Application Design and Software Structure: 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

2.1 The REST API Specification

The base URL for REST API Endpoints is http://mapshare.me/api/

Authentification(RU):
http://mapshare.me/api/user/login

Registration (C):
http://mapshare.me/api/user/register

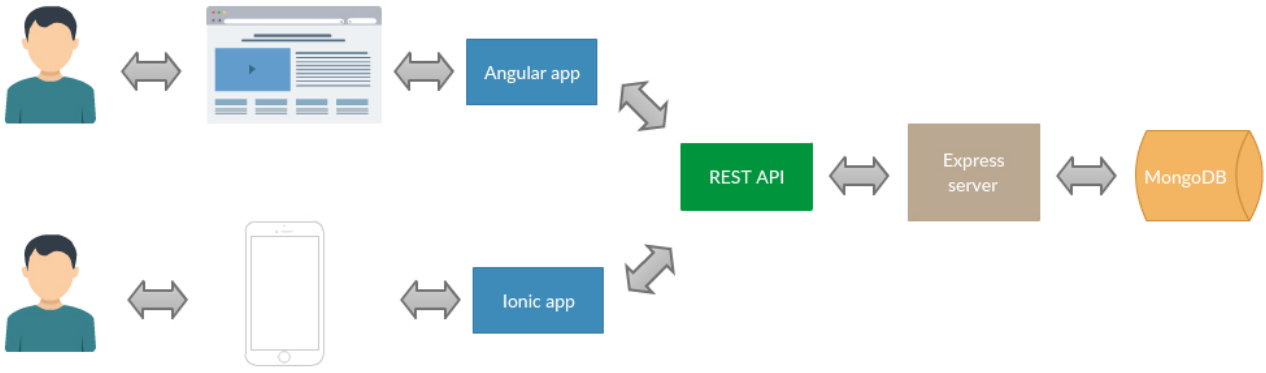
Log off (U):
http://mapshare.me/api/user/logout

Selected user operations (CRUD):
http://mapshare.me/api/user/:userId

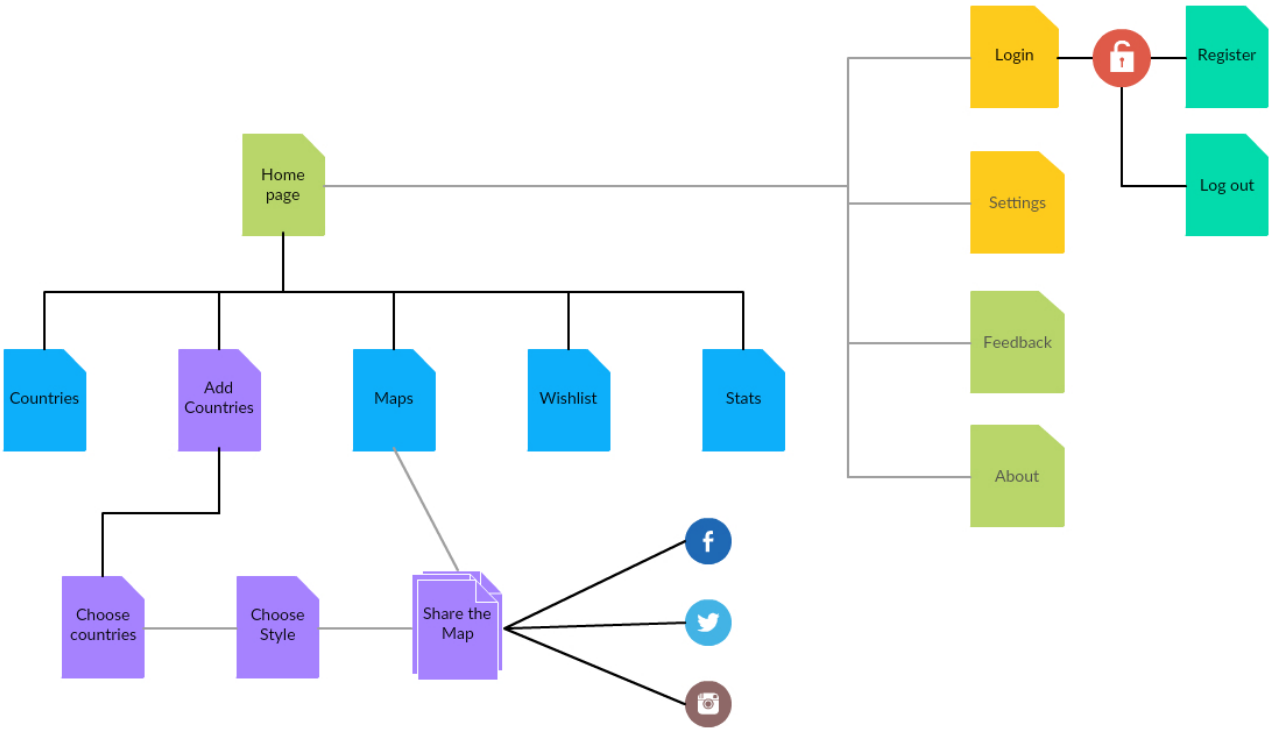
List of countries (R):
http://mapshare.me/api/continents/

2.2 Front-end Architecture Design

Front-end architecture for website is based on Angular and Ionic frameworks MVC architecture.



Site map (both for web and mobile application):



2.3 Database Schemas

Document-based MongoDB is used for store data for app and website.

Data organised in Collections. Because of simplicity, there are only two collections used in our app at the moment:

User

(contains all information about user)

Example:

```
"users": [
    {
      "id": 1,
      "email": "test@test.com",
      "name": "Dmitry Markov",
      "password": "test123",
      "lastLogin": "2016-05-20T17:57:28.556094Z",
      "visited": [
        "BG",
        "CA"
      ],
      "wishlist": [
        "AL"
      ],
      "favs": [
        "BG"
```

```
      ],
      "settings": [
        {
          "language": "RU"
        }
      ],
      "maps": [
        {
          "id": 1,
          "date": "2016-05-20T17:57:28.556094Z",
          "countries": [
            "AL",
            "CA",
            "BG"
          ],
          "styleId": 1,
          "wishlist": [
            "RU"
          ],
          "favorites": [
            "BG"
          ]
        }
      ]
    }
  ]
```

Continents

(contains array of countries and continents)

Example:

```
"continents": [
    {
      "id": "1",
      "name": "Africa",
      "countries": [
        {
          "id": "EG",
          "name": "Egypt"
        },
        {
          "id": "TN",
          "name": "Tunisia"
        }
      ]
    },
    {
      "id": "2",
      "name": "Asia",
      "countries": [
        {
          "id": "CN",
```

```
      "name": "China"
    },
    {
      "id": "JP",
      "name": "Japan"
    }
  ]
}
```

2.4 Communication

The structure of the communication messages will repeat JSON format of database documents. The whole list of countries (with continents) and whole user object will be fetched and posted from and to REST API.

List of countries planned to be immutable from front-end (only get operations provided). User objects will support all kind of CRUD operations: create, read, update and delete.

3. Conclusion

I am trying to make a simple but expandable architecture for the project. There are only two server requests for application's start: user credentials, which includes not only login, but the whole user object and a full lust of countries (because we assume, that a number of countries will not surpass a 200 in a nearest future, this operation will be really fast).

User object may be potentially grow fast for heavy users, but except maps array it will not outstrip tens of elements in its arrays, which will never exceed maximum document size in MongoDB. I am planning to cut the maps array to dedicated collection in future releases and populate this information on a server-side.

4. Referencies

- Project architecture diagram
  http://creately.com/diagram/iosypf8w1/ZLkSBEdsQxQFgJnKoLHA3V0OSt0%3D
- Site map diagram
  http://creately.com/diagram/iosypf8w2/vsZkTCrgJv6SzthQdx9zal4ZY%3D
- JSON format website
- http://json.org/