JFrontend - Technical Background

Task:

The task of the JFrontend is to visualize the result of the execution engine and allow the user to easily control and manage its modules and packages.

Realization:

The JFrontend implements a MVC pattern based on AngularJS.

```
app/
├── fonts/
├── icons/
├── images/
├── js/
├── less/
├── scripts/
├── styles/
├── templates/
├──views/
  ├── index.html
  ├── favicon.ico
```

Main app.js file

App.js files include all external module for angular. Below is a preview of app.js in full version.

```
(function () {
    angular.module('homer', [
        'ui.router',                // Angular flexible routing
        'ui.bootstrap',  // AngularJS native directives for Bootstrap
        'angular-flot',             // Flot chart
        'angles',                   // Chart.js
        'angular-peity',            // Peity (small) charts
        'cgNotify',                 // Angular notify
         'angles',                    // Angular ChartJS
         'ngAnimate',                 // Angular animations
         'ui.map',                    // Ui Map for Google maps
         'ui.calendar',               // UI Calendar
         'summernote',                // Summernote plugin
         'ngGrid',                    // Angular ng Grid
         'ui.tree'                    // Angular ui Tree
     ])
 })();
```

Route config

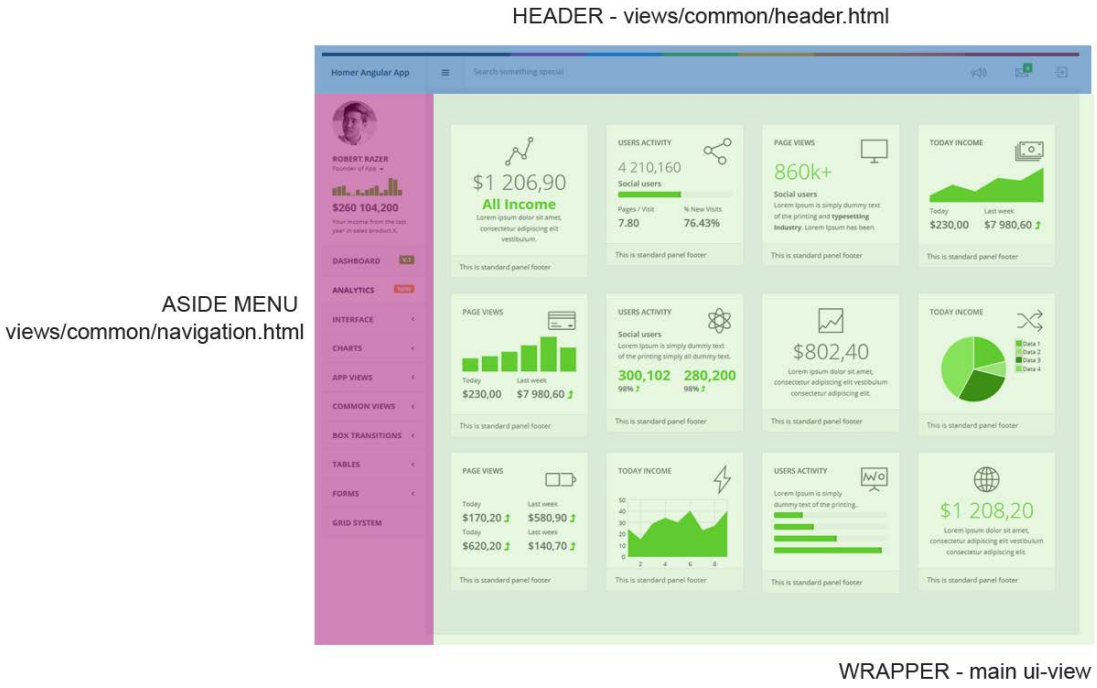
To manage all route we use great plugin Ui.Router. AngularUI Router is a routing framework for AngularJS, which allows you to organize the parts of your interface into a state machine. Below you can see example of configuration ui-view. Configuration routing are in config.js file

```
1. function configState($stateProvider, $urlRouterProvider,
   $compileProvider) {
2.
3.     // Optimize load start with remove binding information inside the
   DOM element
4.     $compileProvider.debugInfoEnabled(true);
5.
6.     // Set default state
7.     $urlRouterProvider.otherwise("/dashboard");
8.     $stateProvider
9.
10.          // Dashboard - Main page
11.          .state('dashboard', {
12.              url: "/dashboard",
13.              templateUrl: "views/dashboard.html",
14.              data: {
15.                  pageTitle: 'Dashboard',
16.              }
17.          })
```

Layout structure

Main index file include only ui-view. Each state has own content layout that can be modified in /views/common/content.html file. Basic layout are created with few main elements:

1. `header` top header.
2. `aside menu` left sidebar navigation.
3. `ui-view` main container for page elements.



Layout structure - file

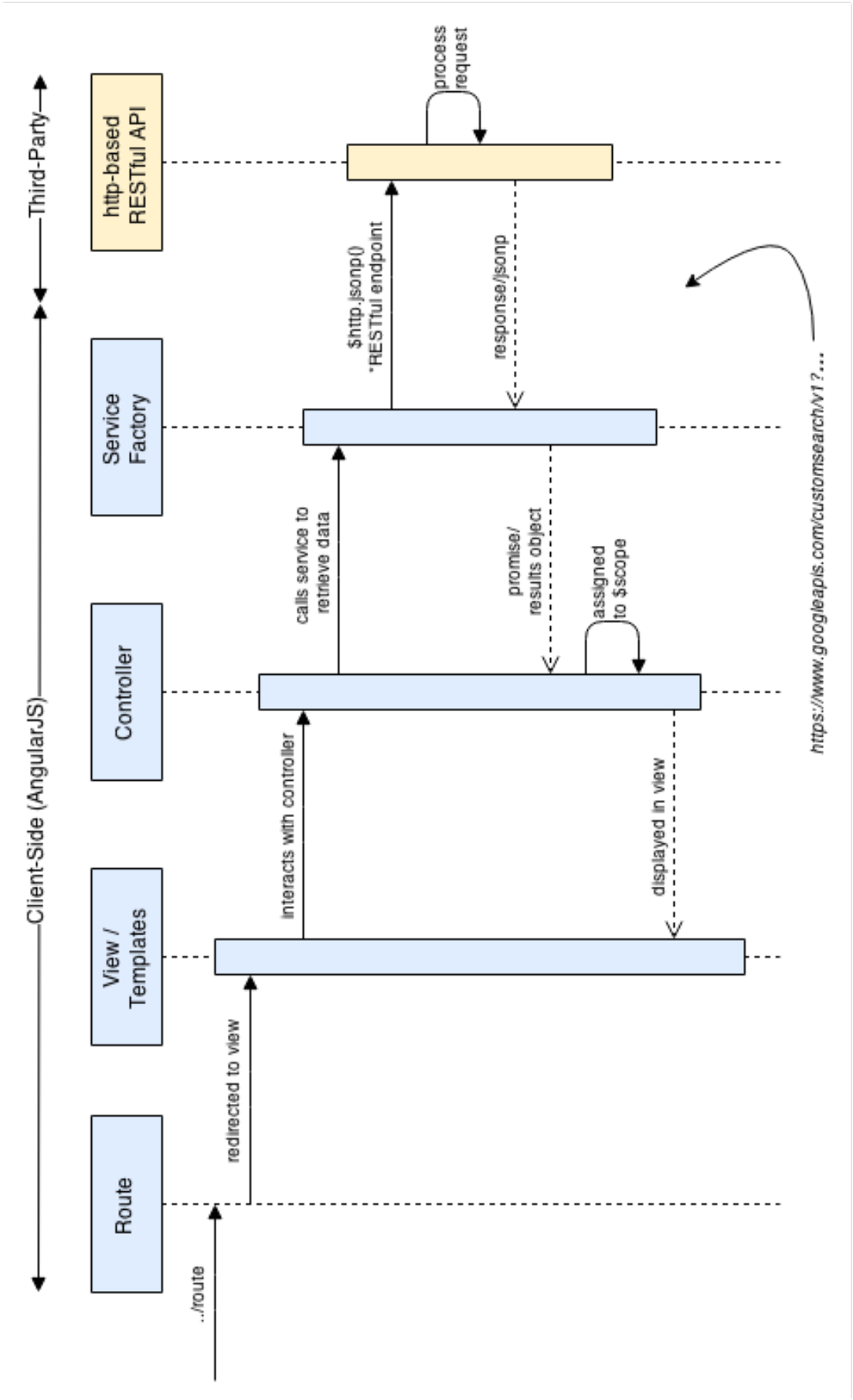
This is example of basic (minimal version) layout structure.

```
1.
2. <!-- Header -->
3. <div id="header" ng-include="'views/common/header.html'"></div>
4.
5. <!-- Navigation -->
6. <aside id="menu" ng-include="'views/common/navigation.html'"></aside>
7.
8. <!-- Main Wrapper -->
9. <div id="wrapper">
10.
11.      <div ui-view ></div>
12.
13.  </div>
14.  </div>
15.  </div>
16.
```

(This tutorial is created by using parts of the documentation of the HOMER template.)

Asynchronous Fetching within AngularJS

AngularJS is fetching its one data asynchronous to improve the user experience. Therefore, when opening the JFrontend, the first requests from angularJS is meant to fetch its own body content and related extension. The following sequence diagram is representing this flow.



(image source: https://programmaticponderings.wordpress.com/tag/angularjs/)