

TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Informatics

Chair of Information Systems

# **Mind map web application for students**

Bachelor's thesis

Student: Konstantin Tukmakov

Student's code: 121109 IAPB

Supervisor: Ingmar Pappel

Tallinn  
2015

## **Author's Declaration**

Herewith I declare that this thesis is based on my own work. All ideas, major views and data from different sources by other authors are used only with a reference to the source. The thesis has not been submitted for any degree or examination in any other university

.....

(date)

.....

(signature)

## Annotatsioon

Infohulk inimeste elus kogu aeg suureneb, kuid inimese mälu piir ei muutu, seega on tekkinud vajadus selle täiendava informatsiooni kuidagi efektiivselt meelde jätta. See puudutab kõigepealt õpilasi, kuna tänapäeval meie näeme haridusasutustes nõuete kalgistamise tendentsi ning omakorda kehtib see ka teiste asutuste suhtes. Antud töö eesmärk on pakkuda lõpptarbijale kasulikku rakendust, mille kasutamise käigus saab kasvatada õppeprotsessi efektiivsust, informatsiooni struktureerimist ja meeldejätmist.

Esiteks, on väga oluline, et rakendus on veebipõhine, nõnda ei pea kasutaja endale mingit tarkvara instaleerima. JavaScript on ainuke võrgust pärinev programmeerimiskeel, mis annab võimaluse suhelda HTML-i graafilise osaga, ilma milleta meie rakendust lihtsalt ei oleks. JavaScripti algupärane graafiline osa on väg nõrk ja vanamoodne, mistõttu on otstarbekas kasutada library-t, mis aitaks tegeleda graafilise osaga.

Teiseks, peab rakendus mõistma meeles kaartide mõtet, mida on põhjalikult kirjeldatud Tony Buzani raamatus "The Mind Map Book: How to Use Radiant Thinking to Maximize Your brain's Untapped Potential".

Valmis projekti tulemuseks on kasutajasõbralik töötav rakendus, mis võib muuta meeles kaarte iga kasutaja jaoks arusaadavaks. Rakendus peab sisaldama kogu võimast meeles kaartide funktsionaalsust ja võib hõlmata ka muid funktsioone, nagu grupid ja mallid, integreerimine ülikoolidesse jne.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 58 leheküljel, 4 peatükki, 21 joonist, 11 tabelit.

## **Abstract**

The amount of information in people's lives continues to increase, yet the limits of the human memory do not change –thus there is a need to somehow effectively memorize the additional information. In particular, this applies to students, because nowadays we can see a tendency to tighten requirements in educational institutions and this, in its turn, applies to all kinds of institutions. Thereby, the aim of the work is to provide the end user with an application, a useful tool, which can be used to significantly grow the efficiency of learning, structuring and memorizing information.

First of all, it is very important, that the application is web-based, so users will not have to install software in their OS. JavaScript is the only web native programming language that provides functionality to interact with HTML graphical part, without which our application just cannot exist. The native support of the graphical part in JavaScript is very weak and old-fashioned, so it is rational to use library that would help us deal with the graphical part.

Secondly, the application must realize the idea of mind maps, which are comprehensively described in Tony Buzan's book "The Mind Map Book: How to Use Radiant Thinking to Maximize Your brain's Untapped Potential" .

The result of the finished project is a user-friendly working application that can make mind maps understandable for every user that uses it. The application must include the whole powerful functionality of mind maps and can include other functions, like group and templates, universities integration and etc.

The thesis is in English and contains 58 pages of text, 4 chapters, 21 figures, 11 tables., etc.

## Abbreviations and glossary of terms

|                 |   |
|-----------------|---|
| API             | Application programming interface. It is a set of tools for building software applications.   |
| JSON            | JavaScript Object Notation. It is a lightweight data-interchange format.  |
| SQL             | SQL is a standardized query language for requesting information from a database.<br><a href="http://www.webopedia.com/TERM/S/SQL.html">[http://www.webopedia.com/TERM/S/SQL.html]</a>   |
| HTTP            | Hypertext Transfer Protocol – application protocol, that do exchange or transfer of hypertext.  |
| CSS             | Cascading Style Sheets - is a style sheet language used for describing the look and formatting of a document written in a markup language.<br><a href="http://en.wikipedia.org/wiki/Cascading_Style_Sheets">[http://en.wikipedia.org/wiki/Cascading_Style_Sheets]</a>   |
| Markup language | A markup language is a system for annotating a document in a way that is syntactically distinguishable from the text.<br><a href="http://en.wikipedia.org/wiki/Markup_language">[http://en.wikipedia.org/wiki/Markup_language]</a>  |
| HTML            | HTML is the standard markup language used to create web pages. It is written in the form of HTML elements consisting of <i>tags</i> enclosed in angle brackets<br><a href="http://en.wikipedia.org/wiki/HTML">[http://en.wikipedia.org/wiki/HTML]</a>   |
| AJAX            | Asynchronous JavaScript and XML - with Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.<br><a href="http://en.wikipedia.org/wiki/Ajax_%28programming%29">[http://en.wikipedia.org/wiki/Ajax_%28programming%29]</a> |

Mind map

A mind map is a diagram used to visually organize information. A mind map is often created around a single concept, drawn as an image in the center of a blank landscape page, to which associated representations of ideas such as images, words and parts of words are added.

[\[http://en.wikipedia.org/wiki/Mind\\_map\]](http://en.wikipedia.org/wiki/Mind_map)

LESS

LESS is a dynamic stylesheet language that can be compiled into Cascading Style Sheets (CSS), or can run on the client-side and server-side.

[\[http://en.wikipedia.org/wiki/Less\\_%28stylesheet\\_language%29\]](http://en.wikipedia.org/wiki/Less_%28stylesheet_language%29)

## Pictures list

|                  |  |
|------------------|--|
| <b>Figure 1</b>  | Node JSON example                              |
| <b>Figure 2</b>  | Link JSON example                              |
| <b>Figure 3</b>  | Domain model diagram                           |
| <b>Figure 4</b>  | Map and node configuration model diagram       |
| <b>Figure 5</b>  | Map and node configuration model diagram       |
| <b>Figure 6</b>  | User registration/authentication model diagram |
| <b>Figure 7</b>  | Group model diagram                            |
| <b>Figure 8</b>  | Template model diagram                         |
| <b>Figure 9</b>  | Index page                                     |
| <b>Figure 10</b> | Registration form                              |
| <b>Figure 11</b> | Sign up user code snippet                      |
| <b>Figure 12</b> | Authentication form                            |
| <b>Figure 13</b> | User authentication code snippet               |
| <b>Figure 14</b> | Map creation form                              |
| <b>Figure 15</b> | Map initialization code snippet                |
| <b>Figure 16</b> | Node creation form                             |
| <b>Figure 17</b> | Node edit form                                 |
| <b>Figure 18</b> | Node content addition/edit window              |
| <b>Figure 19</b> | Node sort code snippet                         |

**Figure 20**

Nodes deletion code snippet

**Figure 21**

Full mind map example



## Tables list

|                 |  |
|-----------------|--|
| <b>Table 1</b>  | Objects definitions  |
| <b>Table 2</b>  | Main model tables definitions  |
| <b>Table 3</b>  | Main model attributes definitions                                    |
| <b>Table 4</b>  | Map and nodes tables definitions                                     |
| <b>Table 5</b>  | Map and nodes attributes definitions                                 |
| <b>Table 6</b>  | User registration/authentication tables definitions                  |
| <b>Table 7</b>  | User registration registration/authentication attributes definitions |
| <b>Table 8</b>  | Group model tables definitions                                       |
| <b>Table 9</b>  | Group model attributes definitions                                   |
| <b>Table 10</b> | Template model tables definitions                                    |
| <b>Table 11</b> | Template model attributes definitions                                |

# Table of contents

|       |  |    |
|-------|--|----|
| 1.    | Introduction .....                               | 12 |
| 1.1   | Research problem and objectives.....             | 12 |
| 1.2   | Tasks formulation .....                          | 13 |
| 1.3   | Methodology .....                                | 14 |
| 2.    | Main tools.....                                  | 15 |
| 2.1   | D3.js JavaScript library .....                   | 15 |
| 2.1.1 | Overview .....                                   | 15 |
| 2.1.2 | Data storage .....                               | 15 |
| 2.1.3 | D3 layout .....                                  | 16 |
| 2.2   | Google .....                                     | 16 |
| 2.2.1 | Overview .....                                   | 16 |
| 2.2.3 | Google Cloud SQL .....                           | 16 |
| 2.3   | The Facebook.....                                | 17 |
| 2.3.1 | Overview .....                                   | 17 |
| 2.3.2 | The Facebook Platform .....                      | 17 |
| 2.4   | Vkontakte.....                                   | 17 |
| 2.4.1 | Overview .....                                   | 17 |
| 2.4.2 | Vkontakte Open API .....                         | 17 |
| 3.    | Design.....                                      | 18 |
| 3.1   | Requirements .....                               | 18 |
| 3.1.1 | Authentication via Facebook API.....             | 18 |
| 3.1.2 | Authentication via VKontakte API.....            | 19 |
| 3.1.3 | Retrieving the data from database .....          | 20 |
| 3.1.4 | Visualizing of gathered data form database ..... | 20 |
| 3.1.5 | On-the-fly node data edition.....                | 20 |
| 3.1.6 | Editing data of certain node.....                | 20 |
| 3.1.7 | Addition of new nodes.....                       | 20 |
| 3.1.8 | Addition and editing node content .....          | 21 |
| 3.1.9 | Group of users with identical mind maps.....     | 21 |
| 3.2   | Domain model.....                                | 22 |
| 3.2.1 | Diagram .....                                    | 22 |

|        |  |    |
|--------|--|----|
| 3.2.2  | Object's definition .....                    | 22 |
| 3.3    | Data model .....                             | 24 |
| 3.3.1  | Main model .....                             | 24 |
| 3.3.2  | Map and node configuration model .....       | 29 |
| 3.3.3  | User registration/authentication model ..... | 31 |
| 3.3.4  | Group model .....                            | 33 |
| 3.3.5  | Template model .....                         | 36 |
| 4.     | Prototype .....                              | 39 |
| 4.1    | Tools .....                                  | 39 |
| 4.1.1  | D3 Library .....                             | 39 |
| 4.1.2  | JavaScript and JQuery library .....          | 40 |
| 4.1.3  | Quill JavaScript library .....               | 40 |
| 4.1.4  | Bootstrap framework .....                    | 40 |
| 4.1.5  | LESS, a CSS pre-processor .....              | 40 |
| 4.2    | Functionality .....                          | 41 |
| 4.2.1  | Overview .....                               | 41 |
| 4.2.2  | Basic architecture .....                     | 41 |
| 4.2.3  | User registration .....                      | 42 |
| 4.2.4  | User authentication .....                    | 45 |
| 4.2.5  | Mind map creation .....                      | 46 |
| 4.2.6  | Mind map deletion .....                      | 47 |
| 4.2.7  | Mind map visualization .....                 | 48 |
| 4.2.8  | Map node creation .....                      | 50 |
| 4.2.9  | Node data update .....                       | 51 |
| 4.2.10 | Node content addition/edit .....             | 51 |
| 4.2.11 | Map node deletion .....                      | 52 |
| 4.3    | Mind map example .....                       | 54 |
|        | Kokkuvõte .....                              | 56 |
|        | Summary .....                                | 57 |
|        | References .....                             | 58 |

# **1. Introduction**

The problem of organization of information for people is relevant these days. On the one hand it is caused by part of information in people's life in nowadays, compared to early age. On the other hand, this is due to human nature that limits our capacity of memorized information.

Being a student, as experienced difficulties of quick and effective memorization of information on my own experience, analysis was conducted of various methods and techniques to achieve the most effective results.

Great efforts have been made in order to facilitate and structure information and a major breakthrough in this area was done by inventing and implementation of mind maps. Tony Buzan was first who popularized this tool and my application will help to ease the process of creating mind maps and extend their functionality.

The purpose of this work is to provide web-based application, which could authenticate each user, create new mind maps for each user, new topics in each map and save content data as formatted text. Each map could be customized as the user wants. The customization includes structuring nodes in map, colorization of topics.

The application uses Google tools to hold data related to the application, including a representative part as well as back-end data. The authentication through third-party like social network services is essential for user experience.

The first chapter represents overview requirements and goals of the application. The second chapter describes main tools which can help to achieve project (thesis) aims. Next one tells about the design of the application. The fourth one represents prototype of the application. And the final part is for the summary.

## **1.1 Research problem and objectives**

This thesis aims to explain the application for creating mind maps, how it can help in learning

new information, structuring already learned information, as well as saving every piece of information in a convenient form for each user and re-use the maps whenever needed. First of all the project should reflect a functionality so-called mind map, offered by author Tony Buzan.

Foremost application is designed for people, who feels the need for memorization and structuring information almost every day. Such people may be students from school to university, scientists and etc. While learning something new or changing old assessment, user can easily modify mind map according to the new information received. This is the difference from the old-fashioned mind maps on paper – user can change it all the time, in the process of learning. But more than that, such a tool as mind maps is significantly multilateral and can contain everyday information, e.g. day schedule, diet graphic and etc., thereby widening the contingent of users from people of school age to elderly.

The application is web-based, so as a result user can use application only with web-browser with the necessity of internet connection. Moreover, it is possible to export mind maps in pdf format, but in this situation mind map becomes static, without possibility to change the information that lies within.

## **1.2Tasks formulation**

The application provides next features:

- Add content to each node as formatted text. The content of the node can be text, formatted according to the user's preferences. Basic text formatting is provided: color, positioning, size, font family, inserted as an unordered list, inserted as ordered list.
- Mind map creation. Each user can create any number of mind maps. There must be no constraints on creation action. The process of creation must be simple.
- Mind map customization. Each user can customize certain mind map according to his preferences. Customization is determined in node background color, text color and the name of the nodes.
- Dynamic data. Data in application must be dynamic. Every change must be applied on-the-fly.

- User identification. Each user can be authenticated using his email. Also, authentication through popular social networks is possible. Chosen such social networks: Facebook and VKontakte. Facebook is the most popular social network in the world, so this choice is predictable. VKontakte is the most popular social network among the Russian population. Both of these social networks have clean and simple API whereby we can apply it to the application easily.

### **1.3 Methodology**

In the prototype were used such technologies as: HTML, CSS, LESS, Bootstrap, JavaScript, JQuery, Quill, PHP, MySQL.

The main data visualization functionality, which is related to mind map creation is done by a using JavaScript library named “D3.js”. The choice was made to this library after careful analysis of existing libraries that could give result with high performance, scalability and good documentation. Surprisingly, there are a very small number of such libraries and among the existing the best performance demonstrated D3 library.

PHP and MySQL stack was chosen as it recommended as very powerful and stable technologies. No libraries for PHP were used, but plain programming language.

### **1.4 Analogs**

There are few applications, which provide mind map functionality within. A lot of them are not web-based and users have to install their software on computers, which sometimes is difficult. This approach imposes some restrictions on the application. One of these restrictions is that developers have to support multiple operation systems. Web analogs, like “Mindmup”<sup>1</sup>, have good functionality, which can go beyond our application’s functionality, but the idea of our application is to provide such an application, which could help users to learn some subject and add to mind map synopsis elements. User can add text and format it as it would be synopsis.

## 2. Main tools

### 2.1 D3.js JavaScript library <sup>2</sup>

#### 2.1.1 Overview

D3.js is very powerful that can manipulate HTML based on data. To bring functionality to life, it uses manipulation through HTML, CSS and SVG. With this library, it is possible to dynamically connect nodes with their children, using such SVG elements as 'line', 'path' and etc.

#### 2.1.2 Data storage

Data is stored in JSON format. Every node represents an object that has it is an own identification number which relies on database data and, as a result, no collision can happen. Also in a node object lies within other data that can be used in map customization.

Example of node data object:

```
[▼ Object 1
  bg_color: "#ffffff"
  content: null
  fixed: true
  id: 91
  index: 0
  label: "Mathematics"
  level: 1
  position_x: NaN
  position_y: NaN
  px: 681
  py: 444.07500000000005
  txt_color: "#4B4B4B"
  user_fixed: 0
  weight: 1
  x: 681
  y: 444.07500000000005
  ► __proto__: Object
```

Figure 1. Node JSON example

Links are stored in JSON format in the same way, but they have only 2 parameters: source and target.

Example of links data object:

```
[▼ Object 1 ]  
  source: 91  
  target: 92  
  ▶ __proto__: Object
```

Figure 2. Node JSON example

### 2.1.3 D3 layout 3

D3 layout determines how data will be visualized in relation to each other. Layout complexity varies from desired task to task. E.g. it can be as stacking bars in a chart or as complex as labeling a map. The layout takes data in JSON data format as input and outputs the result, which has determines positions or shapes for layout visualization.

## 2.2 Google <sup>4</sup>

### 2.2.1 Overview

Google is a multinational technology company with stack of services, which varies from entertainment services to business services. The popularity to Google Company gave its search service, which is currently number one among all search engines in the world, with more than three billion searches every day and 64.5% market share. <sup>5</sup> Provides very various number of robust, flexible, innovative services for developers by low prices.

### 2.2.2 Google Cloud Platform <sup>6</sup>

Google cloud Platform is a cloud computing platform by Google that provides different services, which can be used to create anything from simple websites to complex application, and Google Cloud SQL is one of these services, which in their turn can do various tasks like hosting and computing, storage files and database in the cloud and etc. Characterized as high performance, simple to use and robust tool.

### 2.2.3 Google Cloud SQL <sup>7</sup>

Google Cloud SQL is a MySQL database installed in Google Cloud. Google included different additional settings to configure and control the database, so it has advantages to use Google Cloud SQL than MySQL + PhpMyAdmin database installation. Google Cloud SQL can also



be used with PhpMyAdmin to manage database content, as well as simple CLI access is also possible.

## **2.3 The Facebook**

### **2.3.1 Overview**

Facebook is a top social network, with 1.44 billion monthly active users<sup>8</sup> leading by Mark Zuckerberg as CEO and Chairman. Facebook has grown from social network to a brand very fast from 2004 to present and made Zuckerberg a billionaire and the youngest billionaire in 2012<sup>9</sup>. The amount of people registered in Facebook is absolutely stunning, so, nowadays, it is obvious why we chose Facebook as a login-through system, as the leader in the list.

### **2.3.2 The Facebook Platform <sup>10</sup>**

The Facebook platform is set of tools and services published by Facebook to provide developers and other interested people to gather data and use it in third-party projects.

## **2.4 VKontakte**

### **2.4.1 Overview**

VKontakte social network is established in September of 2006 by Pavel Durov with currently more than 300 million of users. VKontakte is the largest social network among the Russian-speaking users, but there are also persist users, which don't speak Russian at all, but their amount is very small due to predominance of content on Russian language.

### **2.4.2 VKontakte Open API <sup>11</sup>**

The API allows third-parties to exchange data from VKontakte including user's authorization. VKontakte API also give an opportunity to integrate user's data deeper with third-parties applications, though these actions require user's agreement to this action.

## 3. Design

### 3.1 Requirements

In this section will be described how the application must work to achieve the goals, which were mentioned before. I should note, that stack of technologies may vary depending on the architect's vision. Server side programming language can be any other like: Ruby, Python, Java and etc., although unchanged will be always HTML, CSS and JavaScript technologies.

I chose Bootstrap, LESS, Quill and D3 technologies in the application due to their popularity, documentation, community support and functionality that they offer. Of course, the application can be done without these four technologies, but in this case development of the application will be highly complicated.

#### 3.1.1 Authentication via Facebook API

Facebook application authorization uses the OAuth 2.0 open protocol. To access the API, in other words, to authorize user in our application, we have to get "access\_token", which is given if the user completes the VKontakte authorization. Requests to the API are sent via POST or GET method through the specified URL. In our case we have to authorize user by this URL:

```
https://www.facebook.com/dialog/oauth?client_id=APP_ID&redirect_uri=REDIRECT_URI  
&response_type=code
```

Where:

- APP\_ID is our application id, registered in Facebook system
- REDIRECT\_URI – URL where **code** will be passed.

If user logs in through VKontakte API and gives permissions to our application, then we have **code** and we can move forward and receive access token. To receive **access\_token** we have to make one more request to the Facebook API. The request will have 

```
https://graph.facebook.com/oauth/access_token?client_id=APP_ID&redirect_uri=REDIRECT_URI  
&client_secret=CLIENT_SECRET&code=CODE
```

as URL and secret data of the application. Secret data includes in **APP\_ID** and **SECRET\_KEY** which can be fetched in Facebook API application settings.

### 3.1.2 Authentication via VKontakte API

VK application authorization uses the OAuth 2.0<sup>12</sup> open protocol and application has to use so named server authorization. To access the API, in other words, to authorize user in our application, we have to get “access\_token”, which is given if the user completes the VKontakte authorization.

Requests to the API are sent via POST or GET method through the specified URL. In our case we have to authorize user by this URL:

[http://oauth.vk.com/authorize?client\\_id=APP\\_ID&redirect\\_uri=REDIRECT\\_URI&scope=PERMISSIONS&response\\_type=code&v=API\\_VERSION](http://oauth.vk.com/authorize?client_id=APP_ID&redirect_uri=REDIRECT_URI&scope=PERMISSIONS&response_type=code&v=API_VERSION)

Where:

- APP\_ID is our application id, registered in VKontakte system
- PERMISSIONS is requested application access permissions
- REDIRECT\_URI – URL where **code** will be passed.
- API\_VERSION is the version of the API that our application is currently using to access VKontakte API.

If user logs in through VKontakte API and gives permissions to our application, then we have **code** and we can move forward and receive access token. **Code** is a parameter that can be used to receive **access token** within 1 hour, so we must receive it immediately. To receive **access\_token** we have to make one more request to VKontakte API. The request will have [https://oauth.vk.com/access\\_token](https://oauth.vk.com/access_token) URL and secret data of the application. Secret data includes in **client\_id** and **client\_secret**, which can be fetched in VKontakte API application settings..

Important note that calls are made through https secure protocol, which means that transmitted data is secured.

### **3.1.3 Retrieving the data from database**

In the application we use MySQL database as the main storage for data. MySQL database is stored in Google Cloud MySQL and thus data acquisition is faster than if we used local server. Google also provides settings, which can make instances, all around the world, starting from America continent to Asia. The performance of data retrieving must be brilliant.

### **3.1.4 Visualizing of gathered data form database**

To visualize the data we use D3.js JavaScript library. First of all we need to reconstruct data from the database, to capable data format for D3. Then, we should provide to D3 library this data and start visualization process. Starting visualization process also means, that we should set up settings that would make visualization user-friendly.

### **3.1.5 On-the-fly node data edition**

Node data got to be updated on-the-fly, invisibly for the user when it is needed. For example, it is important to give user ability to re-layout the mind map as user wants, and this action must be done “behind the scenes”, so users won’t be distracted on this action. This operation is done using AJAX.

### **3.1.6 Editing data of certain node**

Each node of a certain mind map must be customizable. Without node customization, after it created, we are not able to provide full mind map functionality. In node customization includes at least: background color, text color, label text, positioning. This operation includes in it user intervention. After editing operation ends, new data must be permanently saved in the database until the user edits or deletes this node again. This operation is available only if mind map is chosen.

### **3.1.7 Addition of new nodes**

Operation of addition of new nodes consists of saving data into the database. This operation is available only if mind map is chosen. The user has to fill form with required data and if provided data is correct, the mind map will be reconstructed according to new data inserted. The addition of new node affects nodes links too, because link must be added between parent node and newly created, child node.

### **3.1.8 Addition and editing node content**

In application node can be added content. Node content is formatted text and images that are stored in the database in a formatted way. To format text right in HTML we use Quill<sup>13</sup>, which represents JavaScript WYSIWYG text editor. With this tool we can easily and effectively add, format, edit text and insert images. Text format consists of text positioning, text colorization, font family choosing, text resizing, changing text to bold, italic or underline text. Also unordered and ordered lists are supported to addition.

### **3.1.9 Group of users with identical mind maps**

Group functionality must be carefully designed and implemented. Each group has the same list of maps and everybody has an access to them, although not at the same time. This is an important constraint, that only one user of a group can have an access to edit certain map at each time. This kind of behavior removes the chance of collision in groups, but reduces flexibility of application. After group is created, user can invite collaborators via email and after the invitation is accepted by invited user – he has an access to edit group's mind maps.

### 3.2 Domain model

#### 3.2.1 Diagram

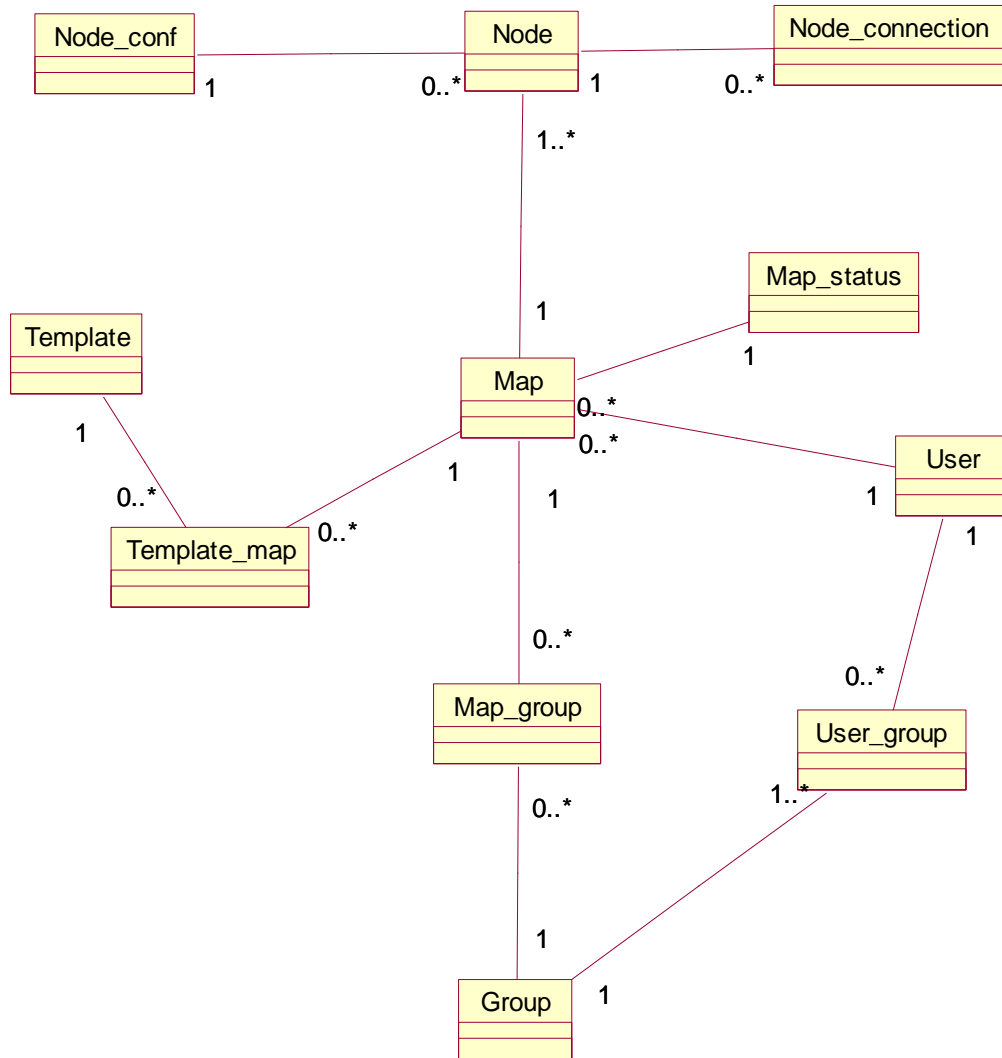


Figure 3. Domain model diagram

#### 3.2.2 Object’s definition

| Object’s name     | Definition  |
|-------------------|---|
| <b>Map</b>        | Map is mind map object, which wraps all nodes in one place.   |
| <b>Map status</b> | Map status is social status format. It can be ‘public’, ‘private’ and ‘shared’. Default value is ‘private’. |

|                           |  |
|---------------------------|--|
| <b>Node</b>               | Nodes are like bricks in building, Map totally composed from Nodes. Contains content as text. Connected with other nodes via Node_connection.  |
| <b>Node connection</b>    | Connection between nodes. Represents source-target relationship, where source equals parent_node_id and target is child_node_id.   |
| <b>Node configuration</b> | Node configuration object contains data for visualization each node determining it by its level. If node level is more than the biggest node level in configuration table, then these nodes will be configured as the biggest level number node. |
| <b>Group</b>              | Group is object contains user's groups within which they can collaborate on the same stack of mind maps.   |
| <b>Template</b>           | Template is object wraps maps in it and can be shared by any user registered in system.  |
| <b>User</b>               | User object contains information about each user. Contains connection information between application and VKontakte API or/and Facebook API.   |

**Table 1. Object's definitions table**

### 3.3 Data model

#### 3.3.1 Main model

##### 3.3.1.1 Diagram

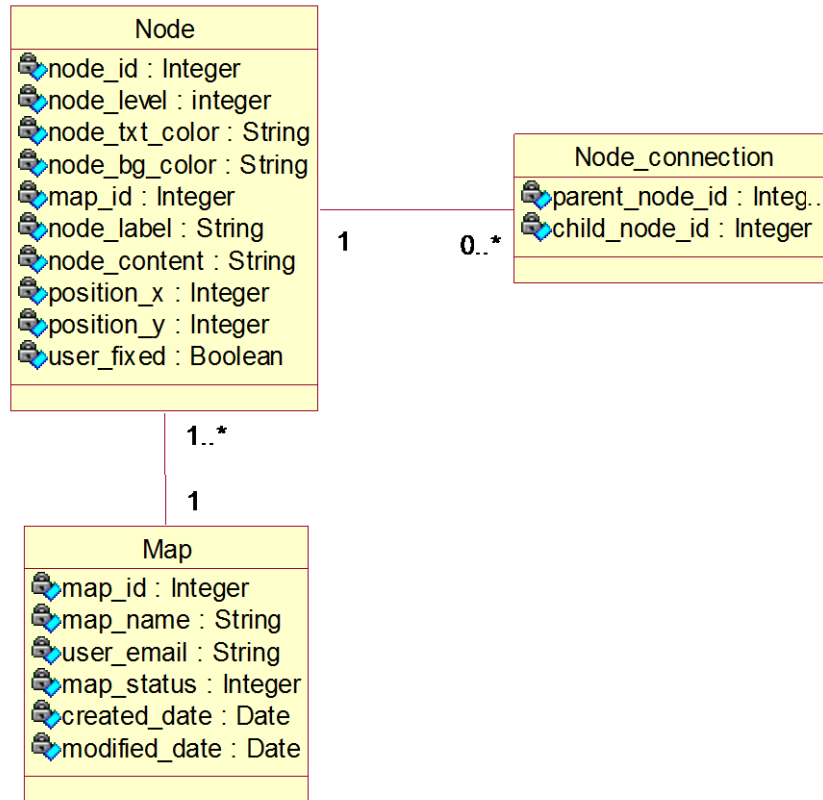


Figure 4. Main model diagram

##### 3.3.1.2 Table's definitions

###### TABLE

###### DEFINITION

|      |  |
|------|--|
| MAP  | Map table contains all data for the each map needs to identify it.         |
| NODE | Node table contains all data that need to be reproduced in user's browser. |



|                 |  |
|-----------------|--|
| NODE_CONNECTION | Node_connection table contains data, that necessary to reproduce links between nodes in map. |
|-----------------|--|

Table 2. Main model tables definitions

### 3.3.1.3 Attributes definitions

| Table | Attribute     | Type         | Definition  | Example  |
|-------|---------------|--------------|---|--|
| Map   | map_id        | INTEGER      | Map's identification number.  | 111  |
| Map   | map_name      | VARCHAR (30) | Name of the map. Also this name spreads on 1-st level node label level.                                 | Mathematics                                      |
| Map   | user_email    | VARCHAR (30) | Foreign key [User.user_email]   | <a href="mailto:test@test.com">test@test.com</a> |
| Map   | map_status    | SMALLINT     | Foreign key [Map_status.map_status_id]  | 1  |
| Map   | created_date  | TIMESTAMP    | Timestamp data to show users when the map was created. Plays role of separator from one map to another. | 2015-05-13 12:19:33                              |
| Map   | modified_date | TIMESTAMP    | Timestamp data to show users when the map was modified last time. This kind information provides        | 2015-05-13 12:19:33                              |

|      |                |             |  |         |
|------|----------------|-------------|--|---------|
|      |                |             | good point to delete old maps if they do not used.   |         |
| Node | node_id        | INTEGER     | Node identification number.  | 919     |
| Node | node_level     | INTEGER     | Node level is a number of remoteness from the main node which level is 1.  | 2       |
| Node | node_bg_color  | VARCHAR (7) | Node background color serves for node customization. This field applies to svg circle element as background color. Background color field stores value as a HEX number with ‘#’ symbol at the beginning. | #000000 |
| Node | node_txt_color | VARCHAR (7) | Node text color server for node customization. This field applies to div element in SVG ForeignObject element. Text color field stores value as a HEX number with  | #ffffff |

|      |              |              |   |  |
|------|--------------|--------------|---|--|
|      |              |              | '#' symbol at the beginning.  |  |
| Node | map_id       | INTEGER      | Foreign key [Map.map_id]  | 1  |
| Node | node_label   | VARCHAR (30) | Field that contains value of node's label. Limited to 30 characters long due to ratio constraint between user's display and circle SVG element possible dimension.                        | Dynamic  |
| Node | node_content | TEXT         | Field contains all content of the node, that user provides. It contains text already as formatted text. Text formatted with Quill JavaScript library which uses HTML tags to format data. | <div style="text-align: center;"><span style="font-size: 18px;"><b>The main section</b></span></div><div>Will save this data for future.</div> |
| Node | position_x   | SMALLINT     | Field serves to save position of the node. If user fixed field value equals to TRUE, then this field  | 291  |

|                 |                |          |   |     |
|-----------------|----------------|----------|---|-----|
|                 |                |          | will be used to save the node's 'x' position coordinate.  |     |
| Node            | position_y     | SMALLINT | Field serves to save position of the node. If user fixed field value equals to TRUE, then this field will be used to save the node's 'y' position coordinate.                           | 400 |
| Node            | user_fixed     | BOOLEAN  | Field serves to save user custom positioning of certain node. If the value of this field equals to TRUE, then position_x and position_y field will be filled with position coordinates. | 1   |
| Node_connection | parent_node_id | INTEGER  | Field serves as link source in source-target relationship between nodes.<br><br>Foreign key [Node.node_id]  | 1   |
| Node_connection | child_node_id  | INTEGER  | Field serves as link target in source-  | 2   |

|  |  |  |                                    |  |
|--|--|--|------------------------------------|--|
|  |  |  | target relationship between nodes. |  |
|  |  |  | Foreign key [Node.node_id]         |  |

Table 3. Main model attributes definitions

### 3.3.2 Map and node configuration model

#### 3.3.2.1 Diagram

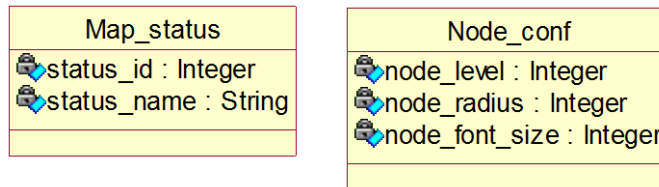


Figure 5. Map and node configuration model diagram

#### 3.3.2.2 Tables definitions

| TABLE      | DEFINITION   |
|------------|--|
| MAP_STATUS | Map status serves to configure different types of Map status. Map status can vary, depending on user settings. |
| NODE_CONF  | Node configuration table contains visualization data depending on node level.                                  |

Table 4. Map and node tables definitions

#### 3.3.2.3 Attributes definitions

| <b>Tables</b> | <b>Attribute</b> | <b>Type</b>  | <b>Definition</b>   | <b>Example</b> |
|---------------|------------------|--------------|---|----------------|
| Map_status    | status_id        | INTEGER      | Status identification number.   | 1              |
| Map_status    | status_name      | VARCHAR (30) | Map status name can vary. Possible values are: 'Private', 'Public', 'Shared'. This data represents statement of the Map and its viewing parameters. | Public         |
| Node_conf     | node_level       | SMALLINT     | Node level field determines the distance from the center / main node. If level equals to 1, then it is main node and positioned in center.          | 2              |
| Node_conf     | node_radius      | SMALLINT     | Node radius field determines SVG circle element radius.   | 75             |
| Node_conf     | node_font_size   | SMALLINT     | Node radius field determines  | 16             |

|  |  |  |                            |  |
|--|--|--|----------------------------|--|
|  |  |  | font size of Node's label. |  |
|--|--|--|----------------------------|--|

Table 5. Map and nodes attributes definitions

### 3.3.3 User registration/authentication model

#### 3.3.3.1 Diagram

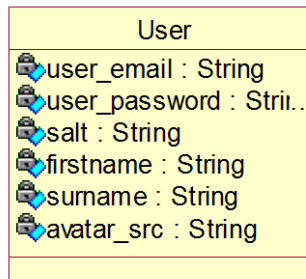


Figure 6. User registration/authentication model diagram

#### 3.3.3.2 Tables definitions

**TABLE**

**DEFINITION**

|      |  |
|------|--|
| USER | User table contains information needed to authenticate user in application. There can be no duplicates of users. User registers with in application via email. |
|------|--|

Table 6. User registration/authentication tables definitions

#### 3.3.3.3 Attributes definitions

| Table | Attribute  | Type         | Definition   | Example  |
|-------|------------|--------------|--|--|
| USER  | user_email | VARCHAR (30) | User email serves to identify user in application. Email must be unique, duplicates are not allowed. Email can | <a href="mailto:test@test.com">test@test.com</a> |

|      |               |               |   |   |
|------|---------------|---------------|---|---|
|      |               |               | be used to identify user also in social networks: Facebook and VKontakte.   |   |
| USER | user_password | VARCHAR (60)  | Password stores in database as hash. Blowfish encryption algorithm of password is used to provide robust password defence from stealing the password. | \$2a\$10\$LOEG<br>PzSeA4tZeVS<br>Zqk7/6OXOL<br>Xxnwx/7ISNa<br>PCIQOI2LVc<br>V8MVTIW |
| USER | salt          | VARCHAR (31)  | Salt field value serves to 'salt' the password with it, therefore we have more robust password value. Salt prevents brutal-force attacks.             | \$2a\$10\$LOEG<br>PzSeA4tZeVS<br>Zqk7/6Q==  |
| USER | firstname     | VARCHAR (30)  | Firstname field serves to identify user by his first name. In stack with surname we can get full name.  | John  |
| USER | surname       | VARCHAR (30)  | Surname field serves to identify user by his surname. In stack with first name we can get full name.  | Doe   |
| USER | avatar_src    | VARCHAR (255) | Avatar_src field serves as source, pointer to the   | images/avatars<br>/humansfriend   |



|  |  |  |                                      |                |
|--|--|--|--------------------------------------|----------------|
|  |  |  | image file of certain user's avatar. | @gmail.com.jpg |
|--|--|--|--------------------------------------|----------------|

Table 7. User registration/authentication attributes definitions

### 3.3.4 Group model

#### 3.3.4.1 Diagram

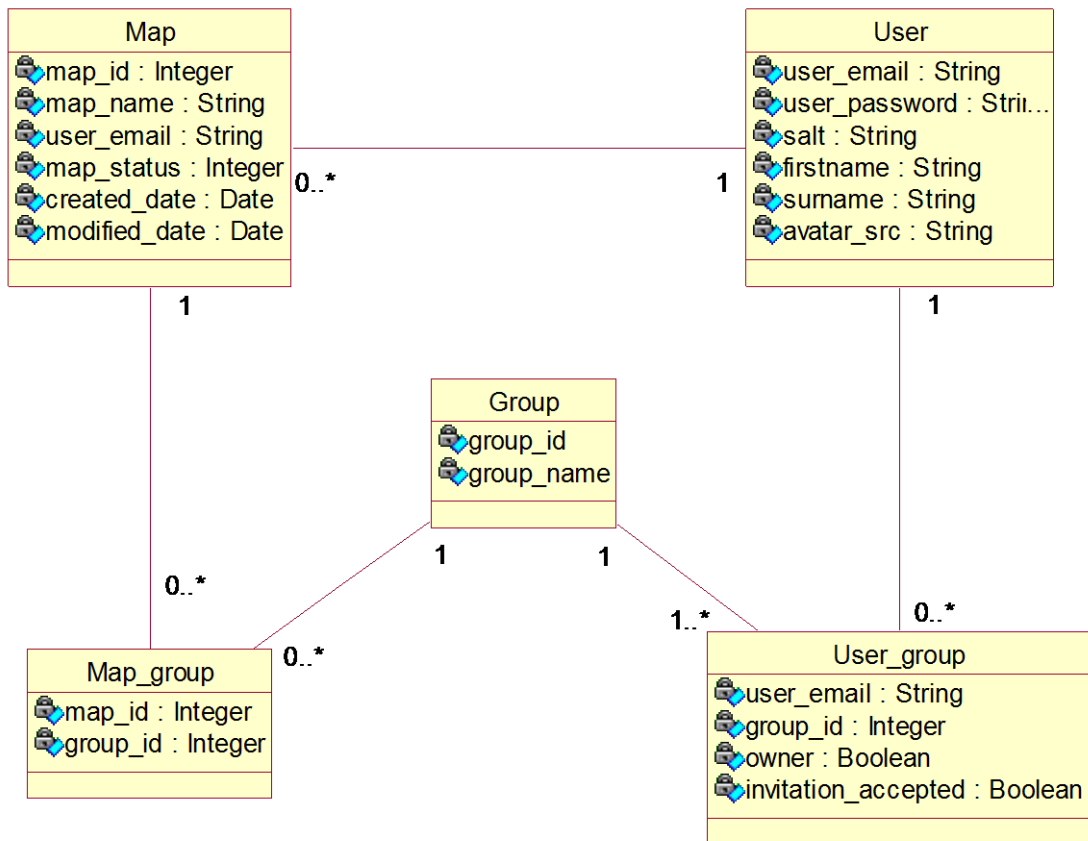


Figure 7. Group model diagram

### 3.3.4.2 Tables definitions

| TABLE             | DEFINITION  |
|-------------------|---|
| <b>MAP_GROUP</b>  | Map_group table's aim is the connection between Map and Group tables.   |
| <b>GROUP</b>      | Group table consists of group's configuration data.                     |
| <b>USER_GROUP</b> | User_group table's aim is the connection between User and Group tables. |

Table 8. Group model tables definitions

### 3.3.4.3 Attributes definitions

| Tables    | Attribute  | Type            | Definition  | Example         |
|-----------|------------|-----------------|---|-----------------|
| Map_group | map_id     | INTEGER         | Foreign key<br>[Map.map_id]   | 152             |
| Map_group | group_id   | INTEGER         | Foreign key<br>[Group.group_id]   | 199             |
| Group     | group_id   | INTEGER         | Group identification number. Must be unique.  | 1001            |
| Group     | group_name | VARCHAR<br>(30) | Group field value represents name of the group. Serves as group identification for users. | TTU<br>Workshop |

|            |                     |                 |   |  |
|------------|---------------------|-----------------|---|--|
| User_group | user_email          | VARCHAR<br>(30) | Foreign key<br>[User.user_email]  | <a href="mailto:test@test.com">test@test.com</a> |
| User_group | group_id            | INTEGER         | Foreign_key<br>[Group.group_id]   | 191  |
| User_group | owner               | BOOLEAN         | Field value serves<br>as indicator if<br>user is the owner<br>of the group.   | 1  |
| User_group | invitation_accepted | BOOLEAN         | Field value serves<br>as indicator if<br>user was invited<br>or already<br>accepted the<br>invitation. If field<br>value equals to<br>FALSE, then<br>invitation was<br>sent, but user has<br>not accepted it<br>yet. If field value<br>equals to TRUE,<br>then invitation<br>was accepted by<br>user. | 1  |

**Table 9. Group model attributes definitions**

### 3.3.5 Template model

#### 3.3.5.1 Diagram

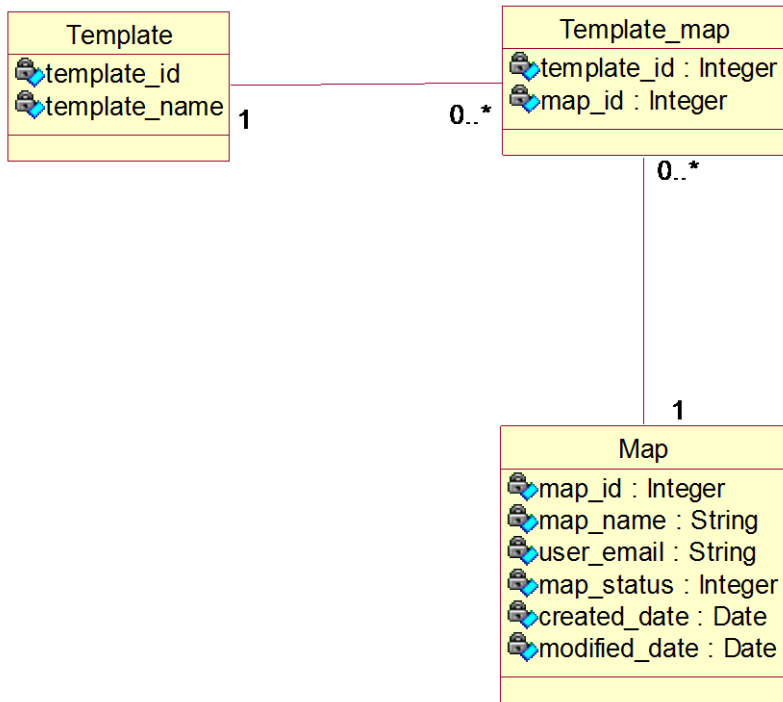


Figure 8. Template model diagram

#### 3.3.5.2 Tables definitions

##### TABLE

##### DEFINITION

| TABLE               | DEFINITION   |
|---------------------|--|
| <b>TEMPLATE</b>     | Template table consists data for template functionality.             |
| <b>TEMPLATE_MAP</b> | Template map table's aim is the connection between Template and Map. |

Table 10. Template model tables definitions

### 3.3.5.3 Attributes definitions

| Tables       | Attribute     | Type         | Definition   | Example             |
|--------------|---------------|--------------|--|---------------------|
| Template     | template_id   | INTEGER      | Template identification number.  | 91                  |
| Template     | template_name | VARCHAR (30) | Template field value represents name of the template. Serves as template identification for users. | ITI0050<br>Template |
| Template_map | template_id   | INTEGER      | Foreign key<br>[Template.template_id]  | 10                  |
| Template_map | map_id        | INTEGER      | Foreign key<br>[Map.map_id]  | 1011                |

Table 11. Template model attributes definitions

## 3.4 The business model canvas

Key partners: as our key partners we can determine schools, universities and other educational systems. Educational systems have to share with our application information about their educational courses, so we could apply them in our application.

Key activities: ideally, application should provide uninterrupted uptime and allow access to its functionality all the time.

Key resources: to hold up our application online for our client or customers we have to maintain servers that we use to hold data and distribute it. For this purpose project needs financial resources. To improve application, to make it more competitive, project needs human resources including application developers.

Value propositions: the application provides tool or service for users to increase their productivity, cost reduction in leaning, memorization or structuration. To be competitive in the market our application should differ from others and we have differences.

Customer relationships: self-service customer relationships is supported by this application as long as we provide our services to clients but we do not interact with them in form of employee-customer relationships.

Channels: to popularize our application will be efficient to use social network ads, as long as this kind of distribution will be the least costly. Also, it is important to make relationships with universities and schools, to expand the boundaries of our application. This kind of distribution will be more costly, than social networking distribution.

Customer segments: as our most important customer we can distinguish people, who take courses in educational institutions. This class has will to learn, structure and memorize information quite often, and, as a result, will use our application more often.

Cost structure: to provide high performance application, which will be hold data for users and retrieve them often, the most costly will be maintenance of the servers. These servers will take more and more load as soon as more and more users will register their selves as our application users and use the application.

Revenue streams: the application is totally free to use and gains profit only from advertisement publishing.

## **4. Prototype**

In this chapter we will consider the application from a practical point of view. The prototype may not reflect all functionality and can have some errors and bugs, but the main point of the prototype is not to provide a finished application, but to provide a partial or a possible implementation of the proposed application. The actual development of fully functional application would spent too much time and resources and cannot be done in terms of this thesis.

To explain the prototype's workflow as much as possible, we will tell about the tools which were used during development first, and then explain how main functionality was implemented in this prototype. Tools descriptions are valuable, because there must not be left any misunderstanding or unsaid. The second chapter can include screenshots of code and prototype itself to provide much complex approach.

To ensure users privacy and security the prototype provides such features like password encryption, profile closure (users' data cannot be provided to third-parties). It would be better to use HTTPS protocol to transfer data between server and user, but in terms of prototype it is not practical.

As the number of users will grow, application must scale and be flexible as much as possible to cover users' needs. As long as Google Cloud SQL is known as very flexible and scalable tool, server, which our prototype uses to transfer data between user and between application and Google MySQL database is not flexible by default. To ensure that our application can hold a high number of users we can increase the number of servers or place our application under Google App Engine. The second approach is better, because App Engine provides progressive scalability, a distributed networks of servers and etc. Currently our prototype does not use App Engine because Google App Engine PHP support is in beta stage now.

### **4.1 Tools**

#### **4.1.1 D3 Library**

As was mentioned before, D3 is the weightiest JavaScript library in this application. D3 library serves to application to visualize data, in our case mind maps data. It can hold data in JSON format, which is very lightweight and easy to use format, and as a result, all its data is processed in D3 library can be easily retrieved from it to store it database.

Important to mention, that every tool is commercially free to use in third-party projects.

#### **4.1.2 JavaScript and JQuery library.**

JavaScript is a programming language that is used by web browsers to interact with the browser to provide static Web page dynamics. It can perform a variety of tasks. JavaScript manipulates Document Object Module to edit page visualization, make Asynchronous JavaScript and XML requests to exchange data with servers, respond to events and fire a custom event or disable/enable them and others.

JQuery library is the most popular JavaScript library, which extends and enhance JavaScript's functionality. As long as users still use old versions of different browser clients like Internet Explorer 8/9, cross-browser support of JavaScript is important and this support JQuery provides successfully.

This prototype uses JavaScript and JQuery to check data input (server-side also checks data input), manipulate events, actions and send AJAX requests to perform basic actions with a database.

#### **4.1.3 Quill JavaScript library**

As was mentioned before, Quill is a WYSIWYG editor that uses JavaScript as its programming language. Like JQuery, Quill has cross-browser support. Also, it support HTML standards, which is important, that it will not make a collision with our HTML code.

#### **4.1.4 Bootstrap framework**

Bootstrap provides tools, with the help of which we can construct a web page without need to make it by ourselves. Often, developers code the same basic elements in HTML and CSS from project to project, but this framework provides these elements, including such elements like: accordion, grid system, dropdowns, button styles and etc.

The prototype uses the full version of bootstrap framework, but it can be shrunk to include only these elements and tools, that we need in our prototype. This operation is usually done when the application is released into production.

#### **4.1.5 LESS, a CSS pre-processor**



LESS is very powerful tool, to write CSS more effectively. LESS is a CSS pre-processor, which can use variables, functions, configurations, mixins, loops and etc. This pre-processor compiles using JavaScript. LESS files can be identified by extension - “.less”.

## 4.2 Functionality

### 4.2.1 Overview

In this chapter, we will start from the beginning of the prototype – user registration and authentication. After this operation, we will create a new mind map that we could use later to create nodes, customize them and add content.

Here is a screenshot that shows the index page, in other words the first page of the prototype.

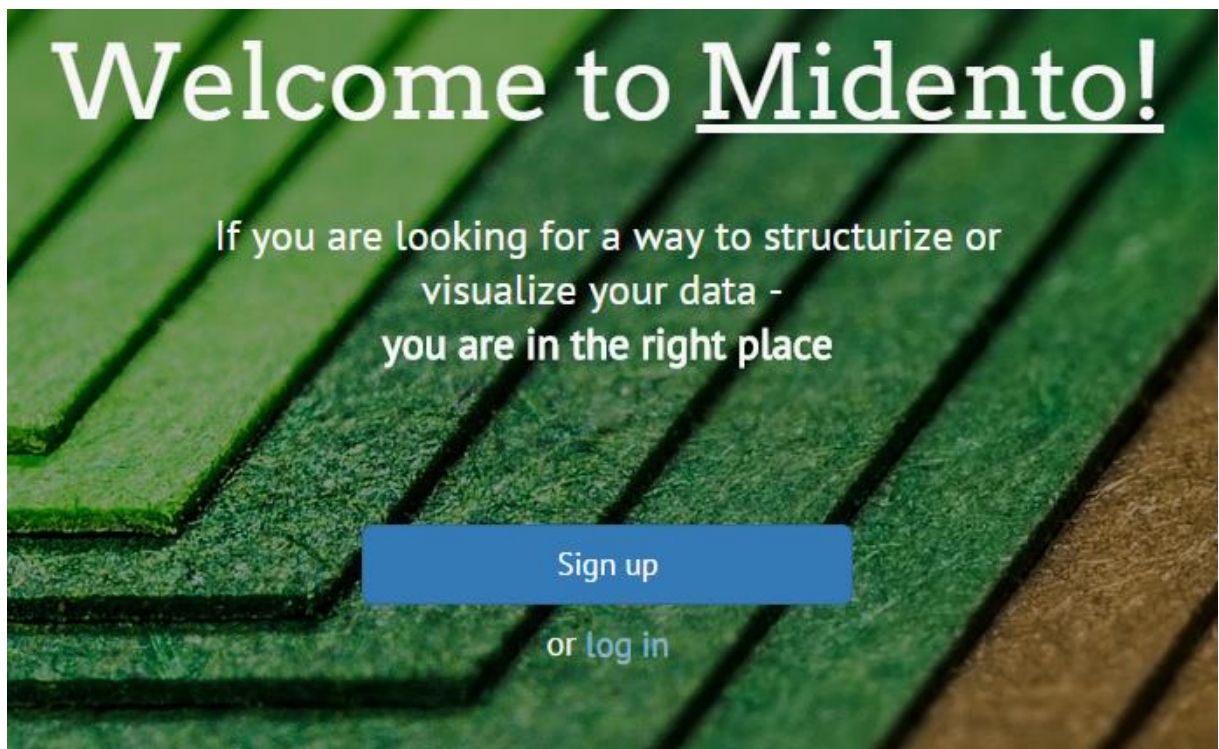


Figure 9. Index page

On this moment prototype does not provide authentication and registration via Facebook or VKontakte APIs.

### 4.2.2 Basic architecture

#### 4.2.2.1 Frontend

Frontend is responsible for everything that is connected with the client-side representation. Prototype is divided for index, main and dashboard pages. The index page is responsible for user authentication functionality. The dashboard page contains all functionality related to mind maps, groups, templates and user profile. And, finally, the main page represents each map separately.

Every page has its own JavaScript file, which controls everything on certain page, including events, AJAX requests and etc. As well, there is a JavaScript file that has functions which can be used everywhere in prototype and named “functions.js”. This file is a common part of JavaScript in this prototype.

#### 4.2.2.2 Backend

Backend is responsible for everything that is connected with the server-side representation. The backend uses a stack of PHP, MySQL technologies. The PHP part contains 3 classes: AjaxHandler, DBHandler and DAO.

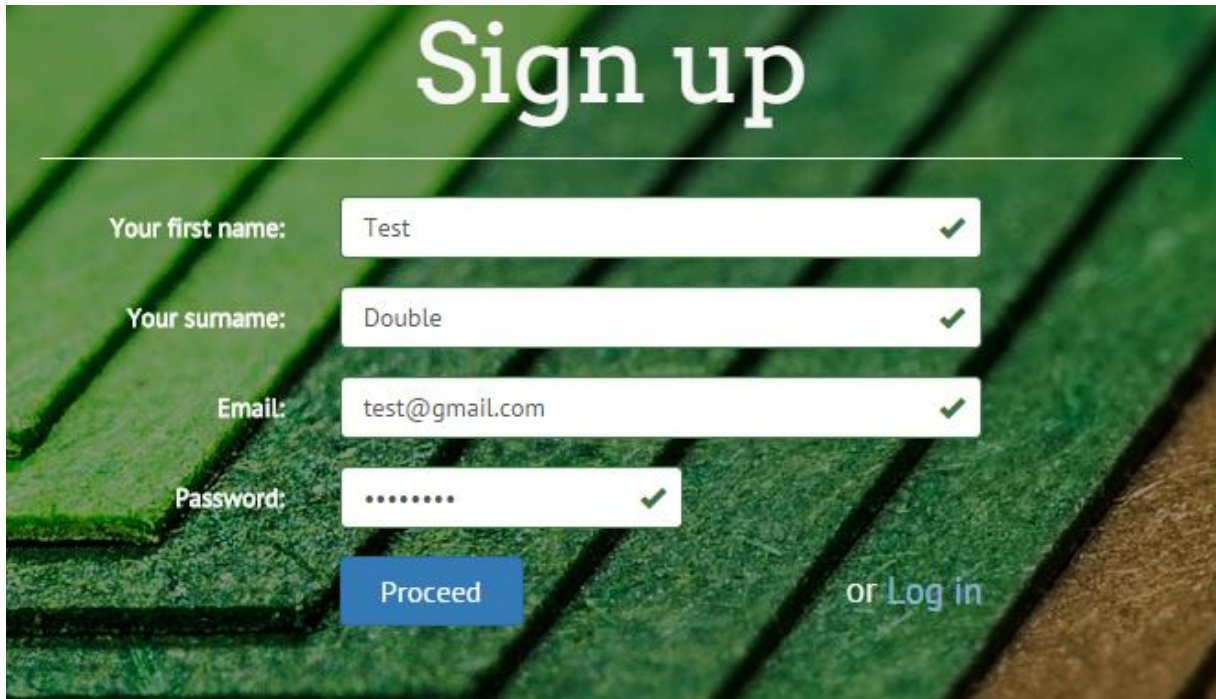
- **AjaxHandler** is responsible for AJAX requests, which come from client-side.
- DAO class is in tight connection with **DBHandler**, as long as all methods of DBHandler are executed in DAO. DBHandler task is to gather post data and check it for correction.
- **DAO** is the class which task is to make queries to database.

AjaxHandler by default returns JSON string, which is converted in JSON object in JavaScript. This JSON object always has “error” parameter with a Boolean type value. If the value of the “error” parameter equals to “False”, then usually additional parameters are set only in case if data is requested from a database. If the value of the “error” parameter equals to “True”, then additional parameters are required, to explain what exactly went wrong.

DAO uses PHP Data Object (PDO) to communicate with the database. PHP Data Object provides service, which has many features to make communication with database secure and robust. One of these features is transactions and the DAO class uses it when we need to make several SQL queries.

#### 4.2.3 User registration

To give an access to the prototype functionality subject must register in the system first. To do that, the user has to navigate to “Create new account” section. The create account section suggests the user to fill the registration form. Let’s fill the form with test data.



The image shows a registration form titled "Sign up" on a green background. The form contains four input fields, each with a green checkmark to its right, indicating successful validation:

- Your first name: Test
- Your surname: Double
- Email: test@gmail.com
- Password: [masked]

Below the fields is a blue "Proceed" button and a link that says "or Log in".

Figure 10. Registration form

Now we can proceed with the registration.

1. If data is okay and user wants to proceed with the registration, then user clicks button “Proceed”.
2. Data from form’s input fields is fetched with JavaScript and JQuery. In the same time as user type, JavaScript checks if value in currently selected input is valid.
3. JavaScript makes an AJAX request to the server with request to create new user with data, that user provided. This operation is made by `sign_up_user()` method in DAO class.

```

public function sign_up_new_user() {
    $msg = $this->handler->validate_sign_up();

    if ($msg['error'] === false) {
        $post = $msg['post_data'];

        extract($post);

        $avatar_src = 'images/avatar.png';

        if ($form_token !== $_SESSION['form_token']) {
            echo array('error'=>true, 'error_msg'=>'Invalid form submission');
        }

        $cost = 10;
        $salt = strtr(base64_encode(mcrypt_create_iv(16, MCRYPT_DEV_URANDOM)), '+', '.');
        $salt = sprintf("$2a$%02d$", $cost) . $salt;

        $hash = crypt($password, $salt);

        try {
            $sql = 'INSERT INTO User (firstname, surname, user_email, user_password, salt, avatar_src)';
            $sql .= ' VALUES (:firstname, :surname, :email, :password, :salt, :avatar_src)';

            $stmt = $this->pdo->prepare($sql);

            $stmt->bindParam(':firstname', $firstname);
            $stmt->bindParam(':surname', $surname);
            $stmt->bindParam(':email', $email);
            $stmt->bindParam(':password', $hash);
            $stmt->bindParam(':salt', $salt);
            $stmt->bindParam(':avatar_src', $avatar_src);

            $stmt->execute();

            unset($_SESSION['form_token']);

            return array('error'=>false);

        } catch (Exception $e) {
            if ($e->getCode() == 23000) {
                $error = array('error'=> true, 'error_msg'=>'User is already exists');
            } else {
                $error = array(
                    'error' => true,
                    'error_msg' => 'We are unable to process your request. Please, try again later'
                );
            }

            return $error;
        }
    } else {
        return $msg;
    }
}

```

Figure 11. Sign up user code snippet

Registration form data is fetched in DBHandler class using validate\_sign\_up() method.

During user sign up operation in PHP, server returns an array which always has key named

“error”. This key shows, if an error, during the operation execution, was thrown or not. If an error was thrown, it has Boolean value “true” and additional keys in array will be provided, which help us to know, what exactly went wrong.

4. If sign up was successful, then user is redirected to login page.

#### 4.2.4 User authentication

As was said earlier, prototype does not include authentication via social networks currently and therefore the user can login with email only. To do that, the user has to fill login form in the authentication section on the index page. Let’s fill the form with the created earlier account.



Figure 12. Authentication form

Now we can proceed with login:

1. If data is okay and user wants to proceed with the authentication, then user clicks button “Login”.
2. Data from form’s input fields is fetched with JavaScript and JQuery. In the same time as user type, JavaScript checks if value in currently selected input is valid.
3. JavaScript makes an AJAX request to the server with request to create new user with data, that user provided. This operation is made by `sign_in_user()` method in DAO class.

```

public function sign_in_user() {
    $msg = $this->handler->validate_sign_in();

    if ($msg['error'] === true) {
        return $msg;
    }

    $post = $msg['post_data'];
    extract($post);

    $sql = 'SELECT firstname, surname, user_email, user_password, avatar_src FROM User WHERE user_email = :user_email';
    $stmt = $this->pdo->prepare($sql);
    $stmt->bindParam(':user_email', $email);
    $stmt->execute();
    $user = $stmt->fetch(PDO::FETCH_OBJ);

    if ( $user == false || !hash_equals($user->user_password, crypt($password, $user->user_password)) ) {
        return array('error'=> true, 'error_msg' => 'No such an user or password is wrong');
    } else {
        $_SESSION['user_email'] = $user->user_email;
        $_SESSION['firstname'] = $user->firstname;
        $_SESSION['surname'] = $user->surname;
        $_SESSION['avatar_src'] = $user->avatar_src;

        return array('error' => false);
    }
}
}

```

Figure 13. User authentication code snippet

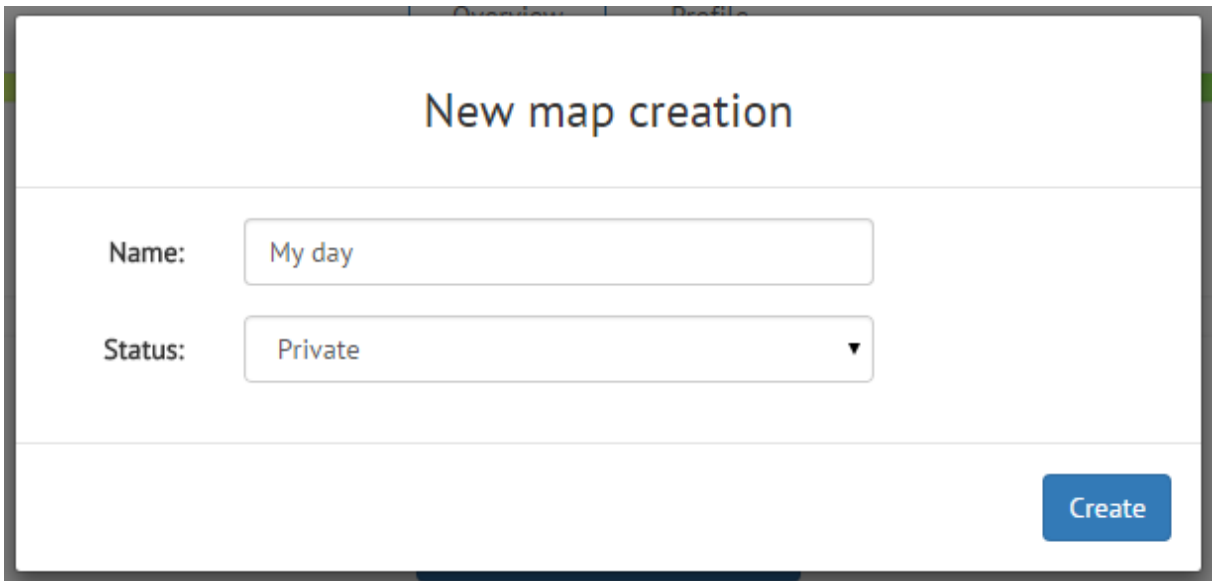
Login form data is fetched in DBHandler class using validate\_sign\_in() method. During user sign in operation in PHP, server returns array which always has key named “error”.

4. If authentication was successful, then user is redirected to dashboard page.

#### 4.2.5 Mind map creation

User is logged in now and by default their mind maps list is empty and the application will print message „You have not created a single map“. The user has to create new mind maps first. To do that, the user just has to click button “Click here to make one” or if the user’s mind maps list is not empty, user has to click button “New map”. Pop-up form will be shown to the user.





New map creation

Name:

Status:

Create

Figure 14. Map creation form

Now we can proceed with map creation:

1. User fills the form and clicks on “Create” button.
2. Data from form’s input fields is fetched with JavaScript and JQuery.
3. JavaScript makes an AJAX request to the server with request to create a new mind map, with user provided data. This operation is made by insert\_new\_map() method in the DAO class. The operation also inserts a new node with the lowest level. This node is the main node and cannot be deleted from the map.
4. If data is ok, then page reloads and new map is listed in the mind maps list.

#### 4.2.6 Mind map deletion

If user does not want to use mind map in future he can delete the map. For this task he can click on trash icon which is on the right side of the map list element. Pop-up will appear that will ask confirmation for this task. Now we can proceed with map deletion:

1. User clicks the accept button.
2. Data from form’s input fields is fetched with JavaScript and JQuery.
3. JavaScript makes an AJAX request to the server with request to delete map. AJAX request also contains an identification number of the map.

4. If request is done successfully, then map will be deleted from the list.

#### **4.2.7 Mind map visualization**

We created map and now it is time to view it. To do this task user has to choose map which want to visualize/edit. To do this action user has to click on edit icon in mind maps listing which is on the right side of the map list element. User will be redirected to main page.

Now map can be visualized:

1. JavaScript makes an AJAX request to the server with request to gather data related to the chosen map.
2. If map does exist, then AJAX request will return nodes data, nodes links and nodes configuration. In other way, if map does not exist, then AJAX request will return error message and JavaScript will redirect user to dashboard page.
3. JavaScript will process AJAX response and retranslate it to D3 compatible links array and nodes object. Configuration file will be also fetched from AJAX response, but it will be used in free format. This operation is made by initializeMapSettings() function.
4. It's time to visualize the map. Function initializeMap() will configure start svg settings, add links group element, nodes group element and initialize layout type. Further, function start() which will add nodes and links to groups created earlier. Start() function uses D3 to do this task.



```

function initializeMap() {
  svgWidth = $(window).width()
  svgHeight = $(window).height() * 0.93;

  d3.select('body').on('keydown', function() {
    if (d3.event.keyCode == 17) {
      ctrlPressed = true;
    }
  })
  .on('keyup', function(event) {
    if (d3.event.keyCode == 17) {
      ctrlPressed = false;
    }
  });

  for (var i = 0; i < nodes.length; i++) {
    if (nodes[i].level == 1) {
      nodes[i].x = svgWidth / 2;
      nodes[i].y = svgHeight / 2;
      nodes[i].fixed = true;
    }
  }

  var svgContainer = d3.select('body')
    .select('#main-svg')
    .call(d3.behavior.zoom().scaleExtent([-1, 2]).on("zoom", zoom));

  var glinks = svgContainer.append('svg:svg')
    .attr('id', 'nodes-svg')
    .attr('width', svgWidth)
    .attr('height', svgHeight);

  var linesContainer = d3.select('#nodes-svg')
    .append('g')
    .attr('class', 'links');

  var nodesContainer = d3.select('#nodes-svg')
    .append('g')
    .attr('class', 'nodes');

  force = d3.layout.force()
    .size([svgHeight - 50, svgWidth - 100])
    //.nodes(nodes);
    //.links(links)
    .on('tick', tick);

  //force.linkDistance(175);
  force.linkDistance(function(d) {
    var level = checkLevelConf(d.source.level);
    var radius = parseInt(nodesConf[level].radius);
    var distance = radius * 2;
    return distance;
  });
  force.linkStrength(1);
  force.gravity(0);
  force.charge(-360);

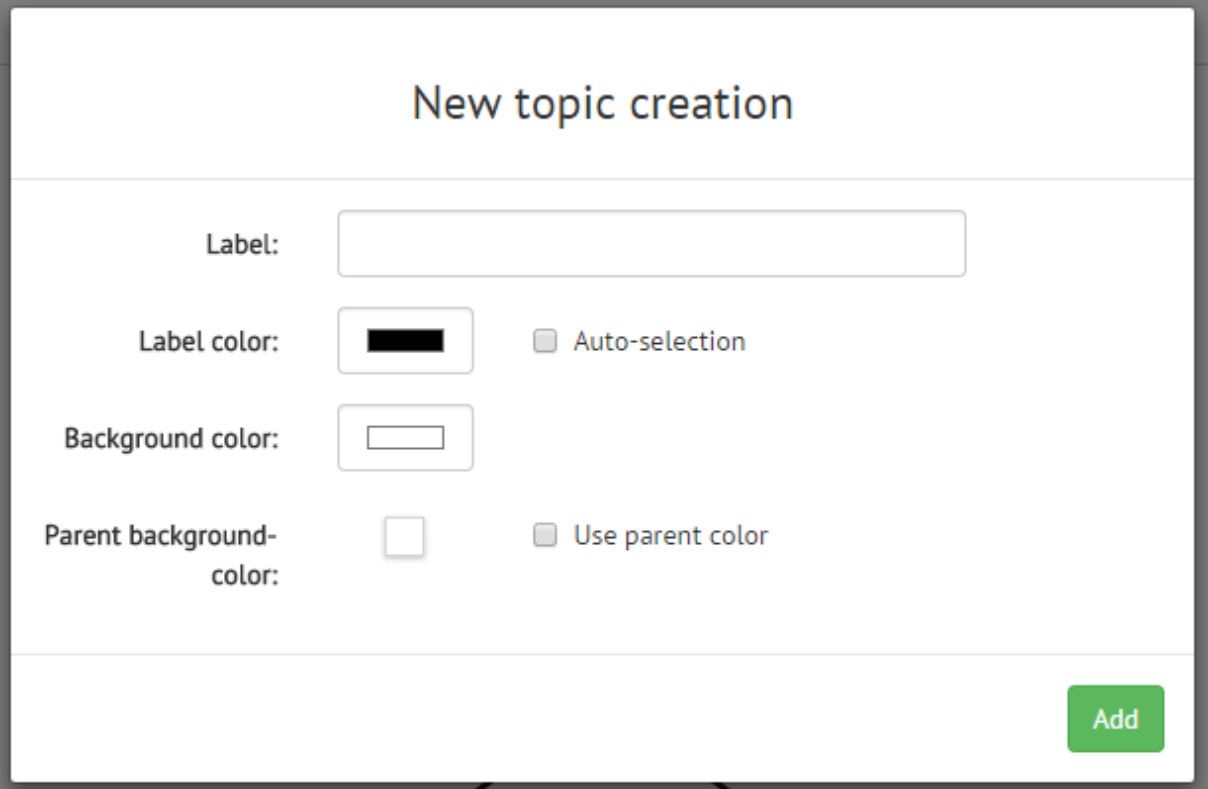
  start();
}

```

Figure 15. Map initialization code snippet

## 4.2.8 Map node creation

Map is visualized and new nodes can be created. To create new node user has to choose „New topic“ in dropdown with name „Actions“. New node will be linked to the one, that is currently chosen as active. If no node is chosen, then alert to user will be thrown. Let's consider that node is chosen an user chose „New topic“ in dropdown list. New pop-up node creation form window will be shown to user.



The image shows a web form titled "New topic creation". It contains the following elements:

- Label:** A text input field.
- Label color:** A color picker showing black, with an unchecked checkbox labeled "Auto-selection".
- Background color:** A color picker showing white.
- Parent background-color:** A color picker showing white, with an unchecked checkbox labeled "Use parent color".
- Add:** A green button located at the bottom right of the form.

Figure 16. Node creation form

Now we can proceed with new map node creation:

1. User fills the form and clicks on „Add“ button.
2. Data from form's input fields is fetched with JavaScript and JQuery.
3. JavaScript makes an AJAX request to the server with request to create new node and link it to certain mind map. Mind map identification number, new node data, parent node identification number will be provided with AJAX request.
4. If no error was thrown during execution, new node adds to the map. Map revisualize.

### 4.2.9 Node data update

We can edit created node if we have such necessity. To do this action user has to choose “Edit node” in dropdown which is named “Actions”. If no node is chosen, then alert to user will be thrown. Let’s consider that node is chosen an user chose „Edit topic“ in dropdown list. New pop-up node edit form window will be shown to user.

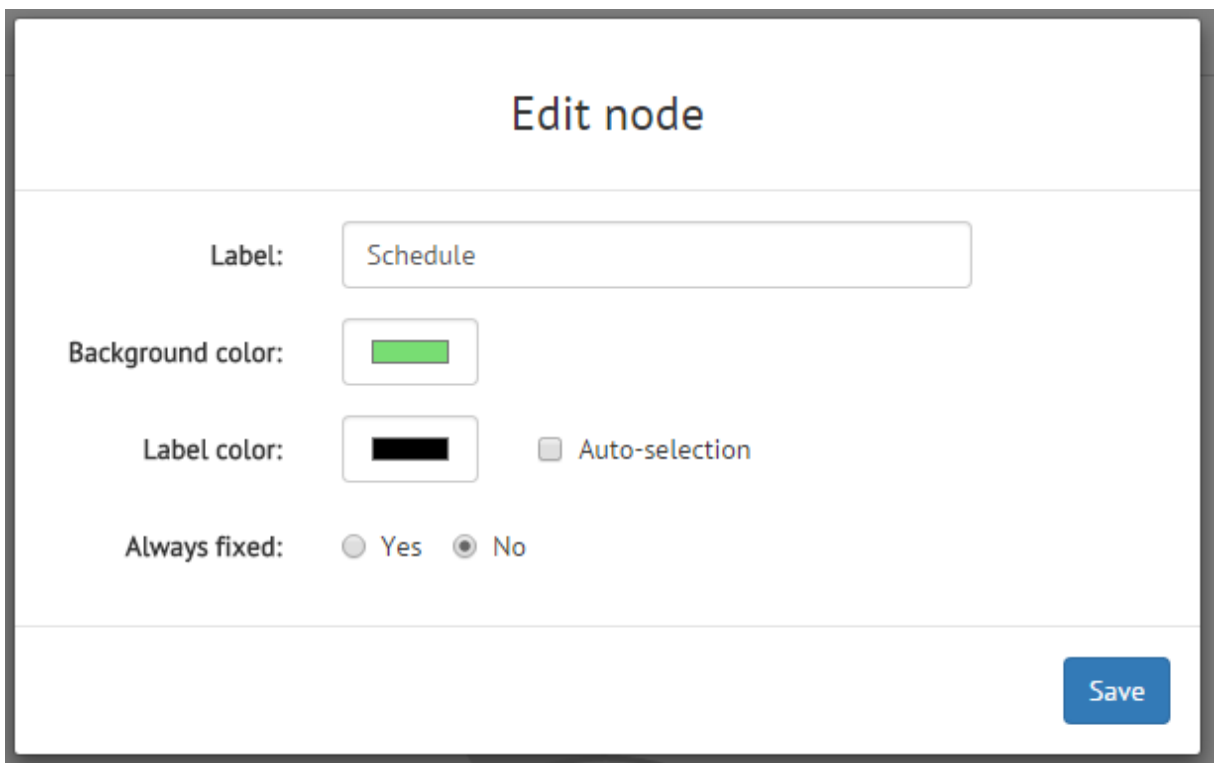


Figure 17. Node edit form

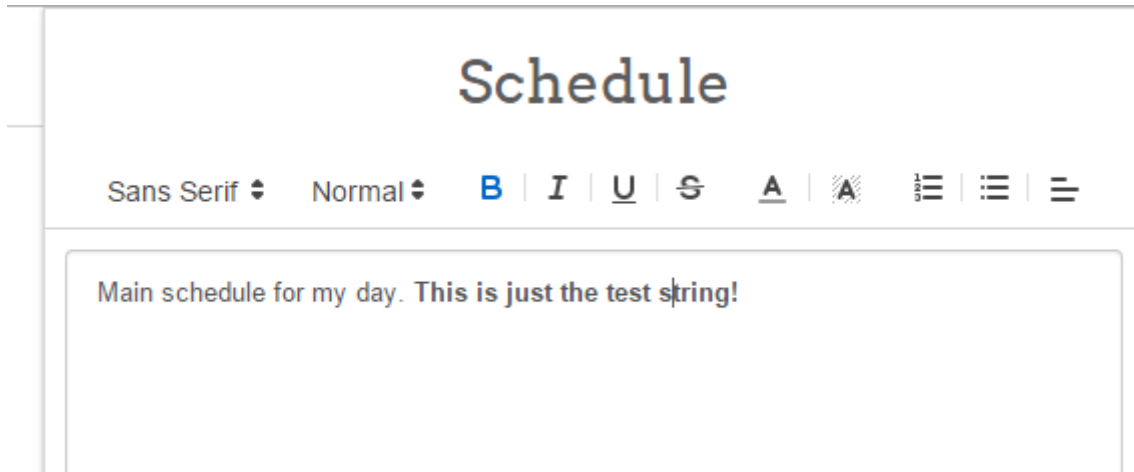
Now we can proceed with node edit:

1. User fills the form and clicks on „Save“ button.
2. Data from form’s input fields is fetched with JavaScript and JQuery.
3. JavaScript makes an AJAX request to the server with request to edit existing node.  
Mind map identification number, new node data will be provided with AJAX request.
4. If no error was thrown during execution, new node adds to the map. Map revisualize.

### 4.2.10 Node content addition/edit

When web is loaded, new Quill instance is instantiated. As we mentioned before Quill is a text editor and prototype uses it to create and edit node’s content. Right side of the page has additional window, which is responsible for the node’s content. To open this window user can

double click on node, which he wants to edit, or to press the open button on the right side of the page.



**Figure 18. Node content addition/edit window**

Now we can proceed with content addition:

1. User fills the input with the content and format it as he likes and clicks “Save” button.
2. Data from form’s input fields is fetched with JavaScript and JQuery.
3. JavaScript makes an AJAX request to the server with request to edit existing node’s content which value is “null” by default. Mind map new node data will be provided with AJAX request.
4. If no error was thrown during execution, content is updated. User can continue to work with the application.

#### **4.2.11 Map node deletion**

User may need to remove nodes from the mind map. The prototype allows to do this action, but it deletes also children nodes. This approach to the task helps user to save time and not to delete every node separately. To do the deletion of node, user has to choose node from which he wants to start deletion chain and then click on “Delete topic chain” in the dropdown list called “Actions”.

If node has no children nodes at all, then the only selected node will be deleted.

Now we can proceed with nodes deletion:

1. User clicks on “Delete topic chain” list element.
2. JavaScript makes an AJAX request to the server with request to delete. Before AJAX transmission we have to find all deletion node identification numbers with their links and send them with AJAX POST data. This action is done by delete\_nodes() function.

```
do {
  var nodeFound = false;
  links.forEach(function(l) {
    if ($.inArray(l.source, nodesToDelete) != -1 && $.inArray(l, newLinks) == -1) {
      nodesToDelete.push(l.target);
      linksToDelete.push(l);
      var nodeFound = true;
    } else if ($.inArray(l.target, nodesToDelete) != -1) {
      linksToDelete.push(l);
    } else {
      newLinks.push(l);
    }
  });
} while (nodeFound);
```

Figure 19. Node sort code snippet

3. If nodes exist in database, then we have delete them and all links related to these nodes. In other way we just have to return an error to the user.

```

public function delete_nodes() {
    $msg = $this->handler->validate_nodes_to_delete();

    if ($msg['error'] === true) {
        return $msg;
    }

    $this->pdo->beginTransaction();

    extract($msg['post_data']);
    unset($msg['post_data']);

    foreach($nodes as $key=>$value) {
        $id = $value;

        $sql = "DELETE FROM Node WHERE node_id = :node_id";
        $stmt = $this->pdo->prepare($sql);
        $stmt->bindParam(':node_id', $id);
        $result = $stmt->execute();

        if (!$result) {
            $msg['error'] = true;
        }
    }

    foreach($links as $link) {
        $src = $link['source'];
        $trg = $link['target'];

        $sql = "DELETE FROM Node_connection WHERE parent_node_id = :parent_node_id AND child_node_id = :child_node_id";
        $stmt = $this->pdo->prepare($sql);
        $stmt->bindParam('parent_node_id', $src);
        $stmt->bindParam('child_node_id', $trg);
        $result = $stmt->execute();

        if (!$result) {
            $msg['error'] = true;
        }
    }

    $this->map_modified($map_id);

    if ($msg['error'] === true) {
        $this->pdo->rollBack();
    } else {
        $this->pdo->commit();
    }

    return $msg;
}

```

Figure 20. Nodes deletion code snippet

4. If no error was thrown during execution, new node adds to the map. Map revisualize.

### 4.3 Mind map example

To show full and final result how mind map can be represented to user in terms of this prototype I made an example. The example shows how can be application used to make mind map on energy subject.

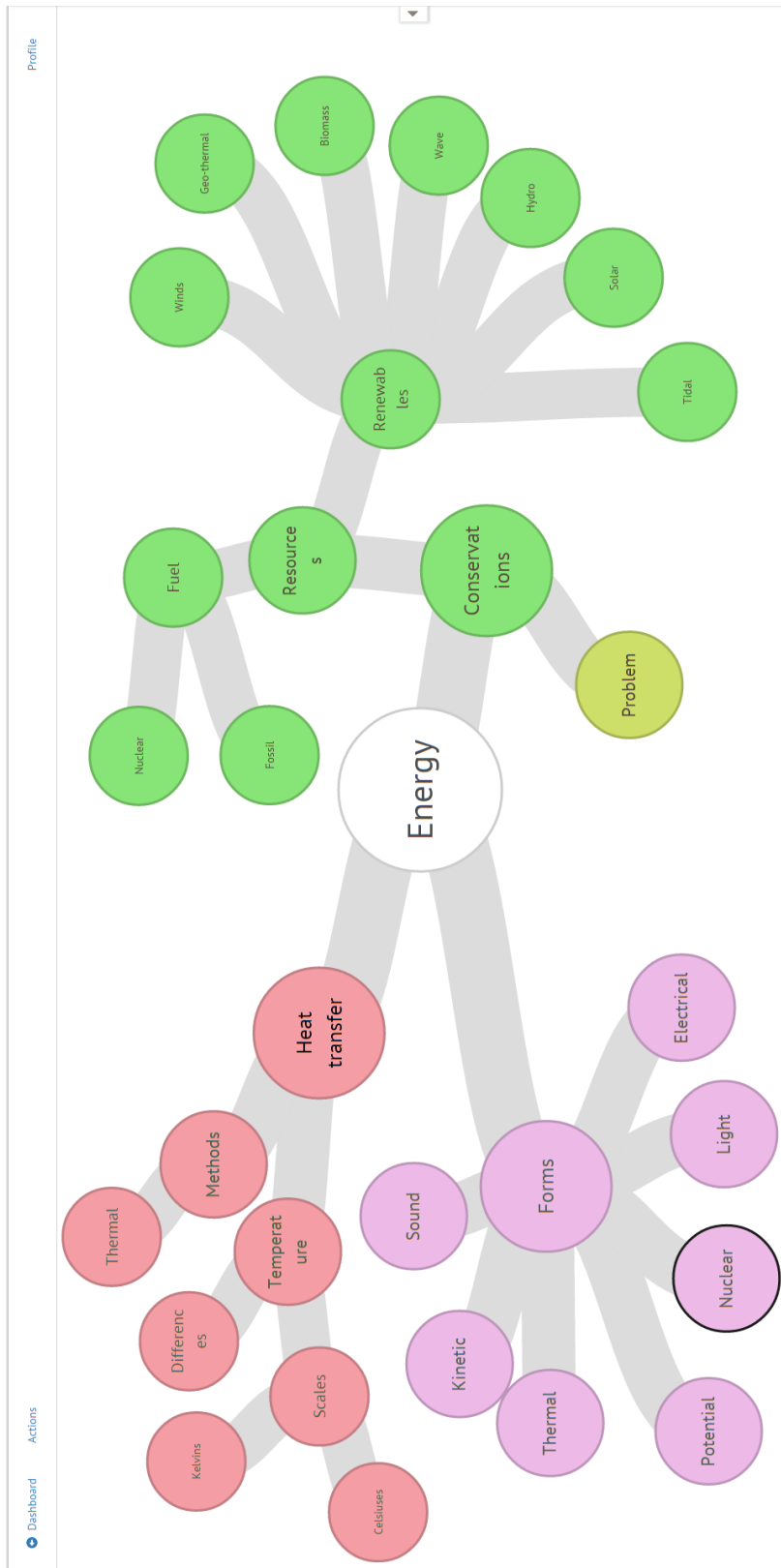


Figure 21. Full mind map example

## Kokkuvõte

Projekti peamine eesmärk on pakkuda igale interneti kasutajale kasutajasõbralikku rakendust, mida oleks lihtne kasutada, kuna see omab võimast funktsionaalsust ja mis täielikult kajastab meeles kaartide ideid.

Veelüks oluline eesmärk on pakkuda suure jõudlusega rakendust, millel on kiire reaktsiooniaeg igale tegevusele mis nõuab andmebaasiga ühendust.

Mõned määratud ülesanded aga jäid tegemata. Rakendus pakub vähemalt põhilist meeles kaartide funktsionaalsust koos kasutaja autentimisega.

Projektil töötamise käigus tekkis selge arusaam, et Google vahendid on tugevamad, tõhusamad ja palju lihtsamad, kui näiteks MySQL andmebaasi kasutamine riigi kohalikul serveril (ka muud teenused tõenäoliselt omavad neid omadusi, kuid ma kasutasin Google MySQL Tools-e). On võimalik, et väiksemate rakenduste puhul oleks selle kasutamine kallim, kuid see on tõhusam keskmiste ja suurte rakenduste puhul. Need olid rakenduse plussid, kuid on vaja mainida, et on olemas ka miinused. Google tööriistade negatiivne külg seisneb selles, et nad ei toeta kõiki tehnoloogiaid. Tehnoloogiate hulk varieerub sõltuvalt teenusest, kuid nad alati toetavad populaarsemaid. Näiteks Google App Engine PHP programmeerimiskeel on hetkel väljatöötamisel, kuid beta versioon on juba kättesaadav.

Antud teema on väga lai. Meeles kaardid on tõhus vahend ning kuigi nende funktsionaalsus ei ole esmatähtis, oleks hea seda rakendusse integreerida. Isegi kui meie ei arvesta meeles kaartide funktsionaalsust, kuid vaatleme maailma võrgu tendentse, siis on selge, et tuleb veel palju tööd teha. Isegi kui meie ei arvesta maailma võrgu tendentse, siis saame pakkuda sellist funktsionaalsust nagu: integratsiooni ülikoolidesse ja koolidesse. See meelitab teadlasi ja õpilasi kasutama rakendust oma igapäevases õpperutiinis.

Samuti oleks väga kasulik teha rakenduse esmakordse kasutamise koolitus uutele kasutajatele. See koolitus tutvustaks lihtsamal viisil rakenduse kasutamise esmased sammud ja ka meeles kaartide ideid, et kasutajad tunneksid ennast palju mugavamalt.



## Summary

The main aim of the project is to provide to every internet available user a user-friendly application, that would be very easy to use as it possesses powerful functionality and which would fully reflect mind map ideas in it.

Another important aim is to provide a high performance application, with high response time to every action that would require a database connection.

Not every task which was defined earlier was done. The application provides at least the basic mind map functionality with user authentication.

While I was working on this project a clear understanding came out, that Google tools are more robust, effective and much simpler instruments, than, for example, a country's local server in case of MySQL database usage (other services are likely to have these characteristics, but I used Google MySQL Tools). Possibly, it is more expensive if your application is small, but it is more efficient on medium and big applications. These are the pros, but I should also mention that there are cons as well. The negative side of Google tools is that they do not support all technologies in their services. The stack of technologies varies from service to service, but they always support popular technologies. For example, Google App Engine PHP programming language support is currently under development, but the beta version is already available.

This subject is very wide. Mind maps are efficient tools and even though their functionality is not primary, it would be good to integrate it with applications. But even if we will not take into account mind map functionality, but will take a look at world web tendencies – there is much work to be done. And even if we will not take into account world web tendencies, we can provide such functionality as: Integration with universities and schools. It will attract scholars and student to use the application in their everyday learning routine.

It would also be very useful to make a tutorial for first time users. This tutorial would make easy first steps into the application and mind maps ideas, so they will feel more comfortable.

## References

- 
- [2] Mindmup application [WWW] – <https://www.mindmup.com/#m:new>
  - [3] D3.js [WWW] – <http://d3js.org/>
  - [4] D3 layout documentation [WWW] – <https://github.com/mbostock/d3/wiki/Layouts>
  - [5] Google [WWW] – <http://google.com>
  - [6] Google Search [WWW] – [http://en.wikipedia.org/wiki/Google\\_Search](http://en.wikipedia.org/wiki/Google_Search)
  - [7] Google Cloud Platform [WWW] – <https://cloud.google.com/>
  - [8] Google Cloud SQL [WWW] – <https://cloud.google.com/sql/docs/introduction>
  - [9] Number of monthly active Facebook users worldwide as of 1<sup>st</sup> quarter 2015 [WWW] <http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/>
  - [10] Facebook's Mark Zuckerberg named world's youngest billionaire [WWW] <http://www.computerworlduk.com/news/it-vendors/facebooks-mark-zuckerberg-named-worlds-youngest-billionaire-3480673/>
  - [11] The Facebook Platform [WWW] – <https://developers.facebook.com/>
  - [12] VKontakte Open API [WWW] – [https://vk.com/pages?oid=-17680044&p=Open\\_API](https://vk.com/pages?oid=-17680044&p=Open_API)
  - [13] VKontakte API Usage [WWW] <https://vk.com/dev/apiusage>
  - [14] Quill JavaScript library [WWW] – <http://quilljs.com/docs/quickstart/>