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.

```
1.    app/
2.      ├── fonts/
3.      ├── icons/
4.      ├── images/
5.      ├── js/
6.      ├── less/
7.      ├── scripts/
8.      ├── styles/
9.      ├── templates/
10.     ├──views/
11.         ├── index.html
12.         ├── 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.

```
1.
2. (function () {
3.     angular.module('homer', [
4.         'ui.router',              // Angular flexible routing
5.         'ui.bootstrap',  // AngularJS native directives for Bootstrap
6.         'angular-flot',           // Flot chart
7.         'angles',                 // Chart.js
8.         'angular-peity',          // Peity (small) charts
9.         'cgNotify',               // Angular notify
10.         'angles',                 // Angular ChartJS
11.         'ngAnimate',              // Angular animations
12.         'ui.map',                 // Ui Map for Google maps
13.         'ui.calendar',            // UI Calendar
14.         'summernote',             // Summernote plugin
15.         'ngGrid',                 // Angular ng Grid
16.         'ui.tree'                 // Angular ui Tree
17.     ])
18. })();
19.
```

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 an 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.         })
```
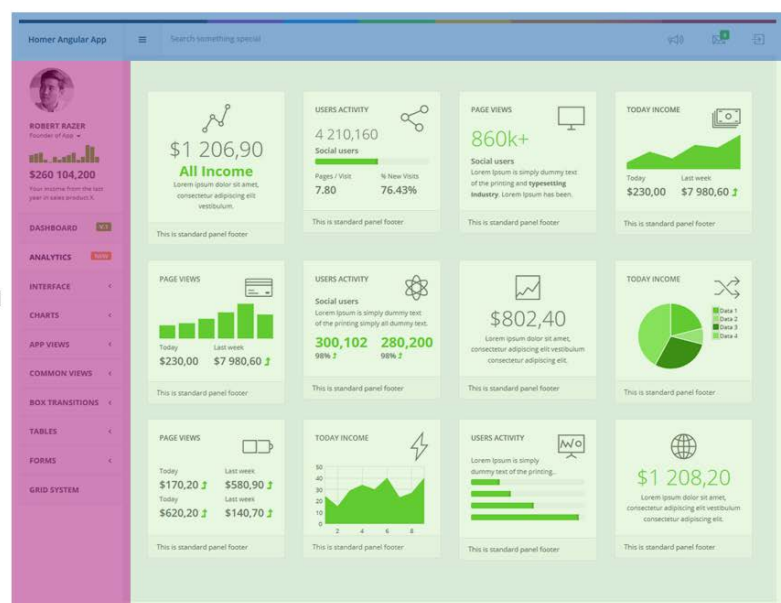
Layout structure

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 is created with few main elements:

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

HEADER - views/common/header.html

ASIDE MENU
views/common/navigation.html

WRAPPER - main ui-view

Layout structure - file

This is an example of basic (minimal version) layout structure.
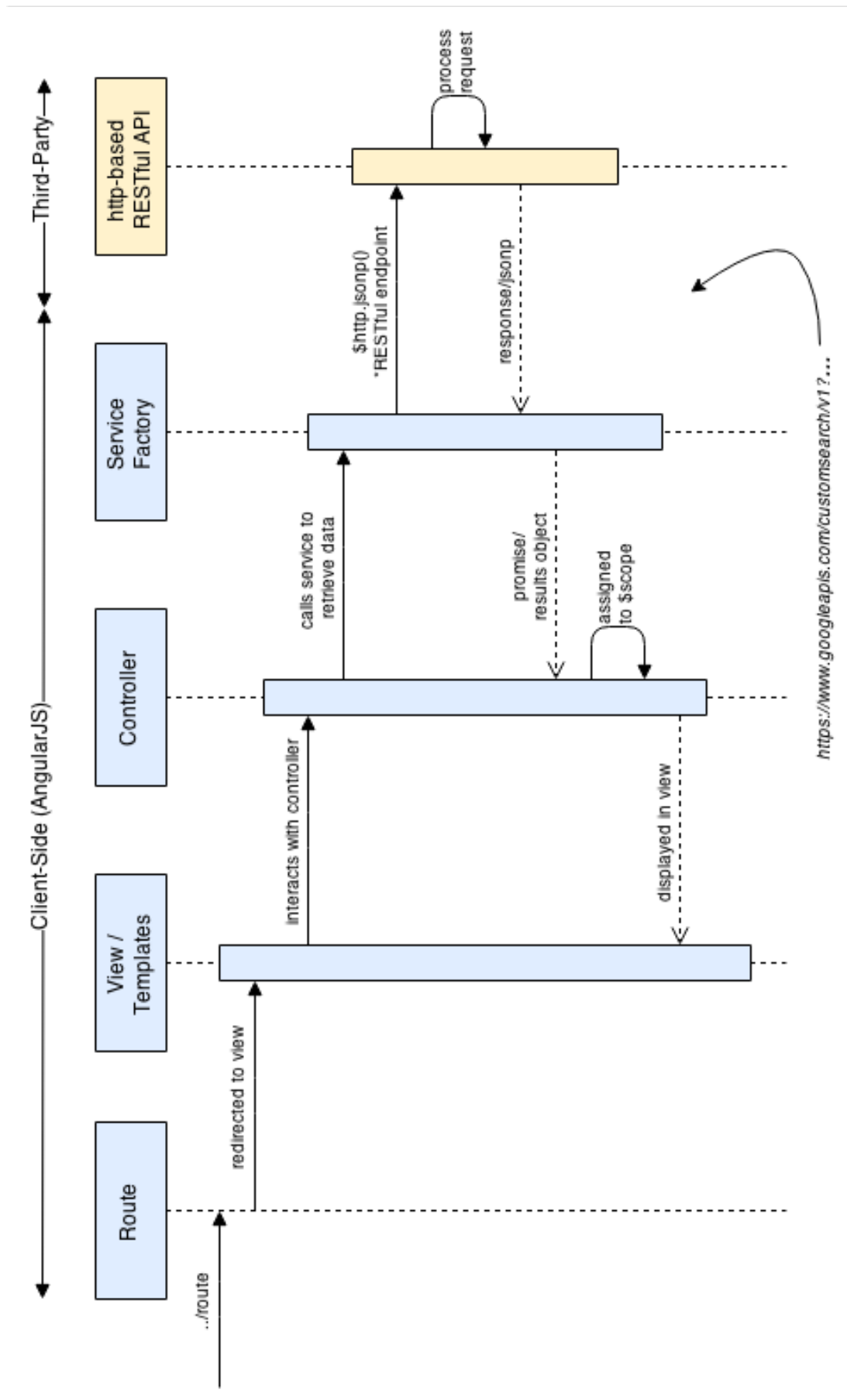
```
1.
2. <!-- Header -->
3. <div id="header" ng-include="'views/common/header.html'"></div>
4.
5. <!-- Navigation -->
6. <aside id="menu" ng-include="'views/common/navigation.html'"></aside>
7.
8. <!-- Main Wrapper -->
9. <div id="wrapper">
10.
11.     <div ui-view ></div>
12.
13. </div>
14. </div>
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 request 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/)