Programmer's Manual
JRC – MYGEOSS AirQ app for viewing AirSensEUR data
project 2016.3701

Table of Contents

1 Technology/Installation

1.1 Introduction

As identified in the Software Architecture Document (SAD), the application uses both Ionic and Cordova. Therefore, the web technologies, HTML, CSS, and JavaScript are used with the AngularsJS framework.

To be able to change, modify, update, or upgrade the application, the programmer must have experience with these common web technologies, along with AngularJS, Cordova and Ionic to both install plugins and build the application for each platform.

Specific hardware requirements for the development include an Apple Mac for the iOS.

1.2 Installation of your workspace

To set up your working environment, you will need to import the following into your working directory:

- Config.xml
- Hooks
- Ionic.project
- Package.json
- Resources
- Scss
- www

Once imported run "`npm install`". This will install all of the modules that ionic needs. Once this has completed run "`ionic prepare`" (or `cordova prepare`). Platforms versions and the plugins version saved in the `config.xml` file will then be installed.

This project uses Sass to generate the CSS files. You can add your own CSS file but the main one `www/css/ionic.app.css` is generated by `./scss/ionic.app.scss`. Please refer to the Ionic documentation to know how to setup Sass on your workspace.

2 Plugins, libraries and platforms

While updating the HTML, CSS, and JavaScript files will, in principle, pose no issue to the application, updates to the platforms and plugins used must be carried out with caution as updates to the actual application may also be required to ensure continued compatibility. This section explains which plugins can, and should, be updated, and the steps to do so safely.

2.1 Platforms

The application uses the Cordova iOS platform v.4.2.1 and the Cordova Android platform v.5.2.2. The developer must evaluate the impact to potentially outdated plugins, Ionic libraries, or the core application, prior to updating the platform.

2.2 Plugins

2.2.1 Plugins list

The list of all Cordova plugins used is saved in the `config.xml` file inside the root folder. These plugins can be found inside the `plugins` folder and are listed below:

```
cordova-plugin-compat 1.0.0 "Compat"
cordova-plugin-console 1.0.3 "Console"
cordova-plugin-datepicker 0.9.3 "DatePicker"
cordova-plugin-device 1.1.2 "Device"
cordova-plugin-geolocation 2.2.0 "Geolocation"
cordova-plugin-inappbrowser 1.4.0 "InAppBrowser"
cordova-plugin-network-information 1.2.1 "Network Information"
cordova-plugin-screen-orientation 1.4.2 "Screen Orientation"
cordova-plugin-splashscreen 3.2.2 "Splashscreen"
cordova-plugin-statusbar 2.1.3 "StatusBar"
cordova-plugin-whitelist 1.2.2 "Whitelist"
cordova-sqlite-storage 1.4.8 "Cordova sqlite storage plugin"
ionic-plugin-keyboard 2.2.1 "Keyboard"
```

2.3 JavaScript / AngularJS modules

Custom added JavaScript libraries/modules and font can be found under the application `www/lib` folder.

ionic-native-transitions.min (https://github.com/shprink/ionic-native-transitions)
leafet.js v0.7.7 (http://leafletjs.com/)
D3.js v3.5.17 (https://d3js.org/)
NVD3.js v3.5.17 (http://nvd3.org/)
Angular-nvd3.min.js v1.0.9 (https://github.com/krispo/angular-nvd3)

2.4 Update

To update a plugin, simply replace the existing file with the new one, keeping the same name.

As the plugins can have many changes, fixes, or even deprecation, the recommendation is to test the application after each individual update. This will ensure the update will not break the application.

For the Cordova plugins, please use the Cordova or Ionic command line tool to add/remove/update a Cordova plugin.

For the Cordova platforms please use the Cordova or Ionic command line tool to add/remove/update a platform.

Command line usage:

https://cordova.apache.org/docs/en/latest/guide/cli/index.html

3 Architecture

The application has been entirely written using a Model View View Model (MVVM) architecture, based upon AngularJS. This allows application dynamic binding, which means that the views can refresh without reloading the application.

3.1 app.js

In the `app.js` file we find:

- The router. Showing the controllers used by the template and the corresponding state.
- The .run function. The code that is first executed.
- The .config functions. Functions used to configure the angular and the ionic defaults values.

3.2 constants.js

This file contains the configuration constants of the application and the static text that can be found in the application.

In this file you can configure which server the app will connect to and the parameters for the alerts.

- jsonServer: The ENDPOINT with all configuration and update for the app
- radiusAlert: Distance to check for alert from the user's position. (in km)
- looktresholdDuring: looking alert duration. (in ms)
- threshold: alert limit value in ug/mg3 (not used anymore)

3.3 directives.js

This file contains the custom directives.

`bindUnsafeHtml`	The directive inside this file enables the data-binding in the template with an HTML code retrieved from an external source.

3.4 controllers.js

Generally, there is one controller per template; please refer to the routing source code in `app.js` to have more details.

Template	Controller
`app.html`	`AppCtrl`
`home.html`	`HomeCtrl`
`about.html`	`AboutCtrl`
`links.html`	`LinksCtrl`
`contact.html`	`ContactCtrl`
`alerts.html`	`AlertsCtrl`
`pollutants.html`	`PollutantsCtrl`
`meteo.html`	`MeteoCtrl`
`network.html`	`NetworkCtrl`
`map.html`	`MapCtrl`
`timeseries.html`	`TimeSeriesCtrl`

Controller	Description
`AppCtrl`	Generic controller for the entire app, used to setup the application and the menu modal.
`HomeCtrl`	Controller for the home page.
`AboutCtrl`	Controller for the about page.
`LinksCtrl`	Controller for the links page.
`ContactCtrl`	Controller for the contact page.

AlertsCtrl	Controller for the alerts page and subscribe to the alert checking event.
PollutantsCtrl	Display the list of selected pollutants parameters.
MeteoCtrl	Display the list of selected weather parameters.
NetworkCtrl	Manage the list of selected air networks.
MapCtrl	Display all the station with the selected parameters on a map.
TimeSeriesCtrl	Display chart and array with all data for a parameter in a station.

3.5 Service.js

This file contains the services of the app. This is the common function and logic of the app.

$AirSensEUR	REST call to the networkd to retrieves data, station, timeseries… (Server API here http://sensorweb.demo.52north.org/sensorwebclient-webapp-stable/api-doc/index.html)
$networksSources	List of selected and default networks sources.
$networkStatus	Check if device is offline/online
$geolocationFactory	Retrieve user's geolocation.
$database	Manage the SQLite local database. (Tables can be found on the Software architecture manual)
$dataConfig	Service to retrieve the config data from the server and update the local data with it.
$alerts	Manage alerts when a threshold for a parameter is reached.
$localStorage	Use the device local storage to store persistent data.

4 Config.xml

This file at the root is used to configure the application. Some of this information can be overridden prior to application store publication by opening the iOS xcodeproject, the android graddle files or the Windows visualstudio project files.

<widget> set the bundle.id and the version number. More information can be found inside the Publications Office document "*Guidelines for mobile apps publishing on the common European Union accounts*".

You can configure some native parameters on each platform <platform name="ios"><platform name="android"> such as the minimum OS version device etc. See the Cordova documentation for this file:

https://cordova.apache.org/docs/en/latest/config_ref/index.html

You can retrieve the platform version and the plugin version from this file. See plugin and platform management from the doc:

https://cordova.apache.org/docs/en/latest/platform_plugin_versioning_ref/

5 Local databases

The application uses an SQLite local database to store persistent data as cached documents or saved documents. This is handled by the `cordova-sqlite-storage` plugin. More information about the fields of the database can be found in the Software Architecture document.

6 Assets

All images and icons are inside the `www/img` folder. The fonts are inside the `www/lib/fonts` folder.

CSS files are generated from SASS. (Please refer to the Ionic document to know how to enable scss in ionic project).

Icons and splashscreens are in the root folder into `./resources`. The `config.xml` file defines the path to these assets for each platform

7 Upgrade

To upgrade the app, once you have modified your files, use the `cordova/ionic` command line tool to provide the files for each platform and build it to provide the native application file. Ensure the same `bundle.id` is in `config.xml` and change the version number.

Command line usage:

https://cordova.apache.org/docs/en/latest/guide/cli/index.html

Configuration information:

https://cordova.apache.org/docs/en/latest/config_ref/index.html

Plugin and platform versioning:

https://cordova.apache.org/docs/en/latest/platform_plugin_versioning_ref/

8 Updating data

For each update of the files you will need to add a new timestamp in the update.json file: “lastupdate” key. The app will update only if the “lastUpdate” value has changed. All other keys are optional

The JSON format needs to be valid, although valid JSON does not ensure that the data are compatible with the app.

“ char in text needs to be escaped with: \”

Wrongly formatted JSON can crash the application.

8.1 Different files

8.1.1 generalContent.json

In this file you can modify the content of the “About page”, “Links page” and “Contact page” of the app. (You cannot add new values as the app will not evaluate new keys on this file).

The description field is what will be displayed within the application. You can use HTML tags to add formatting. (for example:
 for line break).

To add links you need to write a custom attribute on an HTML <a> tag:

```
<a ng-click=\"openExternalLink('http://YOUR_LINK')\">TEXT</a>
```

To open mails application when user click on an email address:

```
<a                ng-click=\"openExternalLink('mailto:YOUR_MAIL@MAIL.COM')\">
mygeoss@jrc.ec.europa.eu</a><br/>"
```

8.1.2 airParameters.json

In this file you manage the air parameter descriptions such as weather (meteo) and pollutants. This is what the user sees when he visits the “Meteoroligcal params.” or “Air pollutants” page.

- Pollutants:

```
{
      "id": "11",
     "label": "Nitrogen monoxide",
     "formula": "NO",
     "unit": "ppm",
     "source": "High temperature combustion processes (such as those occurring in car engines and power plants) are the major sources of nitrogen oxides, the term used to describe the sum of NO and NO2. In occupational situations both gases are mostly encountered together, with higher concentrations of NO which is the main component of the NOx emissions. NO is also manufactured for the production of nitric acid used in the synthesis of nitrate fertilisers. It is also used in nitration reactions and as a respiratory stimulant in hospital intensive care therapy.",
     "effects": "Unlike NO2, NO is not an irritant gas. No major role is attributed to an oxidation of inhaled NO into NO2 in the lungs since, after inhalation, NO is eliminated faster than it is oxidised. Whereas emphysema-like alterations are the main acute toxic effects of NO2, vasodilatory effects and, at high concentrations, methaemoglobin (MetHb) formation, are observed in the case of NO but at concentration levels in the mg/m³ range that are generally not observed in ambient air.",
     "limitValues": "Neither the European air quality directive nor the World Health Organisation have set any limit value for NO while the occupational exposure limits for human working conditions of NO corresponds to 2 ppm (parts per million) (8-hour, time weighted average).",
     "description": "",
     "icon": "chemistry"
}
```

- Meteorological parameter

```
{
     "label": "Temperature",
     "id": "14",
     "unit":  "(in degrees centigrade)",
     "description": "",
     "icon": "thermometer"
}
```

"label" has to exactly match the label used in monitoringNetworks.json file.

"id": The id of the phenomena in the JRC network.

"formula": (only pollutant). The formula displayed on the "Air pollutants" page.

"unit": Only the unit displayed on "Meteoroligcal params." and "Air pollutants" pages.

"source": (only pollutant). Text displayed on the "Air pollutants" page.

"effects": (only pollutant). Text displayed on the "Air pollutants" page.

"limitValues": (only pollutant). Text displayed on the "Air pollutants" page.

"Description": Not used for the moment. Could be used to add additional description on "Meteoroligcal params." and "Air pollutants" pages.

"icon": Name of the icon displayed description on "Meteoroligcal params." and "Air pollutants" pages. (Further information on how to change the icon can be found in Section "Change air parameters icons and network flag icons" .)

8.1.3 monitoringNetworks.json

This file manages all the different networks for the data sources and the mapping for each parameter. (Air parameters have different IDs on different networks)

```
{
        "APIurl": "http://sossvr1.liberaintentio.com:8080/52nSOS/api/v1/",
        "label": "AirSensEUR (JRC Ispra)",
        "flag": "European-Union.png",
        "defaultSelected": "true",
        "airParamsMapping": {
            "Nitrogen monoxide": {
                "id": "11",
                "distance": "",
                "treshold": "",
                "tresholdUnit": "",
                "averagingPeriod": "",
                "text": ""
         }
}
```

"APIurl": The base URL of the network.

"label": Identifier of the network as it will be displayed on the "Monitoring networks" page.

"flag": Name of the flag icon displayed on the "Monitoring networks" page. (Further information on how to change the icon can be found in Section "Change air parameters icons and network flag icons")

"defaultSelected": "true" or "false". Define the network always selected by default, it cannot be unselecting by the user. Adding more than one defautSelected network could cause issues.

"airParamsMapping": All the air parameters with the id corresponding in the network and the information for the alerts. (Further information on how to map and add an air parameter can be found in Section "Adding new air parameters")

8.2 Adding new air parameters

To add a new air parameter you need to first add it in the airParamerters.json file, in the "pollutants" or "meteo" section.

Then you have to add it in the monitoringNetworks.json file **for each network**.

```
"airParamsMapping": {
            "Nitrogen dioxide": {
                "id": "12",
                "distance": "0.100",
                "treshold": "400",
              "tresholdUnit": "mg/m3",
                "averagingPeriod": "3 hours",
                "text":  "A concentration of @@pollutant@@ exceeding @@treshold@@ was observed near you. It might breach this Infomation treshold defined by the European Union Air Quality Directive (2008/50/EC) if the remains above @@treshold@@ for @@averagingPeriod@@."
            }
}
```

The air parameter key, here "Nitrogen dioxide", **needs to be exactly the same as the label in the airParameters.json file and also needs to be exactly the same in each network, even if the parameter does not exist in that network. In this case though the parameter can be left empty.**

If at least one network defines a value for "distance", "threshold", "averagingPeriod" or "text", you need to define values for this parameter in each network.

"id": ID of the parameter in the network. (Leave it empty if it does not exist in the network)

"distance": Distance for the alerts in km. (Leave it empty "" if there are no alerts for this parameters)

"threshold": Threshold for the alert in µg/m³

"averagingPeriod": Averaging period of exposure displayed in the alerts.

"text": Text of the alert. Variable "pollutant", "treshold" and "averagingPeriod" need to be encapsulate with @@variable@@.

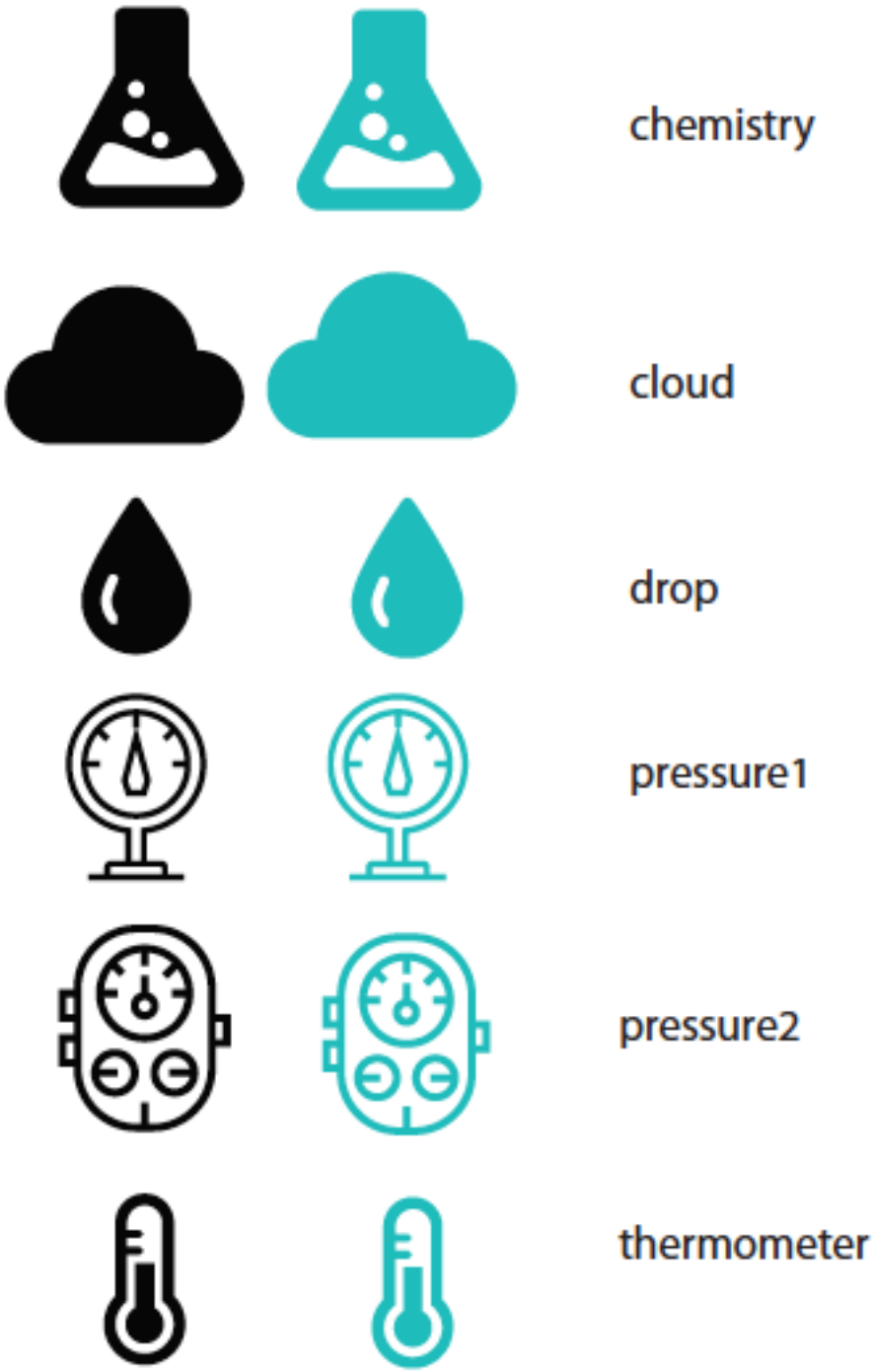
8.3 Adding a new monitoring network

To add a new monitoring network copy and paste a previously added monitoring network from the monitoringNetworks.json file. You need to add each of the air parameters, even if the network does not use all the parameters, although these can be left empty "".

8.4 Change air parameters icons and network flag icons

To change the icon of one of the air parameters, replace the “icon” name in the airParameters.json with one of the available icons on the app.

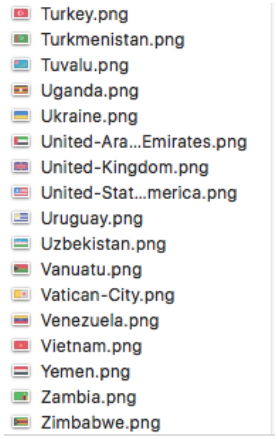
Available icons:



To change the flag icons of the monitoring networks, replace the “flag” label in the monitoringNetworks.json file with the country flag you would like to use and add .png.

Available flags:


Afghanistan.png
Albania.png
Algeria.png
Andorra.png
Antigua-an...Barbuda.png
Argentina.png
Armenia.png
Australia.png
Austria.png
Azerbaijan.png
Bahamas.png
Bahrain.png
Bangladesh.png
Barbados.png
Belarus.png
Belgium.png
Belize.png
Benin.png
Bhutan.png
Bolivia.png
Bosnia-and...egovina.png
Botswana.png
Brazil.png
Brunei.png
Bulgaria.png
Burkina-Faso.png
Burundi.png
Cambodia.png
Cameroon.png
Canada.png
Cape-Verde.png
Central-Afri...epublic.png
Chad.png
Chile.png
China.png
Colombia.png
Comoros.png
Congo-(Democratic).png
Congo-(Republic).png
Costa-Rica.png
Cote-d'Ivoire.png
Croatia.png
Cuba.png
Cyprus.png
Czech-Republic.png
Denmark.png
Djibouti.png
Dominica.png
Dominican-Republic.png
East-Timor.png
Ecuador.png
Egypt.png
El-Salvador.png
Equatorial-Guinea.png
Eritrea.png
Estonia.png
Ethiopia.png
European-Union.png
Fiji.png
Finland.png
Finland.png
France.png
Gabon.png
Gambia.png
Georgia.png
Germany.png
Ghana.png
Grecee.png
Grenada.png
Guatemala.png
Guinea-Bissau.png
Guinea.png
Guyana.png
Haiti.png
Honduras.png
Hungary.png
Iceland.png
India.png
Indonesia.png
Iran.png
Iraq.png
Ireland.png
Israel.png
Italy.png
Jamaica.png
Japan.png
Jordan.png
Kazakhstan.png
Kenya.png
Kiribati.png
Korea,-North.png
Korea,-South.png
Kosovo.png
Kuwait.png
Kyrgyzstan.png
Laos.png
Latvia.png
Lebanon.png
Lesotho.png
Liberia.png
Libya.png
Liechtenstein.png
Lithuania.png
Luxembourg.png
Macedonia.png
Madagascar.png
Malawi.png
Malaysia.png
Maldives.png
Mali.png
Malta.png
Marshall-Islands.png
Mauritania.png
Mauritius.png
Mexico.png
Micronesia-...erated).png
Moldova.png
Monaco.png
Mongolia.png
Montenegro.png
Morocco.png
Mozambique.png
Myanmar.png
Namibia.png
Nauru.png
Nepal.png
Netherlands.png
New-Zealand.png
Nicaragua.png
Niger.png
Nigeria.png
Norway.png
Oman.png
Pakistan.png
Palau.png
Panama.png
Papua-New-Guinea.png
Paraguay.png
Peru.png
Philippines.png
Poland.png
Portugal.png
Qatar.png
Romania.png
Russia.png
Rwanda.png
Saint-Kitts-...d-Nevis.png
Saint-Lucia.png
Saint-Vince...nadines.png
Samoa.png
San-Marino.png
Sao-Tome-...Principe.png
Saudi-Arabia.png
Senegal.png
Serbia.png
Seychelles.png
Sierra-Leone.png
Singapore.png
Slovakia.png
Slovenia.png
Solomon-Islands.png
Somalia.png
South-Africa.png
South-Sudan.png
Spain.png
Sri-Lanka.png
Sudan.png
Suriname.png
Swaziland.png
Sweden.png
Switzerland.png
Syria.png
Taiwan.png
Tajikistan.png
Tanzania.png
Thailand.png
Togo.png
Tonga.png
Trinidad-and-Tobago.png
Tunisia.png



9 General comments

More information about the logic can be found in comment inside the source code, and the function and variable have explicit names.

9.1 Charts

To display the timeseries data, we use the angular nvd3 js library that provide an Angular friendly way to use the NDV3 js libray based on the D3.js library.

The default windows to diplay the data is 24 hours.

9.2 Alerts

The app call a function checkAlert at an interval of 30s to check new alerts in the user area.

END OF DOCUMENT

EPUB_Lot1_MAN_2016.3701_PM_v010