Sii-Mobility Dev App Kit

Aula caminetto, Scuola di Ingegneria, Firenze

24/01/2017

Ing. Claudio Badii

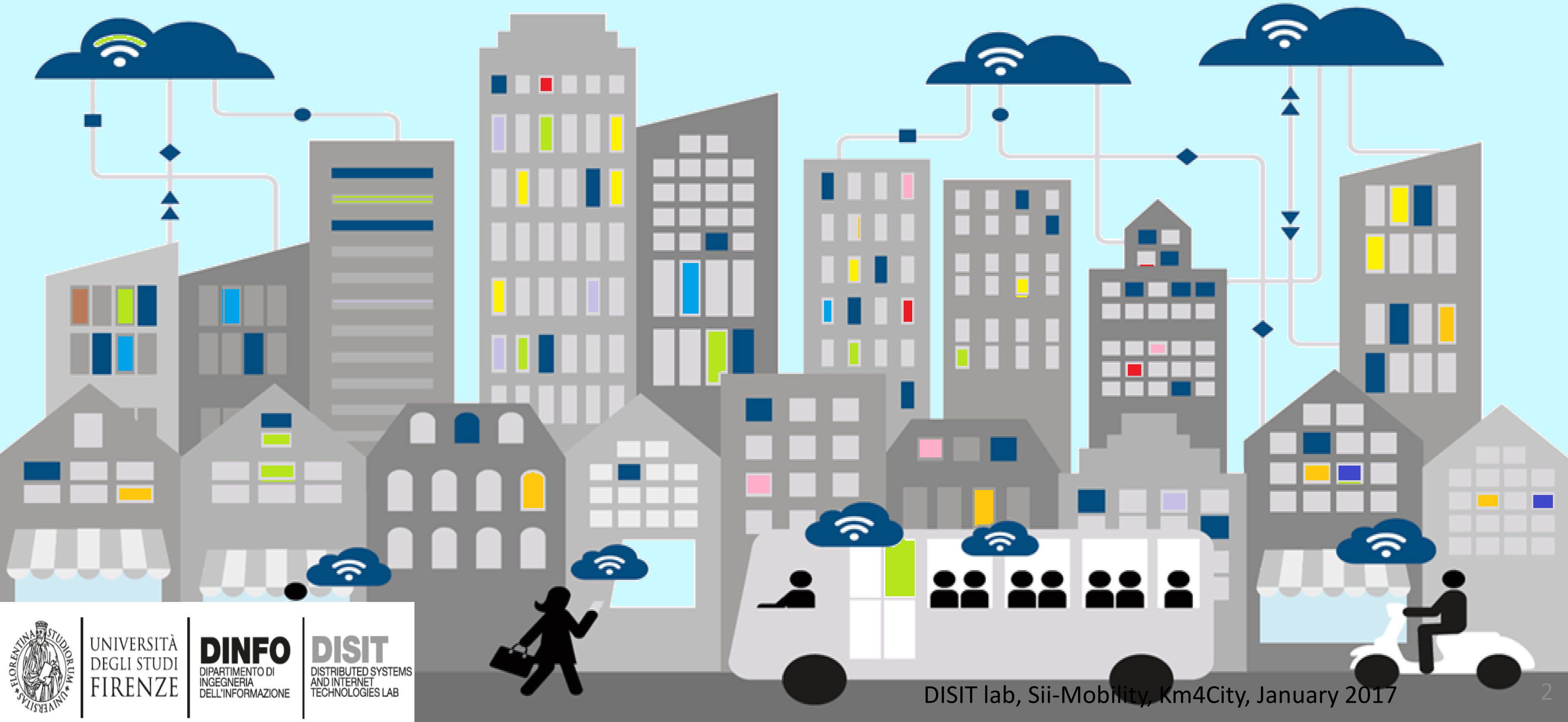
Dipartimento di Ingegneria dell’Informazione, DINFO
Università degli Studi di Firenze
Via S. Marta 3, 50139, Firenze, Italy
Tel: +39-055-2758515,   fax: +39-055-2758570

DISIT Lab

http://www.disit.org

For help desk send email to: km4city@disit.org

Mobile App Kit



UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB

Agenda

- 9:00-10:30 http://www.disit.org/6993
  - Sii-Mobility Overview
  - General Model
  - Engaging Users via Apps
  - Overview of development tools for Apps
  - How to Legally work with SDK
  - Planned Additional Modules for the Apps
  - Info and Documents
- 10:30-11:30
  - ServiceMap usage http://www.disit.org/6994
  - Smart City API http://www.disit.org/6995
- 11:00-12:30 http://www.disit.org/6992
  - Sii-Mobility Mobile App Kit on GitHub
  - How to develop new module
- 12:30-13:30: lunch
- 13:30-17:00:
  - Workshop on development: exercitations

Mobile Apps

- **Usually** when we think of **applications** for the mobile devices, we think of **native applications**, that is downloaded and installed on a specific device.

- In fact the development of **native applications** still remains the **best way** to fully exploit the **potential** of smartphones and tablets and offer integrated and consistent user experience between all applications.

- The **problem** with this approach is that, to support more than one operating system, is necessary to develop **different applications**, thus **multiplying the implementation** code and **times to maintain**.

- The best answer to this problem now comes with the **framework for mobile web applications**, that have as their primary objective to provide a user experience as close as possible to the native while ensuring support for a **large number of devices** and operating systems.

APACHE CORDOVA™

- **Apache Cordova** is a set of **JavaScript APIs** that enable the devices to the application developer to access native features of the device such as the camera or accelerometer, storage, network, gps ....
- Combined with a user **interface framework** such as Dojo Mobile or jQuery Mobile or Sencha Touch, allows the development of smartphone applications using only **HTML, CSS and JavaScript**.
- When using the Cordova API, an application can be built without any native code (Java, Objective-C, C# etc.). The **web technologies** used are **hosted in the same application** at the local level (usually not on a remote http server).
- These **JavaScript API** are **consistent** and **valid** for the **different platforms** of mobile devices, in this way the application built on the Web standard, should be **portable** with a **minimum of changes**.

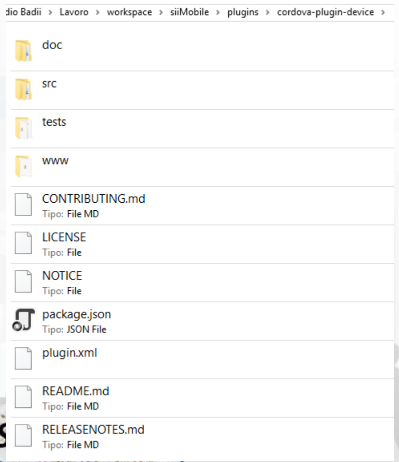
APACHE CORDOVA™

- **Cordova-based apps** are implemented as package applications using the SDK platform, and can be made available for installation from the App Store for each device.
- Cordova offers a set of **JavaScript libraries** that can be invoked **transparently** using the native code of the device.
- Cordova is available for iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, Symbian etc.
- Documentation and downloads are available on the official website: http://cordova.apache.org/

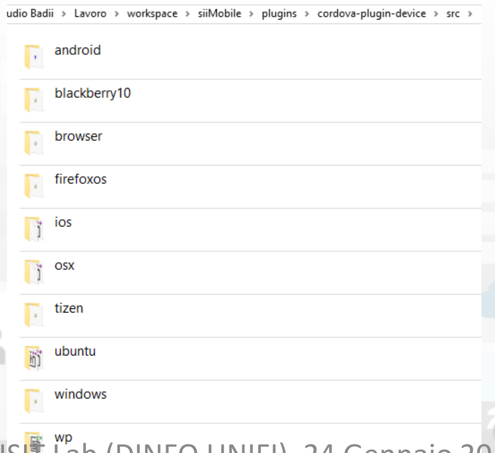
Apache Cordova™ plugin

- cordova-plugin-battery-status: 1.2.1
- cordova-plugin-camera: 2.3.1
- cordova-plugin-console: 1.0.5
- cordova-plugin-contacts: 2.2.1
- cordova-plugin-device: 1.1.4
- cordova-plugin-device-motion: 1.2.3
- cordova-plugin-device-orientation: 1.0.5
- cordova-plugin-dialogs: 1.3.1
- cordova-plugin-file: 4.3.1
- cordova-plugin-file-transfer: 1.6.1
- cordova-plugin-geolocation: 2.4.1
- cordova-plugin-globalization: 1.0.5
- cordova-plugin-legacy-whitelist: 1.1.2
- cordova-plugin-media: 2.4.1
- cordova-plugin-media-capture: 1.4.1
- cordova-plugin-network-information: 1.3.1
- cordova-plugin-splashscreen: 4.0.1
- cordova-plugin-statusbar: 2.2.1
- cordova-plugin-test-framework: 1.1.4
- cordova-plugin-vibration: 2.1.3
- cordova-plugin-whitelist: 1.3.1
- cordova-plugin-wkwebview-engine: 1.1.1

It is possible to create own plugins following the guide for development.


dio Badii › Lavoro › workspace › siiMobile › plugins › cordova-plugin-device ›

doc
src
tests
www
CONTRIBUTING.md
Tipo: File MD
LICENSE
Tipo: File
NOTICE
Tipo: File
package.json
Tipo: JSON File
plugin.xml
README.md
Tipo: File MD
RELEASENOTES.md
Tipo: File MD


udio Badii › Lavoro › workspace › siiMobile › plugins › cordova-plugin-device › src ›

android
blackberry10
browser
firefoxos
ios
osx
tizen
ubuntu
windows
wp

Mustache JS

- The library is **independent** from specific framework but there are plugins for the integration with jQuery, Dojo, and YUI.
- Possibility to work with **javascript objects** and then exploit the communication of **data** in **JSON format from a REST** call via AJAX.
- The **templates** for Mustache may be assigned or loaded as a string to a variable and the placeholder are identified by two braces, for example: {{miopplaceholder}}.
- One of the most interesting of the library feature is support in **enumerable values**
- Documentation and downloads are available on the official website: http://mustache.github.io

Mustache JS

Template

```
<h1>{{titolo}}</h1>
<p>{{descrizione}}</p>
{{#risultato}}  //solo se risultato è true
            <ul>{{#citta}}
               <li> {{nome}} ({{sigla}})</li>
             {{/citta}}</ul>
 {{/risultato}}
 {{^risultato}}  //altrimenti...
            <p><em>Nessuna città trovata!</em></p>
{{/risultato}}
```

JSON

```
var data = {
   risultato: true,
   titolo: Città italiane,
   descrizione: Lista delle città italiane,
   citta: [
      {nome: Milano, sigla: MI},
      {nome: Roma, sigla: RM}
   ]
};
```

JSON

```
var data2 = {
   risultato: false,
   titolo: Città italiane 2,
   descrizione: Lista delle città italiane 2,
   citta: []
};
```

Mustache JS

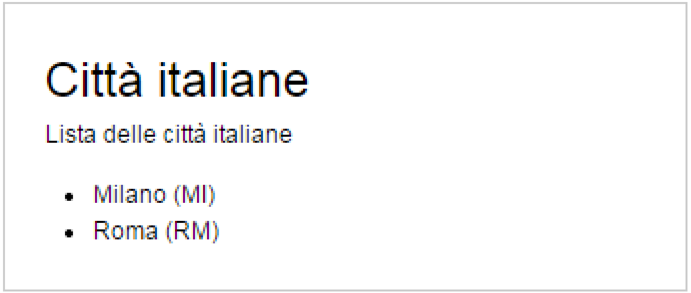
Template →

```
<h1>{{titolo}}</h1>
<p>{{descrizione}}</p>
{{#risultato}}  //solo se risultato è true
            <ul>{{#citta}}
               <li> {{nome}} ({{sigla}})</li>
             {{/citta}}</ul>
 {{/risultato}}
 {{^risultato}}  //altrimenti...
            <p><em>Nessuna città trovata!</em></p>
{{/risultato}}
```

JSON →

```
var data = {
   risultato: true,
   titolo: Città italiane,
   descrizione: Lista delle città italiane,
   citta: [
      {nome: Milano, sigla: MI},
      {nome: Roma, sigla: RM}
   ]
};
```

Template + JSON + Mustache



Città italiane

Lista delle città italiane

- Milano (MI)
- Roma (RM)

Mustache JS

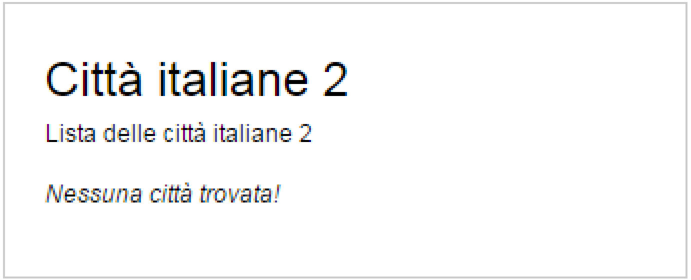
Template

```
<h1>{{titolo}}</h1>
<p>{{descrizione}}</p>
{{#risultato}}  //solo se risultato è true
            <ul>{{#citta}}
               <li> {{nome}} ({{sigla}})</li>
             {{/citta}}</ul>
 {{/risultato}}
 {{^risultato}}  //altrimenti...
            <p><em>Nessuna città trovata!</em></p>
{{/risultato}}
```

JSON

```
var data2 = {
   risultato: false,
   titolo: Città italiane 2,
   descrizione: Lista delle città italiane 2,
   citta: []
};
```

Template + JSON + Mustache



Città italiane 2

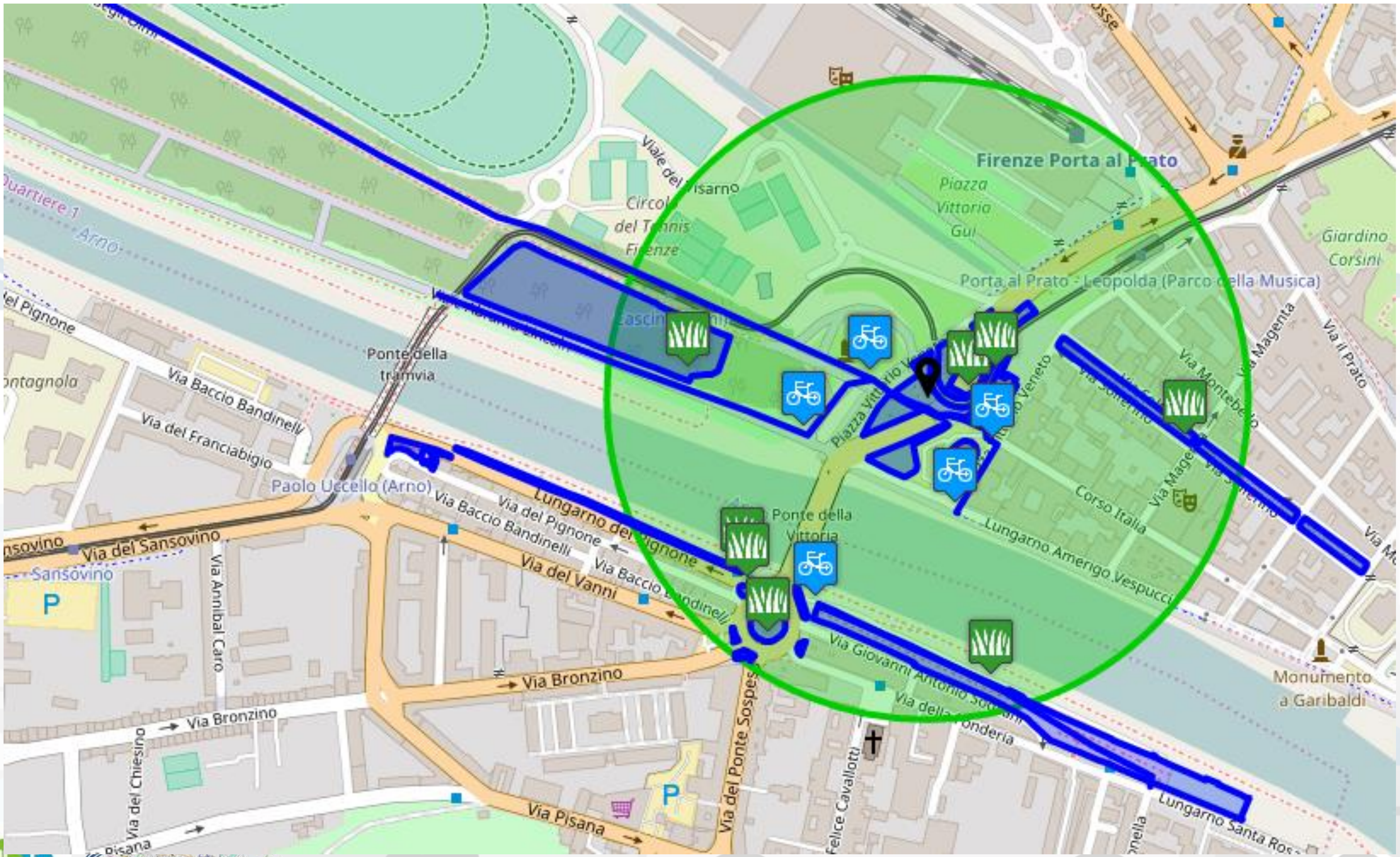
Lista delle città italiane 2

Nessuna città trovata!

OpenLayers 3.0

- **OpenLayers** is an open source **JavaScript library** for **displaying map data** in web browsers and can be used with a hybrid application developed with Cordova
- In the **early versions** of the app, the map was managed by **Leaflet.js** library. This was replaced because it didn't support the rotation, which is required to insert navigation functions within the app
- In addition, OpenLayers 3.0 builds the map and objects added to it with a **canvas** renderer, which is **very efficient** when objects are **numerous and small** as the markers displayed for each search done with the app
- Documentation and downloads are available on the official website: **http://openlayers.org**

OpenLayers 3.0

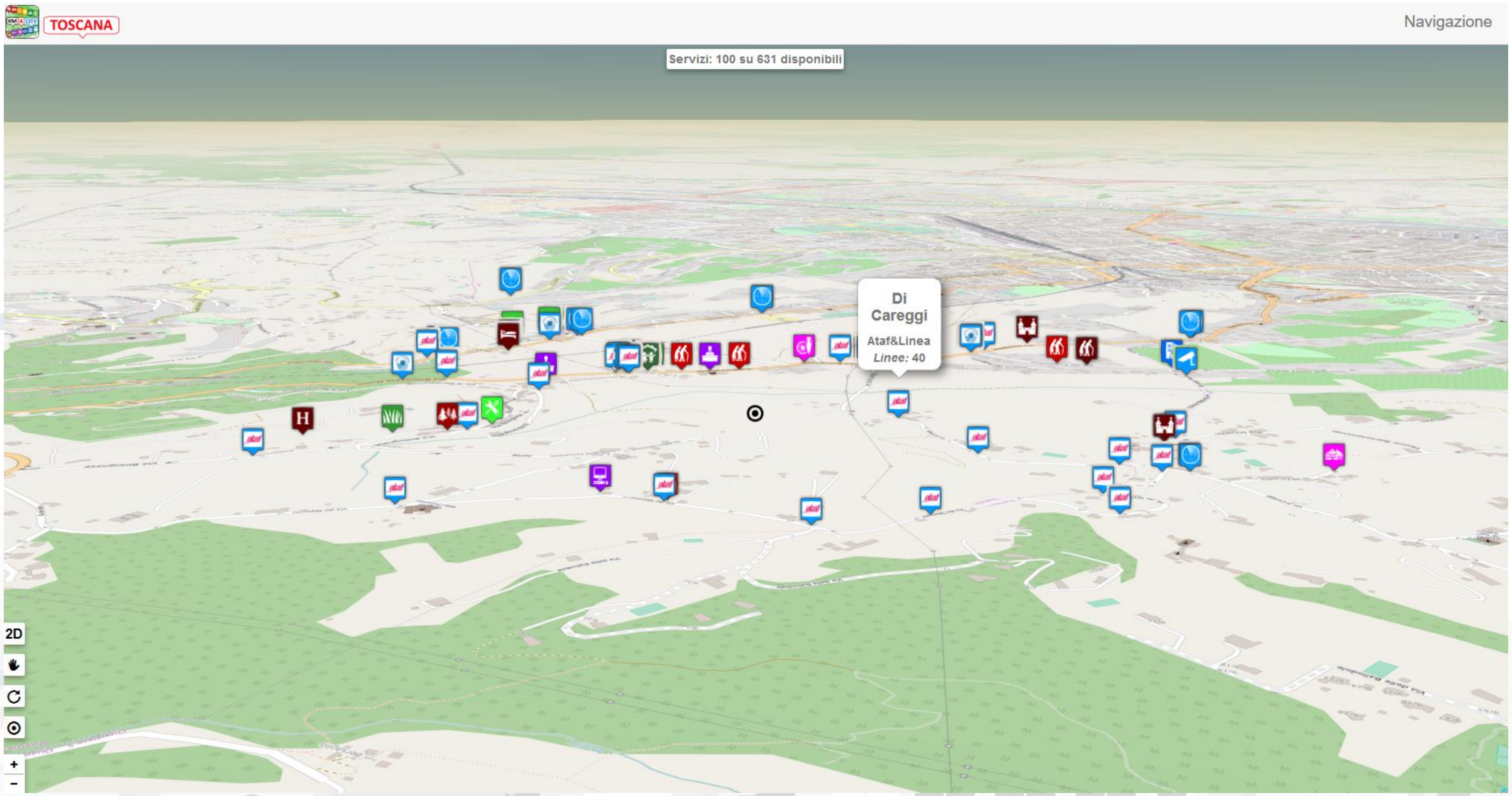


OL3-Cesium

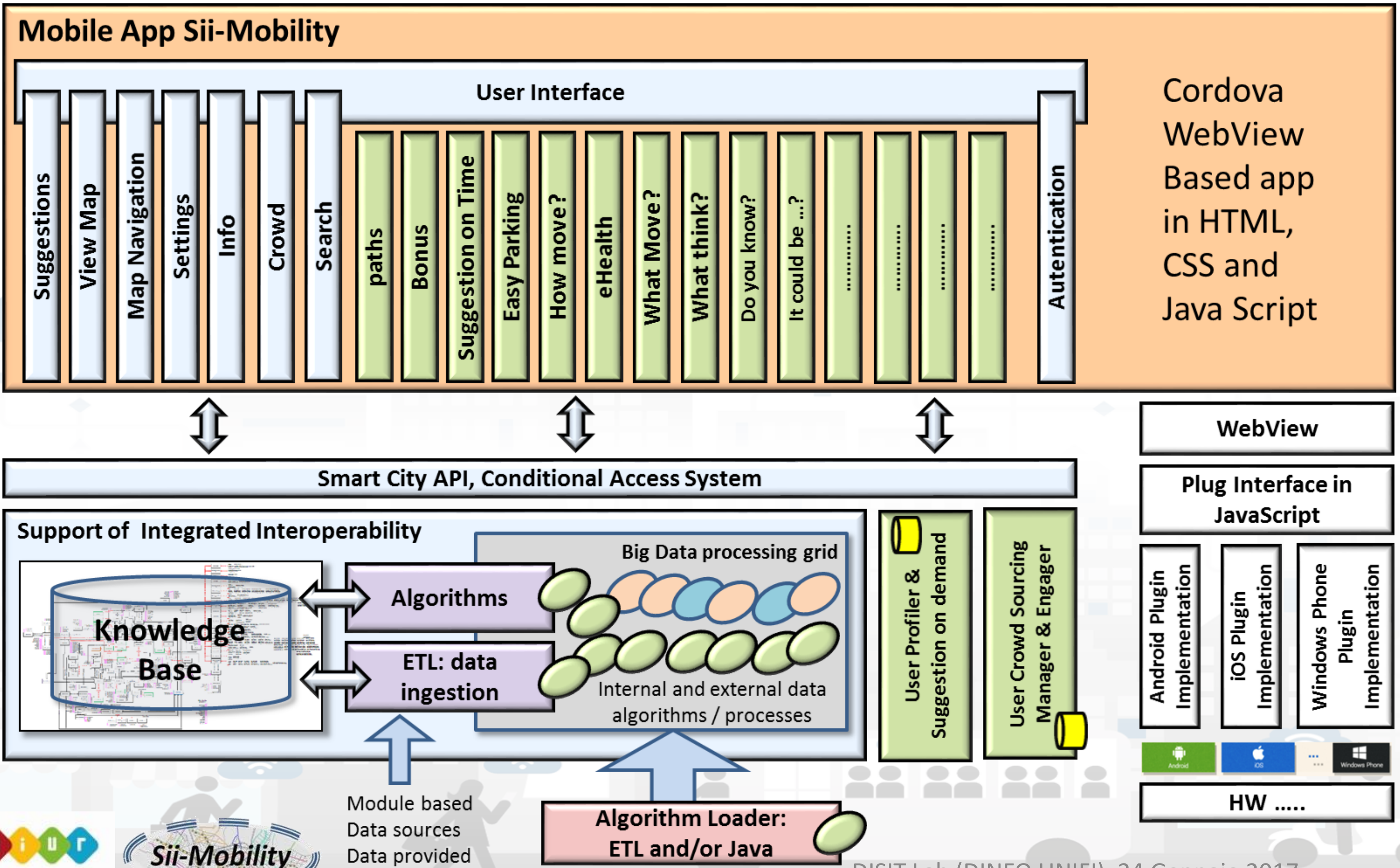
Third dimension for OpenLayers

- **Cesium** is an **open-source JavaScript library** for world-class **3D globes and maps**. Our mission is to create the leading 3D globe and map for static and time-dynamic content, with the best possible performance, precision, visual quality, platform support, community, and ease of use.
- To use Cesium over map created with OpenLayers using the library **OL3-Cesium** (developed by OpenLayers's Team) that allows you to switch easily from 2D to 3D.
- The **3D version** of the map is displayed only on **devices able to execute** it, and only in **navigation mode**. This choice at the moment is due to the high utilization of resources by Cesium and some problems in library OL3-Cesium.
- Documentation and downloads are available on the official website: https://cesiumjs.org and http://openlayers.org/ol3-cesium/

OL3-Cesium
Third dimension for OpenLayers



General architecture



Mobile App Sii-Mobility
User Interface
Suggestions
View Map
Map Navigation
Settings
Info
Crowd
Search
paths
Bonus
Suggestion on Time
Easy Parking
How move?
eHealth
What Move?
What think?
Do you know?
It could be ...?
Autentication
Cordova WebView Based app in HTML, CSS and Java Script
WebView
Plug Interface in JavaScript
Android Plugin Implementation
iOS Plugin Implementation
Windows Phone Plugin Implementation
HW .....
Smart City API, Conditional Access System
Support of Integrated Interoperability
Knowledge Base
Algorithms
ETL: data ingestion
Big Data processing grid
Internal and external data algorithms / processes
User Profiler & Suggestion on demand
User Crowd Sourcing Manager & Engager
Module based Data sources Data provided
Algorithm Loader: ETL and/or Java
Sii-Mobility

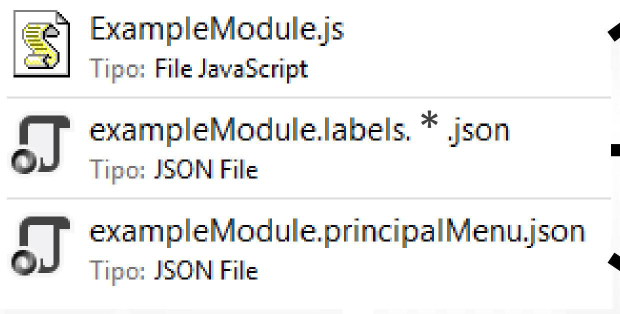
Create ParkingSearcher Module

In the slides following there is an **example** of how to **add a module** to the app.

The goal of this example is to create a **new module** that in addition to viewing the list of car parks as is already the case for the button named “Parking" will **show directly** the **number of free parking lots** for each car park found

Create ParkingSearcher Module

- **Files required** for creating a new module are as follows


ExampleModule.js
Tipo: File JavaScript
exampleModule.labels. * .json
Tipo: JSON File
exampleModule.principalMenu.json
Tipo: JSON File

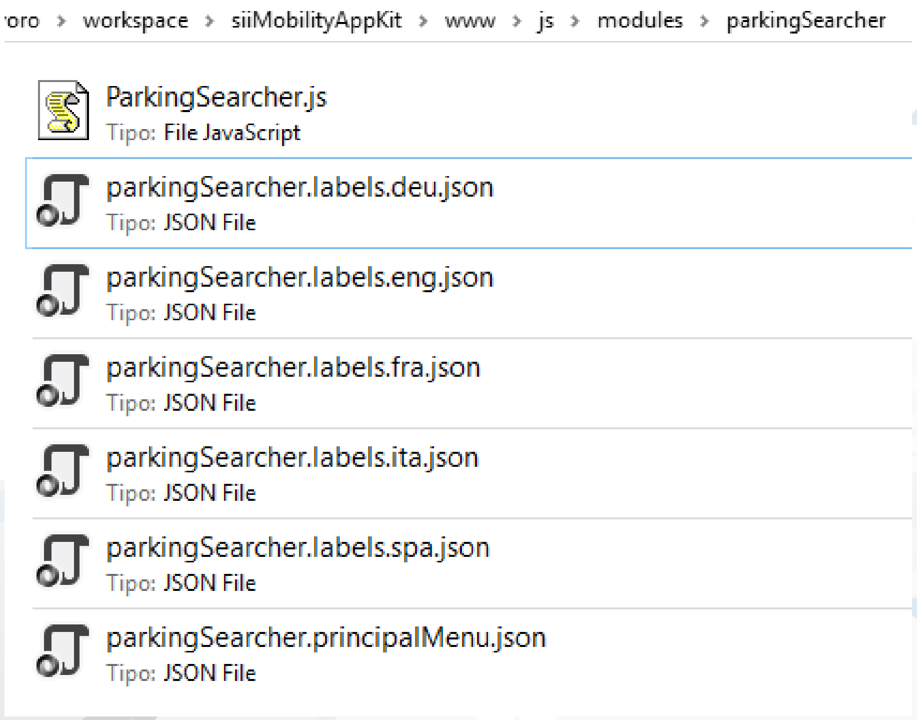
A **Javascript** file containing the **logic**

5 JSON files (**ita**, **eng**, **esp**, **deu**, **fra**) containing **labels** to be included in the new interface

A JSON file that contains one or more **buttons** to be added to **principal menu** to allow the user to interact with the newly created module

Create ParkingSearcher Module

- Copy these files to a **new folder** that will have the **name of the new module** (i.e., **ParkingSearcher**): the **names of the files** copied have to be changed to get the **module name as a prefix**


oro > workspace > siiMobilityAppKit > www > js > modules > parkingSearcher

ParkingSearcher.js
Tipo: File JavaScript

parkingSearcher.labels.deu.json
Tipo: JSON File

parkingSearcher.labels.eng.json
Tipo: JSON File

parkingSearcher.labels.fra.json
Tipo: JSON File

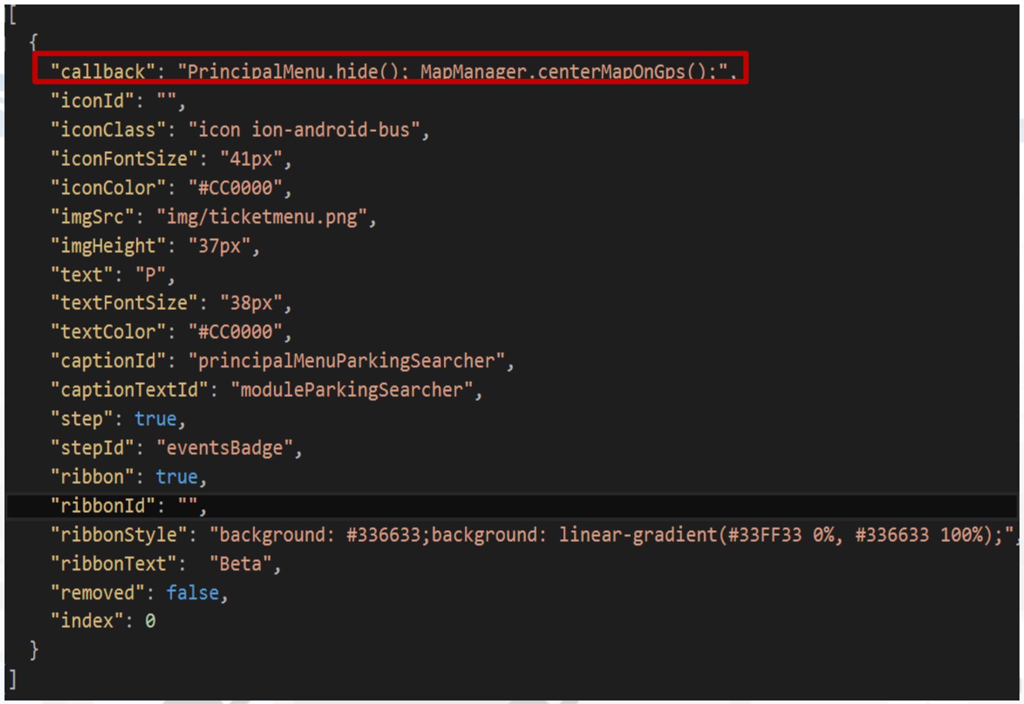
parkingSearcher.labels.ita.json
Tipo: JSON File

parkingSearcher.labels.spa.json
Tipo: JSON File

parkingSearcher.principalMenu.json
Tipo: JSON File

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

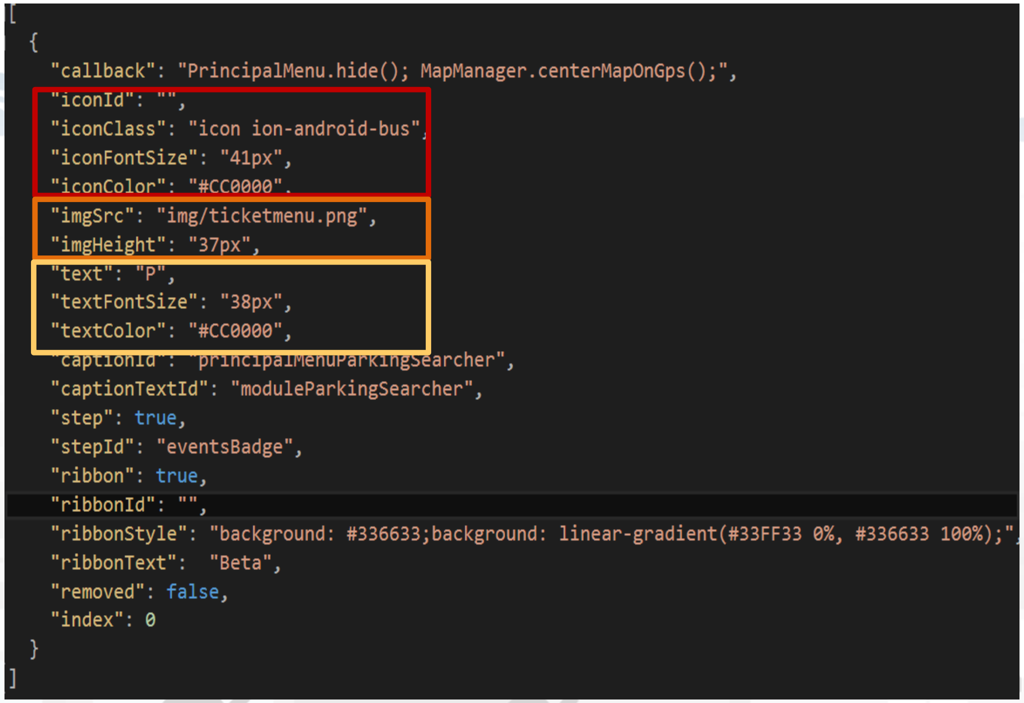
parkingSearcher.principalMenu.json

This field contains the **callback** for the new module.

The present callbacks should be left, because they serves to **close the main menu** and to **center the map on the GPS**

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

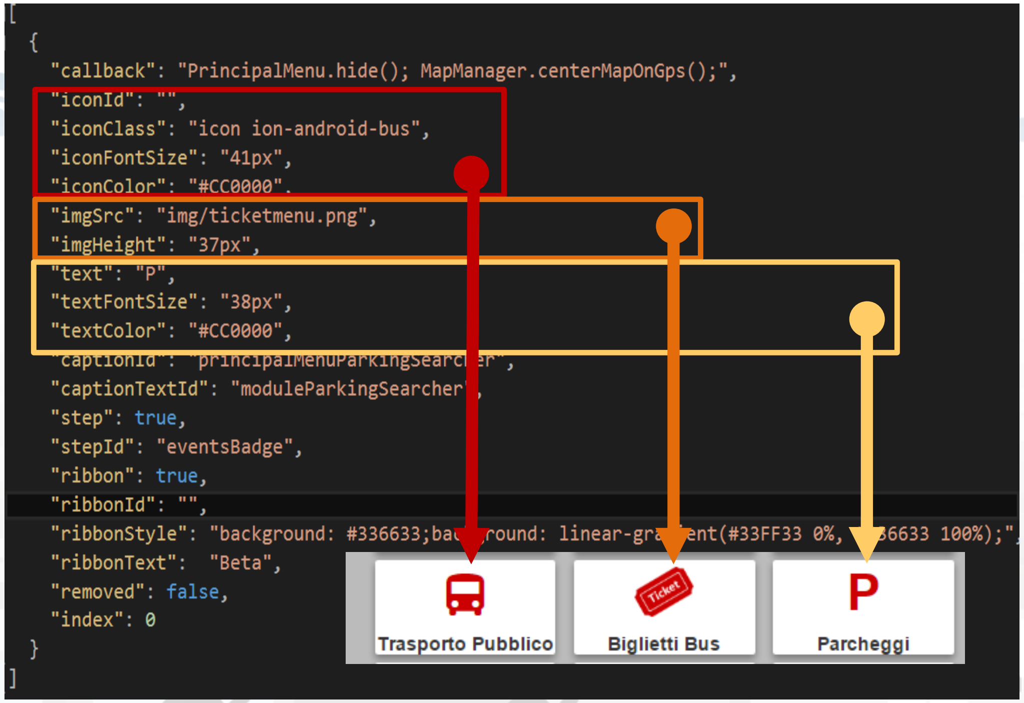
parkingSearcher.principalMenu.json

These blocks of fields are **mutually exclusive**. Allow you to choose the icon that will identify the button that you are creating. This icon can be chosen as an **image**, a **text**, a **glyphicon** (Bootstrap) or **ionicons** (ionicons.com).
N.B. Field **iconId** can be useful if you plan to edit the selected icon **dynamically**

UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



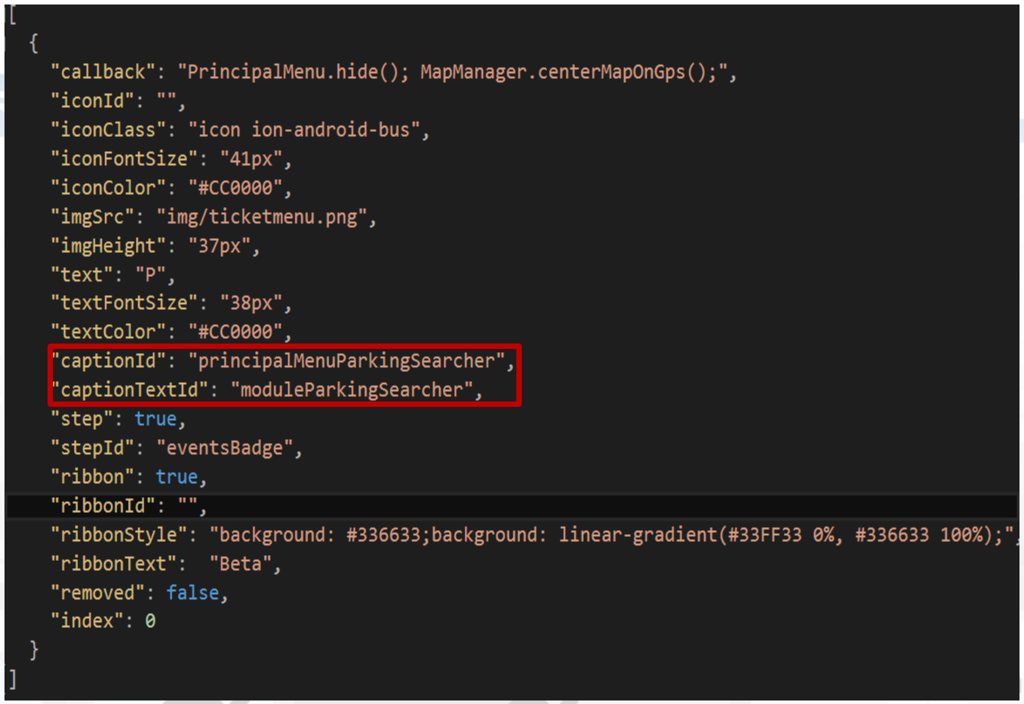
```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

These blocks of fields are **mutually exclusive**. Allow you to choose the icon that will identify the button that you are creating. This icon can be chosen as an **image**, a **text**, a **glyphicon** (Bootstrap) or **ionicons** (ionicons.com).
N.B. Field **iconId** can be useful if you plan to edit the selected icon **dynamically**

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

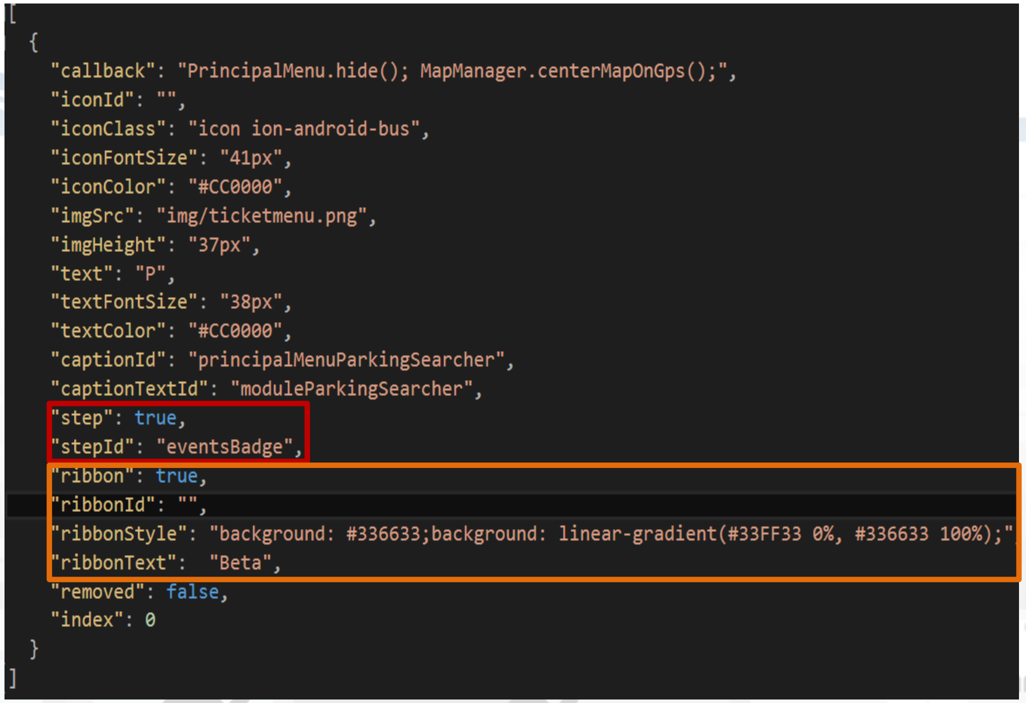
parkingSearcher.principalMenu.json

captionId serves to indicate the **container tag** of the text that is located at the bottom of each button.

captionTextId indicates the name of the field in labels.*.json whose value is the text to be inserted in the previous container.

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



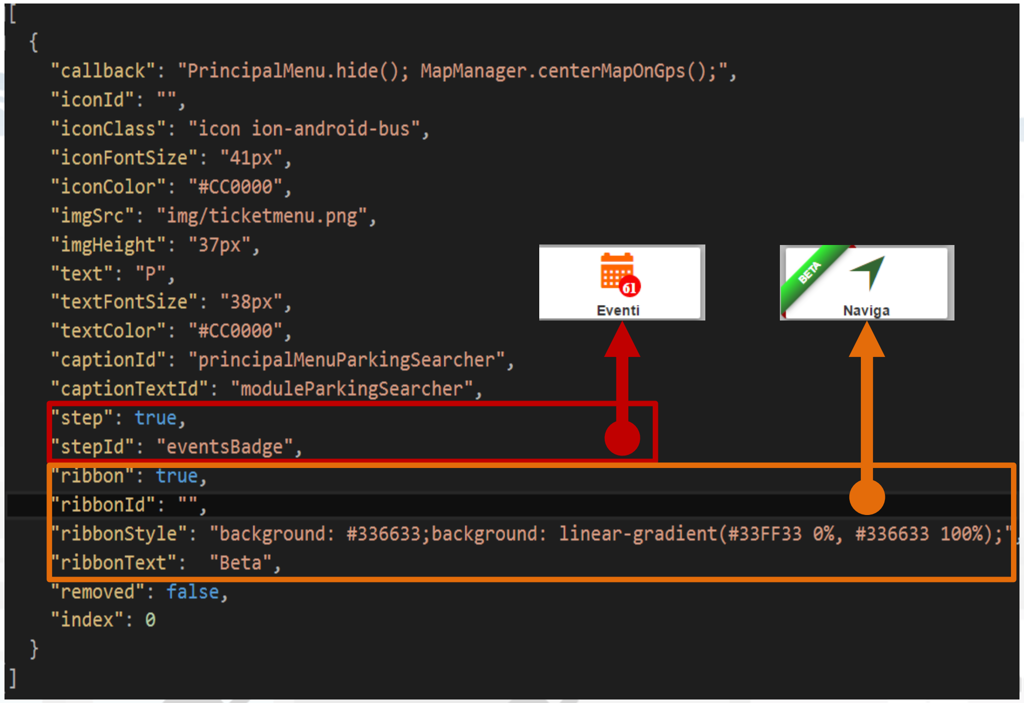
```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);"
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

These blocks of fields are used to show the user **badges containing information** related to the button on which are located

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



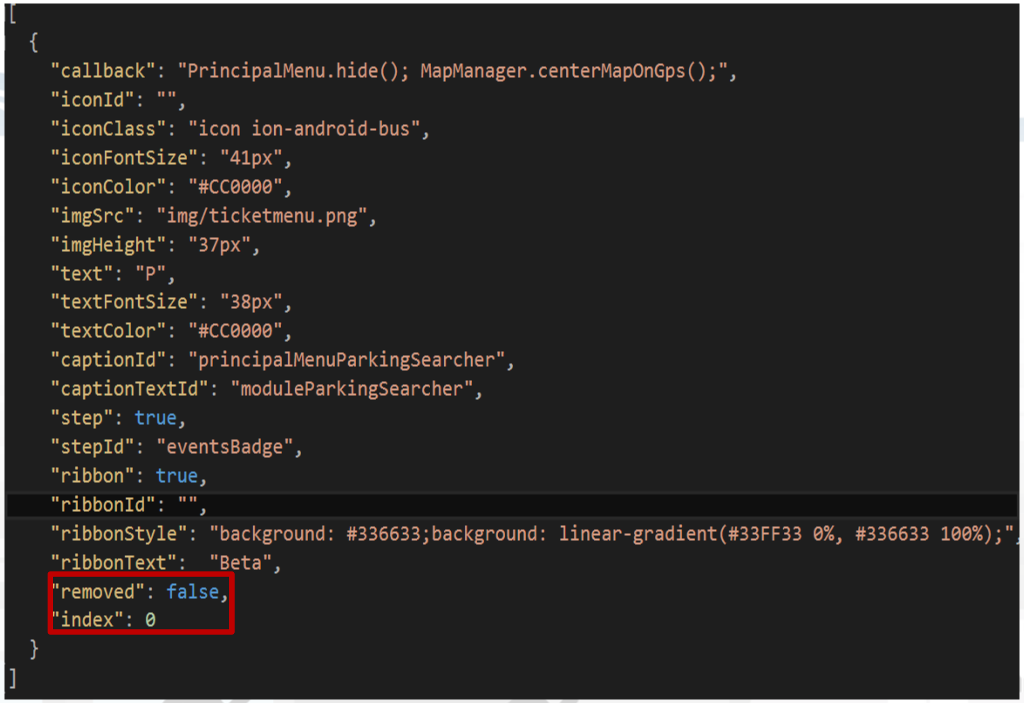
```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

These blocks of fields are used to show the user **badges containing information** related to the button on which are located

ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```

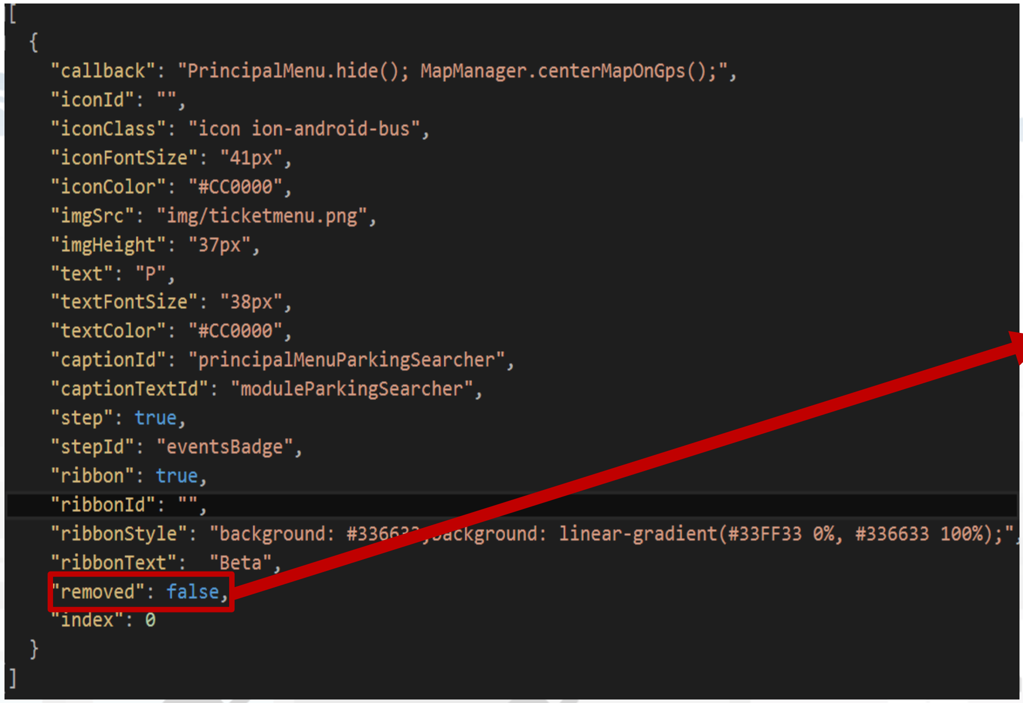
parkingSearcher.principalMenu.json

removed field is useful to allow the removal and the insertion of the buttons in the main menu by the user.

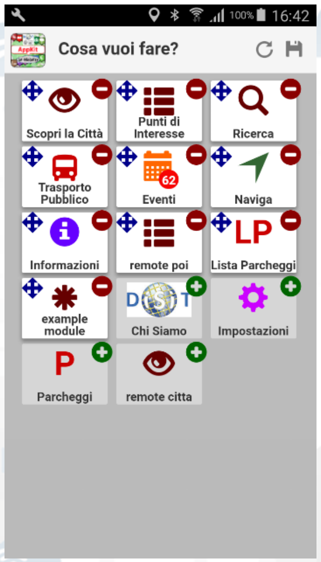
index field is useful for rendering the buttons in the order chosen by the user.

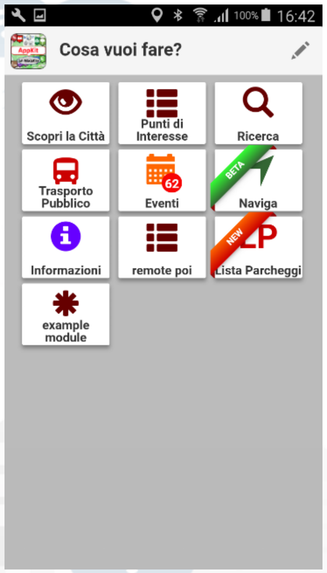
ParkingSearcher in main menu

- **Field descriptions** for creating buttons in the **main menu**



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": true,
    "stepId": "eventsBadge",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #336633;background: linear-gradient(#33FF33 0%, #336633 100%);",
    "ribbonText":  "Beta",
    "removed": false,
    "index": 0
  }
]
```





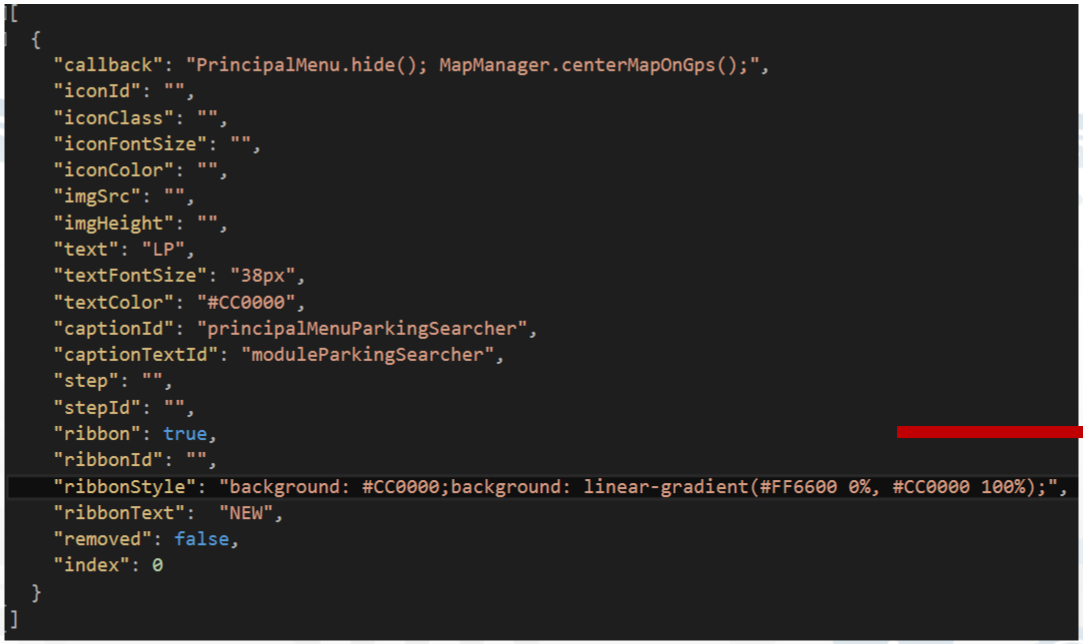
parkingSearcher.principalMenu.json

ParkingSearcher in main menu

- Loading **new buttons modules** within the main menu, takes place by **comparing the captionId** field.
- If the menu already has a button with the **same captionId,** the first is **replaced** with the **new one**.
- To **remove** a button from the main menu (field **removed** hides it) add a **delete** field with value equal to **true**.

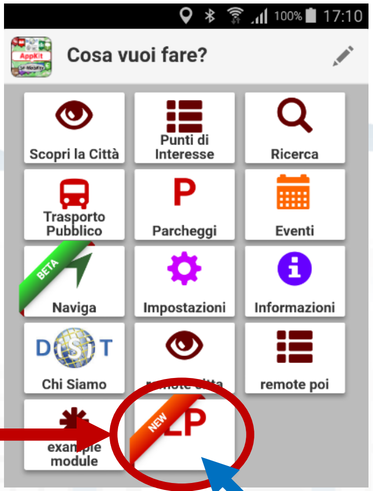
ParkingSearcher in main menu

- First version of the button



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "",
    "iconFontSize": "",
    "iconColor": "",
    "imgSrc": "",
    "imgHeight": "",
    "text": "LP",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": "",
    "stepId": "",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #CC0000;background: linear-gradient(#FF6600 0%, #CC0000 100%);",
    "ribbonText":  "NEW",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

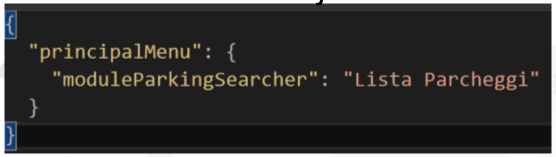


Label missing

Labels of ParkingSearcher

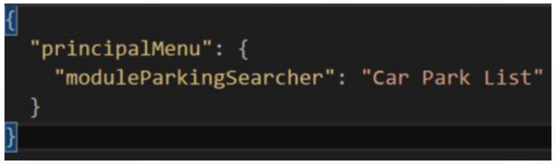
- Description of **label.*.json** files

label.ita.json



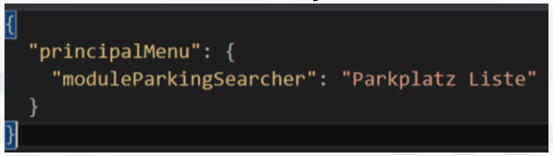
```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista Parcheggi"
  }
}
```

label.eng.json



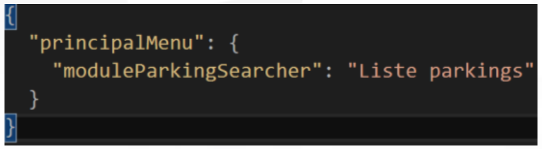
```
{
  "principalMenu": {
    "moduleParkingSearcher": "Car Park List"
  }
}
```

label.deu.json



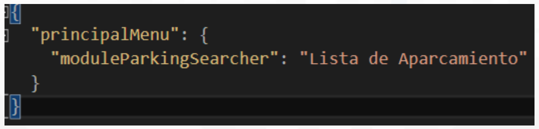
```
{
  "principalMenu": {
    "moduleParkingSearcher": "Parkplatz Liste"
  }
}
```

label.fra.json



```
{
  "principalMenu": {
    "moduleParkingSearcher": "Liste parkings"
  }
}
```

label.esp.json



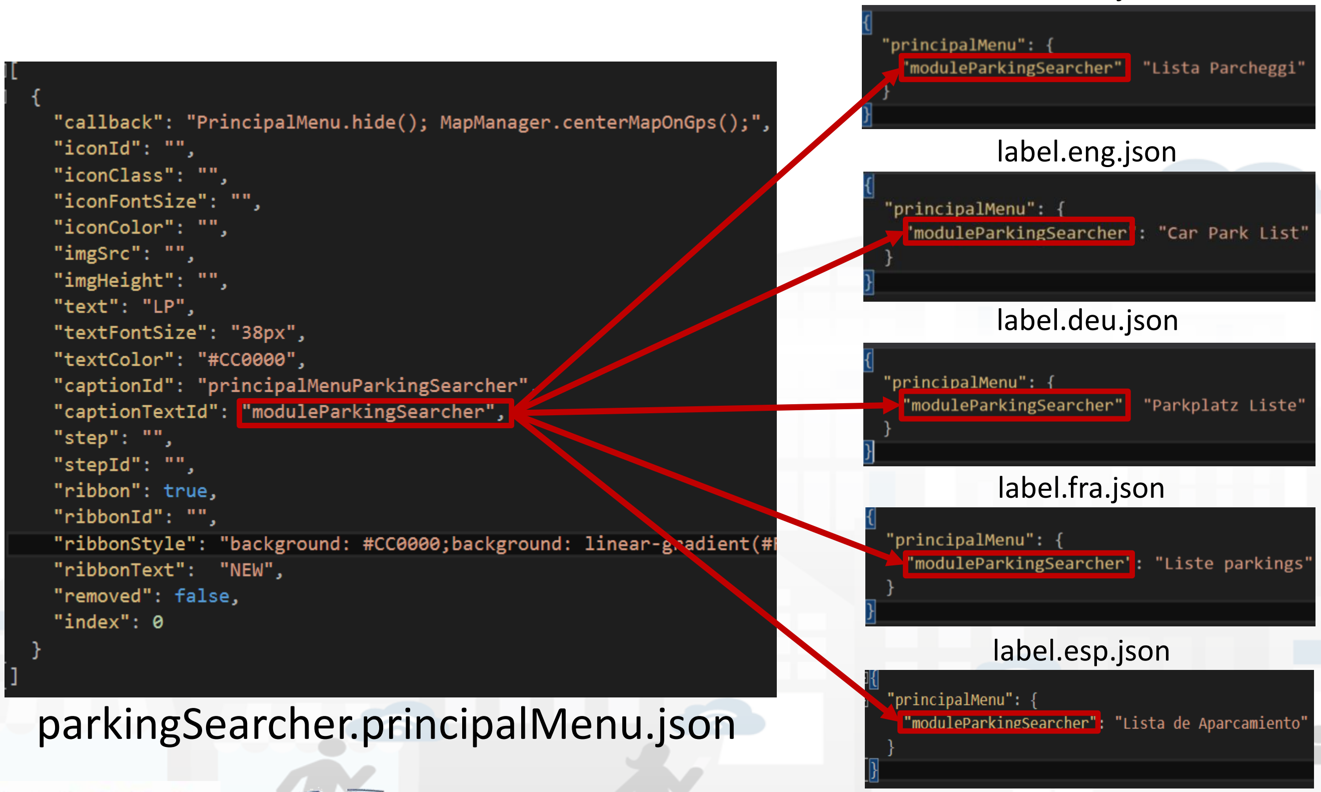
```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista de Aparcamiento"
  }
}
```

Three important things to check:

- Languages shall be indicated by 3 characters: **ita**, **deu**, **esp**, **fra**, **eng**
- The label for the button must be contained within the object "**principalMenu"**
- The name of the field inside "principalMenu“ must be the same of “**captionTextId**” seen before

Labels of ParkingSearcher

- Description of **label.*.json** files



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "",
    "iconFontSize": "",
    "iconColor": "",
    "imgSrc": "",
    "imgHeight": "",
    "text": "LP",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": "",
    "stepId": "",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #CC0000;background: linear-gradient(#
    "ribbonText":  "NEW",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

label.ita.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista Parcheggi"
  }
}
```

label.eng.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Car Park List"
  }
}
```

label.deu.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Parkplatz Liste"
  }
}
```

label.fra.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Liste parkings"
  }
}
```

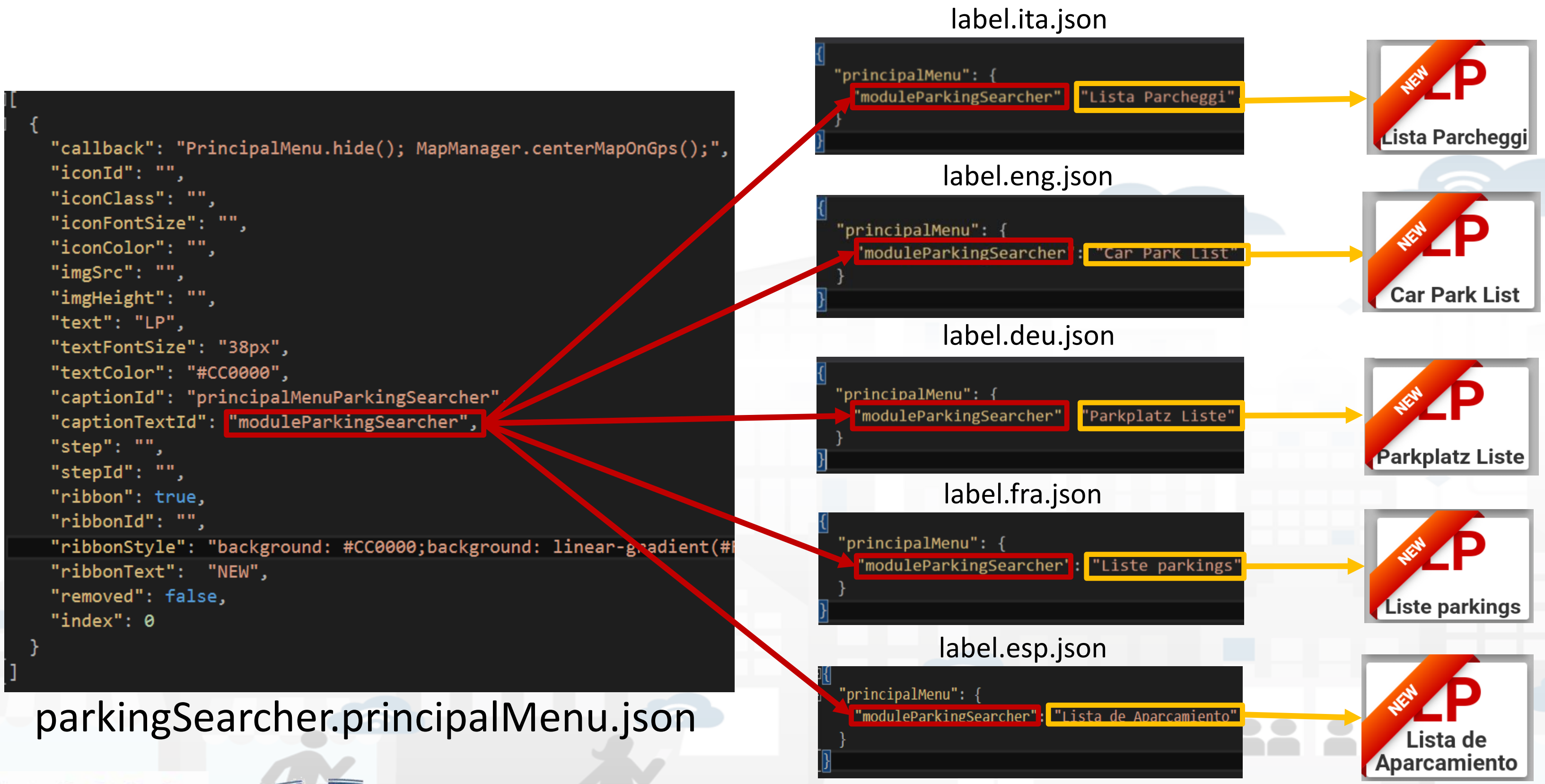
label.esp.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista de Aparcamiento"
  }
}
```

```
$(captionId).html(
labels.principalMenu[
captionTextId]);
```

Labels of ParkingSearcher

- Description of **label.*.json** files



```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "",
    "iconFontSize": "",
    "iconColor": "",
    "imgSrc": "",
    "imgHeight": "",
    "text": "LP",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": "",
    "stepId": "",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #CC0000;background: linear-gradient(#
    "ribbonText":  "NEW",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

label.ita.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista Parcheggi"
  }
}
```

label.eng.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Car Park List"
  }
}
```

label.deu.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Parkplatz Liste"
  }
}
```

label.fra.json

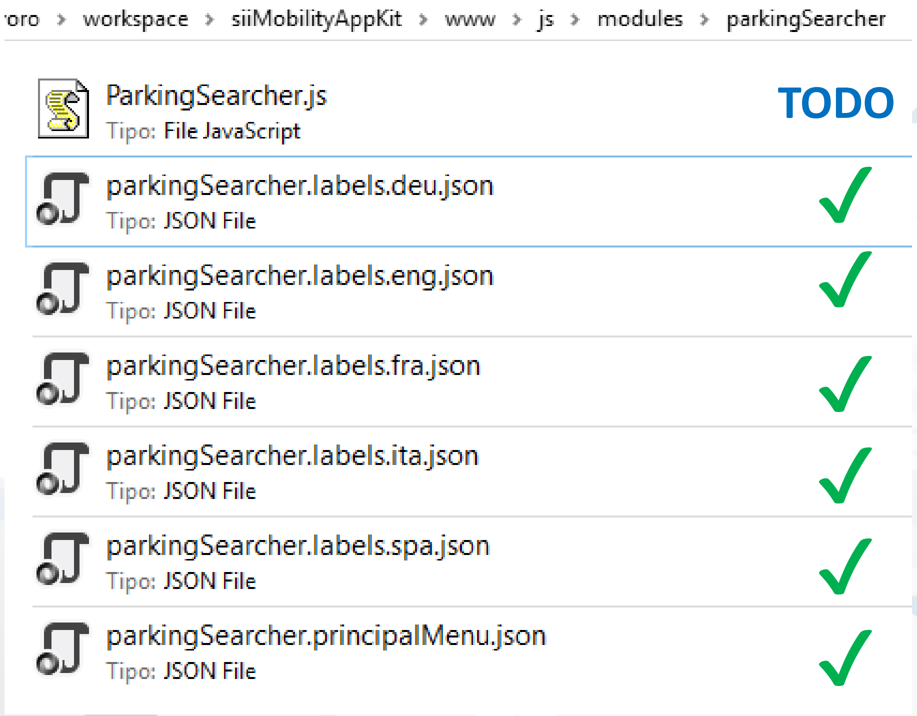
```
{
  "principalMenu": {
    "moduleParkingSearcher": "Liste parkings"
  }
}
```

label.esp.json

```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista de Aparcamiento"
  }
}
```

Create ParkingSearcher Module

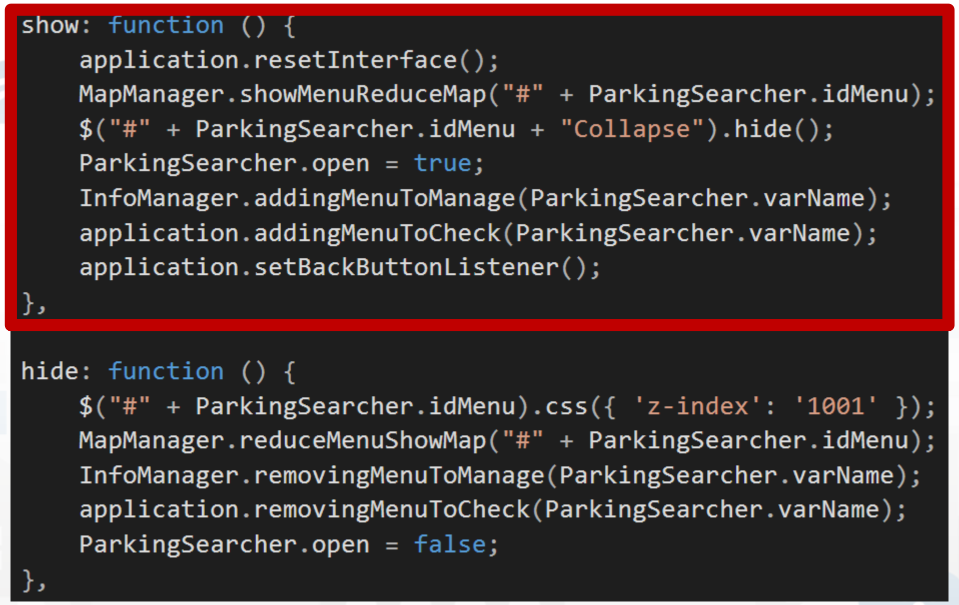
- It is seen as fill most of the files in the folder of new module ParkingSearcher that is developed in this presentation


'oro › workspace › siiMobilityAppKit › www › js › modules › parkingSearcher

ParkingSearcher.js Tipo: File JavaScript	TODO
parkingSearcher.labels.deu.json Tipo: JSON File	✓
parkingSearcher.labels.eng.json Tipo: JSON File	✓
parkingSearcher.labels.fra.json Tipo: JSON File	✓
parkingSearcher.labels.ita.json Tipo: JSON File	✓
parkingSearcher.labels.spa.json Tipo: JSON File	✓
parkingSearcher.principalMenu.json Tipo: JSON File	✓

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**



```
show: function () {
    application.resetInterface();
    MapManager.showMenuReduceMap("#" + ParkingSearcher.idMenu);
    $("#" + ParkingSearcher.idMenu + "Collapse").hide();
    ParkingSearcher.open = true;
    InfoManager.addingMenuToManage(ParkingSearcher.varName);
    application.addingMenuToCheck(ParkingSearcher.varName);
    application.setBackButtonListener();
},

hide: function () {
    $("#" + ParkingSearcher.idMenu).css({ 'z-index': '1001' });
    MapManager.reduceMenuShowMap("#" + ParkingSearcher.idMenu);
    InfoManager.removingMenuToManage(ParkingSearcher.varName);
    application.removingMenuToCheck(ParkingSearcher.varName);
    ParkingSearcher.open = false;
},
```

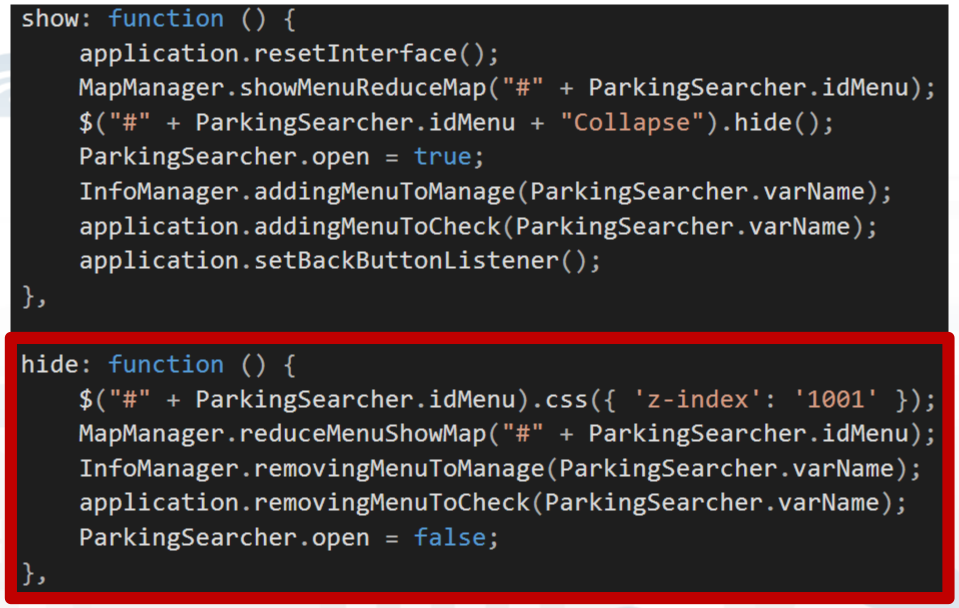
Closes any previously **opened menu**, **shrinks the map** to display the menu, **hides** the **button** to reduce the menu, since it will open already reduced.

Recording to other variables to get notifications when:

- users press the **back button**
- users change the **device orientation**
- must be **closed the menu** opened by this module

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**

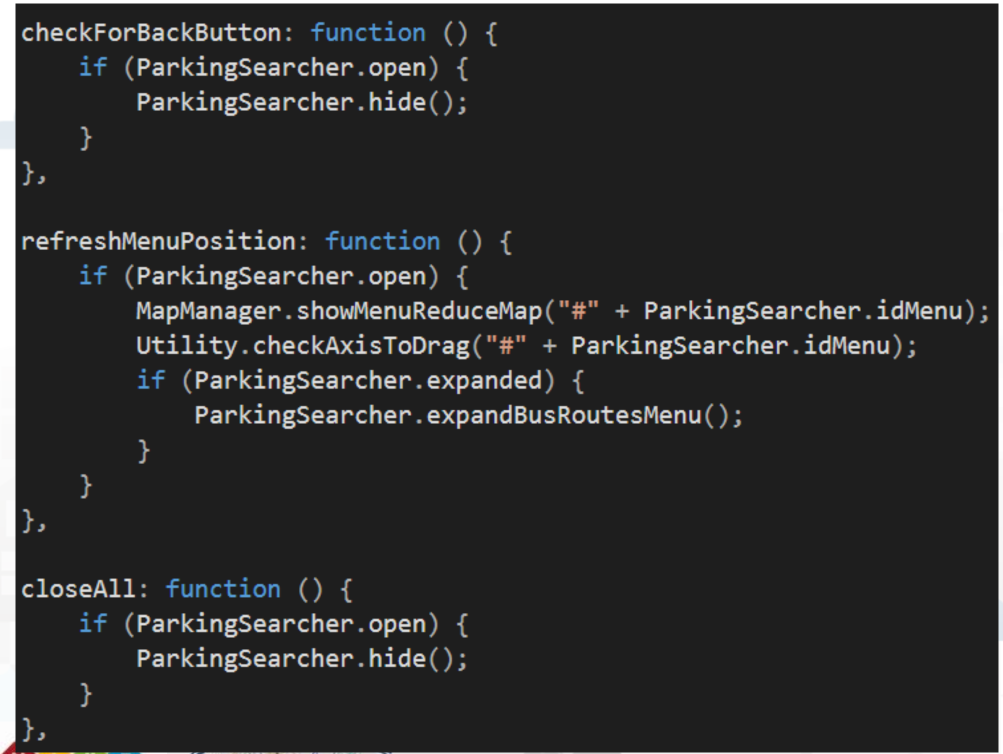


```
show: function () {
    application.resetInterface();
    MapManager.showMenuReduceMap("#" + ParkingSearcher.idMenu);
    $("#" + ParkingSearcher.idMenu + "Collapse").hide();
    ParkingSearcher.open = true;
    InfoManager.addingMenuToManage(ParkingSearcher.varName);
    application.addingMenuToCheck(ParkingSearcher.varName);
    application.setBackButtonListener();
},

hide: function () {
    $("#" + ParkingSearcher.idMenu).css({ 'z-index': '1001' });
    MapManager.reduceMenuShowMap("#" + ParkingSearcher.idMenu);
    InfoManager.removingMenuToManage(ParkingSearcher.varName);
    application.removingMenuToCheck(ParkingSearcher.varName);
    ParkingSearcher.open = false;
},
```

Does the **opposite functions** to those performed by the **function show**, also reset the z-indexof the menu

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**



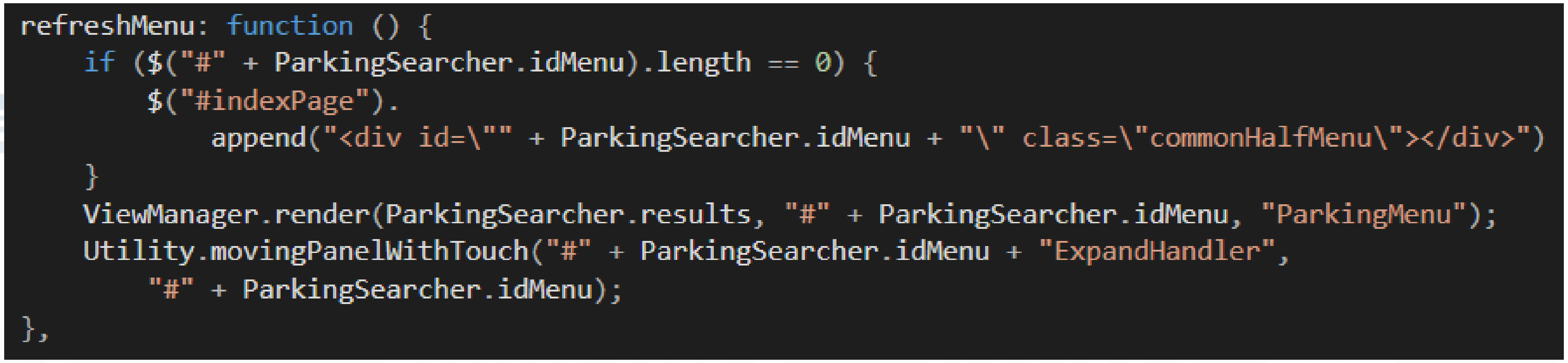
```
checkForBackButton: function () {
    if (ParkingSearcher.open) {
        ParkingSearcher.hide();
    }
},

refreshMenuPosition: function () {
    if (ParkingSearcher.open) {
        MapManager.showMenuReduceMap("#" + ParkingSearcher.idMenu);
        Utility.checkAxisToDrag("#" + ParkingSearcher.idMenu);
        if (ParkingSearcher.expanded) {
            ParkingSearcher.expandBusRoutesMenu();
        }
    }
},

closeAll: function () {
    if (ParkingSearcher.open) {
        ParkingSearcher.hide();
    }
},
```

These are the **callbacks** called to **notify** the occurrence of an event among those described previously (see show function) and for which we recorded the module

- users press the **back button**
- users change the **device orientation**
- must be **closed the menu** opened by this module

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**

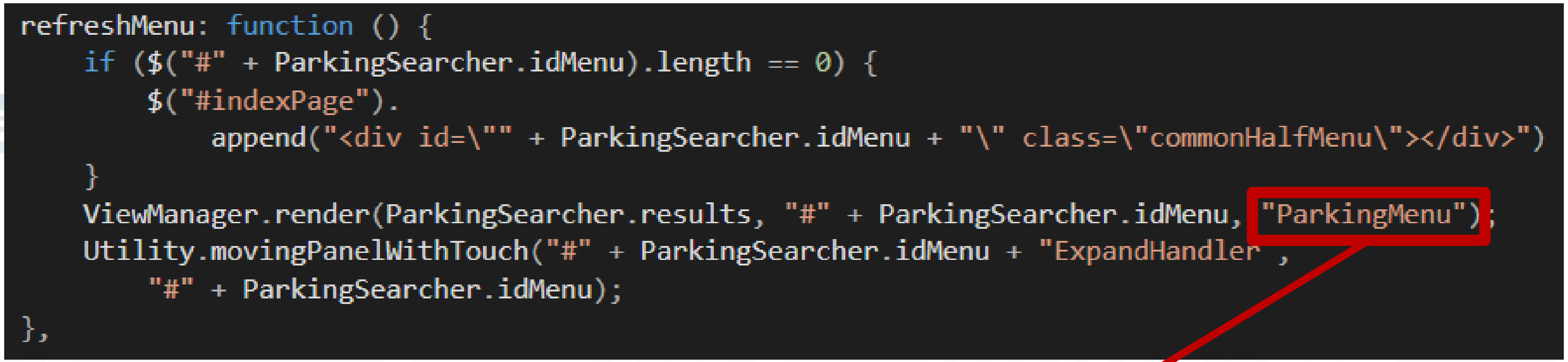


```
refreshMenu: function () {
    if ($("#" + ParkingSearcher.idMenu).length == 0) {
        $("#indexPage").
            append("<div id=\"" + ParkingSearcher.idMenu + "\" class=\"commonHalfMenu\"></div>")
    }
    ViewManager.render(ParkingSearcher.results, "#" + ParkingSearcher.idMenu, "ParkingMenu");
    Utility.movingPanelWithTouch("#" + ParkingSearcher.idMenu + "ExpandHandler",
        "#" + ParkingSearcher.idMenu);
},
```

- Checks if there is the **element** that will **contain the html code** created through the use of **Mustache** library.
- It is generated the html code with **template ParkingMenu.mst.html** and **JSON ParkingSearcher.results** and added to the element container.
- Finally, the **feature** that allows the users **to widen the menu by dragging** the handler is added to it

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**

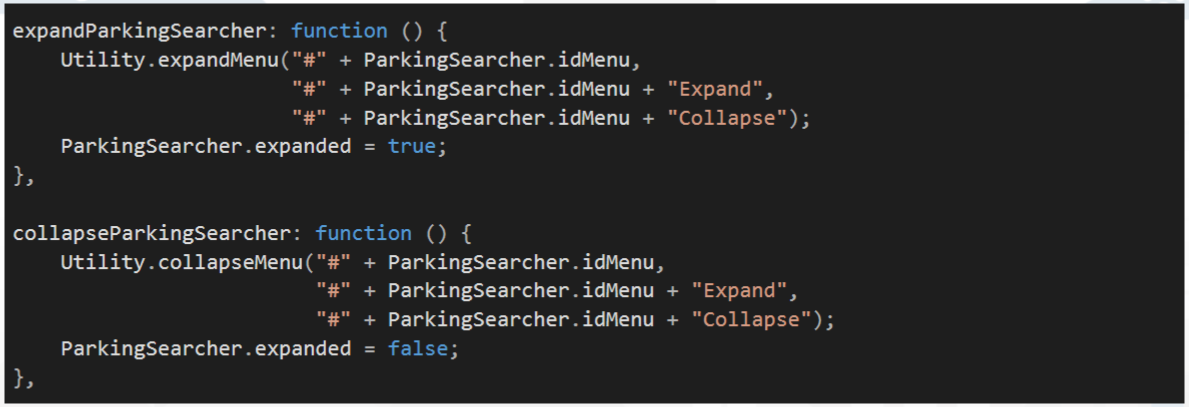


```
refreshMenu: function () {
    if ($("#" + ParkingSearcher.idMenu).length == 0) {
        $("#indexPage").
            append("<div id=\"" + ParkingSearcher.idMenu + "\" class=\"commonHalfMenu\"></div>")
    }
    ViewManager.render(ParkingSearcher.results, "#" + ParkingSearcher.idMenu, "ParkingMenu");
    Utility.movingPanelWithTouch("#" + ParkingSearcher.idMenu + "ExpandHandler",
        "#" + ParkingSearcher.idMenu);
},
```

- Checks if there is the **element** that will **contain the html code** created through the use of **Mustache** library.
- It is generated the html code with **template ParkingMenu.mst.html** and **JSON ParkingSearcher.results** and added to the element container.
- Finally, the **feature** that allows the users **to widen the menu by dragging** the handler is added to it

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**

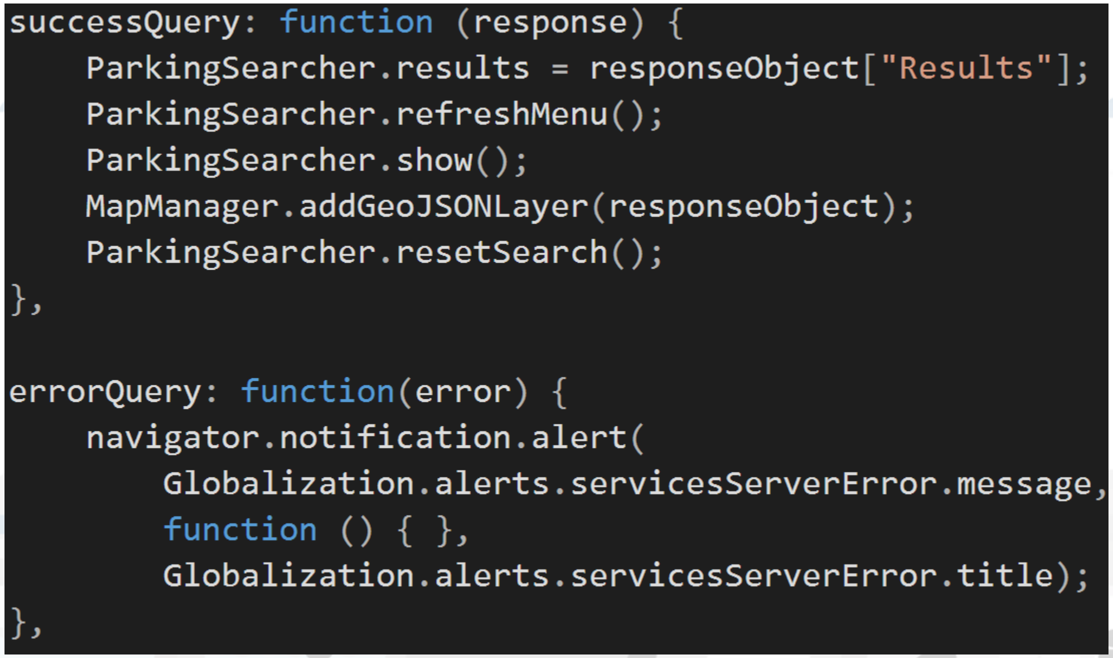


```
expandParkingSearcher: function () {
    Utility.expandMenu("#" + ParkingSearcher.idMenu,
                       "#" + ParkingSearcher.idMenu + "Expand",
                       "#" + ParkingSearcher.idMenu + "Collapse");
    ParkingSearcher.expanded = true;
},

collapseParkingSearcher: function () {
    Utility.collapseMenu("#" + ParkingSearcher.idMenu,
                         "#" + ParkingSearcher.idMenu + "Expand",
                         "#" + ParkingSearcher.idMenu + "Collapse");
    ParkingSearcher.expanded = false;
},
```

They serve to **expand** or **collapse** the menu

ParkingSearcher Module Functions

- **Functions** contained in **ParkingSearcher.js**



```
successQuery: function (response) {
    ParkingSearcher.results = responseObject["Results"];
    ParkingSearcher.refreshMenu();
    ParkingSearcher.show();
    MapManager.addGeoJSONLayer(responseObject);
    ParkingSearcher.resetSearch();
},

errorQuery: function(error) {
    navigator.notification.alert(
        Globalization.alerts.servicesServerError.message,
        function () { },
        Globalization.alerts.servicesServerError.title);
},
```

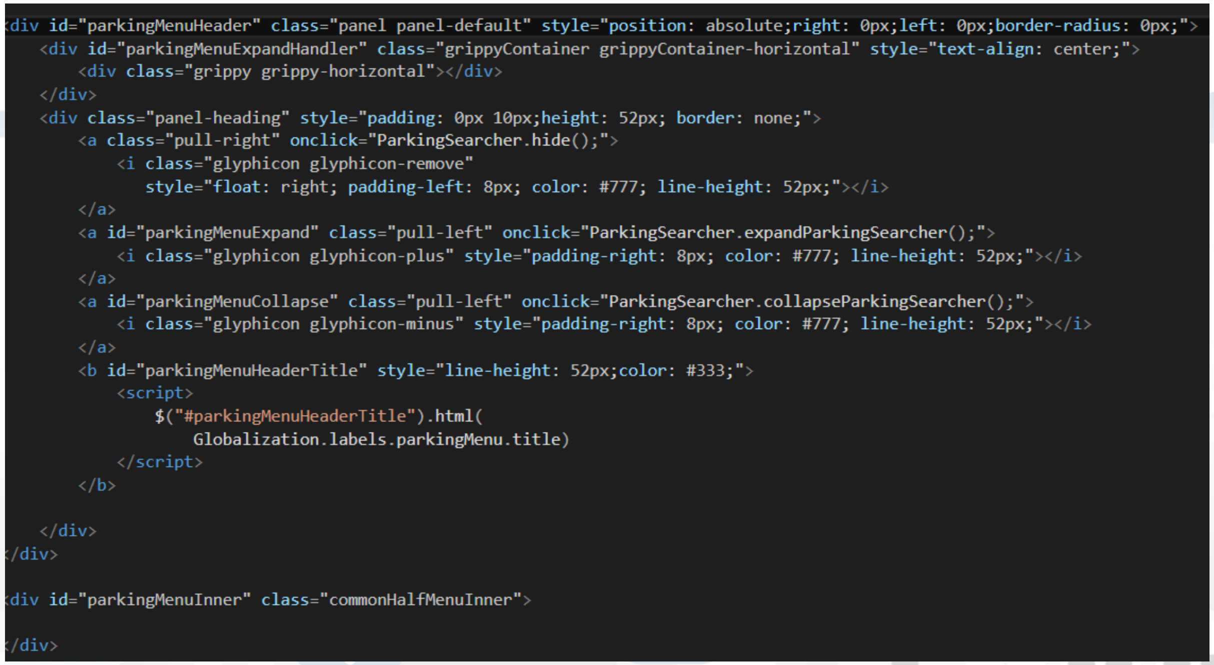
These are the callbacks that should be called once the **JSON**, containing the **data to be displayed** to the user, is created. The **success callback:**

- will locally save the response
- will create the menu
- will show it.

If the menu will contain **elements** that it is possible to **show on the map** they will be added to the map by last function

ParkingSearcher Module Template

- Before adding the logic of the new module, we create the template to be filled with the correct JSON.



```
<div id="parkingMenuHeader" class="panel panel-default" style="position: absolute;right: 0px;left: 0px;border-radius: 0px;">
    <div id="parkingMenuExpandHandler" class="grippyContainer grippyContainer-horizontal" style="text-align: center;">
        <div class="grippy grippy-horizontal"></div>
    </div>
    <div class="panel-heading" style="padding: 0px 10px;height: 52px; border: none;">
        <a class="pull-right" onclick="ParkingSearcher.hide();">
            <i class="glyphicon glyphicon-remove"
               style="float: right; padding-left: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <a id="parkingMenuExpand" class="pull-left" onclick="ParkingSearcher.expandParkingSearcher();">
            <i class="glyphicon glyphicon-plus" style="padding-right: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <a id="parkingMenuCollapse" class="pull-left" onclick="ParkingSearcher.collapseParkingSearcher();">
            <i class="glyphicon glyphicon-minus" style="padding-right: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <b id="parkingMenuHeaderTitle" style="line-height: 52px;color: #333;">
            <script>
                $("#parkingMenuHeaderTitle").html(
                    Globalization.labels.parkingMenu.title)
            </script>
        </b>

    </div>
</div>

<div id="parkingMenuInner" class="commonHalfMenuInner">

</div>
```

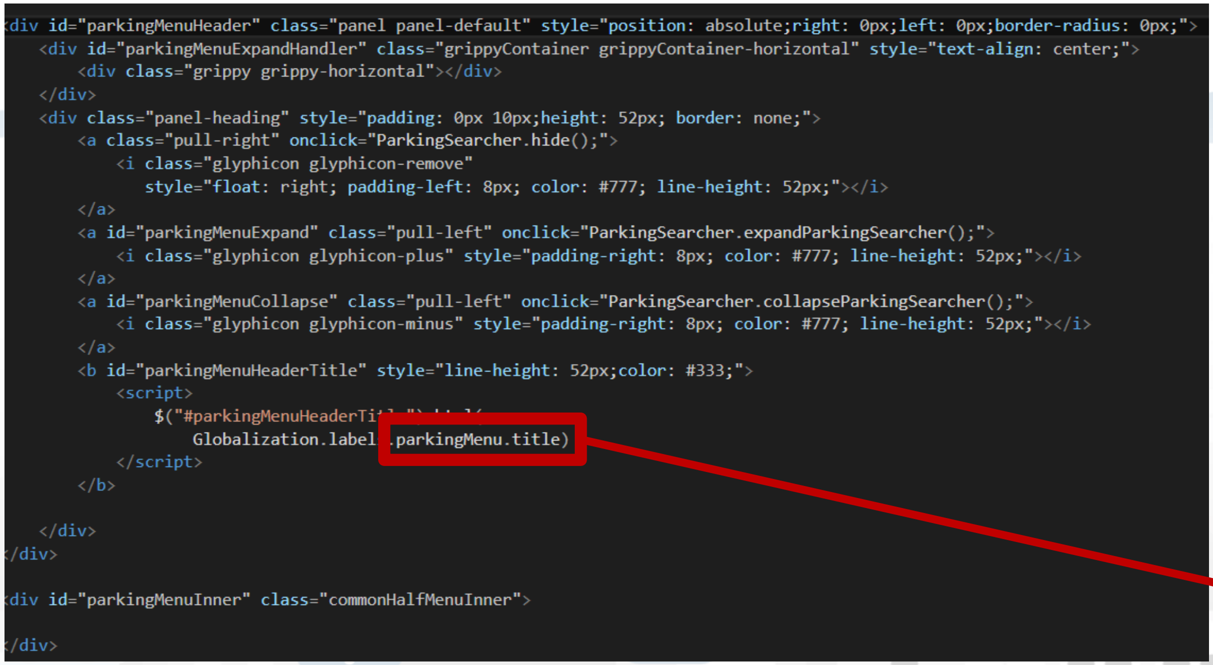
ParkingMenu.mst.html

This default template will **simply show a menu** with a header and body empty. **Must have the same name as the string entered as the third parameter in the call**

ViewManager.render (
  ParkingSearcher.results,
  "#" + ParkingSearcher.idMenu,
"**ParkingMenu**");

ParkingSearcher Module Template

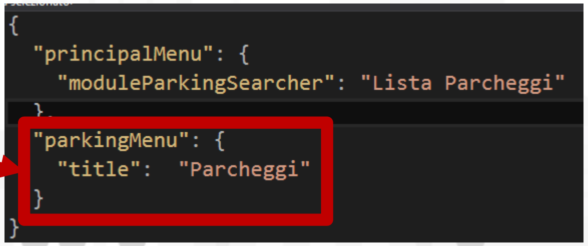
- Before adding the logic of the new module, we create the template to be filled with the correct JSON.



```
<div id="parkingMenuHeader" class="panel panel-default" style="position: absolute;right: 0px;left: 0px;border-radius: 0px;">
    <div id="parkingMenuExpandHandler" class="grippyContainer grippyContainer-horizontal" style="text-align: center;">
        <div class="grippy grippy-horizontal"></div>
    </div>
    <div class="panel-heading" style="padding: 0px 10px;height: 52px; border: none;">
        <a class="pull-right" onclick="ParkingSearcher.hide();">
            <i class="glyphicon glyphicon-remove"
               style="float: right; padding-left: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <a id="parkingMenuExpand" class="pull-left" onclick="ParkingSearcher.expandParkingSearcher();">
            <i class="glyphicon glyphicon-plus" style="padding-right: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <a id="parkingMenuCollapse" class="pull-left" onclick="ParkingSearcher.collapseParkingSearcher();">
            <i class="glyphicon glyphicon-minus" style="padding-right: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <b id="parkingMenuHeaderTitle" style="line-height: 52px;color: #333;">
            <script>
                $("#parkingMenuHeaderTi[illegible]
                    Globalization.label[illegible].parkingMenu.title)
            </script>
        </b>

    </div>
</div>

<div id="parkingMenuInner" class="commonHalfMenuInner">

</div>
```

templates/ParkingMenu.mst.html

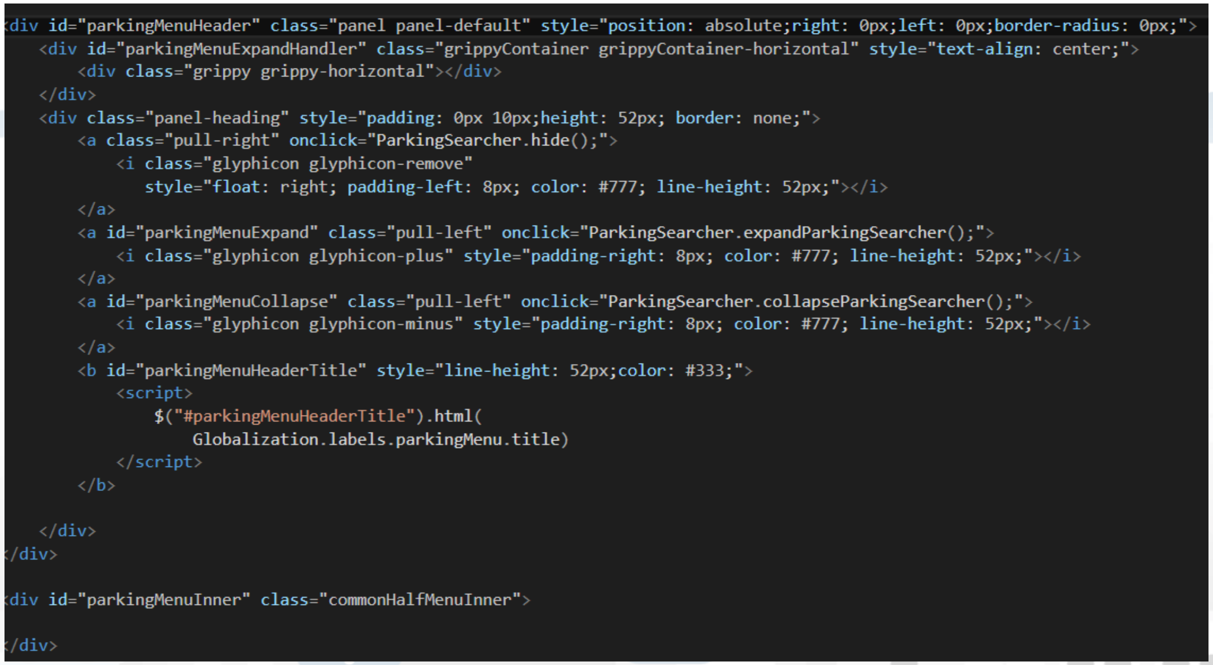
This template will be saved in the folder called «**templates**».
To add a title to the header we should add this item to all files labels.*. Json



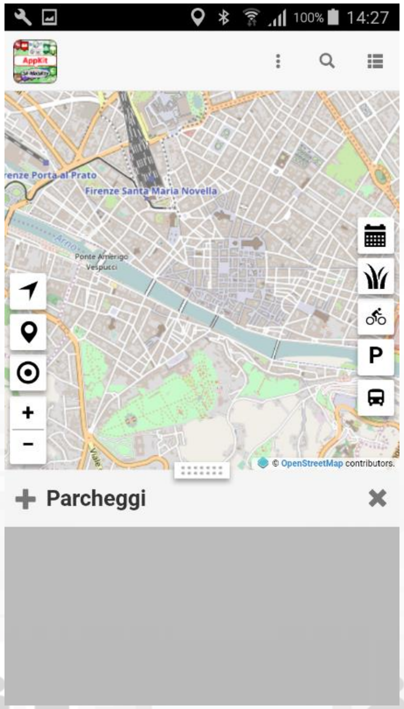
```
{
  "principalMenu": {
    "moduleParkingSearcher": "Lista Parcheggi"
  },
  "parkingMenu": {
    "title":  "Parcheggi"
  }
}
```

ParkingSearcher Module Template

- Before adding the logic of the new module, we create the template to be filled with the correct JSON.



```
<div id="parkingMenuHeader" class="panel panel-default" style="position: absolute;right: 0px;left: 0px;border-radius: 0px;">
    <div id="parkingMenuExpandHandler" class="grippyContainer grippyContainer-horizontal" style="text-align: center;">
        <div class="grippy grippy-horizontal"></div>
    </div>
    <div class="panel-heading" style="padding: 0px 10px;height: 52px; border: none;">
        <a class="pull-right" onclick="ParkingSearcher.hide();">
            <i class="glyphicon glyphicon-remove"
               style="float: right; padding-left: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <a id="parkingMenuExpand" class="pull-left" onclick="ParkingSearcher.expandParkingSearcher();">
            <i class="glyphicon glyphicon-plus" style="padding-right: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <a id="parkingMenuCollapse" class="pull-left" onclick="ParkingSearcher.collapseParkingSearcher();">
            <i class="glyphicon glyphicon-minus" style="padding-right: 8px; color: #777; line-height: 52px;"></i>
        </a>
        <b id="parkingMenuHeaderTitle" style="line-height: 52px;color: #333;">
            <script>
                $("#parkingMenuHeaderTitle").html(
                    Globalization.labels.parkingMenu.title)
            </script>
        </b>

    </div>
</div>

<div id="parkingMenuInner" class="commonHalfMenuInner">

</div>
```



templates/ParkingMenu.mst.html

Create ParkingSearcher Module

The goal of this example is to create a **new module** that in addition to viewing the list of car parks as is already the case for the button named “Car Park" will **show directly** the **number of free parking lots** for each car park found

In ParkingSearcher.js must be made the logic that **retrieves data** from API describer
in previous presentations and creates the **JSON** to fill the **template** and generate the new menu

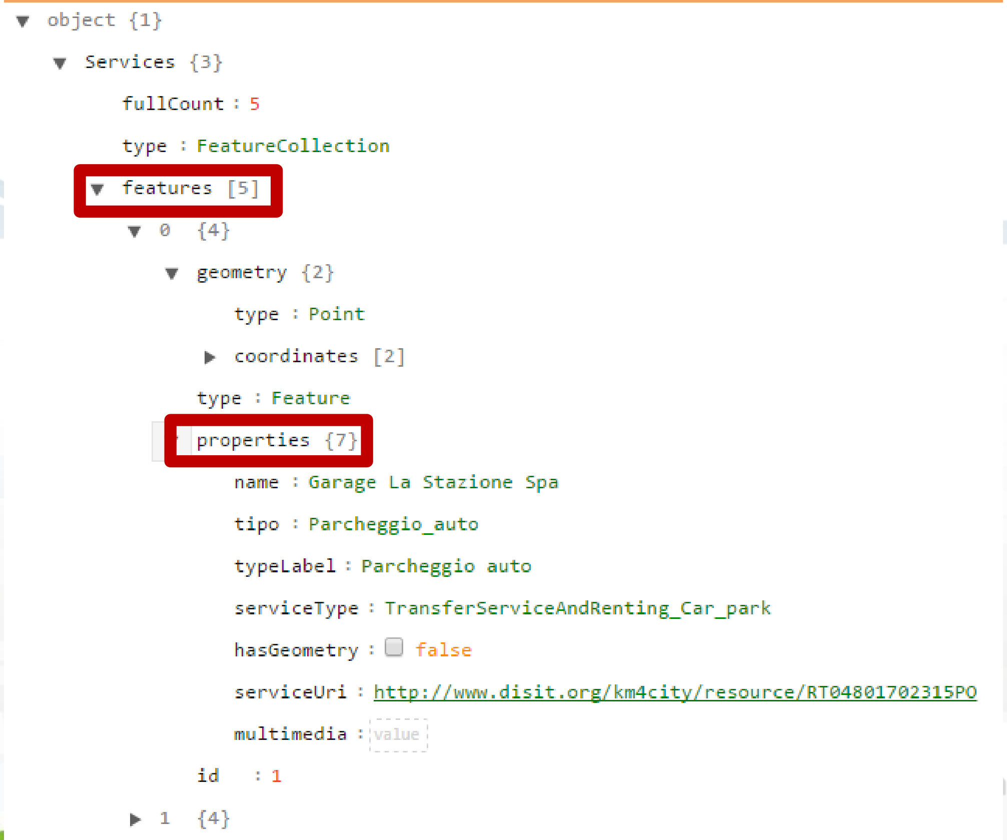
ParkingSearcher Called API

- The following API returns **the list of parking** that are located at a maximum distance of 300 meters from the location sent. The list is limited to 100 items.

http://www.disit.org/ServiceMap/api/v1/?
selection=**43.7778;11.2481**&
categories=**Car_park**&
maxResults=100&
maxDists=0.3&
format=json&
lang=it&
geometry=true

ParkingSearcher Called API

- The returned data are not sufficient to create the final JSON, because these **data are lacking** on the realtime information



```
▼ object {1}
  ▼ Services {3}
      fullCount : 5
      type : FeatureCollection
    ▼ features [5]
      ▼ 0 {4}
        ▼ geometry {2}
            type : Point
          ▶ coordinates [2]
          type : Feature
          properties {7}
            name : Garage La Stazione Spa
            tipo : Parcheggio_auto
            typeLabel : Parcheggio auto
            serviceType : TransferServiceAndRenting_Car_park
            hasGeometry : ☐ false
            serviceUri : http://www.disit.org/km4city/resource/RT04801702315PO
            multimedia : value
          id : 1
      ▶ 1 {4}
```

There are data from **all car parks nearby**, but there are **few properties** that are received

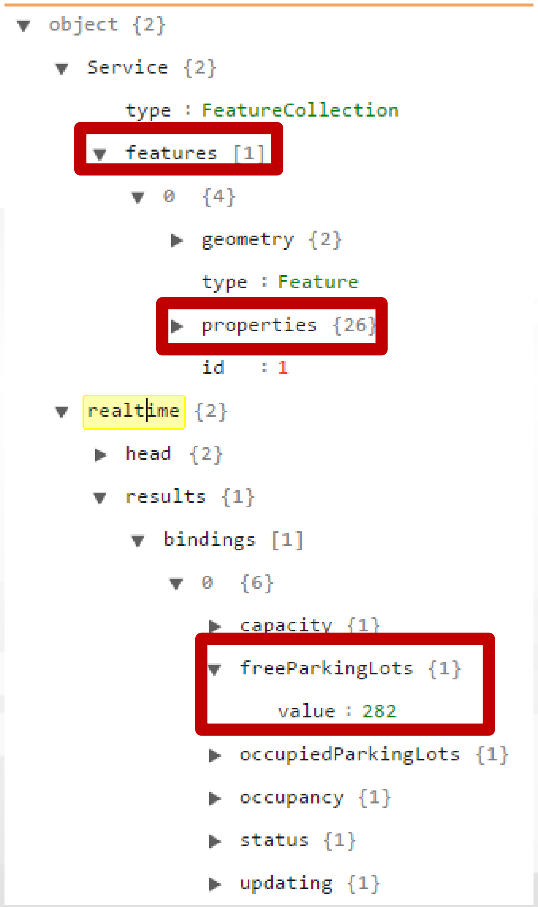
ParkingSearcher Called API

- The following API which returns all information relating to a single service

http://www.disit.org/ServiceMap/api/v1/?
serviceUri=**http://www.disit.org/km4city/resource/RT04801702315PO**&
format=json&
lang=it

ParkingSearcher Called API

- The returned data are not sufficient to create the final JSON, because these data are **relative to only one car park**



```
▼ object {2}
  ▼ Service {2}
      type : FeatureCollection
    ▼ features [1]
      ▼ 0 {4}
        ▶ geometry {2}
          type : Feature
        ▶ properties {26}
          id : 1
  ▼ realtime {2}
    ▶ head {2}
    ▼ results {1}
      ▼ bindings [1]
        ▼ 0 {6}
          ▶ capacity {1}
          ▼ freeParkingLots {1}
              value : 282
          ▶ occupiedParkingLots {1}
          ▶ occupancy {1}
          ▶ status {1}
          ▶ updating {1}
```

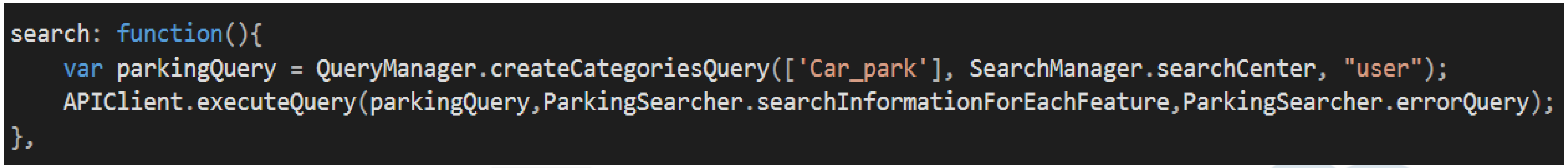
There are data from **one car parks nearby**, but there are **many properties** that are received

ParkingSearcher Module Logic

- The idea is to **call the first API that returns the complete list** of nearby car park, and for each car park in the list **call the second API that returns detailed information** with the number of free parking lots

ParkingSearcher Module Logic

- The first API can be call in the app with the following functions



```
search: function(){
    var parkingQuery = QueryManager.createCategoriesQuery(['Car_park'], SearchManager.searchCenter, "user");
    APIClient.executeQuery(parkingQuery,ParkingSearcher.searchInformationForEachFeature,ParkingSearcher.errorQuery);
},
```

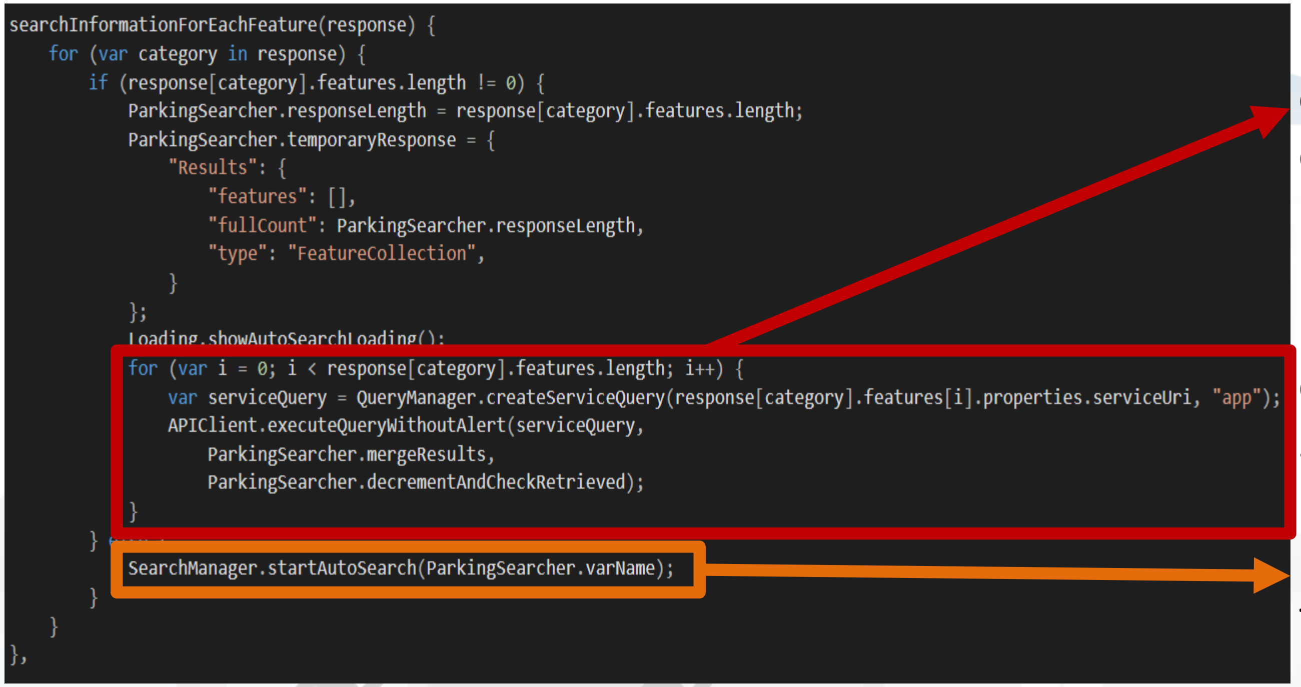
http://www.disit.org/ServiceMap/api/v1/?
selection=**43.7778;11.2481**&
categories=**Car_park**&
maxResults=100&
maxDists=0.3&
format=json&
lang=it&
geometry=true

The **first function** creates the string that contains the **parameters** from "?" to the end.

The **second function** adds the URL of the API and makes the call. When the data has been received calls the error or success callback.

ParkingSearcher Module Logic

- The second API can be call in the app with the following functions



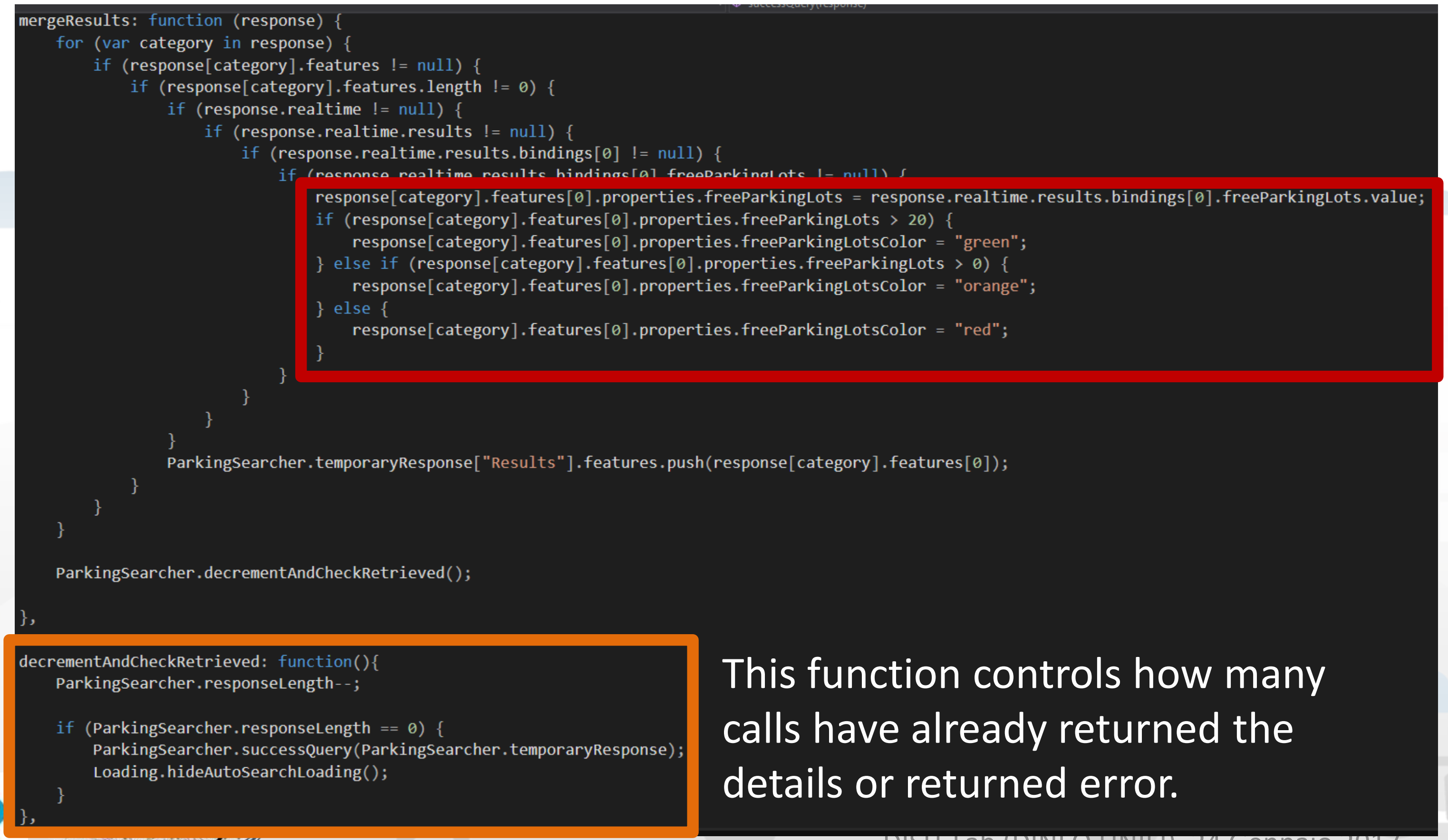
```
searchInformationForEachFeature(response) {
    for (var category in response) {
        if (response[category].features.length != 0) {
            ParkingSearcher.responseLength = response[category].features.length;
            ParkingSearcher.temporaryResponse = {
                "Results": {
                    "features": [],
                    "fullCount": ParkingSearcher.responseLength,
                    "type": "FeatureCollection",
                }
            };
            Loading.showAutoSearchLoading();
            for (var i = 0; i < response[category].features.length; i++) {
                var serviceQuery = QueryManager.createServiceQuery(response[category].features[i].properties.serviceUri, "app");
                APIClient.executeQueryWithoutAlert(serviceQuery,
                    ParkingSearcher.mergeResults,
                    ParkingSearcher.decrementAndCheckRetrieved);
            }
        }
            SearchManager.startAutoSearch(ParkingSearcher.varName);
        }
    }
},
```

For each car park listed is **called the API that returns details**.

If there is **no car park** in the list is called a function which **doubles the radius** of the search area **until at least one car park is in the list** or the radius is greater than 200 km

ParkingSearcher Module Logic

- The number of free parking lots is copied **from realtime object in the properties** to make writing the template easier. Is also added as a property a string that identifies the **text color** based on the number of free parking lots

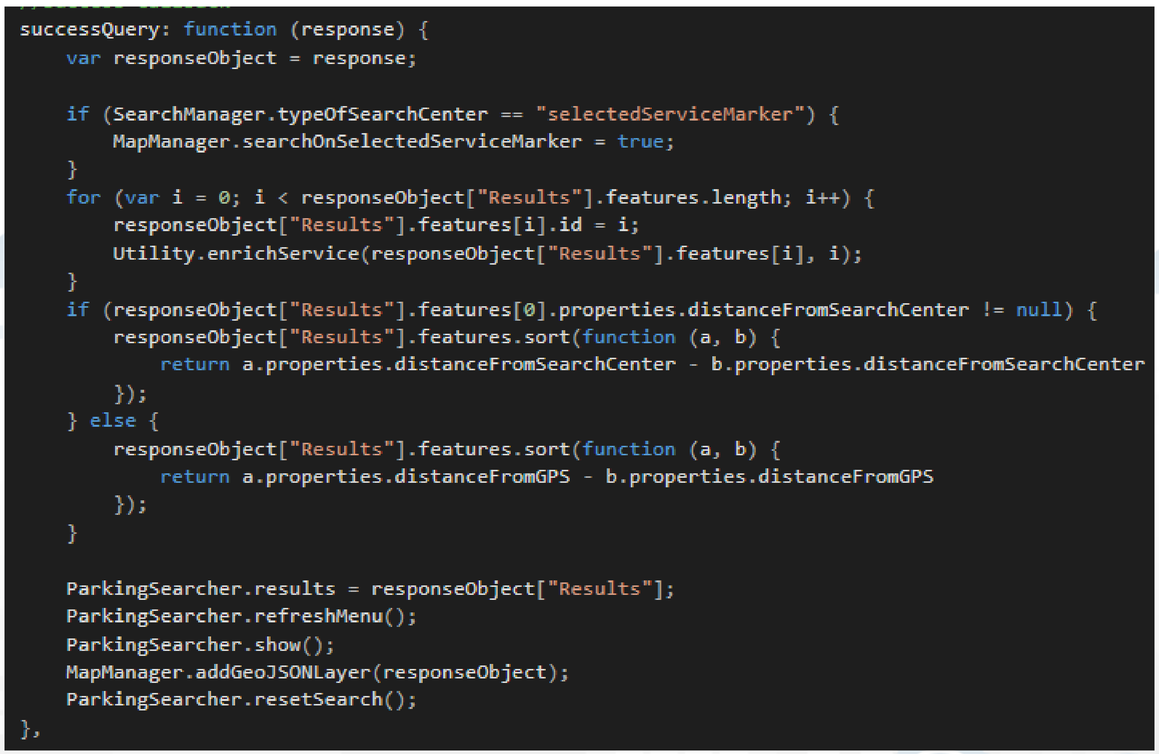


```
mergeResults: function (response) {
    for (var category in response) {
        if (response[category].features != null) {
            if (response[category].features.length != 0) {
                if (response.realtime != null) {
                    if (response.realtime.results != null) {
                        if (response.realtime.results.bindings[0] != null) {
                            if (response.realtime.results.bindings[0].freeParkingLots != null) {
                                response[category].features[0].properties.freeParkingLots = response.realtime.results.bindings[0].freeParkingLots.value;
                                if (response[category].features[0].properties.freeParkingLots > 20) {
                                    response[category].features[0].properties.freeParkingLotsColor = "green";
                                } else if (response[category].features[0].properties.freeParkingLots > 0) {
                                    response[category].features[0].properties.freeParkingLotsColor = "orange";
                                } else {
                                    response[category].features[0].properties.freeParkingLotsColor = "red";
                                }
                            }
                        }
                    }
                }
                ParkingSearcher.temporaryResponse["Results"].features.push(response[category].features[0]);
            }
        }
    }

    ParkingSearcher.decrementAndCheckRetrieved();

},
```

```
decrementAndCheckRetrieved: function(){
    ParkingSearcher.responseLength--;

    if (ParkingSearcher.responseLength == 0) {
        ParkingSearcher.successQuery(ParkingSearcher.temporaryResponse);
        Loading.hideAutoSearchLoading();
    }
},
```

This function controls how many calls have already returned the details or returned error.

ParkingSearcher Module Logic



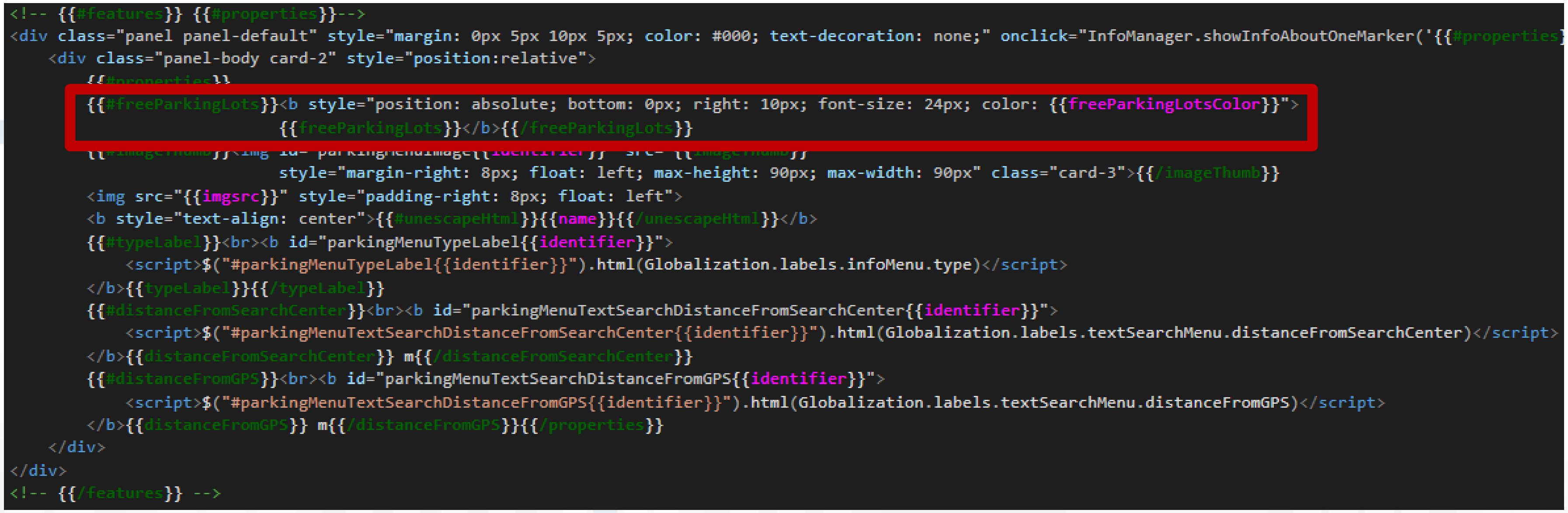
```
successQuery: function (response) {
    var responseObject = response;

    if (SearchManager.typeOfSearchCenter == "selectedServiceMarker") {
        MapManager.searchOnSelectedServiceMarker = true;
    }
    for (var i = 0; i < responseObject["Results"].features.length; i++) {
        responseObject["Results"].features[i].id = i;
        Utility.enrichService(responseObject["Results"].features[i], i);
    }
    if (responseObject["Results"].features[0].properties.distanceFromSearchCenter != null) {
        responseObject["Results"].features.sort(function (a, b) {
            return a.properties.distanceFromSearchCenter - b.properties.distanceFromSearchCenter
        });
    } else {
        responseObject["Results"].features.sort(function (a, b) {
            return a.properties.distanceFromGPS - b.properties.distanceFromGPS
        });
    }

    ParkingSearcher.results = responseObject["Results"];
    ParkingSearcher.refreshMenu();
    ParkingSearcher.show();
    MapManager.addGeoJSONLayer(responseObject);
    ParkingSearcher.resetSearch();
},
```

This is the **function** that receives the **end JSON and shows it to the user**, by creating the marker on the map and **populating** the **list** through the **template**.

The **JSON is enriched** with additional information such as **distance from GPS** or from a manual search and **list is sorted** according to these values.

ParkingSearcher Module Template

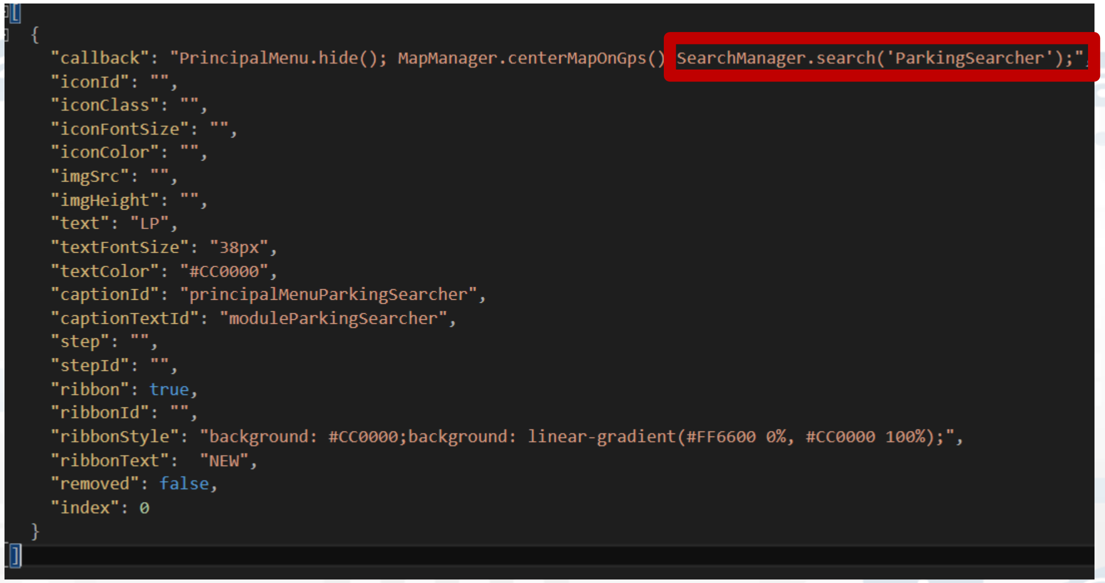
- This is the **final template** that allows you to show the user a list of car parks in its vicinity with an **indication of the number of free parking lots**



```
<!-- {{#features}} {{#properties}}-->
<div class="panel panel-default" style="margin: 0px 5px 10px 5px; color: #000; text-decoration: none;" onclick="InfoManager.showInfoAboutOneMarker('{{#properties}
    <div class="panel-body card-2" style="position:relative">
        {{#properties}}
        {{#freeParkingLots}}<b style="position: absolute; bottom: 0px; right: 10px; font-size: 24px; color: {{freeParkingLotsColor}}">
                          {{freeParkingLots}}</b>{{/freeParkingLots}}
        {{#imageThumb}}<img id="parkingMenuImage{{identifier}}" src="{{imageThumb}}"
                          style="margin-right: 8px; float: left; max-height: 90px; max-width: 90px" class="card-3">{{/imageThumb}}
        <img src="{{imgsrc}}" style="padding-right: 8px; float: left">
        <b style="text-align: center">{{#unescapeHtml}}{{name}}{{/unescapeHtml}}</b>
        {{#typeLabel}}<br><b id="parkingMenuTypeLabel{{identifier}}">
            <script>$("#parkingMenuTypeLabel{{identifier}}").html(Globalization.labels.infoMenu.type)</script>
        </b>{{typeLabel}}{{/typeLabel}}
        {{#distanceFromSearchCenter}}<br><b id="parkingMenuTextSearchDistanceFromSearchCenter{{identifier}}">
            <script>$("#parkingMenuTextSearchDistanceFromSearchCenter{{identifier}}").html(Globalization.labels.textSearchMenu.distanceFromSearchCenter)</script>
        </b>{{distanceFromSearchCenter}} m{{/distanceFromSearchCenter}}
        {{#distanceFromGPS}}<br><b id="parkingMenuTextSearchDistanceFromGPS{{identifier}}">
            <script>$("#parkingMenuTextSearchDistanceFromGPS{{identifier}}").html(Globalization.labels.textSearchMenu.distanceFromGPS)</script>
        </b>{{distanceFromGPS}} m{{/distanceFromGPS}}{{/properties}}
    </div>
</div>
<!-- {{/features}} -->
```

ParkingSearcher in main menu

- Final version of the button with call to module logic

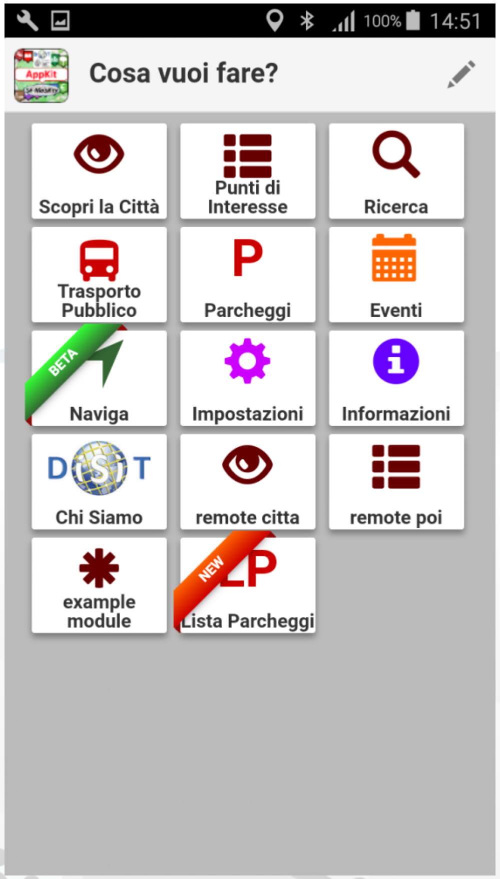


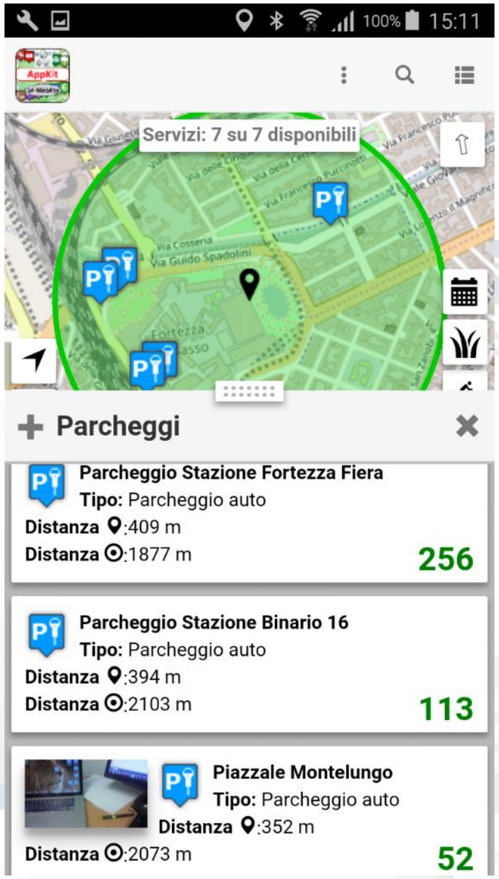
```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps() SearchManager.search('ParkingSearcher');",
    "iconId": "",
    "iconClass": "",
    "iconFontSize": "",
    "iconColor": "",
    "imgSrc": "",
    "imgHeight": "",
    "text": "LP",
    "textFontSize": "38px",
    "textColor": "#CC0000",
    "captionId": "principalMenuParkingSearcher",
    "captionTextId": "moduleParkingSearcher",
    "step": "",
    "stepId": "",
    "ribbon": true,
    "ribbonId": "",
    "ribbonStyle": "background: #CC0000;background: linear-gradient(#FF6600 0%, #CC0000 100%);",
    "ribbonText":  "NEW",
    "removed": false,
    "index": 0
  }
]
```

parkingSearcher.principalMenu.json

The search function of the variable SearchManager **asks the user where want search** (GPS, Manual or Last Service) and then call the **search function** of the variable which is passed as string

ParkingSearcher Module Finished





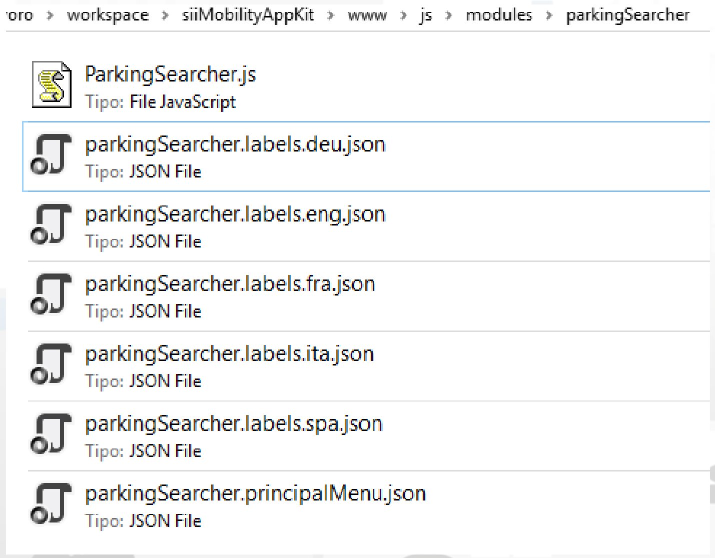
Parcheggio Stazione Fortezza Fiera
Tipo: Parcheggio auto
Distanza :409 m
Distanza :1877 m
256

Parcheggio Stazione Binario 16
Tipo: Parcheggio auto
Distanza :394 m
Distanza :2103 m
113

Piazzale Montelungo
Tipo: Parcheggio auto
Distanza :352 m
Distanza :2073 m
52

Create FuelStationSearcher Module

- Now you can try to **exercise to create the module** that **finds fuel stations** and show users a list of fuel stations that contains the **current prices of each fuel served** by each fuel station.
- To start: copy these files to a **new folder** that will have the **name of the new module** (i.e., **FuelStationSearcher**): the **names of the files** copied have to be changed to get the **module name as a prefix.**



oro › workspace › siiMobilityAppKit › www › js › modules › parkingSearcher

ParkingSearcher.js
Tipo: File JavaScript

parkingSearcher.labels.deu.json
Tipo: JSON File

parkingSearcher.labels.eng.json
Tipo: JSON File

parkingSearcher.labels.fra.json
Tipo: JSON File

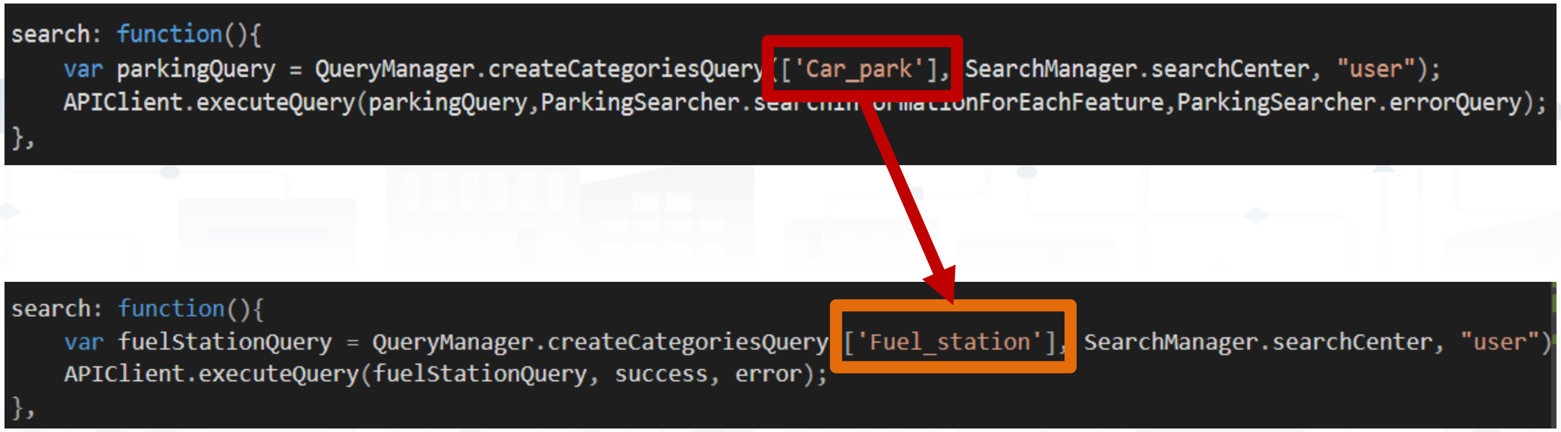
parkingSearcher.labels.ita.json
Tipo: JSON File

parkingSearcher.labels.spa.json
Tipo: JSON File

parkingSearcher.principalMenu.json
Tipo: JSON File

Create FuelStationSearcher Module

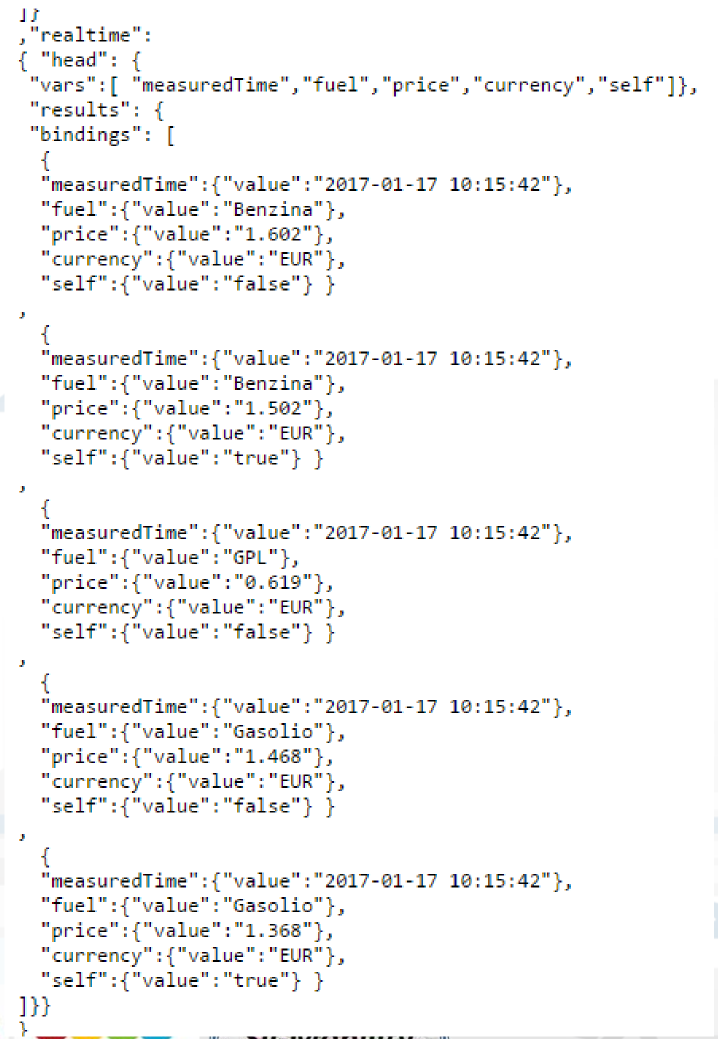
- Follow the slides from page 18 to the end.
- Remember that the ID of fuel stations is **Fuel_station,** then the search function will change this way:



```
search: function(){
    var parkingQuery = QueryManager.createCategoriesQuery(['Car_park'], SearchManager.searchCenter, "user");
    APIClient.executeQuery(parkingQuery,ParkingSearcher.searchInformationForEachFeature,ParkingSearcher.errorQuery);
},
```

```
search: function(){
    var fuelStationQuery = QueryManager.createCategoriesQuery(['Fuel_station'], SearchManager.searchCenter, "user")
    APIClient.executeQuery(fuelStationQuery, success, error);
},
```

Create FuelStationSearcher Module



```
]}
,"realtime":
{ "head": {
 "vars":[ "measuredTime","fuel","price","currency","self"]},
 "results": {
 "bindings": [
  {
  "measuredTime":{"value":"2017-01-17 10:15:42"},
  "fuel":{"value":"Benzina"},
  "price":{"value":"1.602"},
  "currency":{"value":"EUR"},
  "self":{"value":"false"} }
,
  {
  "measuredTime":{"value":"2017-01-17 10:15:42"},
  "fuel":{"value":"Benzina"},
  "price":{"value":"1.502"},
  "currency":{"value":"EUR"},
  "self":{"value":"true"} }
,
  {
  "measuredTime":{"value":"2017-01-17 10:15:42"},
  "fuel":{"value":"GPL"},
  "price":{"value":"0.619"},
  "currency":{"value":"EUR"},
  "self":{"value":"false"} }
,
  {
  "measuredTime":{"value":"2017-01-17 10:15:42"},
  "fuel":{"value":"Gasolio"},
  "price":{"value":"1.468"},
  "currency":{"value":"EUR"},
  "self":{"value":"false"} }
,
  {
  "measuredTime":{"value":"2017-01-17 10:15:42"},
  "fuel":{"value":"Gasolio"},
  "price":{"value":"1.368"},
  "currency":{"value":"EUR"},
  "self":{"value":"true"} }
]}}
}
```

An **additional difficulty** with respect to the example of the car parks is given by:

- the **type of fuels**
- fuels may have a "**self**" and a **regular price**

It is possible to decide whether:

- **ask in advance** what fuels users want to find
- **allow filtering** the results according to fuels retrieved

Maybe, in **first case** you should be able to save **users preferences** for **future searches**

Documentation

Sii-Mobility

- **Documentation Smart City API, version 1, January 2017**
  - http://www.disit.org/6991 (document from Sii-Mobility)
- **App Kit development page:**
  - http://www.disit.org/6977 (slides ready, and video to appear)
- **Ontology and Km4City Tools:**
  - Http://www.km4city.org
  - http://www.disit.org/6506 Ontology and documentation
- **Sii-Mobility is Open Source on GitHub as DISIT lab:**
  - https://github.com/disit
  - https://github.com/disit/siiMobilityAppKit (mobile App kit)
- **Data Ingestion processes and tools, tutorial:**
  - http://www.sii-mobility.org/index.php/documentazione/slide-e-altro
  - http://www.disit.org/6690
- **Deliverables of Sii-Mobility:**
  - http://www.sii-mobility.org/index.php/documentazione/deliverable
- **FAQ of Sii-Mobility:**
  - http://www.sii-mobility.org/index.php/il-progetto/faq
- **Promotional Kit for «Toscana dove cosa App»**
  - http://www.sii-mobility.org/index.php/documentazione/kit-promozionale-app-toscana

Open Source

Km4City, Sii-Mobility, RESOLUTE, REPLICATE: smart city big data open source tools. Km4City is a knowledge base and a research line of DISIT lab mainly developed before the start of Sii-Mobility, RESOLUTE, REPLICATE projects. While it has been mainly improved by them. Those projects are complementar each other and almost all of them use and contribute the Km4City research line.ServiceMap smart city knowledge base tool: smart city service tool (mainly developed for Sii-Mobility project) for accessing to km4city knowledge base, for service browsing and query, for **Smart City API** for mobile and for mobile development tool, http://www.disit.org/km4city

- **Km4City ontology model and files** are accessible from http://www.disit.org/km4city improved with the support of projects as Sii-mobility, REPLICATE and RESOLUTE
- **SCE, Smart City/Cloud Engine front end interface,** SCE is part of DISCES a Distributed SCE Scheduler Tools (SCE: Smart City/Cloud Engine), a DISIT tool for smart environments. It is currently in use in Km4City tools, and in ICARO Cloud project and service, see CLOUD page. Developed for ICARO, and then improved for Sii-Mobility, and used in many other projects
- **SCE, Smart City/Cloud Engine backend,** SCE is part of DISCES a Distributed SCE Scheduler Tools (SCE: Smart City/Cloud Engine), a DISIT tool for smart environments. It is currently in use in Km4City tools, and in ICARO Cloud project and service, see CLOUD page. Developed for ICARO, and then improved for Sii-Mobility, and used in many other projects
- **DIM-RIM**: Data Ingestion Manager and RDF Indexing Manager, WEB page on DISIT lab with user manuals, DIM and RIM area used in Km4City and tools, Sii-Mobility smart city national SCN project, RESOLUTE H2020
- **Dashboard Builder and executo**r: a tool for creating dashboard for smart city. See Km4City example of dashboard as reported in http://www.km4city.org/controlroomtools.html for the documentation see http://www.disit.org/6935 which is manual with examples regarding widgets. Developed for REPLICATE Project, and used in others as Sii-Mobility, RESOLUTE.
- **Sii-Mobility Dev Kit Mobile AppKm4city**: Open Source version of the Sii-Mobility mobile and web app, open modular (the full version is operative and accessible on all stores as "Firenze dove cosa", or " Toscana dove cosa", you can install from http://www.km4city.org/app). Developed for Sii-Mobility, adopted for the training and development meeting of the 24 January 2017, and as a basis for the Hackathon of 7-8 April 2017 in Florence.

Acknowledgement

- Thanks to the European Commission for founding. All slides reporting logo of **RESOLUTE H2020** are representing tools and research founded by European Commission for the RESOLUTE project. **RESOLUTE** has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 653460).
- Thanks to the European Commission for founding. All slides reporting logo of **REPLICATE H2020** are representing tools and research founded by European Commission for the REPLICATE project. **REPLICATE** has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 691735).
- Thanks to the MIUR for co-founding and to the University of Florence and companies involved. All slides reporting logo of **Sii-Mobility** are representing tools and research founded by MIUR for the Sii-Mobility SCN MIUR project.
- **Km4City** is an open technology exploited by those projects and line of research of DISIT Lab. Some of the innovative solutions and research issues developed into the above mentioned projects are also compliant and contributing to the Km4City approach and thus are contributing to the open Km4City model of DISIT lab.

Horizon 2020
European Union Funding
for Research & Innovation

Horizon 2020
European Union Funding
for Research & Innovation