**“Mobile Applications Development”**

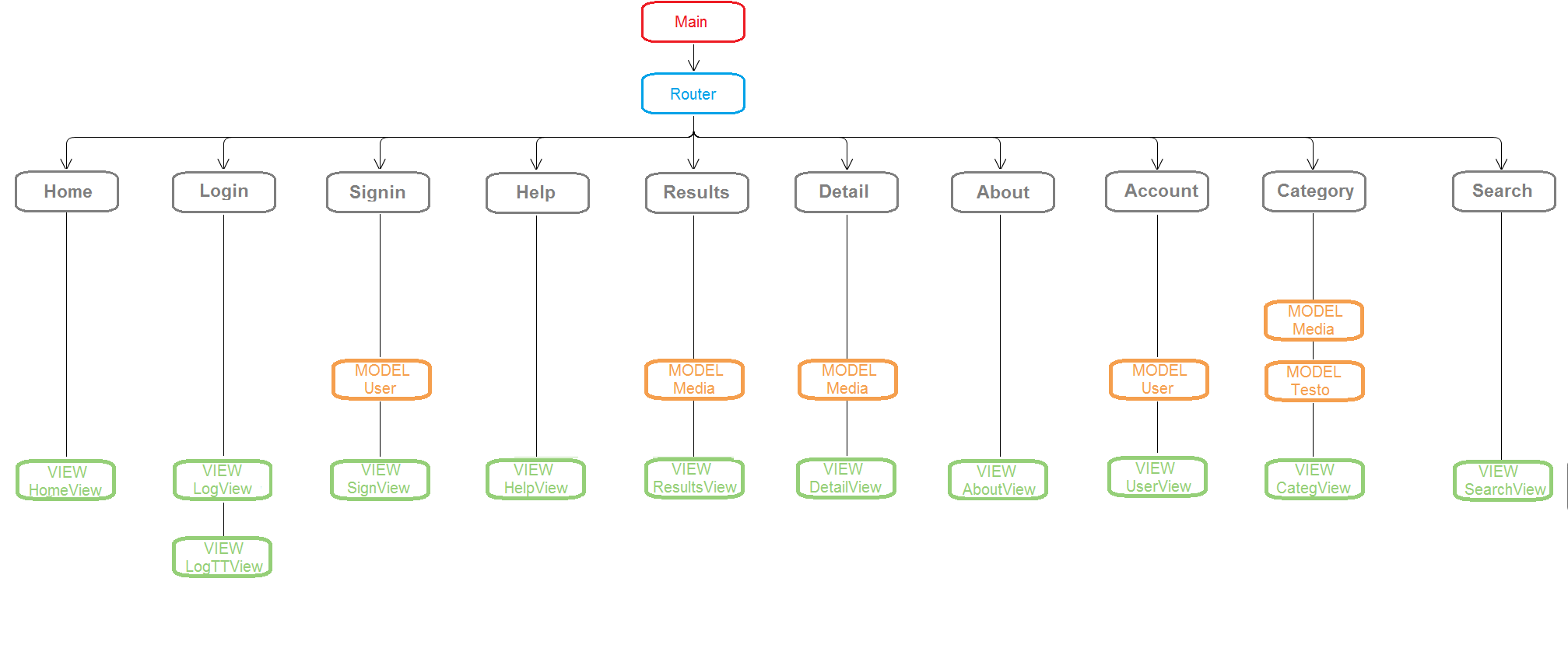
**course**

**a.y. 2014/2015**

**TRY THIS!**

**Technical Documentation[[1]](#footnote-1),[[2]](#footnote-2)**

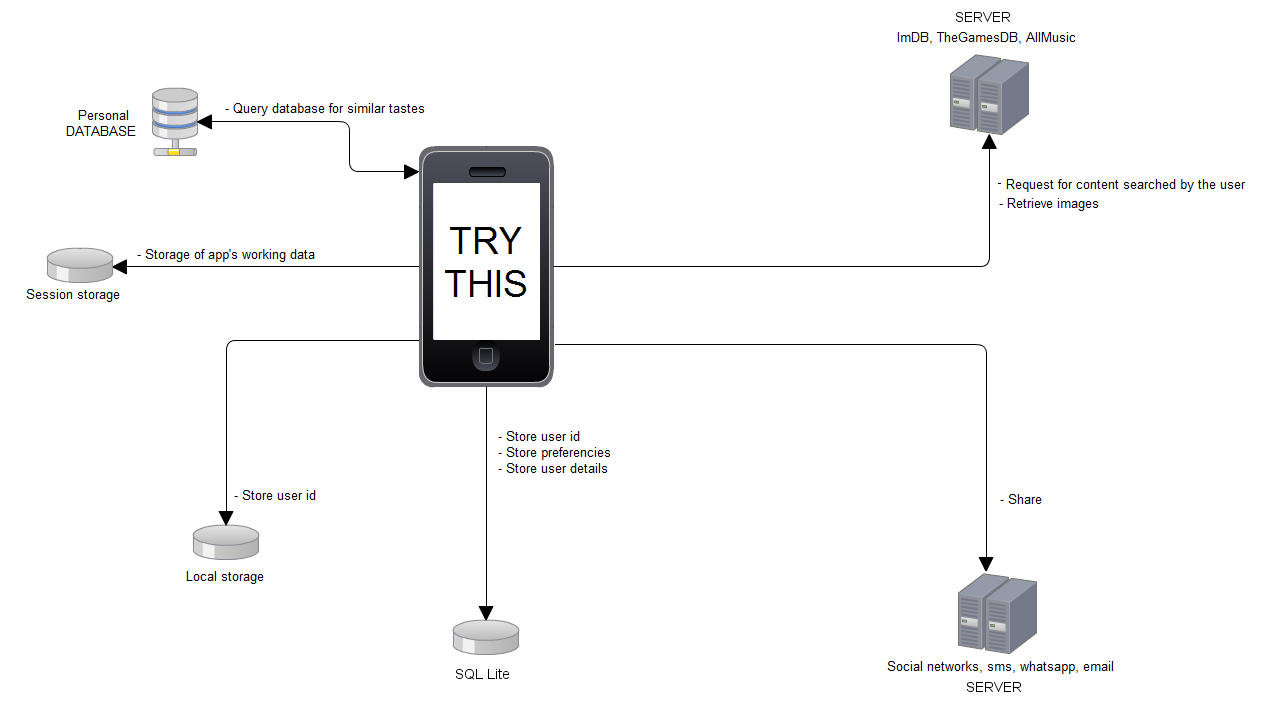
|  |  |  |
| --- | --- | --- |
| **Team Members[[3]](#footnote-3)** | | |
| **Name** | **Student Number** | **E-mail address** |
| Mattia Ciolli | 219263 | mattia93c@gmail.com |
| Michele Iessi | 222123 | iessimichele@gmail.com |

Architecture

# MVC description:

The main launches the router. In the router we have a model (DB.js) used to create the database if necessary at startup. From the router we access all other views. The first view at the first launch is LogView. In it you can choose to login or signin. In both cases changes the view to LogTTView or SignView that uses a User model to create a user and pass his values trough views via localStorage. It also put him in the database if a new user. If the user already used the app, the first view will be HomeView, skipping Logview and doing automatic login. HomeView is the “central” view of the app. From here you can access via the router to SearchView, UserView, HelpView and Aboutview. You can also logout from here. You can select the category to search and it passes via sessionStorage the parameters to SearchView to automatically enable the search in the right category. The ResultsView uses the Media model to let DetailView know which element search in database and show. ResultsView stores the Media model in sessionStorage and DetailView retrieves it to show all its details. The AccountView uses the User model just to show in the title the name of the current user. Passed via router. The CategoryView uses two models: Testo and Media. The first to show in the title the name of the category selected by the user such as “My movies”. The second similar to ResultsView, saves in sessionStorage the media result of the DB query and passes it to DetailsView to show everything. Search view doesn’t use a model but passes all fundamental elements for ResultsView’s query such as item searched and category. Passed via sessionStorage. HelpView and Aboutview are “passive” views, they only show some informations.

# Data sources



# Description

Stored all user’s details and preferencies with embendded cordova’s sqlite

Used local storage to store user id to keep the current user also between different accesses.

Used session storage to store app’s dat, like query results or category chosen to pass it trough views and do a cache-like action.

Access to database with search to retrieve similar tastes.

Access to social services, sms, email with socialshare plugin.

Access to external servers to retrieve content informations like images.

# Used libraries and frameworks

Backbone.js: for mvc pattern. Used to give order to the code

RequireJS: to improve the speed and quality of code. Used to load scripts

Handlebars: template engine. Used to simply build semantic templates

Underscore: Backbone depends on it

Zepto: faster than jQuery. Used for jQuery functions

Slideout: used for the side scrolling menu in the home. Used because the menu had only 4 voices, so

an entire template and view was eccessive. Also adds some style

Ratchet: entire app’s graphics. Useful because simple and effective. Used about everything of it

Icecream: to size the content in entire page

Preloader: to preload images at startup

Utils: to enhance the speed of page change

Spin: loading spinner, to avoid blank spaces while loading images

SocialShare: to share over about everything. Used because simple to implement and very effective, also autoinstalling via npm

Development activities

# Smart solutions

Not particular solutions. Used ratchet slider instead of using an external library for home and help. Saved time, code and space.

# Difficulties

Adapting Ratchet style to ours: just changed its css.

Side scrolling menu with Slideout: after many attempts, figured out how to implement in Backbone. Disabled support to swipe interactions due to the carousel in the home page. They were both scrolled while swiped, not so good to see.

Carousel: tried to use Slick carousel library but impossible to implement. Solved by heavly modifying Ratchet’s slider.

Libraries: zero documentation, zero examples and it seems no one is using them with Backbone or requirejs, figured out with brute force attempts :-D

Backbone and requirejs coexistence: it’s difficult to implement every external library in a short time, solved with a lot of time and patience. Moreover their documentation takes for granted a lot of things, giving only basic informations.

# Used tools

Coding:

Code writer, Sublime text 2.

Used for coding. Chosen because they support pretty all types of scripts

Testing and debugging:

Opera browser, Chrome, Ripple emulator.

Used developer tools to check if error occurred. Used Ripple to test app’s functionality on different devices. Used element inspector to know why something was not displaying or doing it bad.

Graphics:

Game Maker (image editor section), Photoshop, Paint.

Used all to create the UI elements such as background, TryThis logo and images in documents. Chosen because we are familiar with them.

1. The max length of this document is 10 pages [↑](#footnote-ref-1)
2. The structure of this document is fixed, it cannot be changed in any way [↑](#footnote-ref-2)
3. The team leader is listed as first member in this list [↑](#footnote-ref-3)