SAVONIA

THESIS - **BACHELOR'S DEGREE PROGRAMME**
TECHNOLOGY, COMMUNICATION AND TRANSPORT

IMPLEMENTATION OF THE SPEECH-SCENARIOS WEB APPLICATION FOR THE RUSSIAN PROVIDER COM-PANY

Author/s: Vitalii Ponich

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Vitalii Ponich			
Title of Thesis Implementation of the Speech-Scenarios Web Application for the Russian Provider Company			
Date	31 June 2018	Pages/Appendices	34
Supervisor(s) Mr Mikko Pääkkönen, Lecturer and Mr Arto Toppinen, Principal Lecturer			
Client Organisation /Partners The Citylink Provider Company, Mr Igor Pellinen and Mr Alexander Shulgin, Mentors			

Abstract

The rising of Internet companies came to the widespread usage of web pages for searching an information. At the same time, many technologies and tools were developed and introduced to help programmers make their web applications much faster. However, high-load projects in IT companies are always one step back from the technological evolution. Programmers need to spend more time to uphold those projects and integrate new functionalities. Fortunately, administration of companies understands the situation and gives time to include modern technologies in new subprojects and even the possibility to migrate the whole project to the newest standards. new subprojects and even possibility to migrate the whole project to the newest standards.

The purpose of the thesis was to research and present the main aspects of modern web development technologies, including chosen popular frameworks and basic tools and creating a solution to reduce time consumption for the Citylink provider company in making speech scenarios.

The thesis includes basic concepts of a web application, a research of modern front-end frameworks and build tools, a description of chosen software libraries and an implementation of the Speech-Scenarios web application for the Russian Internet provider company – the Citylink. During the development, the required skills and techniques were obtained and implemented. These included a Vue Javascript framework, routing and FLUX-system, which were unfamiliar to the author.

As a result, a project was created to relieve programmers from additional work and let them concentrate on tasks that are more important. on. The thesis and the project development process are presented as a research and further implementation of Speech-scenarios web application.

Keywords
The Citylink company, Laravel, Vue, Vuex, Eloquent, ORM, Web development, Speech scenarios, SPA, Module system, Frameworks

CONTENTS

1 INTRODUCTION

Web development is a complex structure of different processes. Almost 30 years later after the invention of Hyper Text Markup Language, it has reached an incredible size – starting from a one page on URL info.cern.ch including only text, with further creation of first browsers (Mosaic and NetScape) and becoming a bunch of websites, mobile apps, web services based on a huge amount of progressive web technologies. There are about 2 billion websites in the world currently. Daily usage of social media portals, blogs and Google search engine is a part of human life. It will not be possible without an evolution of programming tools and techniques. Moreover, the web development organization process has become as a set of rules of success plan how to implement perfect solutions. Modern web application developer should be familiar with these processes and technologies.

There are many sources where it is possible to find a lot of information about modern web-technologies. Many of them include the current situation what to learn and what to choose. However, it is difficult to choose which one to use and why if you are starting to work on a new project within the company to solve a task, if no one has worked with them before. Necessity of research created a motivation to present these technologies and mention their pros and cons. At the same time, the implementation of a new web application needs to be done and documented.

The Citylink company is a huge Internet and TV provider in the North-West region of Russian Federation (Karelia Republic). The thesis aim is to create the application with the ability to create, update and delete speech-scenarios for several situations, which happen within customers' and engineers' dialogs.

A solution to integrate a new functionality for the speech-scenarios creation process requires to overview instruments to pick up and use in an implementation of a new application. As a result, the research of the modern technologies and development tools was done.

2 WEB TECHNOLOGIES RESEARCH

Evolution of web development technologies came up with an amazing amount of different solutions. Therefore, programmers need to choose which of them will handle companies' requirements. However, sometimes it is a tricky question. Programmers need to choose scalability support, powerful hardware and which of technologies to use to make process of development comfortable, maintainable, supportable and well optimized.

To answer those questions, the reader needs to know what is a web application in general. It is essential to understand basic concepts.

2.1 Basics

A web application is a computer program, which is stored on a server side (back-end), accessed through a web browser as its client (front-end) and performs specific tasks. In the modern world there are two types of platform where is possible to use web applications – static like desktops or infrastructure devices and mobile like smartphones, tablets and e-watches. (Moreau 2018-02-22)

Front-end is a client side of a user interface, where the user is able to overview content and perform some activities. As an example, any of websites in browsers (example - google.com or facebook.com) present client part of a web application. The Front-end includes three main components: HTML templates, CSS styles and JavaScript (dynamic programming language).

1) HTML – Hyper Text Mark-up Language – describes a structure of web pages, presented as elements with tags (<html></html>), which are used by browser to show a content.

2) CSS – Cascading Style Sheets – provides and controls layout of a web page, tells HTML elements how to be displayed in a browser.

3) JavaScript – is a dynamic programming language, which provides a behaviour of a web page – handle user interactions, send/catch an information to/from web server. In modern web applications the AJAX technology is performed. It allows calling server actions asynchronically and presenting answers without a page re-rendering.

Back-end is a server side of a program, which handle requests from client, perform business logic and interaction with a database and send response back to the client. Usually, includes one server programming language such as Java, C#, PHP, Python, Ruby or even JavaScript, possible to have a mix of them. To store important data for an application a database is applied. There are two types of databases – SQL (examples - MySQL, PostgreSQL) and non-SQL (example - MongoDB). The difference between them first use Structured Query Language to represent and store data in relative tables when the second use a creation of database implementation as a collection of JSON documents. (Homan 2014-05-14.)

Front-end and back-end are interacted through http/https protocols. Typical representations of layers and their interactions are shown below (Figure 1).

HTTP – Hypertext Transfer Protocol - is an application-layer protocol to execute transmitting hypermedia documents between client and server. This protocol is implemented in the web architecture in June, 1999. (MDN team member, 2016)

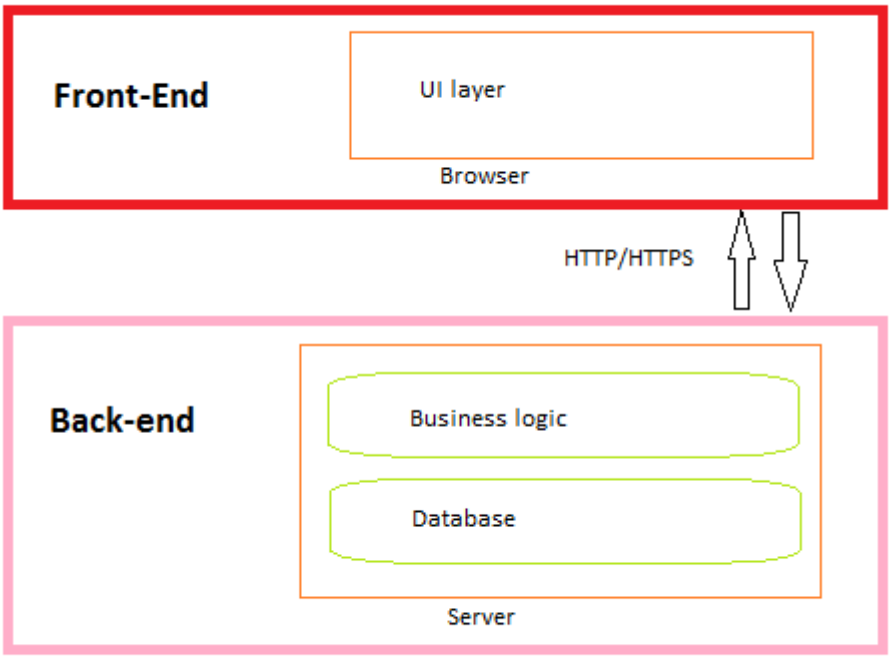

FIGURE 1. The presentation of the Front-end and Back-end layers and interaction between them.

In a modern world, to create a web application there are solutions to make programming comfortable and faster – libraries. These are solutions created by other programmers or companies to provide efficient instruments to decide routine tasks, which are used quite often. For example, programmer needs to implement hardcore math equations to make calculations. Of course, it is always possible to create functionality by itself and spend a lot of time for that. However, a better choice would be to use reusable components, which are libraries. Therefore, programmer will spend time to more non-trivial tasks rather to reinvent the wheel.

Sometimes a group of enthusiasts decides to evolve a library and create from it something bigger, which will be a trend for a web development. Frameworks are those libraries. By definition, these are application libraries, which provide a pre-defined code structure and reduce amount of trivial tasks appear in a web-based application development process. The most valuable advantages of

frameworks usage are quality of the application using reusable code and design, good code maintenance by centralizing the general code in a single location, faster development environment expansion. (Technopedia, 2013)

2.2 Front-end frameworks

As mentioned above, front-end is a client part of the application and this part consists of three elements. Front-end frameworks are applied for all of those elements – a module system and patterns for the templates, a design and JavaScript code. However, the most significant role attends to a code. Those frameworks are specialized to create a dynamic web application – SPA (Single Page App) or PWA (Progressive Web App). Both of types provide possibility to display a new data by calling API (Application Programming Interface or just a web server) without page re-render, but PWA applications allow using them without Internet connection or just for an offline usage. The reader needs to become familiar with popular front-end frameworks and their pros and cons.

FIGURE 2. The emblem of AngularJS framework

AngularJS – is an open-source JavaScript framework made by Google (Figure 2). It helps programmers develop dynamic web applications. The framework is built on the famous architecture concept of MVC (Model-View-Controller). This is a design pattern used in all modern web applications. This pattern is based on dividing the business logic layer, the data layer, and the presentation layer into separate parts. It is done to provide a management more easily (Figure 3). (Guru 99, 2012)

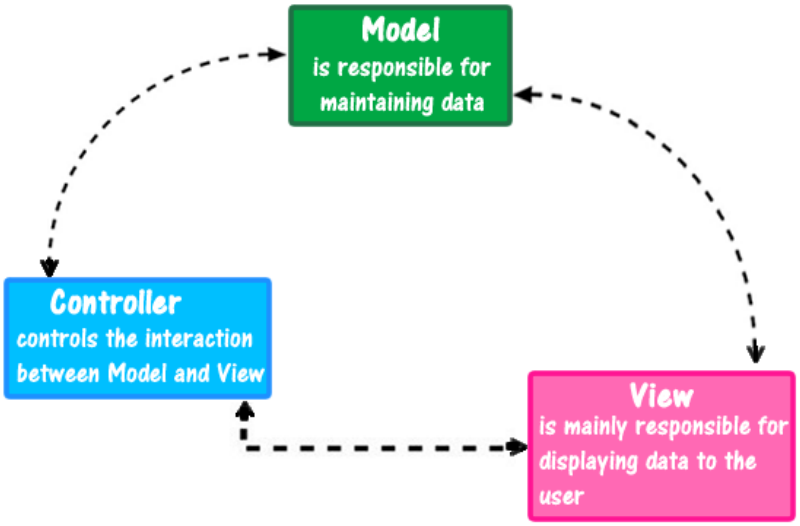

FIGURE 3. The presentation of Model-View-Controller pattern.

Pros:

- Dependency injection. This is an operation of supplement a required resource of code in an another code part.
- Two-way data binding – synchronization of Model and View layers. When something changes on the model, it also affects the view the same as vice versa.
- Suitable for medium or large projects
- Support of testing – Angular allows programmers to implement unit-tests and integrational testing.
- Current version is Angular5, which is introduced with newest features like faster compilation and HttpClient launch.

Cons:

- Time of studying – programmer needs to spend more time to understand how to work with AngularJS because of a difficulty rather than with other frameworks.
- Limitation in watchers (only 2000)
- Typescript – a statical JavaScript language. It allows using classes and static types for variables (number, string) and methods. It is a good to learn that tool, but to use the last version of the Angular it requires to know Typescript too.

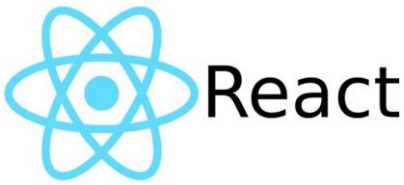

FIGURE 4. The React`s emblem.

ReactJS – is an open source JavaScript framework introduced by the team from Facebook (Figure 4). Released after the Angular framework it became popular and many companies require hiring engineers with React experience. In contrast to the Angular React is mostly capable for View of MVC pattern means to create templates and user interface logic when the Model and Controller layers are placed on a server. Nevertheless, Angular is allowed to use the same approach. (Altexsoft, 2018)

Pros:

- Virtual DOM (Document object model). DOM is a logical structure of templates in HTML, XHTML or XML formats. To update the whole DOM structure, it needs to consume more time, which leads to a long response of a web application, the Virtual DOM allows updating and rendering only specific part – time spends less and the web application becomes reactive.
- Reusable components. As an example, when requires to use some widget-service in different parts of the application, which is written on ReactJS, programmers could implement it as a module.

- Suitable for all project sizes.
- Learning curve. Comparing with the Angular, programmers need to spend less time to get knowledge of ReactJS basics.

Cons:

- One-way data binding. ReactJS provides only update templates via components and sometimes it requires implementing handlers for event calls. However, this type of data binding allows to write a stable code.
- JSX – a template based on mix of JavaScript and HTML. Most of the studying time of the ReactJS framework is consumed to understand what is JSX.
- View-orientation. No Model and Controller layers implemented from the box.

FIGURE 5. The VueJS`s emblem.

VueJS – is an open-source JavaScript framework made by Evan You, who worked on Google with an AngularJS in an amount of projects. He made a decision to get best parts of Angular and create a lightweight analog. The framework is used as an MV* pattern of the MVC architecture, which means it provides Model-View layer for the client-side part of the application. The VueJS is chosen by author of the thesis as a framework to implement Speech-Scenarios web application for the Citylink company and described more deeply in the Chosen Software Overview section. The framework was released in 2014. It is young, but already popular.

Pros:

- Small size of the framework – comparing with Angular and ReacJS frameworks, Vue is the smallest one, which leads to a less size of an application
- Less learning curve – programmers spend less time to understand VueJS basics. The best choice for beginners to learn frameworks.
- Two-way data-binding
- Virtual DOM
- Transitions – handle timing of adding/removing classes to components/elements

Cons:

- Language barrier – a majority of the users are non-English speakers
- Size of community – comparing to other frameworks, usually more difficult to find solution for different issues (9 Series, 2018)

2.3 Build tools

Build tools for applications provide an opportunity to compile all parts of the project all together and to put in a specific folder and to do script tasks. There are not so much these tools and they are divided in two types – install package managers and build managers. In this section is described the build managers, where the install package managers are specified in details in the "Chosen Software Overview" section for the Speech-Scenarios web application.

Most popular build tools are Webpack, Grunt and Gulp. The reader needs to know them and their pros and cons.

FIGURE 6. The Webpack`s emblem.

Webpack – is an open-source build tool static bundler of modules of a web application. It provides a dependency graph during the processing - a bundle asset or a build, which maps to every module of the application. Till the version 4.0.0, Webpack did require to configure a file to specify the set of instructions of what and how should be bundled. After that, no need in the configuration file – everything is done automatically with a possibility to set instructions. The tool is chosen as a build instrument for the Speech-Scenarios web application. (9 Series, 2017)

Pros:

- Webpack-dev-server – hot and live reloading support
- Amount of features
- Minification
- Dependency graph

Cons:

- Difficult in configuration (for previous versions)
- Tricky production builds versioning

FIGURE 7. The Gulp`s emblem.

Gulp – is an open-source task-manager, which provides an automatic solution for trivial objectives such as a minification, a compilation and a deployment for the front-end part of the application (Figure 12). All instructions are set into the file called "gulfile.js" in the root of the project. Comparing to Webpack, Gulp provides a more flexible approach in configuration of mentioned operations.

Pros:

- Speed of compiling
- Readable
- Amount of plugins
- Performance
- More control over the flow

Cons:

- Not a comprehensive documentation
- Debugging a tool
- Dependency to external plugins

FIGURE 8. The Grunt`s emblem.

Grunt – is an open-source task manager, which automates routine objectives. Gulp and Grunt look the same, but there two main difference between them (Figure 8). Tuning in Grunt is based on a configuration, while Gulp is based on a stream. The second distinction is in how they run a set of instructions – the first runs in series, while the second one runs in parallel.

Pros:

- Configuration is easy
- Learn is easy
- Great community
- Amount of plugins

Cons:

- Become unreadable and complicated if configuration grows
- Less popular – trend to outdating
- Performance

Comparing all of the mentioned build tools nowadays, the most popular are Webpack and Gulp. The community of programmers' advices to choose Gulp, because of a performance and a control over flow. However, for the Speech-Scenarios web application were chosen the Webpack – the speed of a project expanse is much faster and no more code is required.

3 OVERVIEW OF THE CHOSEN SOFTWARE

This section describes chosen instruments for an implementation of the project and their basics. That is done to prepare a reader to the understanding of how the Speech-Scenarios web application is made.

3.1 NodeJS/NPM

NodeJS – is an open-source JavaScript run-time environment, which allows to use a JavaScript language not only for web pages in a browser, but in a desktop and mobile applications. JavaScript were previously used only for the client-side development. With introduction of NodeJS it is possible to use it also for the server-side scripting like PHP language, which leads to the creation of the application using one language for both front-end and back-end side. (NodeJS 2017)

For the thesis purposes, it is required to install the NodeJS environment just for one important thing – a package manager. The npm package manager allows to download extended JavaScript libraries for the project, manage them with a configuration file and run code-written scripts with a mix of the chosen build system. There are few basic commands, which are used in the Speech-Scenarios web application and ran through the CLI (Command Line Interface). The reader needs to look on them:

1)
```
> npm init
```
The below command will initiate a questionary in the CLI, where the script ask a programmer for specified fields to create a configuration "package.json" file. In this file are mentioned in the JSON format program`s author name, license (MIT is usual), some other author`s information such as an e-mail, an URL address, a version of the program, and a list of specified scripts and included packages. The command is used only once in the beginning of the web application development.

2)
```
npm install [<@scope>/]<name>@<version>
```
The "npm install" command does install packages. To specify which package to install and which version, it is simple to add the package name after the "install" command and a version after the "at" sign. As an example, to install a Vue package with a 2.3.4 version the command is – "npm install vue@2.3.4".

3)
```
npm run-script <command> [--silent] [-- <args>...]

alias: npm run
```
The "npm run" command with a specified name of the script will run this script, which is coded in a "package.json" file. For example, in "script section" is added a script called "test: some test code", a programmer executes this script with a command "npm run test" and that is it. No need in writing code of the script in the CLI – it is already written in the "package.json" file. (NPM 2015)

3.2 VueJS

VueJS is briefly described in the ”Web Application Research” section. However, it is a chosen software for the development process and this framework is used for all of created web pages for the client-side. In this section are specified VueJS features and general concepts.

Every Vue application starts with an instance. It could be done as a call of the class construction:

```
const vuem = new Vue({
    // options
})
```

FIGURE 9. The VueJS`s instance.

As a reader can see, the instance of Vue object is called and assigned to the constant variable (Figure 9). It is done to be sure the instance will not be changed during the development process and methods/function calls. With the mentioned instance call is created a root instance of the entire client-side part of the web application. It also will be a mounted point for the root template of the web page.

Options in the root instance usually present the DOM options for the Vue object. There are usually three of them:

1) *el* – an id of an HTML element to mount on. This element will be assigned as an entry point for the whole Vue application. Example – ”el: #entry”
2) *template* – a markup for the root Vue template of the component. Example – ”template: <App/>”
3) *components* – a characterized template with a logic. Usually it is specified with the same name as the root template, because the root Vue components includes both. Example – ”components: {App}” (VueJS 2018)

Additionally, there is possibility to include two more options for the root instance of the Vue application – a router and a store. The first one – Vue-router is capable for the route configuration of different Vue components (or Web pages) and provides comfortable in use module system for the navigation through an application. The second one – Vuex - allows to store a retrieved information on client-side part, call a server for the specific data, make a mutations of the data state during the process of web application. Both of mentioned libraries are described below in this section.

As the entry point of Vue application is discussed, the reader needs to take a look on a Vue basic component. By the official documentation, the basic pattern for the Vue component file is to use an HTML template, JavaScript and CSS in the one file.

```
<template>
    <!--Your Vue component`s template -->
</template>

<script>
    /* Your Javascript code*/
</script>

<style>
    /* Your CSS style for the Vue component */
</style>
```

FIGURE 10. The Vue component`s pattern.

In a template, a programmer puts an HTML code and in a style – the CSS style, which is simple. However, the most difficult part is the script section – a programmer needs to specify a Vue web page component instance and export it to use in the root Vue instance (Figure 10). The code below shows how it is done (Figure 11):

```
<script>
    import SomeComponent from '../../components/SomeComponent'
    import AnotherComponent from '../../components/AnotherComponent'

    export default {
        name: 'name',
        components: {
            SomeComponent,
            AnotherComponent
        },
        data () {
            return {
                some: null,
                another: 'String',
                someValue: 15,
                someObj: {
                    property: ''
                },
                someArray: [
                    1, 2, 3
                ]
            }
        },
        created () {
            this.someMethod()
        },
        mounted () {
            this.someMethod()
        },
        methods: {
            someMethod () {
                console.log('haha')
            }
        },
        computed: {
            someComputed () {
                return some;
            }
        },

    }
</script>
```

FIGURE 11. The example of the script of Vue component.

There are new options appeared for the Vue instance such as a name, a data function, created and mounted hooks, methods and computed properties. The “Name” specifies the name of the Vue component. The data function returns a set of different variables, which are used in the code. To use some variable from the set in hooks it is required to start an invocation of a variable with “this”

word, because the set of data is in the scope of the Vue component. Example – 'this.someVariable'. The same principle is mandatory for the methods invocation in hooks and computed functions. In the Speech-Scenarios project are used two types of hooks – created and mounted. In the created hook are made some code before the render of the web page. In the mounted hook, an API is called to the server and retrieved data is set. However, in the official documentation is specified call to the API of the server needs to be done from the created hook, but because of features of the project, it is done in a such manner. If the reader is interested in the Vue JS framework, the author suggested to open the VueJS official documentation and through look the guide. (VueJS 2018)

From the code, a reader could understand the dependency injection is used to import two other Vue components and passed them into the "components" object – one of the Vue instance option – and now it is possible to implement them into the template section like this (Figure 12):

```
<template>
    <some-component></some-component>
    <another-component></another-component>
</template>
```

FIGURE 12. Embedded components in a template.

As a reader could mention, the components are injected in a template with the kebab-case style, when they are imported in the Camel Case style, which is a feature of the VueJS framework.

3.3 Vue-router

There is possibility to add a router library in the root entrance of a Vue application. Therefore, the best-case scenario would be to add an official Vue-router library for the navigation between components. It also includes features as a dynamic route matching with params, nested routes, named routes and many other possibilities and approaches to make a navigation system strong. Moreover, the Vue-router provides a modular router configuration to make a project structure intuitively understandable for other programmers. Below the reader can see the example of the router instance (Figure 13):

```
import Vue from 'vue'
import Router from 'vue-router'
import Index from '../pages/Index'

Vue.use(Router)

export default new Router({
  routes: [
      {
          path: '/',
          name: 'main.page',
          component: Index
      },
  ]
})
```

FIGURE 13. The Vue-router instance with a one defined route.

To create a router instance, it is necessary to import the Vue library with the Vue-router, then allow Vue to use this router. After that, to define routes, a JSON component is inserted into *routes* key of the router. A simple named route is defined with a url path for the route, name and component reference for the page. In the example, the "main.page" route is defined with a path "/" and a component reference. When the user goes to the "https://www.website.com/", he/she will get the Index page of this route.

3.4 VueX state machine

Vuex – is the state machine for the VueJS framework. It is defined as a Flux architecture for the application. Flux came from the Facebook team and became popular in ReactJS. After that, the VueJS framework`s team developed the same technology for their own "creature". There are three main entities of the Vuex store – actions, mutations and a state.

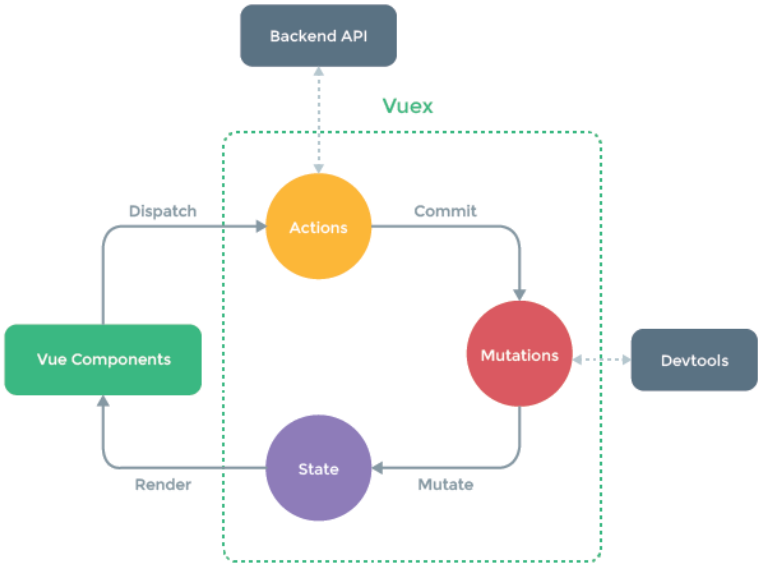

FIGURE 14. The Vuex lifecycle representation.

First, the user makes an action – calls the API of the server and fetch a data. Then the actions are committed to call a mutation, which then saves/mutates the data into the state object. Finally, the state object is observed by Vue components, and if the state is changed, the component will rerender DOM elements of the page (Figure 14).

```
import axios from 'axios'
import {
  SET_TYPES_LIST,
} from './mutation-types'

const actions = {
  fetchTypes ({commit}) {
    return axios.get(route('sa.widget.type.index'))
      .then((response) => {
        commit(SET_TYPES_LIST, response.data)
      })
      .catch(function (error) {
        console.log(error)
      })
  },
};

export default actions
```

FIGURE 15. The Vuex action example.

Actions in the project are used for GET, POST, PUT, DELETE and PATCH queries to the server. The "axios" library is used for actions to simplify queries instance. As a reader can see in the example above, in the "actions.js" file of the "settingsModule" of a Vuex store for the application, the axios and mutations-types are imported (Figure 15). Action with the axios query will return a JavaScript promise, which is an asynchronical operation that could be chained in the promise chain. In the query, if the data is returned from the server, this data will be committed to a mutation to modify the state object.

```
import {
  SET_TYPES_LIST,
} from './mutation-types'

const mutations = {
  [SET_TYPES_LIST] (state, types) {
    state.types = types
  },
}

export default mutations
```

FIGURE 16. The Vuex mutation example.

Mutation`s principle is simple – it is defined as a function with one and more attributes passed in. The state should be always defined as one of the attributes of the mutation (Figure 16).

```
const getters = {
  currentType (state, getters) {
    if (state.current_type_id)
      return state.types.find(type => type.id === state.current_type_id)
    else
      return {}
  },
}

export default getters
```

FIGURE 17. The Vuex getter example.

There one more object of the Vuex store – a getter. The getter returns a data from the state and allows to modify this data before actually return (Figure 17).

To combine actions, mutations, getters and a state into one module the import in one file will do a thing. Moreover, it is better to provide a namespace for the module to simplify the code of the pages and components (Figure 18).

```
import state from './state'
import mutations from './mutations'
import getters from './getters'
import actions from './actions'

const moduleWidget = {
  namespaced: true,
  state,
  mutations,
  getters,
  actions,
}

export default moduleWidget
```

FIGURE 18. The Vuex module instance with allowed namespace.

For the Speech-Scenarios web application there will be two modules. These modules are combined in the Vuex store instance (Figure 19).

```
import Vue from 'vue'
import Vuex from 'vuex'
import settingsModule from './settings'
import moduleWidget from './widget'

Vue.use(Vuex)

const store = new Vuex.Store({
  modules: {
    settings: settingsModule,
    widget: moduleWidget,
  }
})

export default store

```

FIGURE 19. The Vuex store instance for the project.

In the project of the thesis work the Vuex actions, mutations, getters and the state are called via helpers with a namespace module definition by mapping to the store module. This is a better approach rather than using a "dispatch" method of the store. Actions and mutations are always defined into methods of the Vue components. Getters and the state are defined into computed property.

4 SPEECH-SCENARIOUS WEB APPLICATION

In this chapter is described what needs to be done for the web application and how it should look. Moreover, there are installation environment and configuration instructions, project structure for better comfortability and usability from a developer perspective and finally – how the application was implemented and possible ways to improve the Speech-Scenarios web application.

4.1 Plan and Design layout

The reader needs to be familiar on what needs to be in the project. The Speech-Scenarios web application allows customers to create, update, delete and review (go step-by-step) different scenarios. All of these processes involve CRUD (Create, Read, Update and Delete) operations with a server and a database. For those purposes will be used the RESTful API – Representational State Transfer application program interface, which is a set of rules to manage CRUD operations via the HTTP and HTTPS protocols between client and server sides (Zell 2018). The thesis includes only the front-end (client) part of the application. Other programmers have done the server part and RESTful services of the server are consumed by the client application.

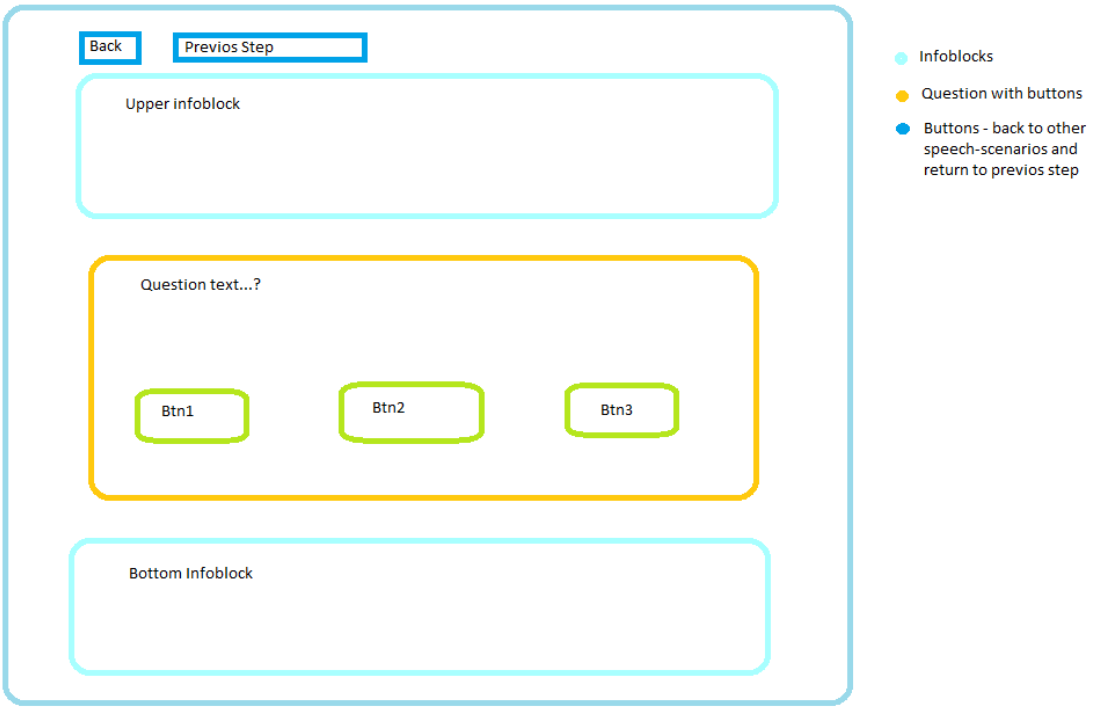

FIGURE 20. The presentation of a single step of the Speech-Scenario.

Figure 20 presents a single step of a speech-scenario – what customer will see at the first use. There are few components per each step – a panel with question text and buttons, which lead to other steps, an info-block (upper and bottom) and two optional buttons on the top of the step page – one returns to a list of other speech-scenarios, another – to the previous step, if the current step is not the first one.

However, to understand the whole application design, the reader needs to take a look on main pages, which will be implemented:

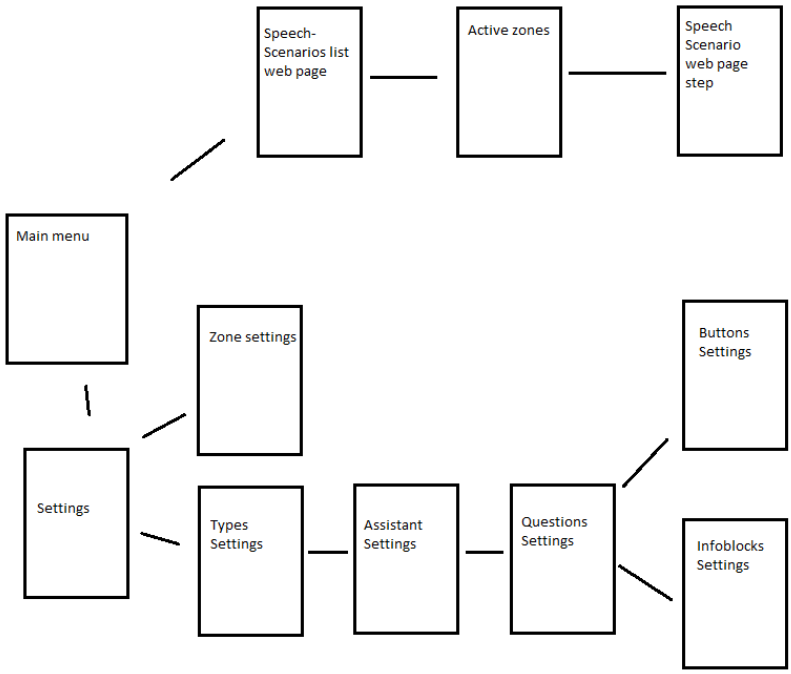

FIGURE 21. The general design of the Speech Scenarios web application.

The application includes next pages:

1) Main menu (Index file) – The first page, which leads to other two pages – one is the list of speech-scenarios and another – to the settings.
2) Speech-Scenarios list – a table with an amount of different speech-scenarios and a search filter. In the settings these scenarios are called as "Types".
3) Active zones – a page with the table of active zones (a zone is a town/region). If there is only one active zone, then after click on the speech-scenario from the list page the customer will be redirected straight forward to a Speech-Scenario first step of an active zone. One zone is an "Assistant' – speech-scenario for the specific zone.
4) Settings – a simple page with two links – one leads to zones settings and another one - to types settings.
5) Speech-Scenario step page
6) Zone settings – a page to create, update and delete zones.
7) Types settings – a page to create, update and delete types. Each type includes a name of the speech-scenario and different scenarios with different zones.
8) Assistants settings – a page to create, update, duplicate, activate/deactivate and delete assistants with different zones (Speech-Scenarios)
9) Questions settings – a page to create, update and delete questions. Think of questions as steps of a speech-scenario. Includes links to buttons and info-blocks settings pages.

10) Buttons settings – a page to create, update and delete buttons. Each button allows to point to a next step (question).
11) Info-blocks settings – a page to create, update and delete different blocks with a useful information. Could be placed on the top or bottom of the step page.

Consider the information above, there will be approximately 11 different pages and few components for the Speech-Scenario step web page – question (with buttons) and info-block. Moreover, the application requires to include a loading component, which will be shown, while a data is retrieving from the server and a page is rendering. For this purpose, will be used a spinner component – it will be included in every page of the application.

4.2 Installation environment

Any developer requires his/her own environment to actually develop applications. For the thesis the first software needs to be installed is the PhpStorm IDE (Integrated Development Environment). It will be used to write, inspect and debug code of the application. Moreover, it will help to create a project structure, to include front and back-end frameworks and different libraries and setup a configuration to build application for development and production (final) versions. The PhpStorm is chosen by the team of developers, because our back-end server is based on the Laravel PHP framework. Installation itself is simple.

Next step is to install Laravel project via the Composer manager – the same as NPM manager for the JavaScript environment - in the PhpStorm and that is done by the team member of developers in the Citylink company. For that required software is already installed for the web-server. It is possible to use a virtual machine system and develop applications everywhere. However, for the company purposes it requires to have a static hardware with a pre-installed software for the work with the server in the building. The installation of the Laravel framework is out of the thesis work.

After the PhpStorm and Laravel are installed, it is required to install the VueJS framework, the Vuex state machine, the Babel package to transpile the JavaScript ES6 standard to the ES5 during the compilation, the Bootstrap-vue for better design view, the vue-router and axios packages for the routing, the vue-simplemde for the info block component configuration (insert images, links and text edit tool). Moreover, the installation of laravel-mix and cross-env packages needs to be done to configure a build for development and production versions of the main JavaScript file.

To install mentioned packages, the NodeJS is required. It is downloaded from the official web-site and installed with a simple set of instructions. After that, it is possible to install packages via the NPM manager.

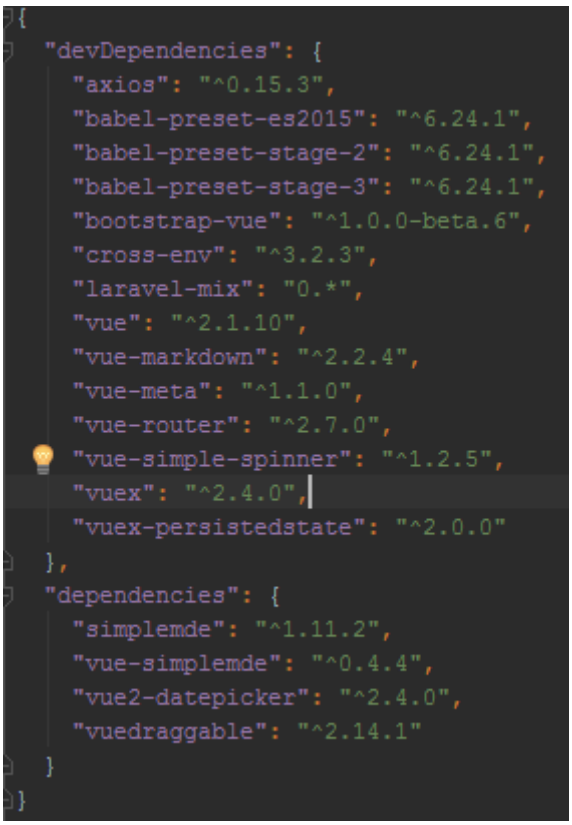

FIGURE 22. The "package.json" file with installed dependencies.

After everything is installed for the development purposes, it is necessary to configure build tools and write some scripts into the "package.json" file to make a building process much faster (Figure 22). Moreover, it requires to configure paths for build tools to make them understand where to collect a code for the compilation and where to put the final version.

4.3 Configuration and Project Structure

Build tools are the WebPack and Laravel-mix libraries. Usually, the front-end team uses the first one and then the build is deployed to the back-end part. However, because of the Laravel framework, it is necessary to configure build tools on the server-side. The Webpack library is a part of the Laravel-mix, it needs to install only the second one, which was done. There is the file called *webpack.mix.js* and here the entry path for the project and output path are configured (Figure 23).

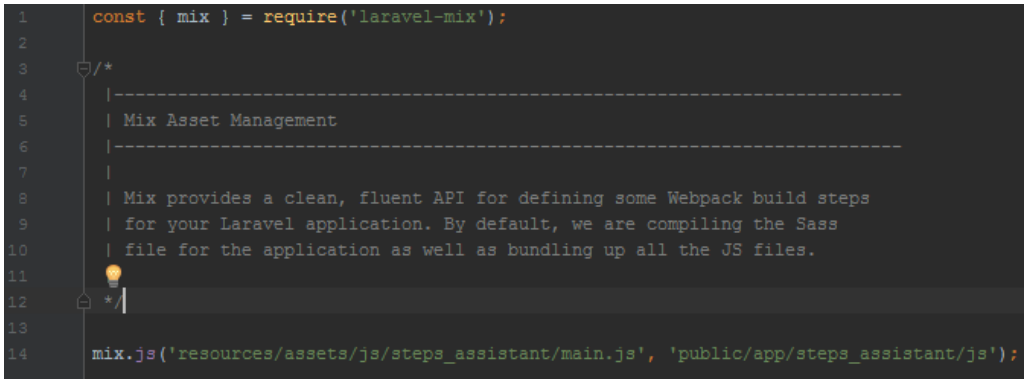

FIGURE 23. The Webpack configuration file for Laravel-mix.

In the file a mix object is defined as the laravel-mix instance and then used the “js” method to compile VueJS files from a one directory to another. A versioning of builds is removed from the initial configuration. As specified in the ”webpack.mix.js” configuration file, the entry point to compile the project is ”resources/assets/js/steps_assistamt/main.js” and the output folder is ”public/app/steps_assistant/js”. The reader can see below the Laravel project structure and especially on the front-end part of project:

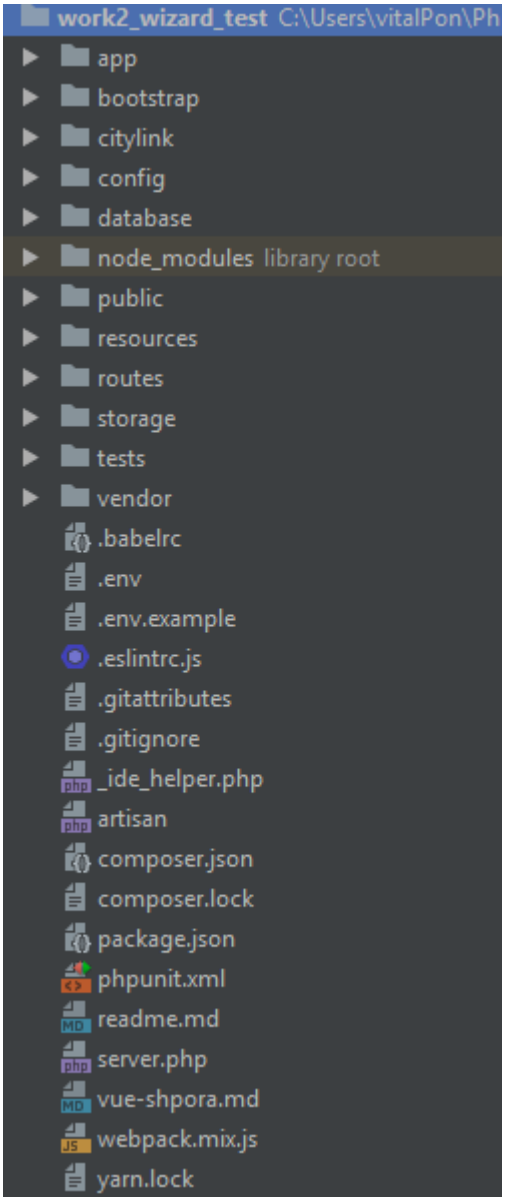

FIGURE 24. The project structure of the Speech-Scenarios web-application – Laravel default.

The implementation of the front-part is started from the ”main.js” file in resources directory. This directory is used to hold Laravel general templates – Blade, which looks like HTML templates, but with another syntax. There are also CSS and JS project files.

After the creation of the configuration file, it is necessary to add few scripts into the ”package.json” file to run development and production build compilations, which will be run via the CLI with NPM using the ”run” command:

1) ”development” – when running ”npm run development”, the development build will be compiled
2) ”watch” - when running ”npm run watch”, the development build will be compiled and the build tool will listen for next changes in the code.
3) ”production” – when running ”npm run production”, the production build will be compiled

All of those scripts are available to through look on the GitHub repository.

In ”Plan and Design layout” amount of pages and components were evaluated. It is possible to initialize VueJS files and divide them between by the abstracts. There are two main abstract entities for the project – speech-scenarios widget and settings. Widget allows leading through the bunch of questions. Widget exists of next entities or pages:

- *Type(s)* – hold the name of a scenario and connection to ”assistants” with questions and buttons. The page for the widget shows a table with different types of scenarios.
- *ActiveAssistant* – if for chosen type exists more than one assistant (zone), and then user will be redirected to this page. The page shows a table with different assistants or zones.
- *Assistant* – the main page for the widget – a user will start his/her walkthrough the scenario. Exists of question and infoblock components, buttons, which returns back to the list of types (Types page) and back to the previous question of the speech-scenario.

However, to create type of scenario, its assistants and questions for them with buttons and info blocks, the settings entity is required. The settings exist of next pages:

- Zones – the page allows to create, update and delete zones entities to use them while the creation of assistant. Zones abstractly are towns and each assistant can be for one zone at a time. It is possible to create a lot of assistants with the same zone, but only one of them could be activated.
- Types – the page allows to create, update and delete types entities. After type is created, it is possible to change its names and lead to the Assistants page.
- Assistants – the page allows to create, update, delete, modificate and duplicate assistants with an attached zone. Leads to Questions page.
- Questions – the page allows to create, update and delete questions for an assistant. Leads to Buttons and Infoblocks pages.
- Buttons – allows to create, update and delete buttons with the transition to the next question for the existing question.
- Infoblocks – allows to create info-blocks with some important data for the question (step).

There is also the main page with the links to the Widget and Settings pages, Settings page with the links to Zones and Types setting pages and “Not found” page if user will try to type an address with nonexistent routes.

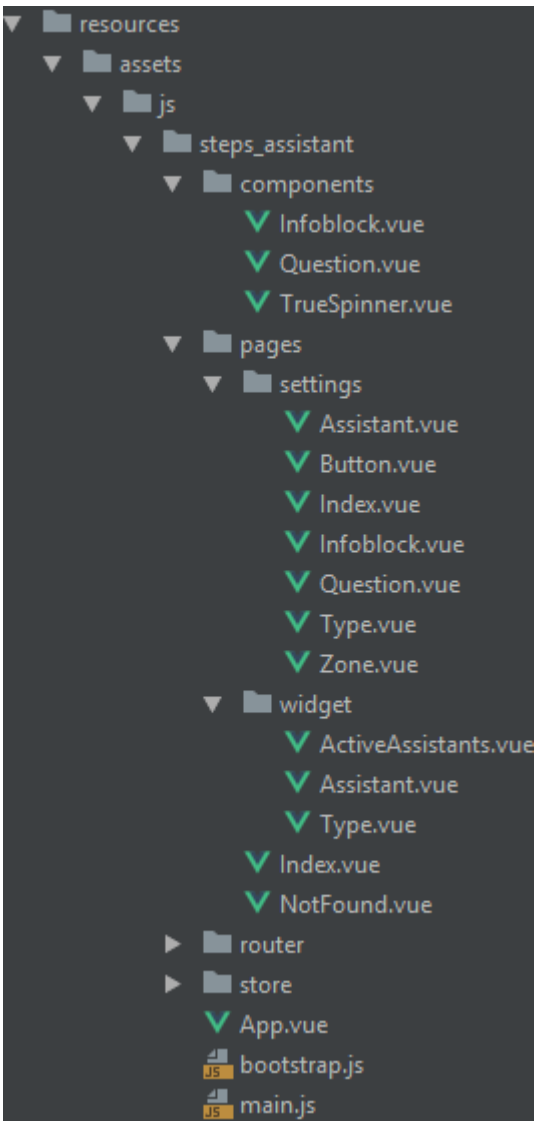

FIGURE 25. The project structure of the front-end part divided by the abstract entities.

In the project structure additional entities are added – router and store. The router directory is capable to hold files for the named routes of widget and settings abstracts. The routing processing is done with Vue-router package.

The store directory includes two modules – one for the widget and one for the settings. Each module includes six files – actions, getters, mutation-types, mutations, state and index file.

There are also the entry page for the whole project - App.vue and the entry file for the application – main.js with the requirment of bootstrap.js file, which is capable for the CSRF protection token assignment. The Cross Site Request Forgery is a type of the attack on users of websited, used to find weak point in HTTP protocol (Shapland 2011). The server will not allow to open the pages without this configuration.

4.4 Implementation

The code is available via this link: https://github.com/ponichWKing/Speech_Scenarios_App
It is impossible to use the project application, because of non-installed server support.

First of all, the entry points for the project were implemented. The ”main.js” file includes import of Vue and Bootstrap-Vue packages, router and store directories with Vue-router and Vuex packages and ”bootsrap.js” file with CSRF token configuration with ”axios” library for the routing. Of cause, the entry page ”App.vue” is also imported and specified as a template and main component of global Vue instance. This instance is provided with router, store and reference element to the Blade template of main HTML file.

FIGURE 26. The first page.

The first working page was the ”Index.vue” file. It includes router-links DOM elements to the widget/Types ans setting/Index pages (Figure 26). In the router the named route for the Index.vue is specified and attached with the component. After that, the ”NotFound” page was implemented and specified to be shown for all routes, which are not specified in the router configuration. The same configuration is done for other pages.

It was impossible to start implementing widget without settings pages creation – there is no meaning to create widget without created types, assistants, question and other entities. That is why, the Settings pages were coded first.

The ”Index.vue” of the settings entity includes just router-links to the ”Zone” and ”Types” pages, but because it is the subpage, it was necessary to implement the breadcrumbs with links on the previous pages and vue-bootstrap directive for the breadcrumbs was used. The breadcrumbs are implemented for each sub-route pages.

The ”Zone” page includes ability to create, update and delete zones. A table is specified to show users existed zones with the ability to update and delete specified zone by the row. Creation and modification of zones is done with the modals windows, which appears from clicks on the ”Create” button and row of the table. The actions – create, update, delete and retrieve existing zones is done with Vuex store and Axios library – the same as for other pages and components.

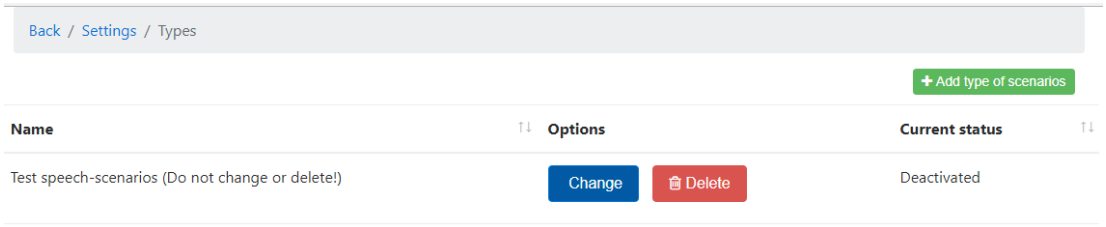

FIGURE 27. The Type page of settings.

The ”Type” page includes ability to create, update and delete types of the scenarios. As in the ”Zone” page interface, a table is implemented with the buttons ”edit” and ”delete” options. Also, the modals windows are appeared when the user clicks on the buttons ”Add” and ”Delete” or the row of existed type in the table. When the user clicks on the ”Change” button, his/her is redirected to the ”Assistant” page settings (Figure 27).

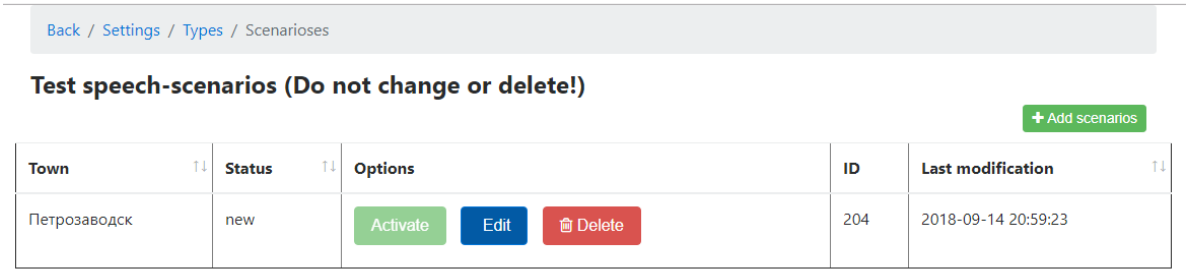

FIGURE 28. The Assistant page.

The ”Assistant” page includes ability to create, update and delete assistants of the scenarios from existing zones. The same functionality for modals and breadcrumbs is implemented. However, by the requirements it should be possible to duplicate an assistant. Therefore, an additional button is added to each row in a table of existing assistants. It is possible to duplicate an assistant only if its activated. In the activated mode it is impossible to edit assistant (Figure 28).

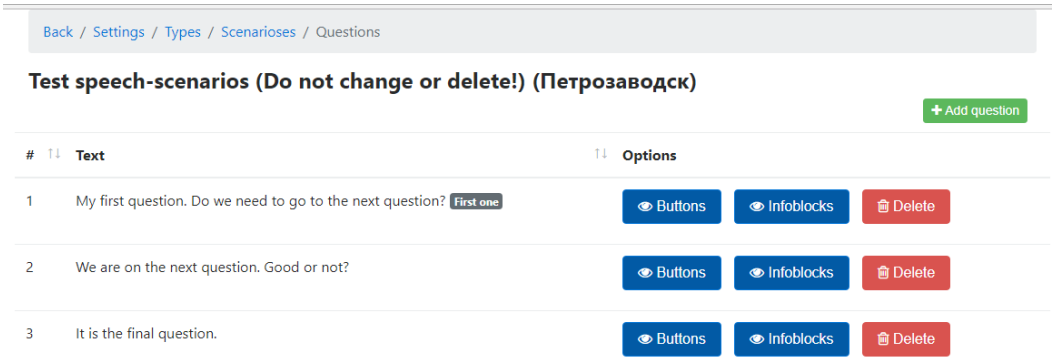

FIGURE 29. The Question page.

The “Question” page includes ability to create, update and delete questions for the existing assistants. Also, the questions are presented as a table, with buttons to delete question and create buttons and info-blocks for the question (Figure 29).

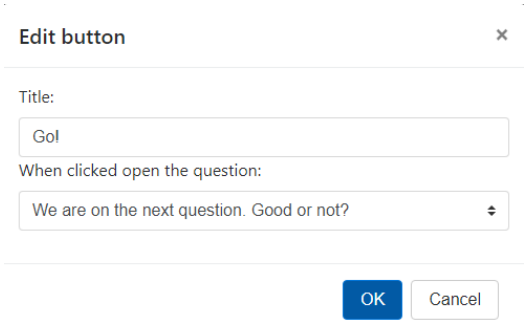

FIGURE 30. The modification modal window of the Button page.

The "Button" page includes ability to create, update and delete buttons for the existing questions. Besides the possibility to specify the name of the button, the main functionality to reference to the different question is implemented. In creation and modification modals a user can specify the next question for the button, so when he/she clicks on the button in widget, it will redirect him/her to the specified question or step (Figure 30).

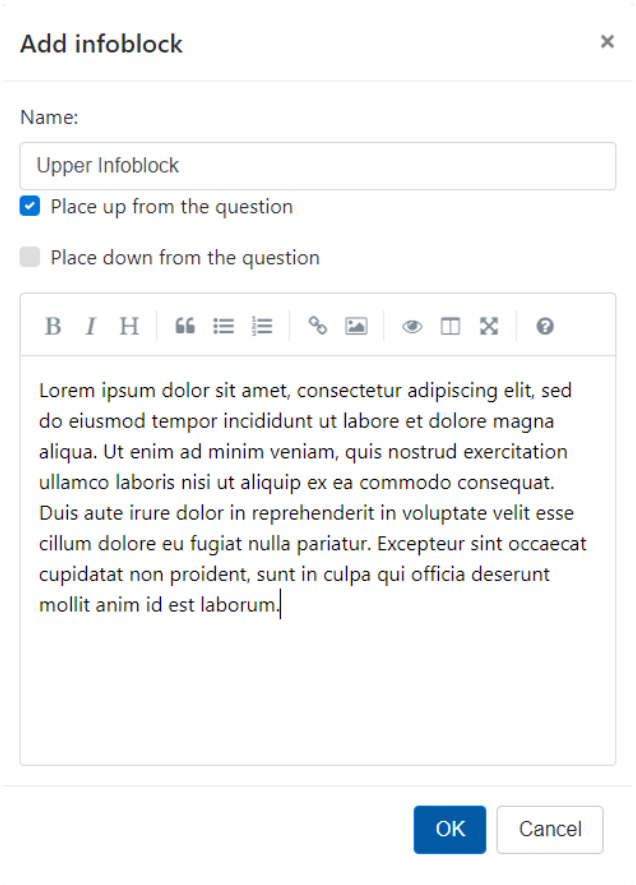

FIGURE 31. The Infoblock page, the creation of the entity.

The "Infoblock" page includes ability to create, update and delete info-blocks for the existing questions. In the page the "vue-simplemde" package is attached to provide simple text-editor for an important data with the ability to add links, images, lists and so on (Figure 31).

All of the CRUD operations (Create, Read, Update, Delete) is done with the Vuex state machine. All calls to the server is specified in "actions.js" file of the settings module of the Vuex store. Each call

is using the axios library with named routes. Its returns the promise – an asynchronical function, which returns data from the server and then uses "mutations" to set data in the state of the store component. Then, after the state is set, the Vue component renders retrieved information on the page via getters and computed methods for the state. For the Settings the Vuex module named as "settingsModule". At this moment, it is possible to create a simple scenario, but impossible to through look it step-by-step. That is why it requires to implement Widget pages.

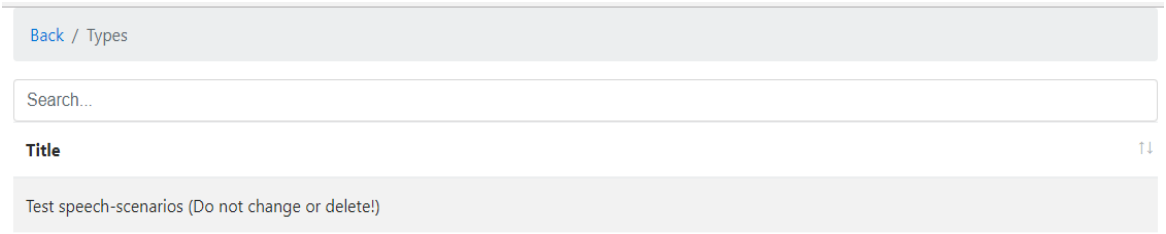

FIGURE 32. The Types page.

The first page of the Widget entity is "Types". The page includes a table with existed types of different speech-scenarios. Click on the row of the table leads to "Assistant" page, if there is only one activated assistant, and to "ActiveAssistants" page, if there are more than one. Moreover, on the top of the table a search input panel was implemented. It allows to search for a type by the name (Figure 32).

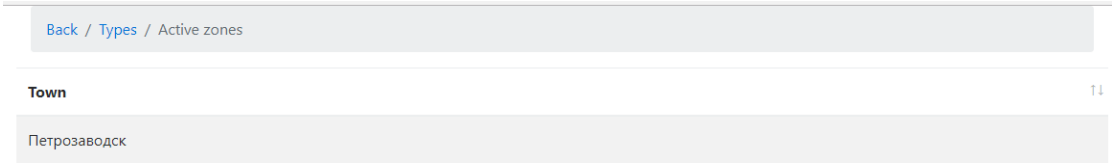

FIGURE 33. The Active-Assistants page

The "ActiveAssistants" page includes a table with existing active assistants. Each row leads to the "Assistant" page (Figure 33).

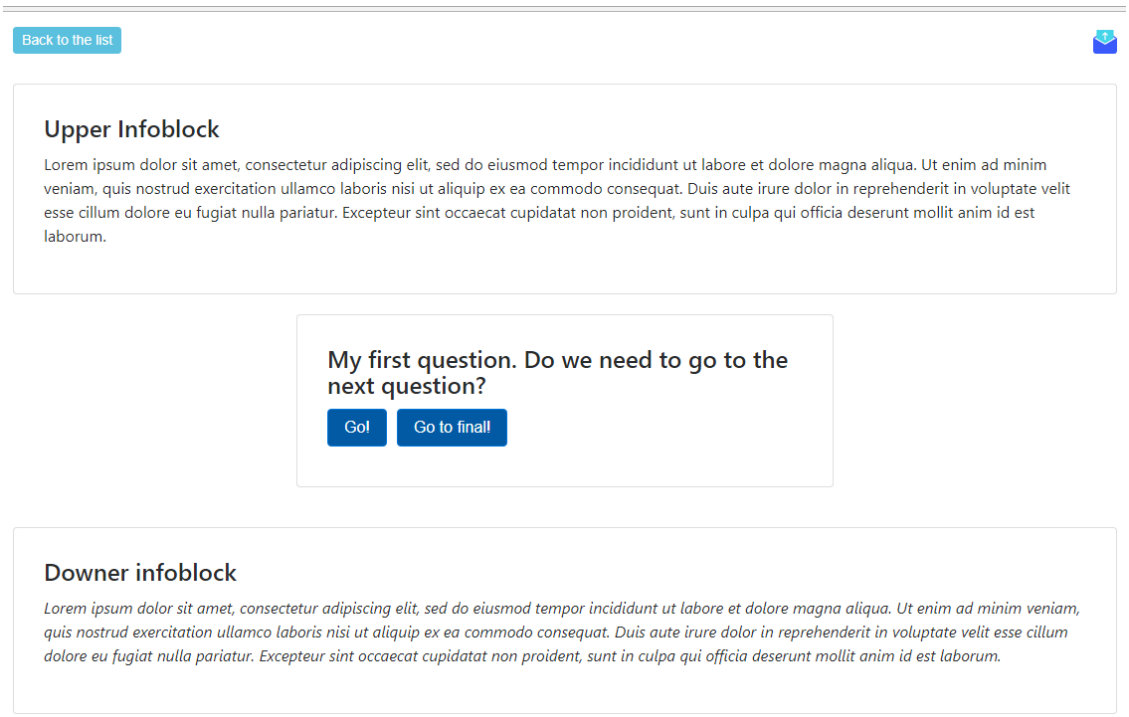

FIGURE 34. The Assistant page – a first step of the speech-scenario.

The "Assistant" includes a lot of components and it is the biggest Vue page of the project. There are imported "Question" and "Infoblocks" components, two buttons – one returns to the widget/Type page and another – to the previous step (question) of the speech-scenario. If it is the first question, the button is hidden, because no previous steps are provided.

Widget pages are using Vuex "widgetModule" and all operations are done with this module. It is imported in each widget page.

5 CONCLUSION

During the thesis process, the research and Speech-Scenarios web application were done and the author earned a huge amount of experience. The study concentrated on the research of modern front-end technologies and libraries, deep understanding on the chosen software and implementation of the web application for the Citylink Company - Russian Internet provider. The solution for the company provides a flexible system to create different speech-scenarios by engineers, without a requirement of code changes. Therefore, programmers of the company shifted their part of work to other engineers of the company and concentrated on other issues.

The project implementation included a systematic approach in the creation of such solutions. The module system of the application allows other programmers understand how it works ”from the box” much faster. During the development process, many technologies became understandable and applicable by the author in other applications.

After the implementation of the web application, engineers of other departments reduced time spending on different speech-scenarios cases. That is done by changing the old service based on pure programming languages to a Single Page Application based on modern VueJS and Laravel frameworks, which allows to spend less time on to load pages.

Apparently, the further work requires a new functionality to be implement such as the possibility to duplicate questions and types, the statistics analysis of the user`s experiments, integration of the application in other parts of the inner system.

REFERENCES AND SELF-PRODUCED MATERIALS

9 SERIES, Handcrafted Technology solutions. Why choose VueJS? [Accessed 2018-06-27.]

Available: https://www.9spl.com/blog/why-choose-vuejs-pros-and-cons/

ALTEXSOFT, blog portal. The Good and the Bad of ReactJS and React Native. [Accessed 2018-06-25.] Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/

Guru 99 portal. What is the AngularJS? [Accessed 2018-06-24.] Available: https://www.guru99.com/angularjs-introduction.html

HOMAN, Jacqueline. Pluralist. Relation and non-relational databases. [Accessed 2018-06-24.]

Available: https://www.pluralsight.com/blog/software-development/relational-non-relational-databases

MDN web docs, HTTP protocol documentation. [Accessed 2018-06-21.] Available: https://developer.mozilla.org/en-US/docs/Web/HTTP

MOREAU, Elise. Life wire. Browsers basics. [Accessed 2018-06-20.] Available: https://www.lifewire.com/what-is-a-web-application-3486637

NODEJS, official portal. About NodeJS. [Accessed 2018-07-10.] Available: https://nodejs.org/en/about/

NPM, official documentation. CLI commands. [Accessed 2018-07-10.] Available: https://docs.npmjs.com/cli

SHAPLAND, Rob 2017. Cross-site request forgery: Lessons from a CSRF attack example. [Accessed 2018-10-20.] Available: https://www.computerweekly.com/tip/Cross-site-request-forgery-Lessons-from-a-CSRF-attack-example

TECHNOPEDIA, blog portal. Application framework. [Accessed 2018-06-24.] Available: https://www.techopedia.com/definition/6005/application-framework

VUEJS, official documentation. API guide – Options/DOM. [Accessed 2018-07-28.] Available: https://vuejs.org/v2/api/#Options-DOM

VUEJS, official documentation. API guide – Options/Data. [Accessed 2018-07-30.] Available: https://vuejs.org/v2/api/#Options-Data

WEBPACK, official documentation. Webpack concepts. [Accessed 2018-07-05.] Available: https://webpack.js.org/concepts/

ZELL, Liew 2012. Understanding and Using REST APIs. [Accessed 2018-10-07.] Available: https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/