Final Report: MapShare

1. Introduction

This website and application offer easy way to save your travel history and share it with you friends.

Main goals of the project:

- ☑ create a permanent, but easily maintained database of visited countries and a corresponding wish-list;
- ☑ provide easy way to create and share beautiful-looking custom visual maps based on available data
- ☑ provide infographic statistics related to travel history and other users on the project.

The main value of the project is to give users opportunity to save and share their travel history in a good-looking way.

2. Design and Implementation

From Design to Realization

When I made sketches and UI prototype I had started to make an HTML layout and CSS styles for different views. I'd chose Bootstrap CSS framework and Material Design for Bootstrap to implement mobile app look to my web application and to force development on standard Google styles. It supports responsive grid up to large desktop screens, thanks to Bootstrap 4.

Then I'd started to implement views and corresponding controllers to Angular framework project. I'd tried to follow ***John Papa*** *Angular style guides* when scaffolding an app. First, I'd tried to minimise using of *$scope* and used a construction

```
controller  : 'HeaderController',
controllerAs: 'vm'
```

in ***ui-router*** and

```
var vm = this;
vm.saveMap = saveMap;
vm.copyLink = copyLink;
function copyLink(url) {
      …
}
```

in controllers to clearly understand what this is referred to. So, that I can use functions in views like this:

```
ng-click="vm.copyLink(url)"
```

In parallel, I'd made a ***Gulp*** automation environment and write two different scripts: for live preview with *BrowserSync* and for publishing to production:

```
gulp.task('publish', ['copy','uglify', 'sass', 'css', 'html']);
gulp.task('default', [‘serve']);
```

I'd tried to escape function callbacks in Controller definitions and used this syntax to make easy code understanding and reading:

```
angular
      .module('mapshare')
      .controller('MainCtrl', MainCtrl)
      .controller('OtherCtrl', OtherCtrl)
      …;

/* @ngInject */
function MainCtrl($rootScope, $state) {
      //some code
}
```

Then I'd tried to implement *multi-language User Interface*. I contrive stand-alone ***JSON*** files with language translations to every interface element. The structure of language file is flat. Except from *english* version:

```
{
"error" : "something goes wrong",
"btn_close": "close"
…
}
```

From *russian* language:

```
{
"error" : "что-то пошло не так",
"btn_close": "закрыть"
…
}
```

This realisation makes extension very easy. Just put translated file to directory and it will support new language. User language stored in user settings in database and browser *localStorage*. When user open website or app, it's loading values to Javascript object and putting it in $rootScope variable, that accessible from all parts of application. Then I can use it in Views in Angular expressions like

```
{{ ::init.messages.btn_share }}
```

or in Controllers as

```
$rootScope.header = $rootScope.init.messages.menu_countries;
```

Then I'd started to implement ***REST API*** and DB document format. After several attempts, I'd stopped on two paths: ***countries*** (list of countries) and ***users*** (everything, that related to user, like settings, visited countries, wishlist and generated maps).

Example of REST API JSON format for *countries*:

```
{
  "continents": [
    {
```

```
      "id": "1",
      "name": "Africa",
      "countries": [
        {
          "id": "EG",
          "name": "Egypt"
        },
        {
          "id": "TN",
          "name": "Tunisia"
        }
      ]
    }
…
}
```

Users example:

```
"users": [
    {
      "id": "s0MqEGKCLh",
      "email": "test@test.com",
      "name": "Test Man",
      "home_country": "RU",
      "language": "RU",
      "password": "",
      "lastLogin": "2016-05-20T17:57:28.556094Z",
      "visited": [
        "BG",
        "CA",
        "AU",
        "US",
        "BR",
        "EG",
        "JP"
      ],
      "wishlist": [
        "AL",
        "NZ"
      ],
      "favs": [
        "BG"
      ],
      "feedback": [
        {
          "name": "Dmitry Markov",
          "email": "test@test.com",
          "textarea": "eqweqweqw"
        }
      ],
      "maps": [
              {
```

```
            "id": "1RtwCKCASbNnG9",
            "date": "2016-06-09T11:17:16.364Z",
            "countries": [
              "BG",
              "CA",
              "AU",
              "US"
            ],
            "wishlist": [
              "AL",
              "NZ",
              "JP"
            ],
            "styleId": 2
          }
    ]
}
```


The design and show of created map produced by ***Highmaps.js*** - a *jQuery* plugin with great customisation abilities. It supports different themes, so I'd made some for example. Lately it will be more elaborated from the design perspective.

I am planning to write my own engine to work with *SVG* maps, but for the development speed **I'd chose** 3rd-party solution for now.

The difficulties with *Statistics* faction of app is concerning about caching values corresponding to selected stats. For example, when the user base become huge, it will be inefficient to made a query to database to select number of users, so, I'm considering to cache some figures.

Server-side implemented with ***Node Express Generator***. Authorisation made with ***JWT*** and ***Passport***.

Modules and Libraries

jQuery - necessary for Bootstrap and Highmaps.js

Bootstrap 4 - I had switched to version 4 because of support for Card component, better grid support and MDB framework support

Material Design for Bootstrap - framework to make bootstrap classes looks like Google Material Design. I prefer this project to Angular Material and Materialize.css because of Bootstrap support

HighMaps.js - a library to create SVG maps with flexible customisation

Easy Pie Chart - Angular module to make a donut infographics

ngStorage - Angular module to implement localStorage functionality in the *angular way*

ui-router, ngSanitise, ngResource - additional modules to scaffold Angular app

Screenshots of web application and hybrid mobile app

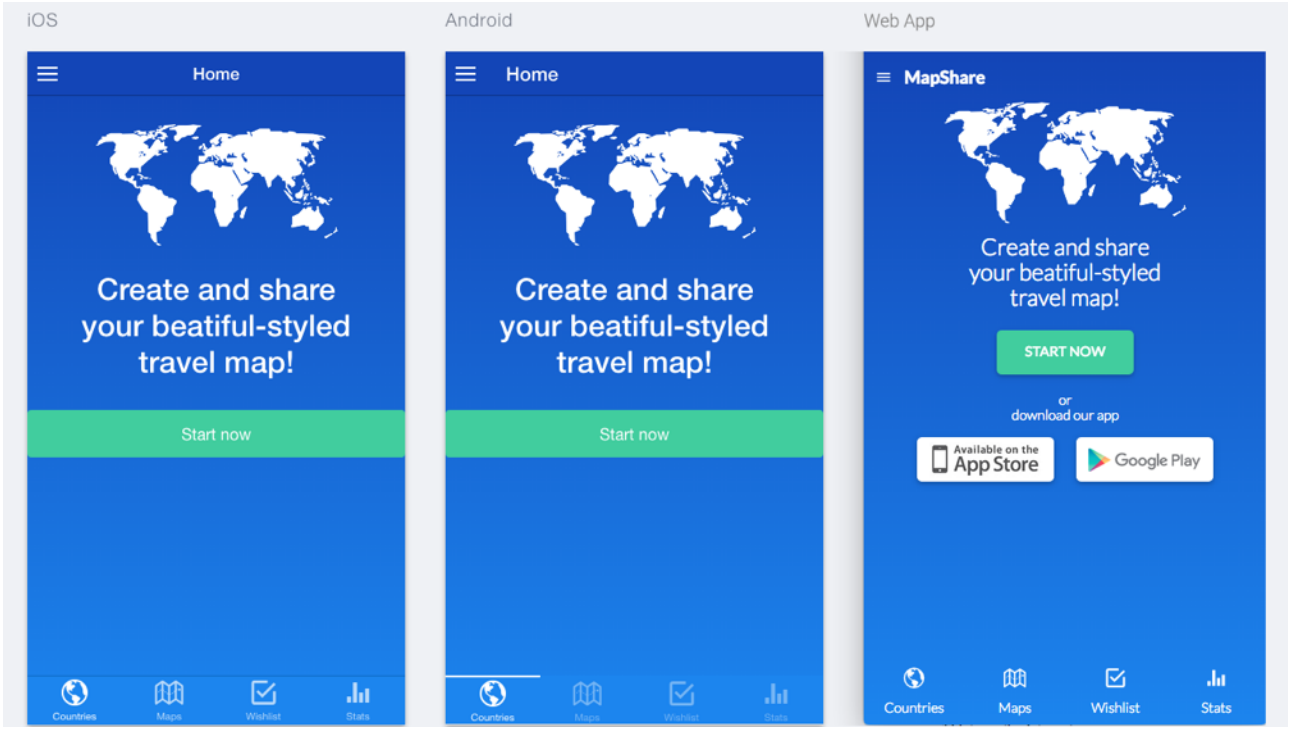

Figure 1: app start page

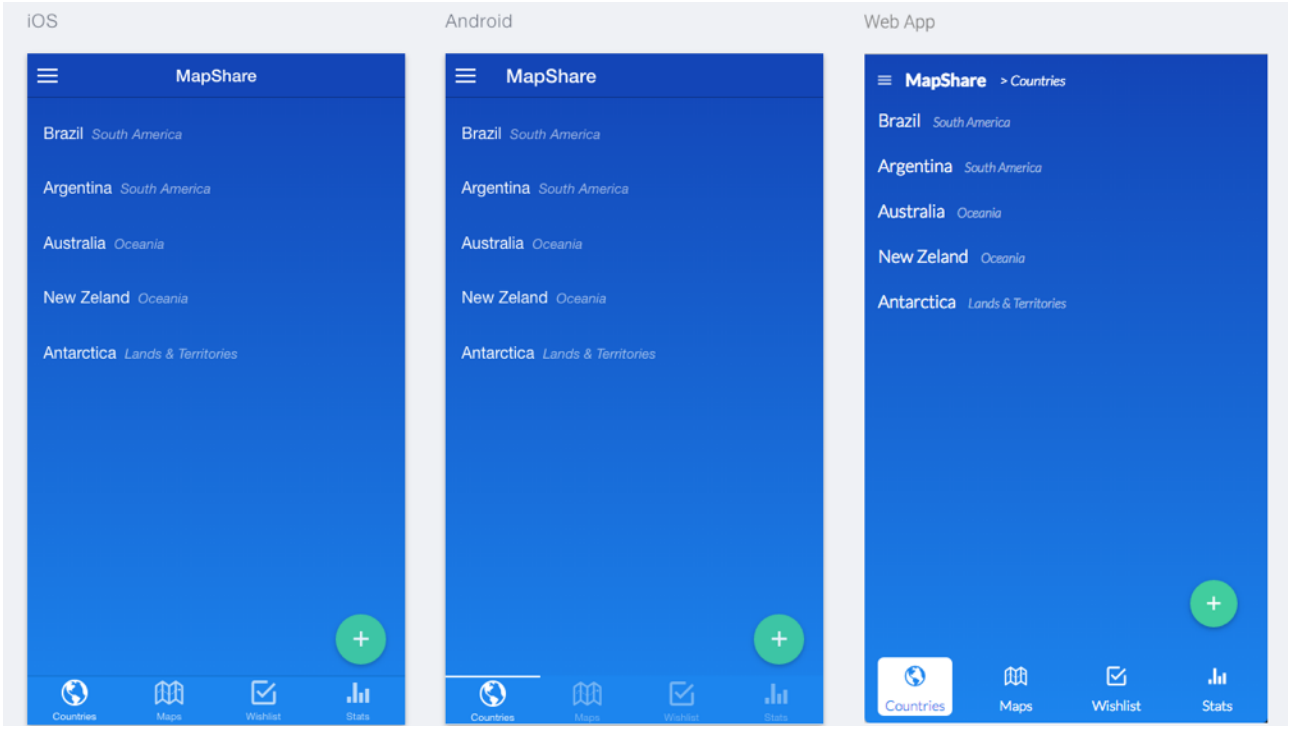

Figure 2: countries list

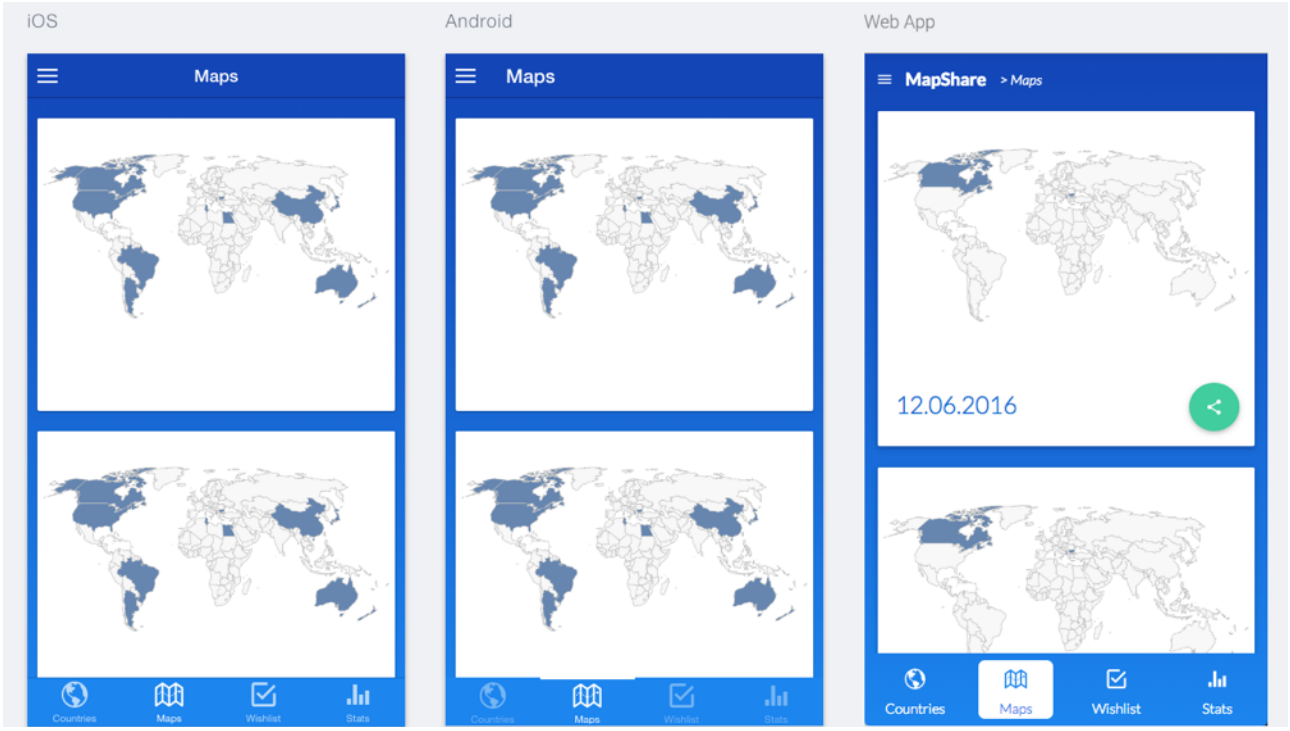

Figure 3: list of maps

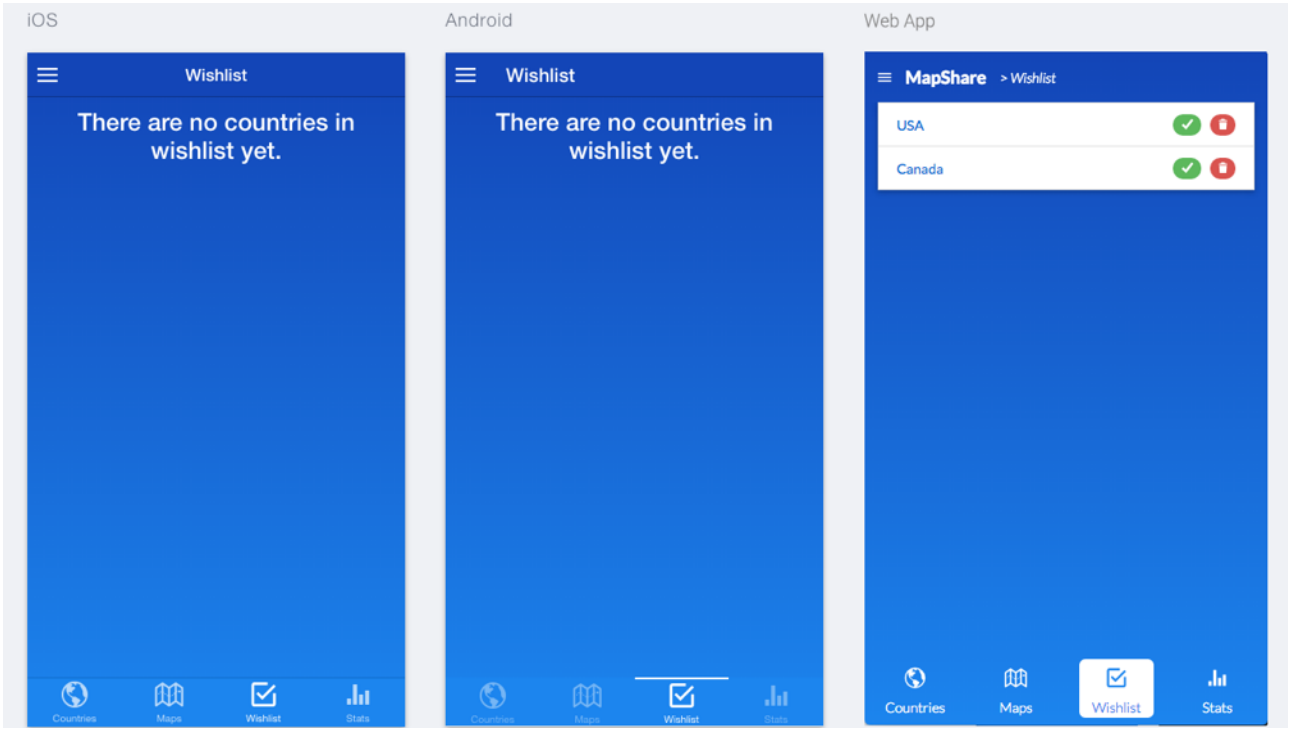

Figure 4: wishlist view

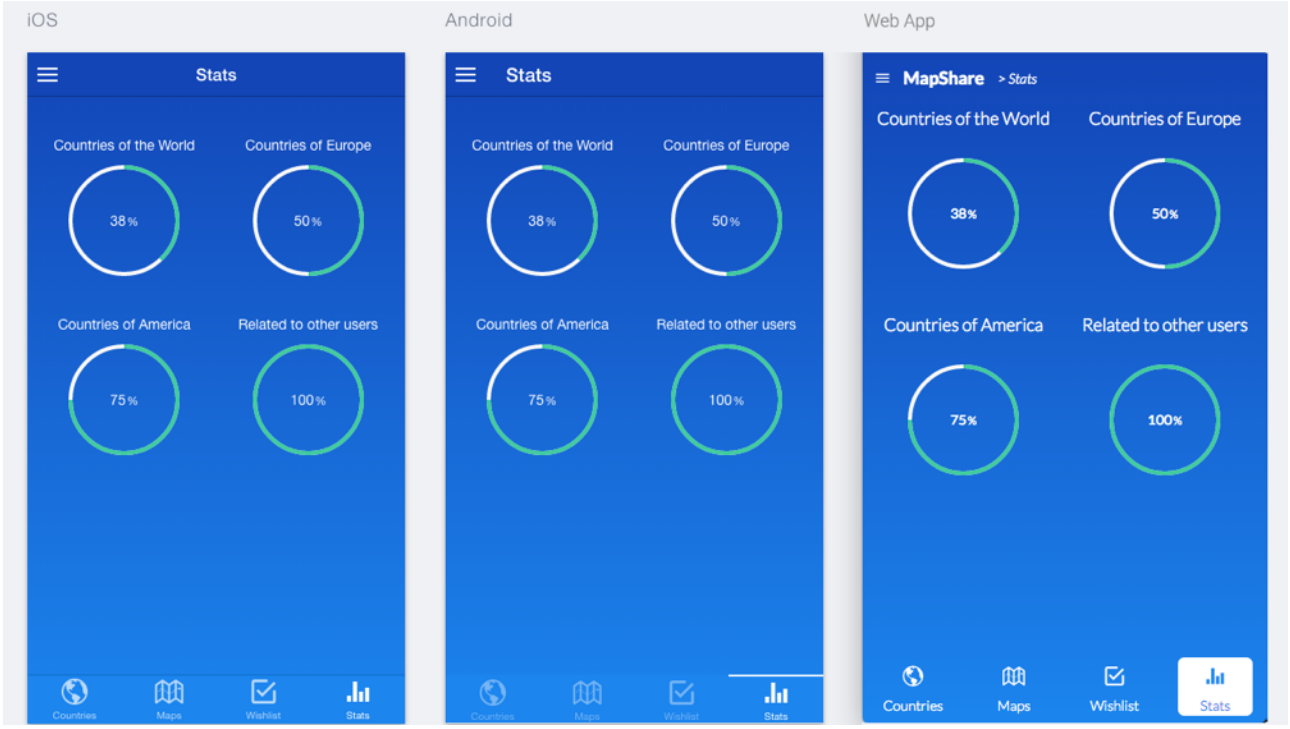

Figure 5: statistics view

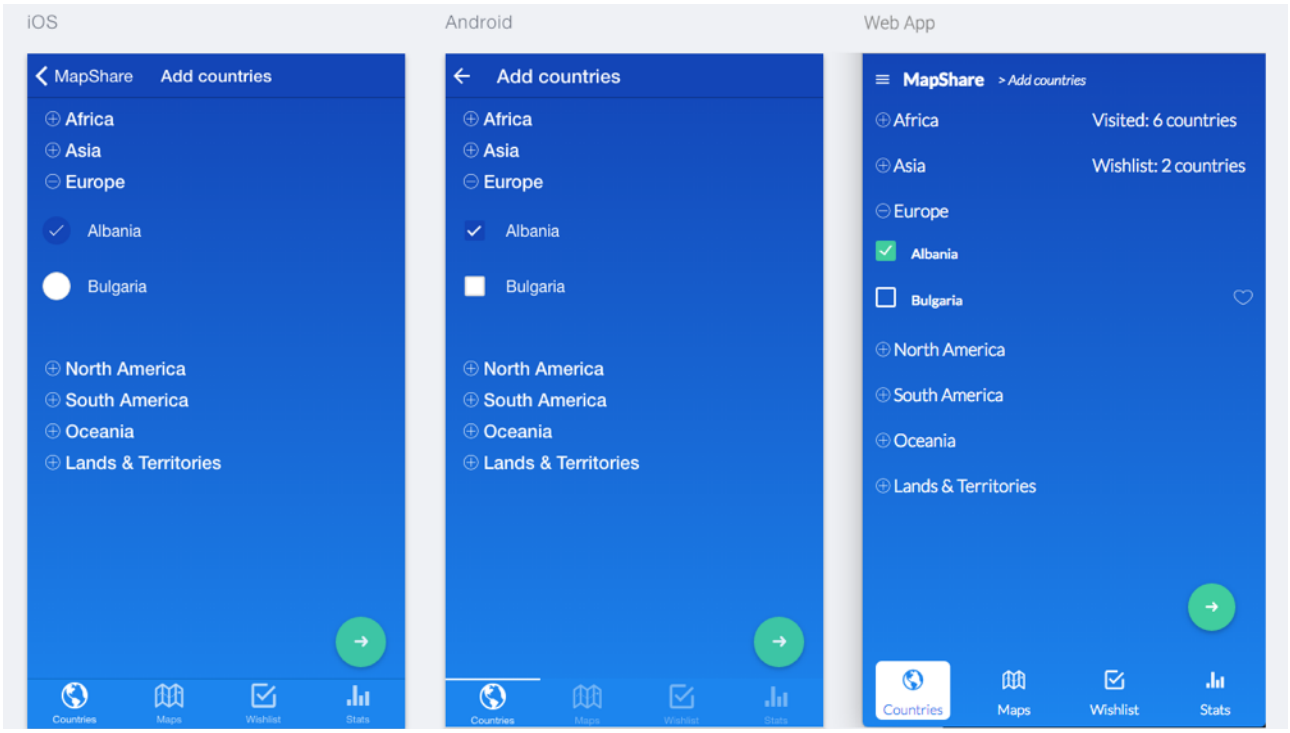

Figure 6: add countries

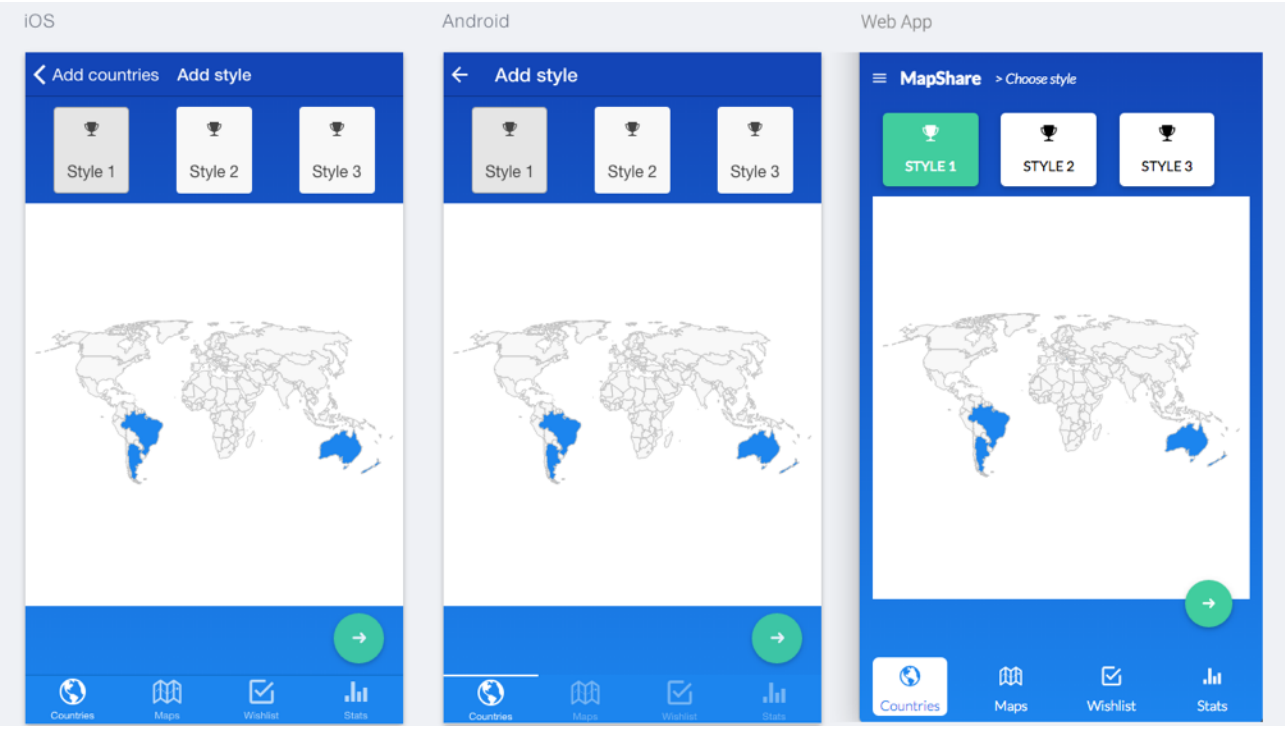

Figure 7: choose map style

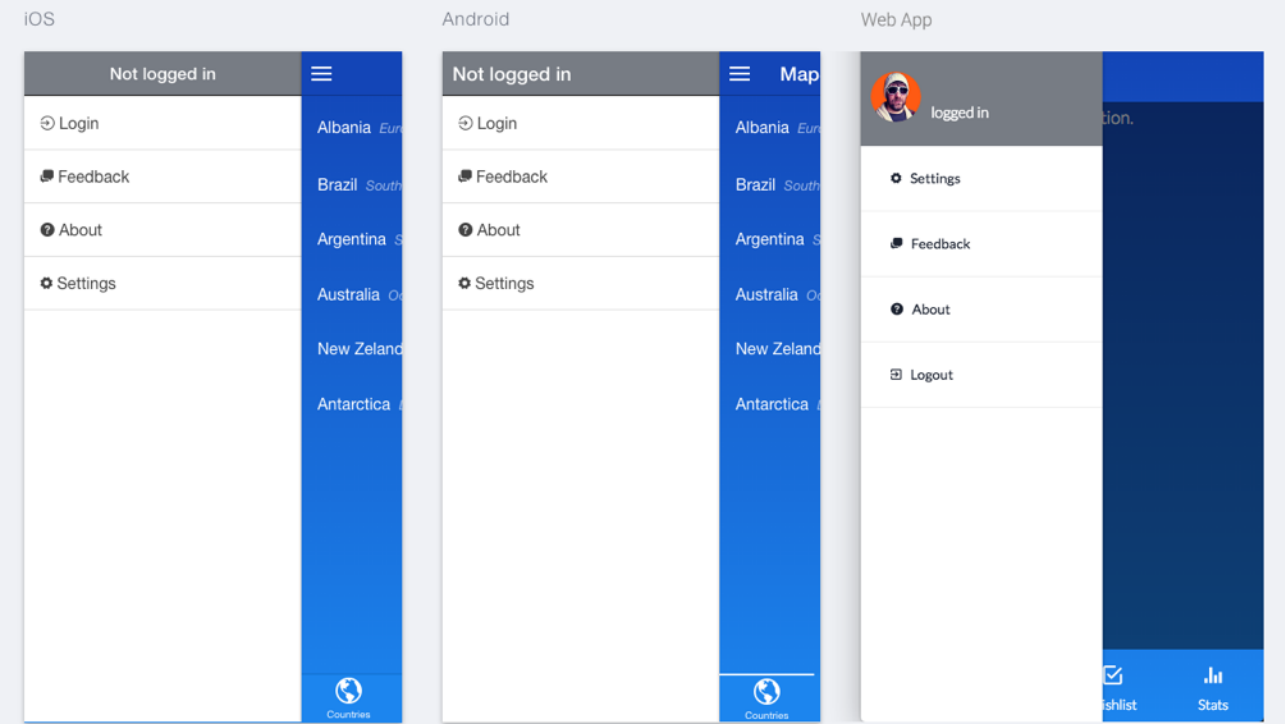

Figure 8: side menu

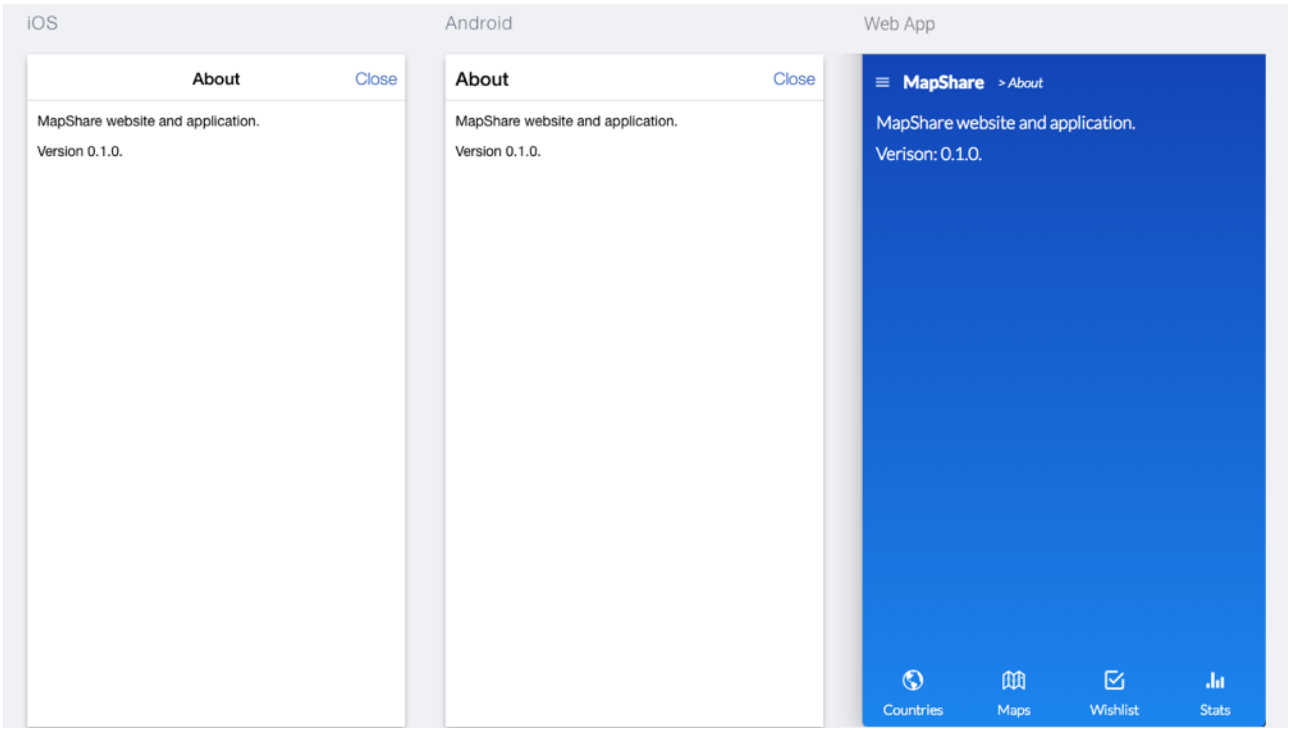

Figure 9: about page

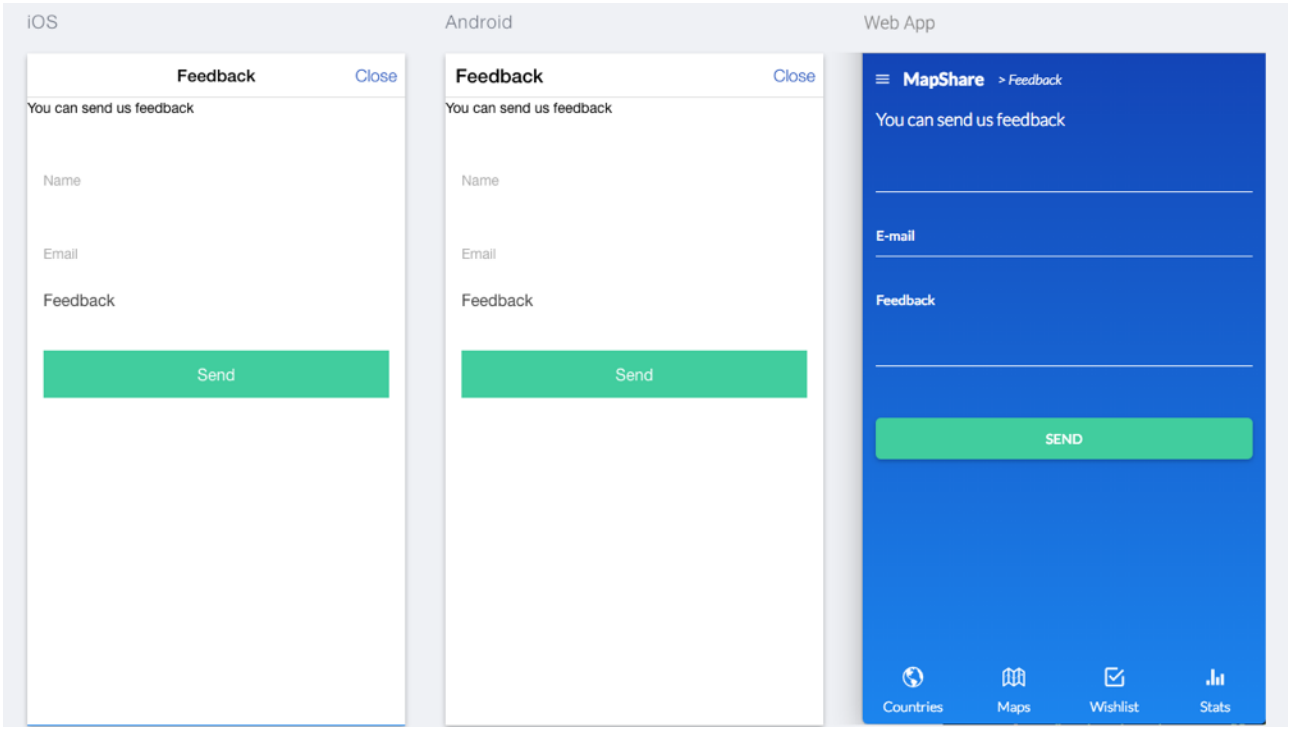

Figure 10: feedback page

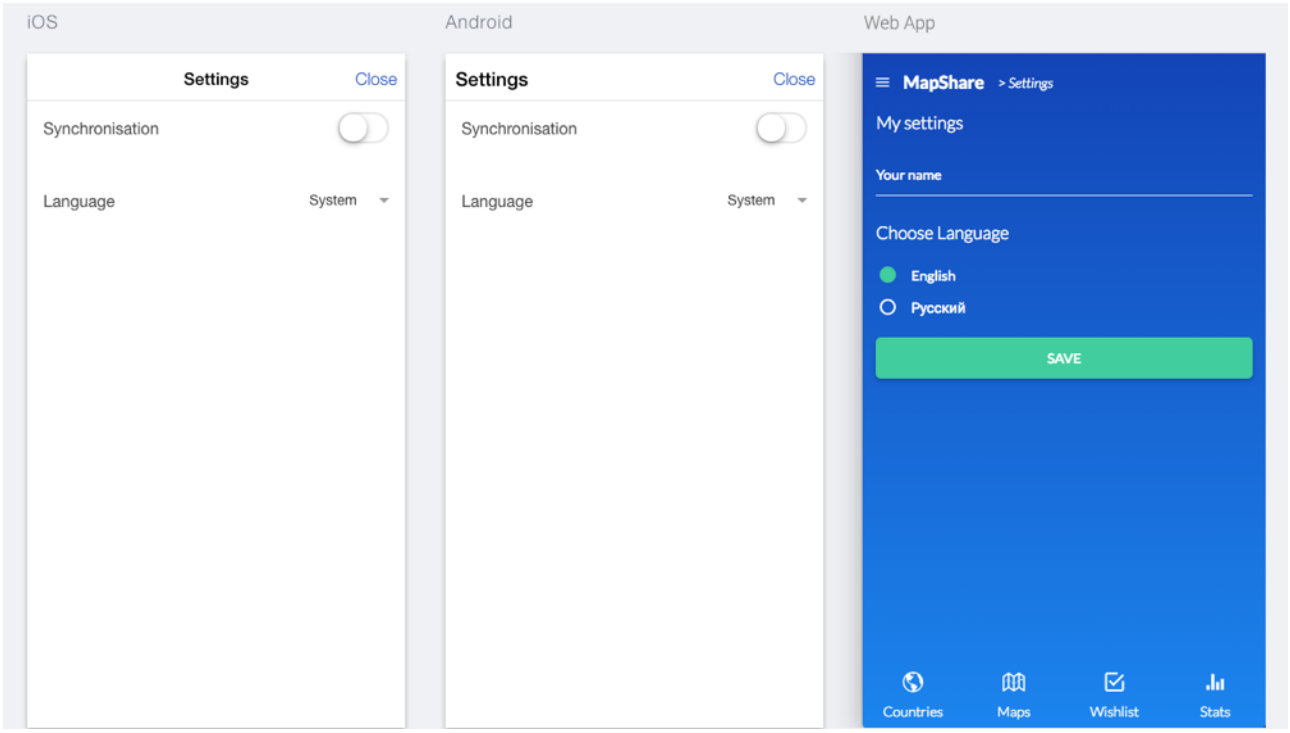

Figure 11: settings page

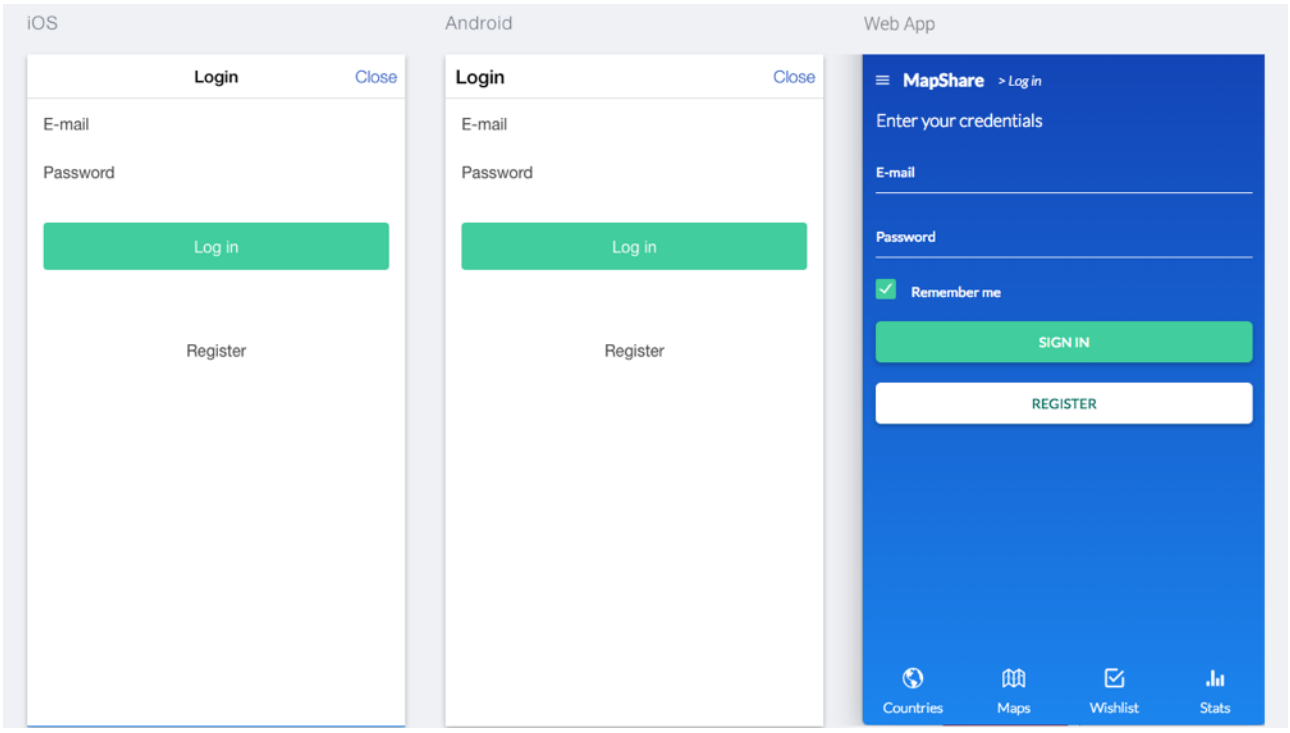

Figure 12: login page

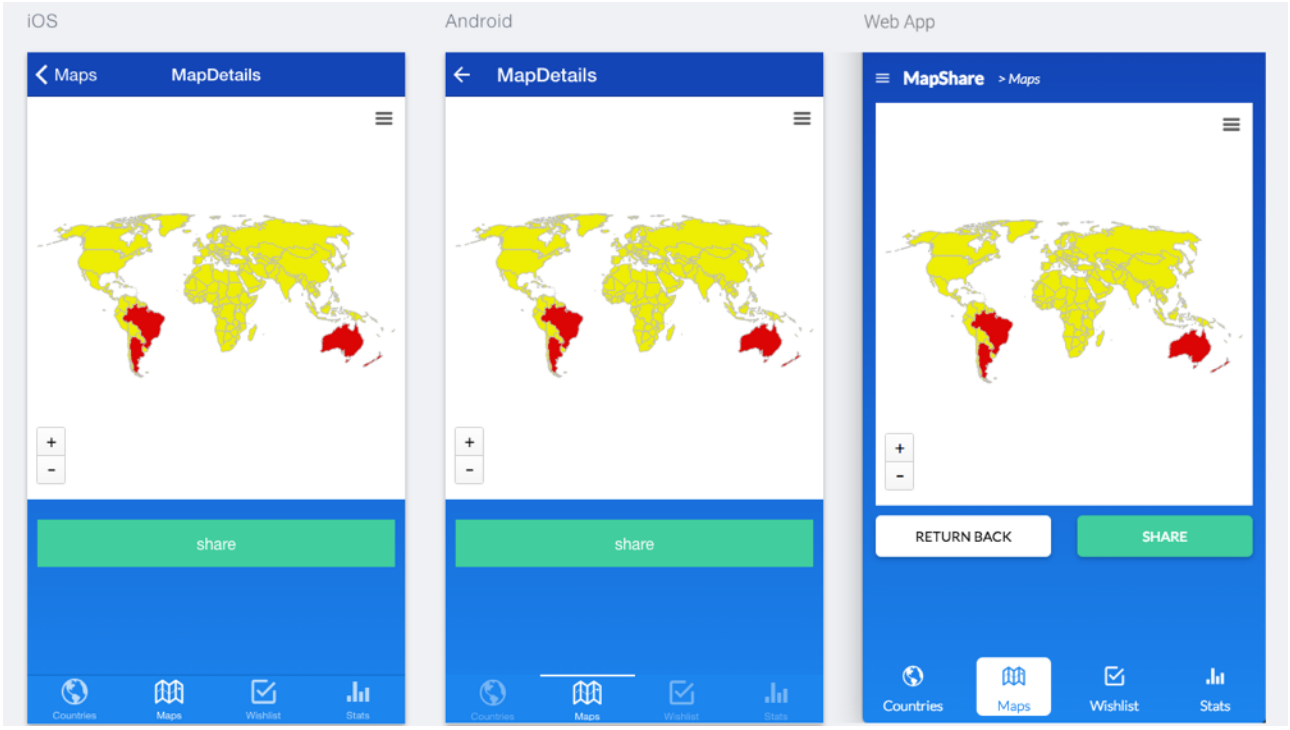

Figure 13: map details view

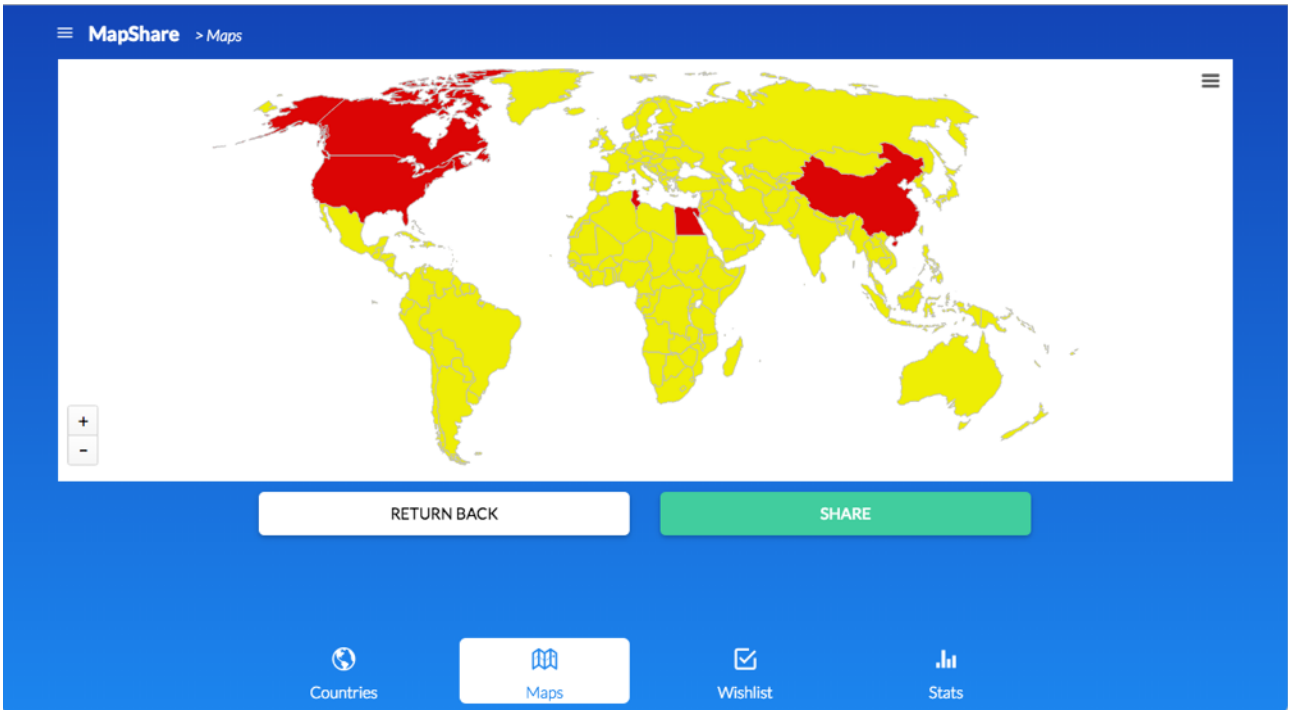

Figure 14: desktop screen map details view

3. Conclusion

Results I obtained

I have come a long way from the idea of the project to practical realization. Being a full-stack developer isn't easy task, especially for the first time. I had designer's background, so the first steps like *UI design*, prototyping and *HTML layout* were comparatively easy to me. Then, every next step demand more and more effort to realise. Maybe, I choose too ambitious task for the very first project as web developer. I had stuck several times when my knowledge was insufficient to solve a task. Then I chose alternative decision or found hints on the internet.

Features and shortcomings

The main benefit from this project that I get priceless experience to work on the real project. And the main disadvantage is to make this alone. There is no real opportunity to put every feature to the perfect level, when you are so condensed on the time scale.

Choices, I might have made differently

To make project getting done, I heavily used 3rd-party libraries and frameworks and that's be a challenge to make things work the right way. If I had more time, I could try to get rid some of them, and I am planning to continue developing the project in the near future. First, I'll try to abandon the famous *jQuery* library, cause it's behaviour ruin project structure. But before I need to implement my own library to work with canvas and sag to draw maps, cause *Highmaps.js* is dependant on jQuery.

Then I am planning to migrate to *Angular* version 2 and Ionic 2, cause Ionic 1 is lack of critical UI components like *Floating Action Button (FAB)* and *Navigation Drawer* (that I'd implemented in web version).

Desktop and tablet version also need more elaborated design, as well as new features, that I am planning to implement to the project.

Final words

Anyway, that was an educating experience and I am glad for what I had achieved. The journey just began, and I want to say "thank you" to all, who make this possible, especially prof. Jogesh Muppala.

4. Referencies

- Project website http://mapshare.me/dev
- GitHub page https://github.com/DmitryMarkov/MapShare
- jQuery http://jquery.com/
- Highmaps.js http://www.highcharts.com/docs/maps
- Bootstrap 4 http://v4-alpha.getbootstrap.com/
- Material Design for Bootstrap http://mdbootstrap.com/
- Easy Pie Chart https://rendro.github.io/easy-pie-chart/
- Angular Style Guide by John Papa https://github.com/johnpapa/angular-styleguide