**Code Structure**

**Client Side**

Our app is a single-page app, written with AngularJS using ui-router. Files by type (and folder) -

**Partials:**

* **login\_signup** – Landing page, user login (by username only) and signup form in case of new user.
* **main** – Holds the navbar and a change location autocomplete box.
* **tab\_content** – Primary content HTML, fills up according to selected tab

**Services:**

* **LoginService** – Provides API access to the server for all user-related actions (login, signup, update).
* **PlacesService** – API access for place related queries, such as getting a list of restaurants/hotels/etc, searching for a place by review text, getting full place details and so on.

**Controllers:**

* **LoginController** – Controller for login/signup flow.
* **MainController** – Injects general functions and utils to the scope, not that major.
* **PlacesController** – Primary app controller, used for controlling the various tabs and full place view, reacting to user actions (such as changing search radius, central address, etc).

**Other:**

* **app.js** – App initialization and configuration.
* **utils.js** – Utilities, such as a function collecting the HTML files from the server.
* **index.html** – The parent HTML file.
* **CSS** folder – Holding several css files we use.

**Server Side**

Our server is Django based, with all the DB-related actions written separately using python’s MySQLdb library.

**tauwebsite**: Main module.

* **settings** – Django settings file. Added our DB connection and static file serving for the HTML files.
* **urls** – Routing file, connects API calls to the relevant views.
* **utils** – Holds the DBUtils class, our implementation for the SQL queries.
* **serializers** – Basic serialization implementation for serializing query results before sending them back to the client.

**users**: User-related module. Only views.py file is relevant.

* **LoginView** – GET call returns a user by his username. POST call is used for both registering a new user and updating an existing one, differentiated by is\_update parameter.

**places**: Place-related module. Only views.py file is relevant.

* **PhotograficPlacesView** – POST call retrieves places around central coordinates with at least ‘num’ pictures in our DB.
* **GeneralPlacesView** – GET call retrieves places with the highest average ranking type, POST call retrieves the best result of each type in radius 3km around the user.
* **PlacesByReviewView** – POST call gets all places with a text string in their reviews.
* **FoodView** - POST call retrieves restaurants around central coordinates in a certain radius.
* **BarView** - POST call retrieves bars around central coordinates in a certain radius.
* **ClubView** - POST call retrieves clubs around central coordinates in a certain radius.
* **HotelView** - POST call retrieves hotels around central coordinates in a certain radius.
* **ShopView** - POST call retrieves shops around central coordinates in a certain radius.
* **PlaceDetailsView** – GET call retrieves the full details of a place, including all pictures, reviews and opening hours.