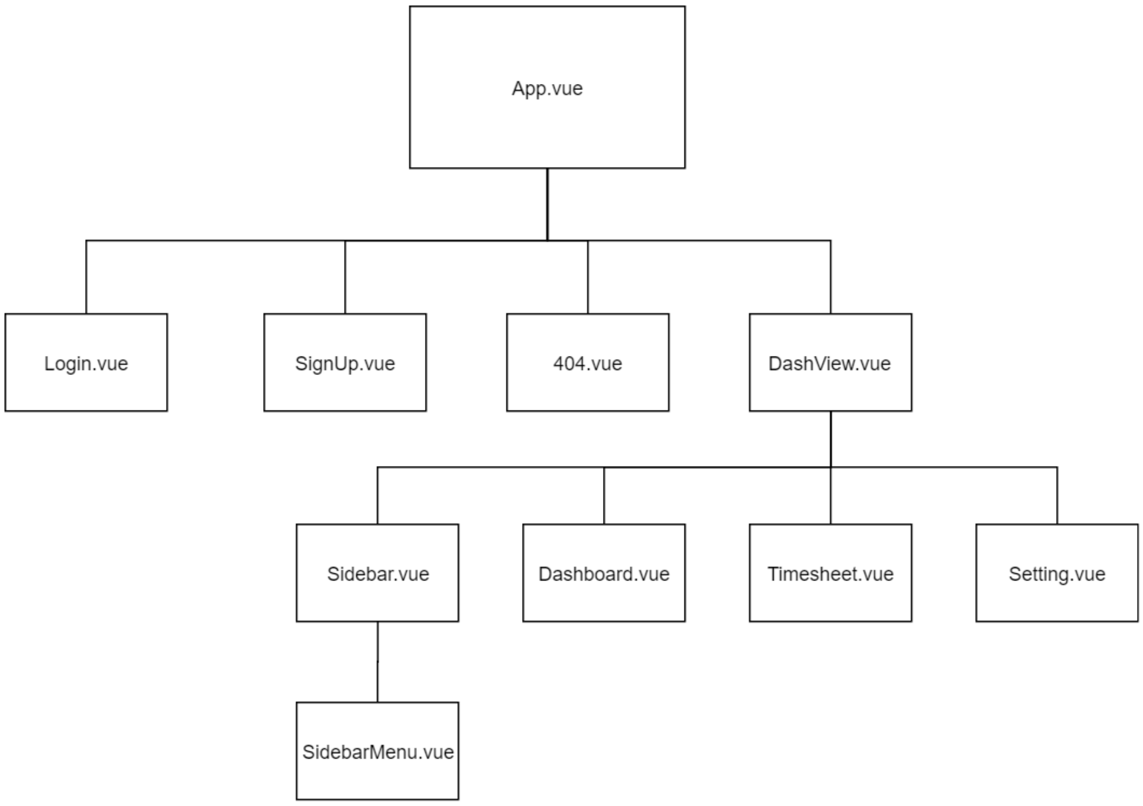


Figure 22: Application components hierarchy

4.5.7 Components overview

App.vue – this component is the core element of components tree. It contains vue-router entrance to the application.

404.vue – a component for displaying HTTP 404 error.

Login.vue – login form display component.

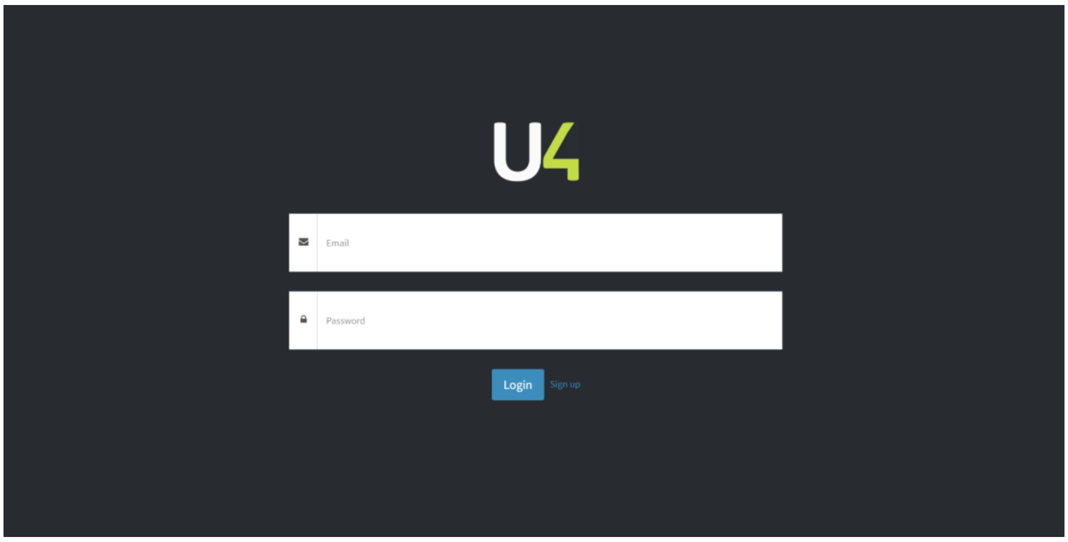


Figure 23: Login page view

Signup.vue – registration form display component.

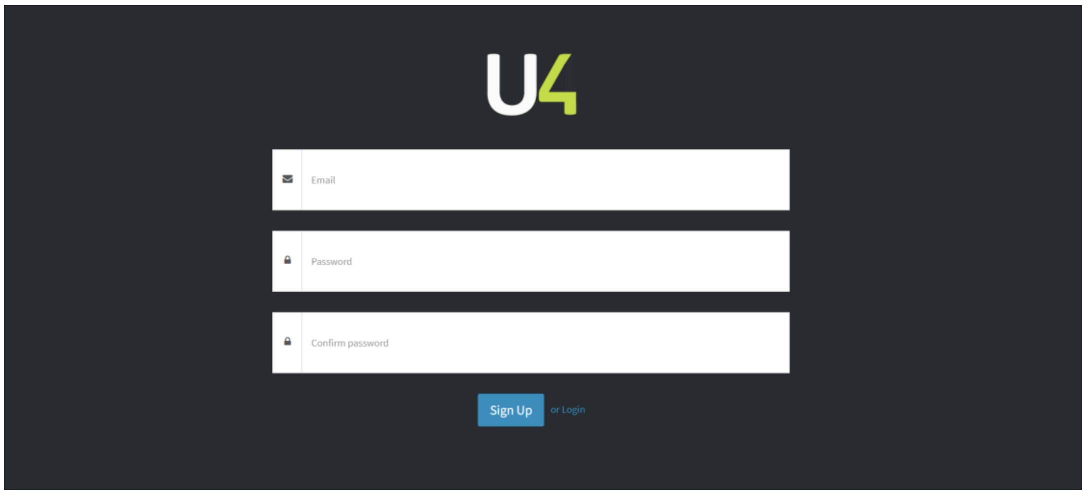


Figure 24: Sign up page view

SidebarMenu.vue – contains menu links to other components.

Sidebar.vue – is the sidebar of the application, including user display name, user avatar, search in menu field, online status and SidebarMenu.vue.

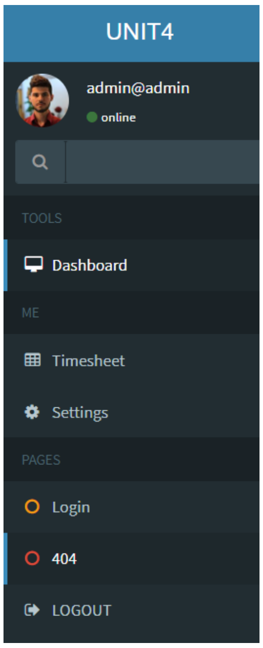


Figure 25: Sidebar view

Dash.vue – takes a role of content-container of the application, it contains toggle menu navigation with Sidebar.vue, user avatar, header and vue-router entrance for other elements.

Dashboard.vue – is a component for statistics and other useful information display. This is the first view, which user sees.

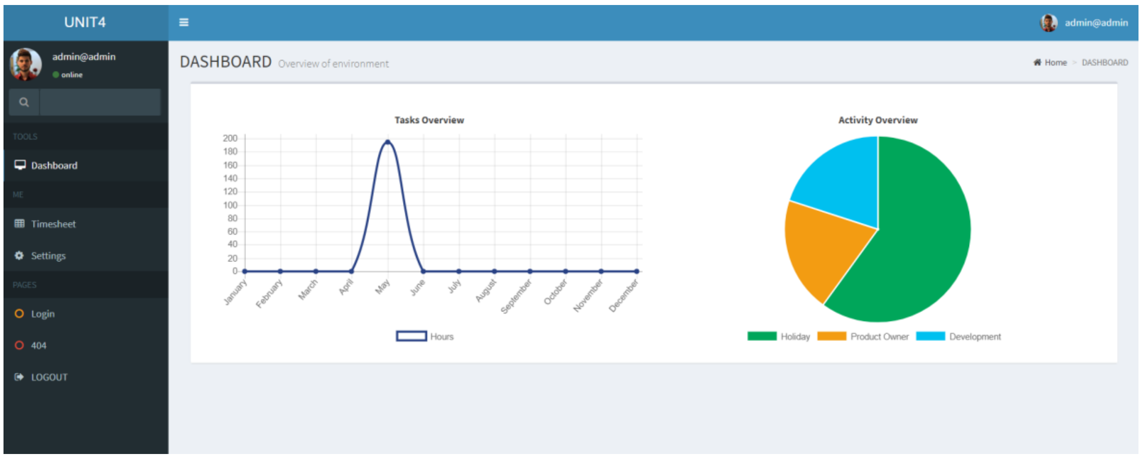
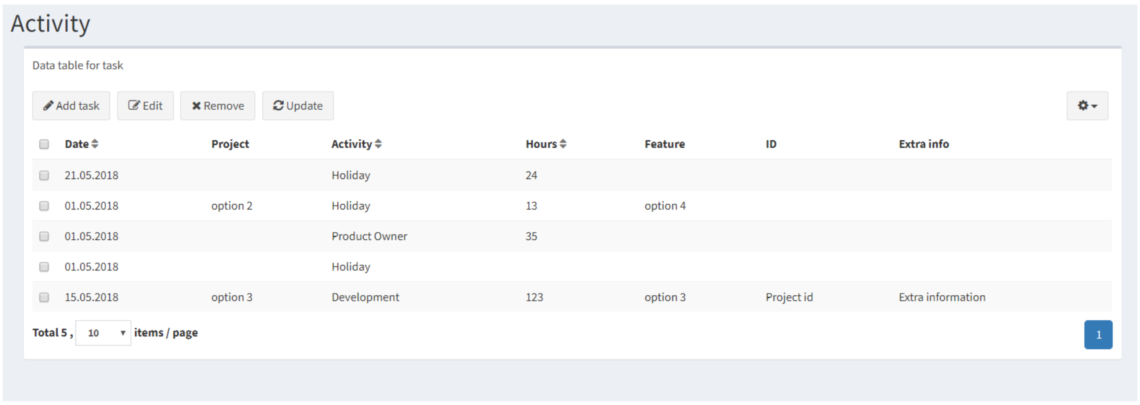


Figure 26: Dashboard page view

Timesheet.vue – is a component where user works with DataTable. Create, edit, remove data in it.

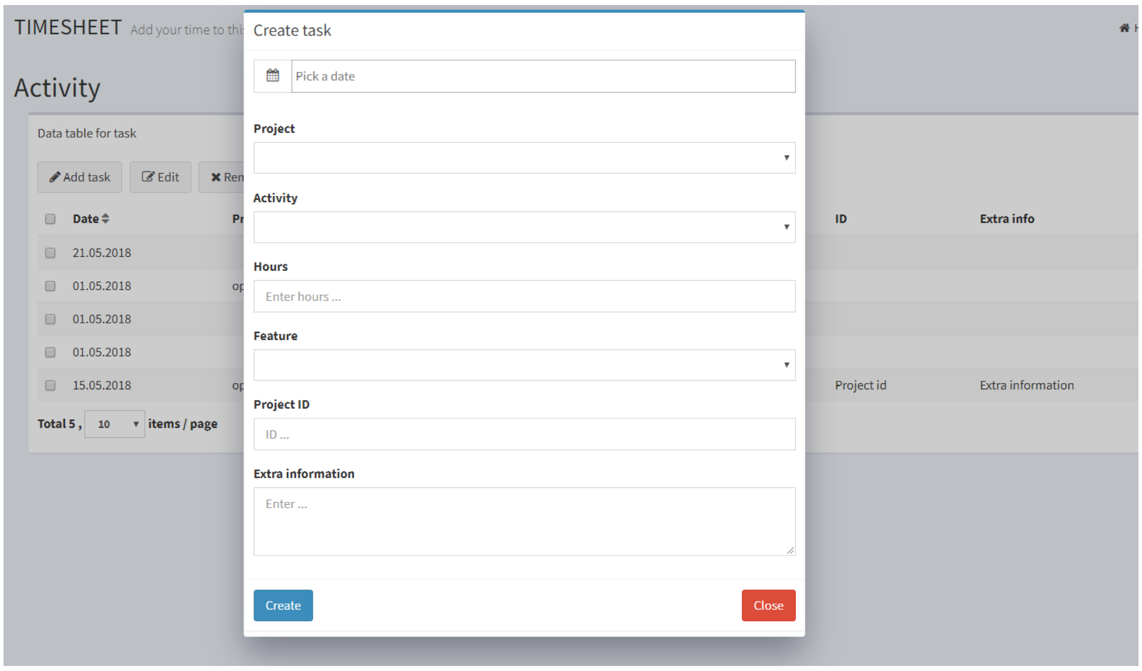

Activity

Data table for task

	Date	Project	Activity	Hours	Feature	ID	Extra info
☐	21.05.2018		Holiday	24			
☐	01.05.2018	option 2	Holiday	13	option 4		
☐	01.05.2018		Product Owner	35			
☐	01.05.2018		Holiday				
☐	15.05.2018	option 3	Development	123	option 3	Project id	Extra information

Total 5 , 10 items / page

Figure 27: Tasks table view



Create task

Pick a date

Project

Activity

Hours

Enter hours ...

Feature

Project ID

ID ...

Extra information

Enter ...

Figure 28: Task create modal view

Setting.vue – is a component where user can change email and password.

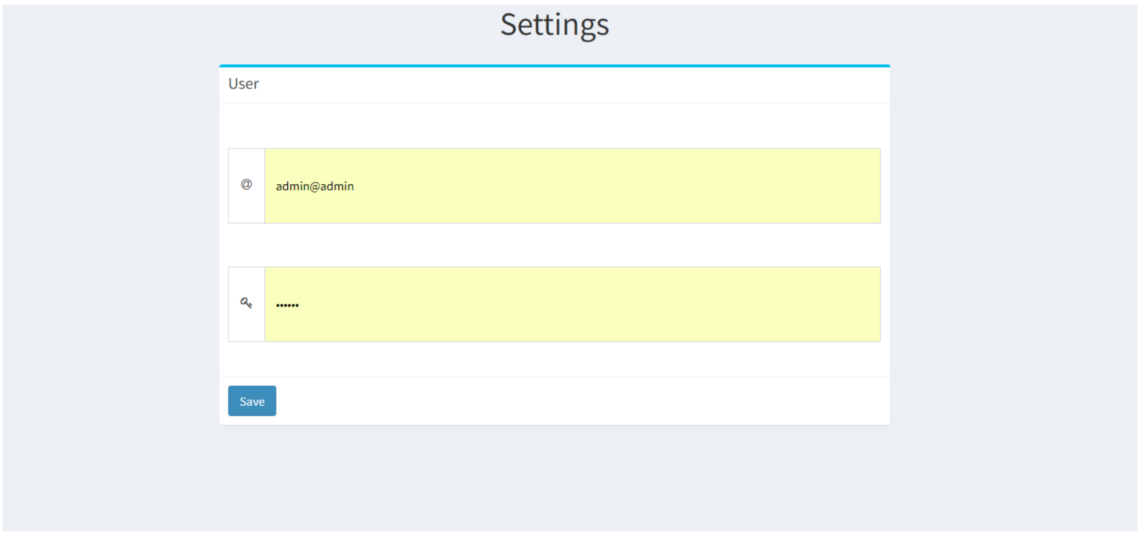


Figure 29: Settings page view

4.6 Future development

- Admin user – who can remove, create, edit any of users and their data.
- Detailed statistics – more detailed statistics with reports.
- User notifications and messages – important user notification from administrator or personal messages about tasks.

5 Summary

The main goal of this thesis was to discover and apply a new solution for task given by Unit4 Eesti OÜ with all functional and non-functional requirements. During this thesis there have been reviewed a lot of aspects of Single Page Application specifically Vue.js such as project initialization, modules organization, Vue.js application architecture, integration with third-party JavaScript libraries.

The back end of the application went through development as well, there were reviewed different back end frameworks and databases, finally we have Node.js and mongoDB as a database.

As the result the final product is a full stack web application, running with Vue.js on front end and Node.js with mongoDB including all requirements of functionality like mobile-responsive, fast, safe and independent.

References

[1] E. You, “Vue.js introduction docs,” vue.js, 2014. [WWW]. Available: https://vuejs.org/v2/guide/. [Accessed 10 May 2018].

[2] Wikipedia.org, “Express.js,” 22 04 2018. [WWW]. Available: https://en.wikipedia.org/wiki/Express.js.

[3] Wikipedia.org, “Single Page Applications,” 1 05 2018. [WWW]. Available: https://en.wikipedia.org/wiki/Single-page_application.

[4] N. Foundation, “Node.js docs,” 14 05 2018. [WWW]. Available: https://nodejs.org/en/docs/.

[5] D. Dastanaron, “Vue.js and how to know it,” habrahabr, 24 04 2018. [WWW]. Available: https://habr.com/post/351882/.

[6] “Vue-router docs,” Vue.js , 7 05 2018. [WWW]. Available: https://router.vuejs.org/ru/.

[7] Wikipedia, “Model-View-ViewModel (MVVM),” 27 04 2018. [WWW]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel.

[8] “MongoDB Docs,” 3 05 2018. [WWW]. Available: https://docs.mongodb.com/.

[9] S.T.Huang, “How not to get desperate with MVVM implementation,” Medium.com, 13 05 2018. [WWW]. Available: https://medium.com/flawless-app-stories/how-to-use-a-model-view-viewmodel-architecture-for-ios-46963c67be1b.

[10] Microsoft, “MVC Overview,” 21 05 2018. [WWW]. Available: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx.

[11] jwt.io, “Introduction to JSON Web Tokens,” 21 05 2018. [WWW]. Available: https://jwt.io/introduction/.

[12] P. Heard, “How To Architect Enterprise Single Page Application,” logicroom.com, 29 04 2018. [WWW]. Available: https://www.logicroom.co/how-to-architect-enterprise-single-page-applications-part1/.

[13] npmjs.org, “NPM docs,” 13 05 2018. [WWW]. Available: https://docs.npmjs.com/.