**“Mobile Applications Development”**

**Course**

**a.y. 2014/2015**

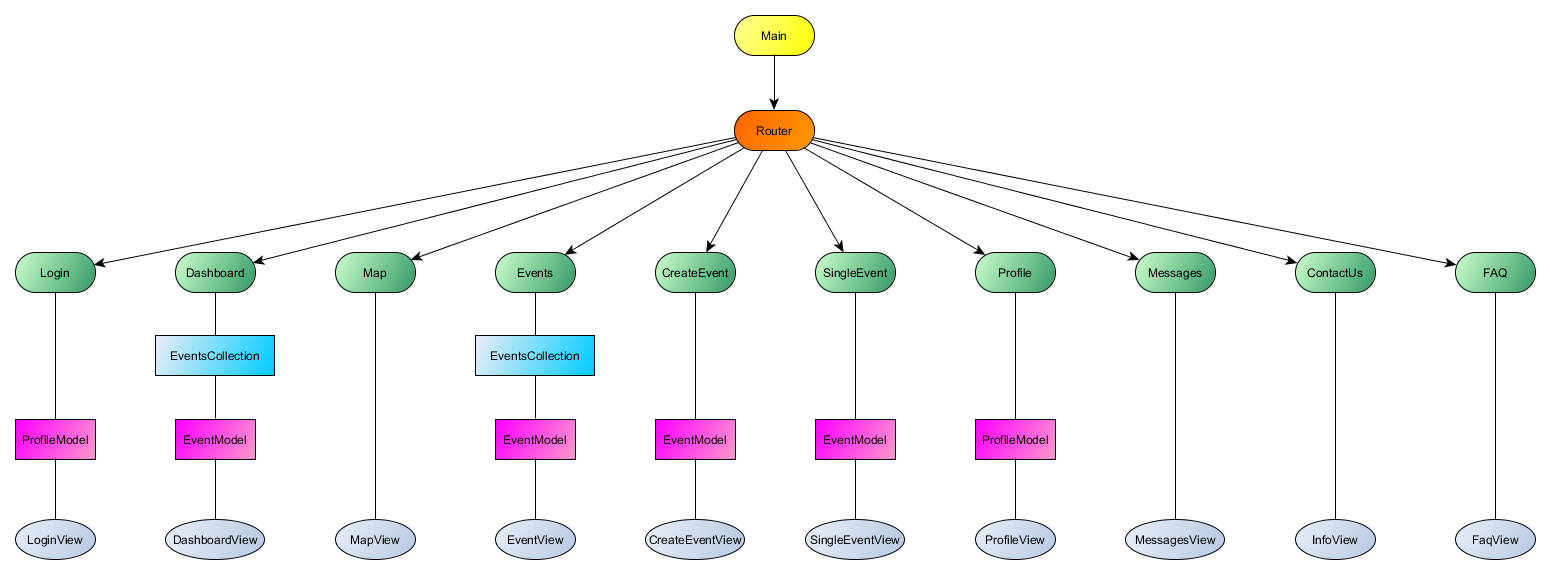
**Meet You There**

**Technical Documentation**

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name** | **Student Number** | **E-mail address** |
| **Martin Doychev** |  | **martin.doychev@abv.bg** |
| **Milos Darmanovic** |  | **milosdarmanovic@yahoo.it** |

Architecture

# MVC architecture



The architecture of the application is built on the MVC pattern. One of the main controlling points is the Router, which distributes the requests to the different views and their matching controllers.

The **Login** module is responsible for allowing the user to login to the application. It also allows registration in the same view via a different form with several fields. It uses the **LoginView**.

The **Dashboard** module is the main entry point of the application after login. It shows the main navigation units and some events information, as well as interface, linking to the creation of new events. It uses the **EventsCollection**, **EventModel** and **DashboardView**.

The **Map** module is used for visualizing events on the map (the functionality is not yet developed). Currently, it is only showing a map. It uses the **MapView.**

The **Events** module is used for visualizing a list of events. It also allows filtering of the events by the users’ preferences. It uses the **EventsCollection**, **EventModel**, **EventsView** and **EventView**.

The **CreateEvent** module is used for creating and recording new events in the database. It provides all the required interface for inserting the data. It uses the **EventModel** and the **CreateEventView**.

The **SingleEvent** module is used for visualizing information about event. It is named like this due to the existing EventView, used for populating the EventsView table. It also provides the interface for attending and canceling attending to an event. It uses the **EventModel** and the **SingleEventView**.

The **Profile** module is used for visualizing the user’s profile. It uses the **ProfileModel** and the **ProfileView**.

The **Messages** module is used for allowing the user send messages to other users. It uses the **MessagesView**.

The **ContactUs** module is used for allowing the user contact the developers of the application. It provides the required interface for explaining any issues and saves them in the database. It uses the **InfoView**.

The **FAQ** module contains all pre-set frequently asked questions. It uses the **FaqView**.

# Data sources



Our application is only depending on BaasBox as user management platform and database provider. All other functionalities are provided by the application itself.

# Used libraries and frameworks

* **Baasbox.js** – as the main database and user management provider;
* **Backbone.js** – as the main MVC framework provider;
* **Handlebars.js** – helpful for templating the content of the views;
* **Icecream** – helpful for arranging the views in a manageable grid view;
* **JQuery.js** – for simplifying object access and JS in general;
* **Ratchet** – for frontend issues;
* **Require.js** – for modularizing the application;
* **Spin.js** – used in order to simplify creating the spinner, which shows during loading periods;

Development activities

# Smart solutions

We believe that the usage of Handlebars provided us with the opportunity to reuse good templates and populate them quite easily with data, which seems like an elegant and nice solution.

Also, we try to use the same pattern for views that have common purpose (like the “Contact us” and “FAQ” views) and provide small navigation unit to switch easily between them. We use the same pattern in our login page.

# Difficulties

[LOW] **Initial setup and familiarization with the tools and frameworks used** – since we didn’t have any experience with most of the tools used, we needed some time to get used with the use and logic behind them; also, we had trouble with the setup of the development environment due to imperfect guides for setup and other side errors;

[LOW] **Differences between Ripple and mobile device visualization** – we noticed several very significant differences in the visualization, but we managed to quickly fix them relying on the mobile view;

[MEDIUM] **General BaasBox usage** – a lot of issues came across when we started implementing the database; for example, in order to access data the user must be of role “backoffice” or more privileged in a default situation; otherwise, on every object creation, we have to explicitly grant permission to that object to all users;

[HIGH] **BaasBox sign-up** – the sign-up doesn’t work because of an authentication error; we couldn’t fix it; we found a last minute solution, but we couldn’t implement it in time;

[HIGH] **BaasBox delete queries** – the queries return an OK answer, but don’t actually delete the record it the DB; we found a last minute solution and we managed to implement it, but we have a small visual issue which we couldn’t implement in time;

# Used tools

* **Notepad++** – as the main development environment for JS;
* **Ripple Emulator** – as the main debugging environment;
* **GitHub** – as a team synchronization tool;
* **Apache Cordova, Android SDK** – for building the project and as the connection point for deploying the application on a real mobile device;
* **Microsoft Office Word** – for writing documentation;
* **yEd Graph Editor** – for creating the graphs for documentation;