100% OPEN SOURCE

EU GDPR COMPLIANT

PEN Test Passed

Powered by FIWARE

SNAP4CITY on EUROPEAN OPEN SCIENCE CLOUD MARKETPLACE

Node-RED

SNAP4CITY

KM4CITY

SNAP4 Appliances and Dockers Installations

LIVING LAB

Be smart in a SNAP!

SMARTCITY EXPO WORLD CONGRESS

15 - 17 NOV 2022
BARCELONA & ONLINE

Visit our stand:
Pavillon 2, stand B86

Advanced Smart City API, ASCAPI Web and Mobile App development

December 2022, Course
https://www.snap4city.org/577

SCALABLE SMART ANALYTIC APPLICATION BUILDER FOR SENTIENT CITIES

UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO | DISIT

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

SNAP4CITY

Powered by KM4CITY

scalable Smart aNalytic APplication builder for sentient Cities: for Living Lab and co-working with Stakeholders

https://www.Snap4City.org

December 2022, Course
https://www.snap4city.org/577

Paolo Nesi, paolo.nesi@unifi.it
https://www.Km4City.org
https://www.disit.org

100% OPEN SOURCE

Powered by FIWARE

FREE TRIAL

PEN Test Passed

EU GDPR COMPLIANT

SNAP4 Appliances and Dockers Installations

EUROPEAN OPEN SCIENCE CLOUD

Node-RED

JS Foundation

E015 digital ecosystem

NVIDIA

SNAP4CITY KM4CITY

SMART SOLUTIONS AND DECISION SUPPORT SYSTEMS

CONTROL ROOMS - DECISION SUPPORT SYSTEMS - WHAT-IF ANALYSIS - BUSINESS INTELLIGENCE - SIMULATIONS - SMART APPLICATIONS

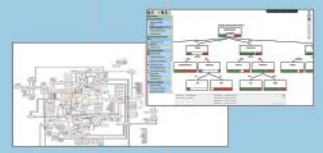
DASHBOARDS - VISUAL ANALYTICS - SYNOPTICS - DIGITAL TWIN - GRAPHICAL WIDGETS - ANALYTICS - GUI CUSTOM STYLES - VISUAL PROGRAMMING

DASHBOARDS, WIDGETS
TEMPLATES

PREDICTION - ANOMALY DETECTION - CLUSTERING - ROUTING - SENTIMENT NLP - TRAFFIC FLOW
PEOPLE FLOWS - SDG - 15 MIN CITY INDEX - KPI - HEATMAPS - ORIGIN DESTINATION - ETC...

API - MICROSERVICES - GIS - BPM
VIDEO - REPORTS - MAPS - 3D ...

ANY: DATA, BROKER, NETWORK AND VERTICAL



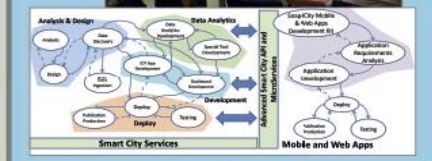
EXPERT SYSTEM, KNOWLEDGE BASE
SEMANTIC REASONING
SMART DATA MODEL
IOT DEVICE MODELS, STORAGE

BIG DATA ANALYTICS, ARTIFICIAL INTELLIGENCE
EXPLAINABLE AI, MACHINE LEARNING
OPERATIVE RESEARCH, STATISTICS

VISUAL PROGRAMMING, ADAPTERS
DATA FLOWS, WORKFLOWS
PARALLEL DISTRIBUTED PROCESSING
DATA DRIVEN

Native and External Applications

- Smart Parking
- Smart Light
- Smart Waste
- Smart Energy
- Social Media Analysis
- ...



METHODOLOGIES
LIVING LABS
COURSES AND COMMUNITY
DEVELOPMENT TOOLS

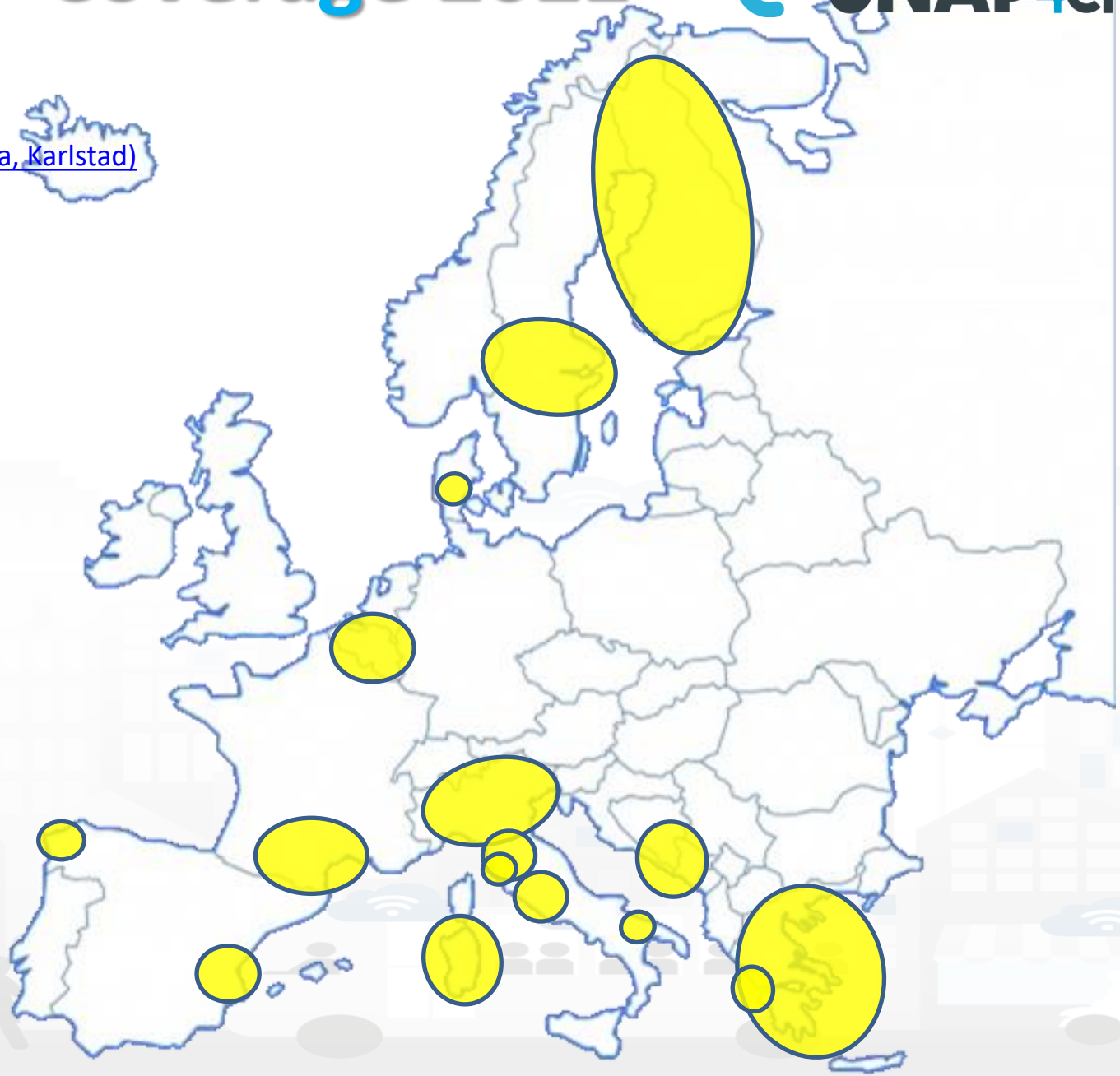
Snap4City/Industry structure

- The **Snap4xxxx** solution is released in Open Source, VM and Docker with fully support of MultiTenant/multiple-Organizations
  - Each Organization may be configured for a separate environment with a set of Maps, Menus, Users, Data, Dashboards, IOT Apps, MicroApplications, Custom Widgets, Models, resources, open data, etc.
- **Https://www.Snap4City.ORG** is the main instance of Snap4xxxx solution managed by DISIT Lab. The main documentation is located and updated on Snap4City.org, GitHUB, dockerHub and Node-Red Library. Snap4City.org is where the last tools are tested and news published.
  - Organizations on Snap4City.org have been created with contracts as for *Platform as a Service*, for testing and for providing *SmartCity as a Service* as well as *Industry 4.0 as a Service*

Coverage 2022

Main Organizations/areas

- Antwerp area (Be)
- Bologna (I)
- Capelon (Sweden: Västerås, Eskilstuna, Karlstad)
- DISIT demo (multiple)
- Dubrovnik, Croatia
- Firenze area (I)
- Garda Lake area (I)
- Greece (Gr)
- Helsinki area (Fin)
- Livorno area (I)
- Lonato del Garda (I)
- Modena (I)
- Mostar, Bosnia-Herzegovina
- Oslo & Padova (Impetus)
- Pisa area (I)
- Pistoia (I)
- Pont du Gard, Occitanie (Fr)
- Prato (I)
- Roma (I)
- Santiago de Compostela (S)
- Sardegna Region (I)
- Siena (I)
- SmartBed (multiple)
- Toscana Region (I), SM
- Valencia (S)
- Venezia area (I)
- WestGreece area (Gr)



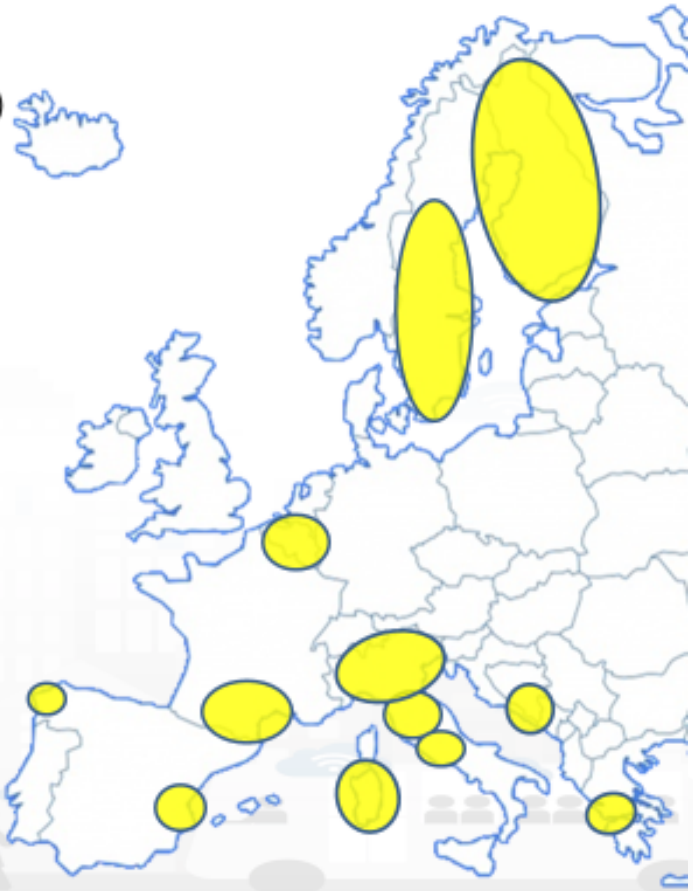
Snap4City/Industry Community

- Most of Organizations on Snap4City.org also correspond to companies or institutions that have an installation of Snap4City tools on their Premise,
  - such as: Pisa, SmartGarda Lake, Snap4, ALTAIR, etc.
- This double way allows them to:
  - test the news,
  - share experiences with other groups,
  - get visibility,
  - work in the collaborative environment, and
  - be better supported by Snap4City.org and DISIT Lab personnel.
- Each instance of Snap4xxxx solution ***can decide to join the federation*** of SmartCity API to exploit shared data.
  - This allows to exploit regional data for city installations applications (web, mobile, dashboards, etc.) without reloading them for example.

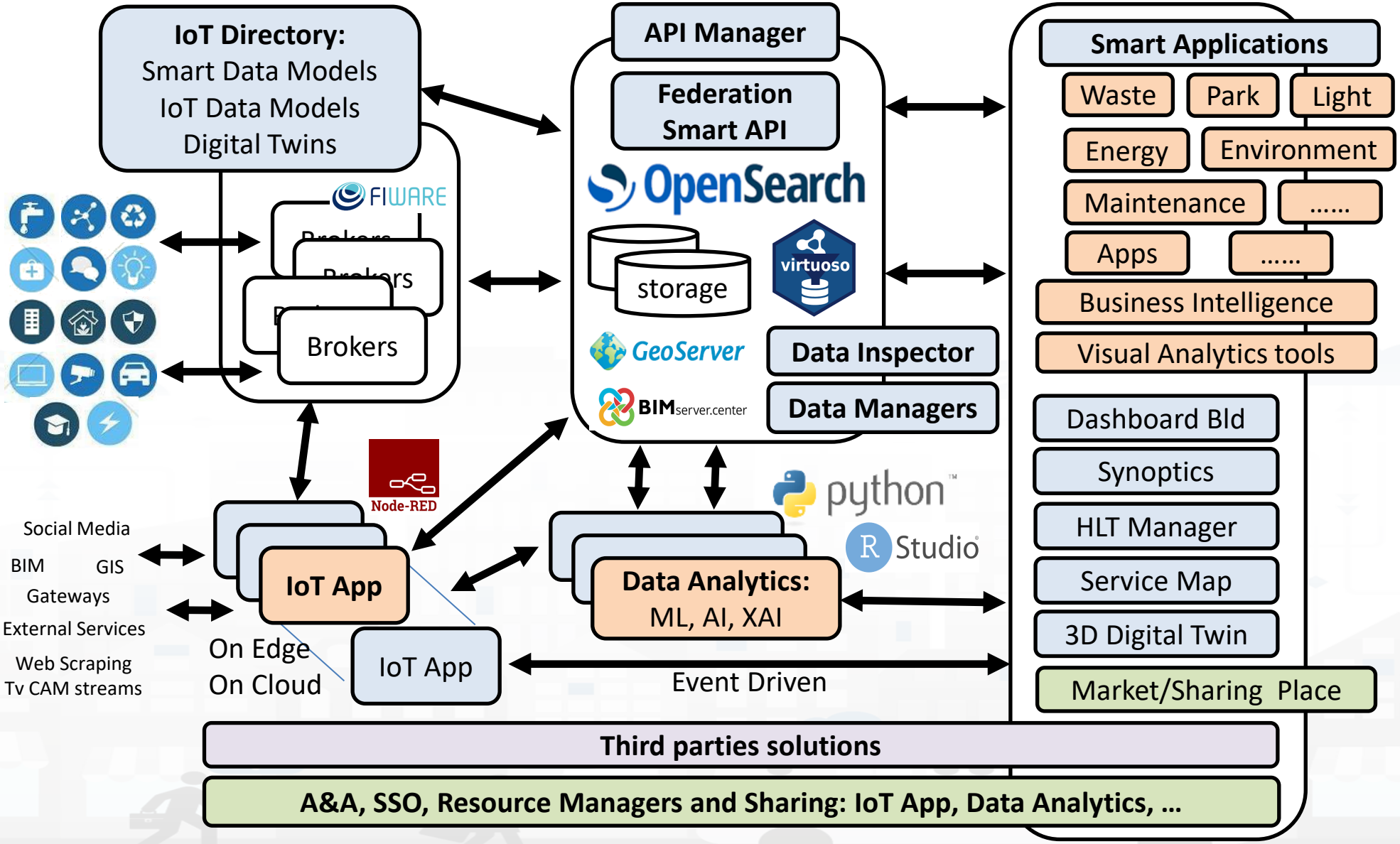
Main Organizations/areas

- Antwerp area (Be)
- Capelon (Sweden: Västerås, Eskilstuna, Karlstad)
- DISIT demo (multiple)
- Dubrovnik, Croatia
- Firenze area (I)
- Garda Lake area (I)
- Helsinki area (Fin)
- Livorno area (I)
- Lonato del Garda (I)
- Modena (I)
- Mostar, Bosnia-Herzegovina
- Pisa area (I)
- Pont du Gard, Occitanie (Fr)
- Roma (I)
- Santiago de Compostela (S)
- Sardegna Region (I)
- SmartBed (multiple)
- Toscana Region (I), SM
- Valencia (S)
- Venezia area (I)
- WestGreece area (Gr)

Snap4City (C), October 2020

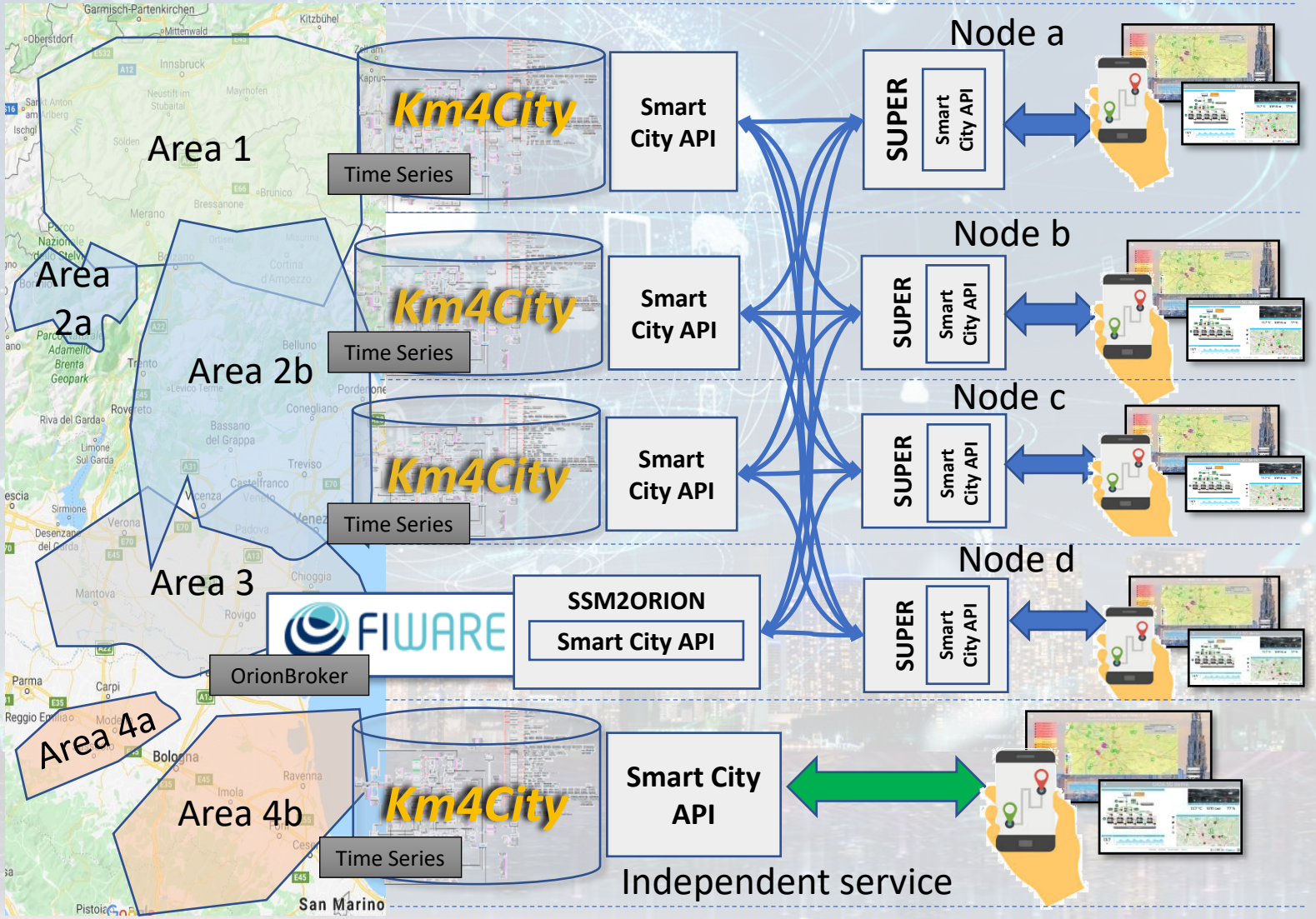


Tech Arch - 2023



Federation of Smart City Services

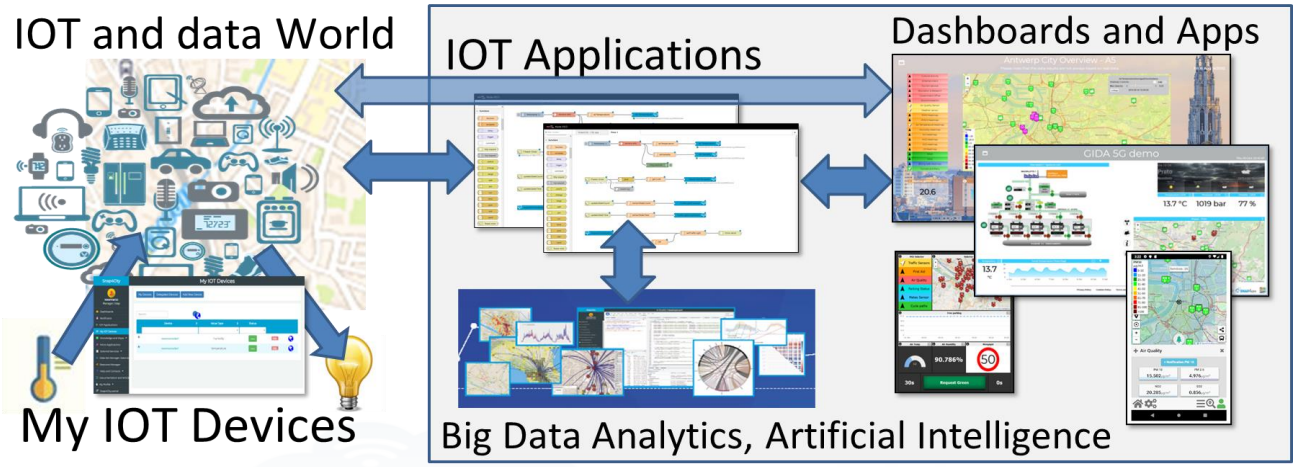
Knowledge Base
Semantic Reasoners



- Km4City **Semantic Reasoner**
- **ServiceMap** interoperability
- **Seamless for multiple Mobile Apps**
- **Smart City API**
- **Super**:
  - distributed access and sharing services
  - Each city control its own data
  - Final user can pass from one city / area to another in seamless manner: without changing the mobile Apps

Free Trial

- Register on WWW.snap4city.org
  - Subscribe on **DISIT Organization**
- **You can:**
  - Access on basic Tools
  - Access to a large volume of Data
  - Create Dashboards
  - Create IOT Applications
  - Connect your IOT Devices
  - Exploit Tutorials and Demonstrations



IF you need to go more in deep you can ask us to pass at the next Role becoming full AreaManager with full rights of development, also for Data Analytics, machine learning, etc.

https://www.snap4city.org/577

On Line Training Material (free of charge)

	1st part	2nd part	3rd part	4th part	5th part	6th part	7th part	8th
what	Overview	Dashboards	IOT App, IOT Network	Data Analytics	Data Ingestion processes	System and Deploy Install	Smart City API: Web & Mob. App	Design and Develop Smart Solutions
PDF 2022								
Interactive (2022) with video and animations								

Video1	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube
Video2	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube
Video3	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube	▶ YouTube
Video4	▶ YouTube	▶ YouTube	▶ YouTube	none	▶ YouTube	none	none

General Overview of the full Course

1. *General Overview*
2. ***Dashboards** Creation and Management, **Business Intelligence***
3. ***IOT Applications** development, IOT Devices, IOT Networks*
4. ***Data Analytics**, in R Studio, in Python, how to Exploit and Manage Data Analytics in IOT Applications*
5. ***Data Ingestion**, Data Warehouse, Data Gate, IOT Device Data ingestion, IOT App for Data Ingestion, **Interoperability**, etc.*
6. *Snap4City **Installation, Extension, Administration***
7. ***Smart city API** (internal and external) **Web and Mobile App development** tool kit*
8. *How to **Design and Develop Smart Solutions***

A number of the training sections include exercitations

Updated versions on: https://www.snap4city.org/577

See also courses in ITALIANO: https://www.snap4city.org/485

7th Part Agenda

GO
- **Smart City API: Internal and External**

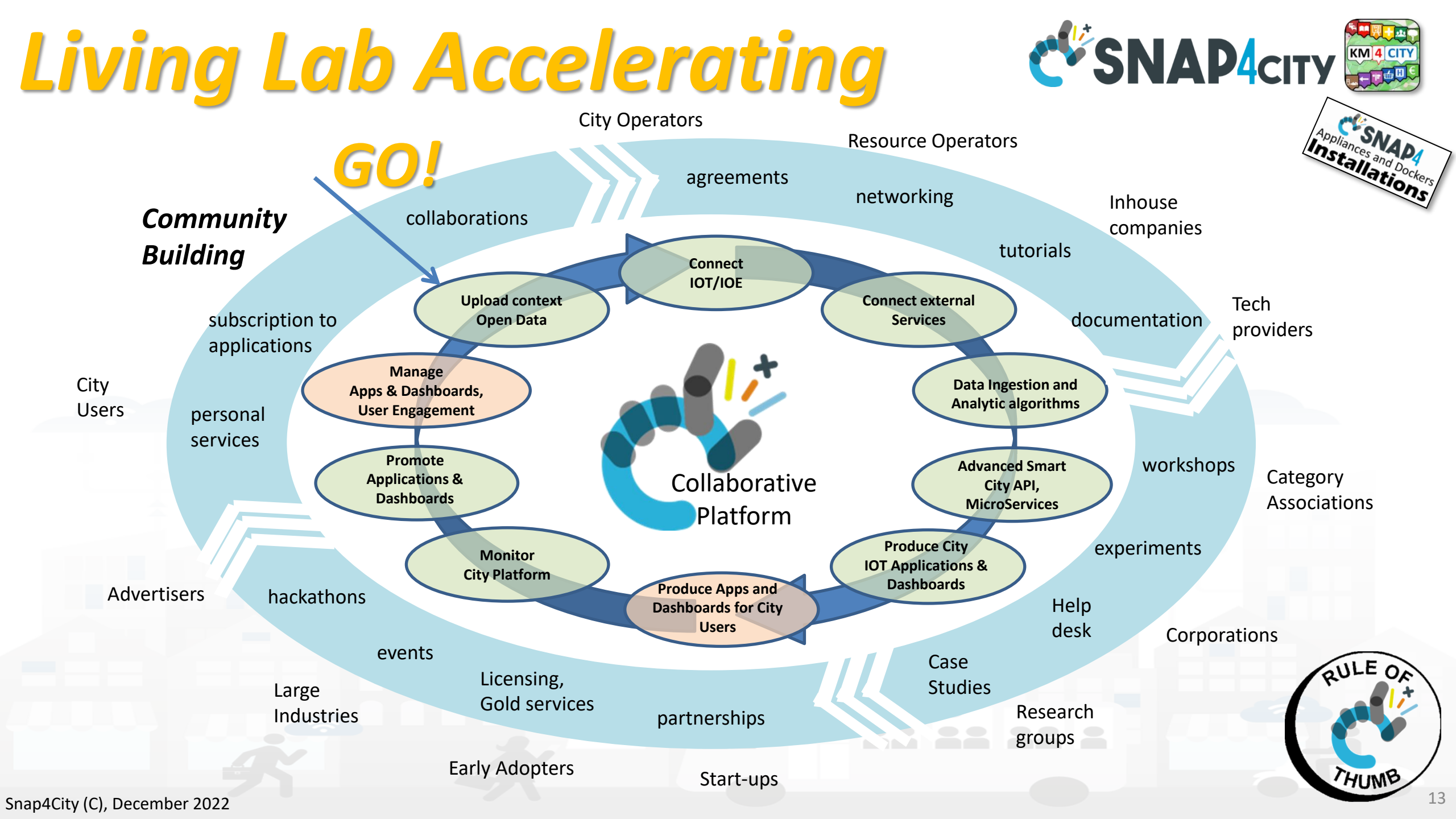
GO
- **Forging and Managing Flexible Mobile Apps, Web Apps and MicroApplications**
  - Web and Mobile App with Open Development Kit
  - Understanding how City User are using the City Services
  - Engagement of City Users, towards a participated attitude
  - Connected Drive
  - Integration with Telegram: SnapBot

GO
- **Advanced Smart City API, MicroServices, Snap4City API**

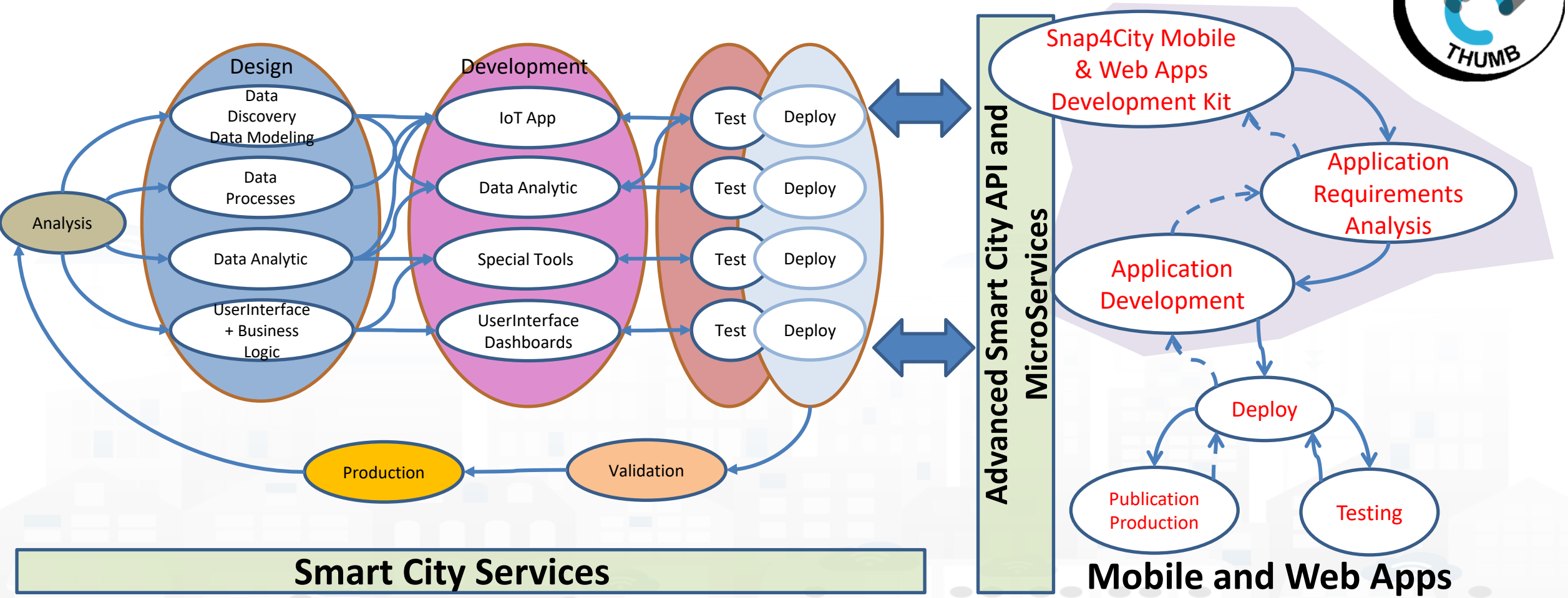
GO
- **Federated Knowledge Base and Smart City API**

GO
- **Web and Mobile App Development Kit**
- **Acknowledgement**

Living Lab Accelerating



Develop Mobile & Web Applications Exploiting Snap4City Smart City Services

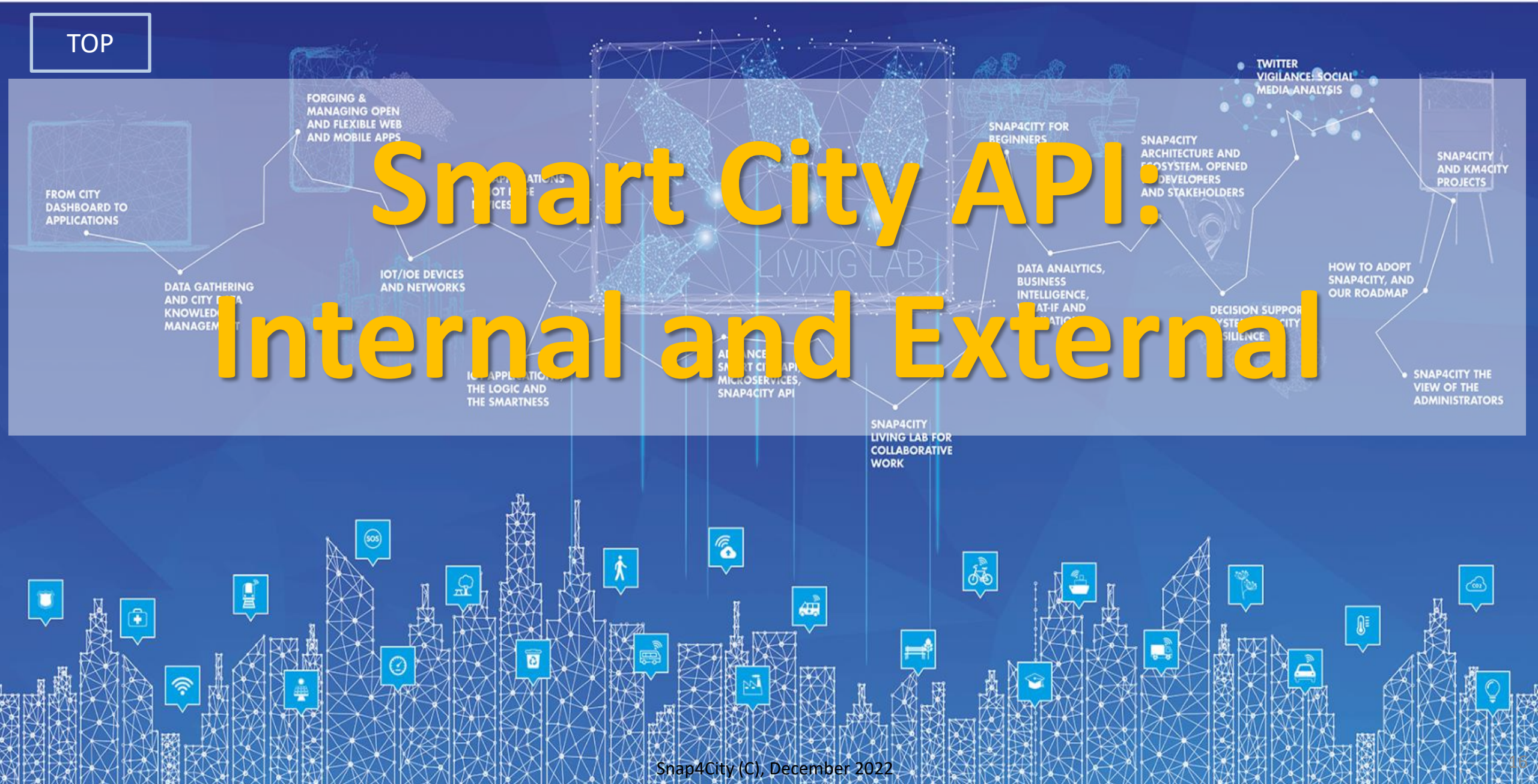


Levels of Difficulty

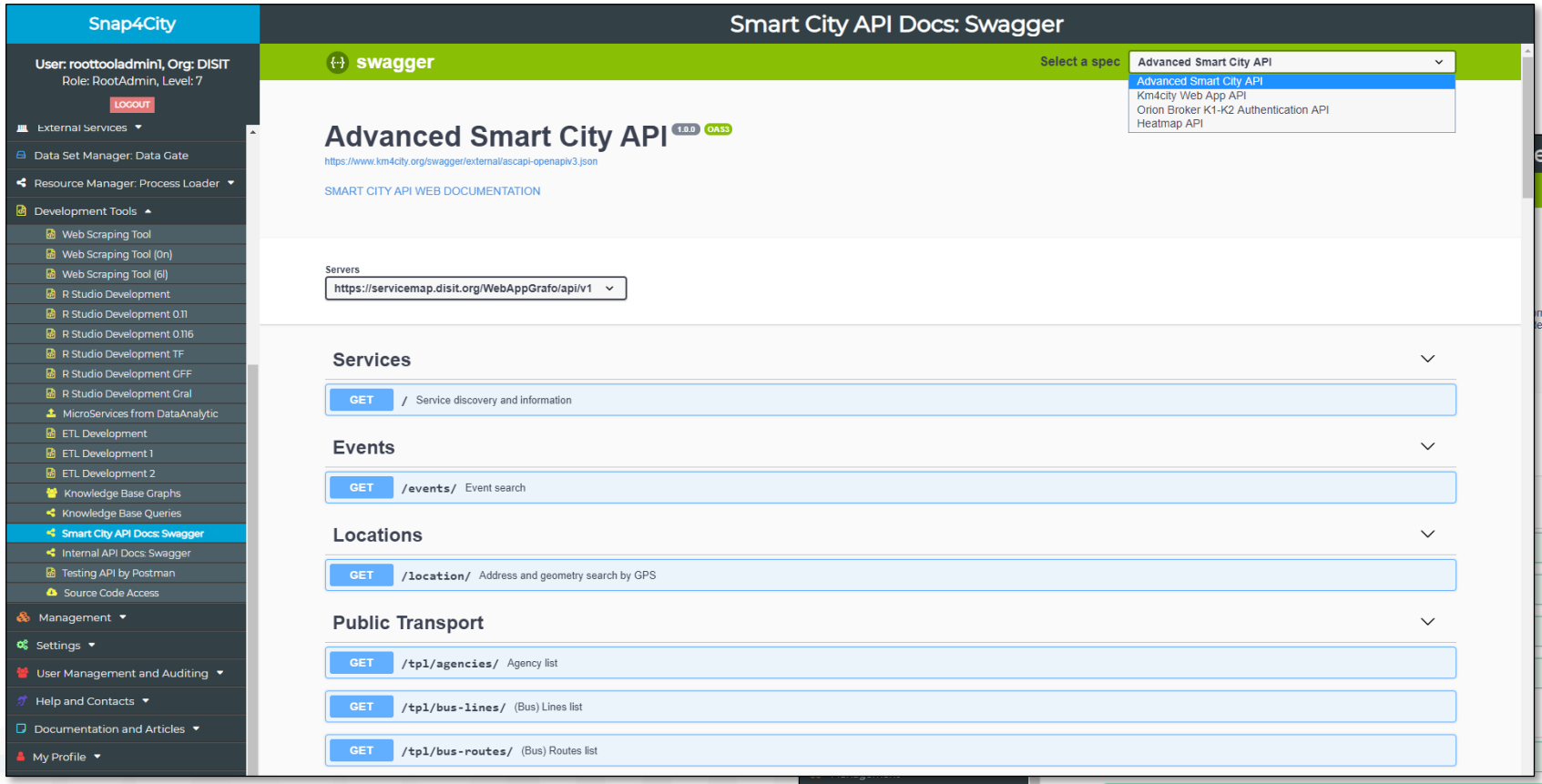
- Easy.  non programmer level
- Moderate.  Some JavaScript rudiment coding
- Good.  JavaScript programming
- Golden.  Programming in R Studio
- Professional.  Exploiting Smart City API
- Excellent.  Developing Full IOT Applications, Dashboard and Mobile Apps

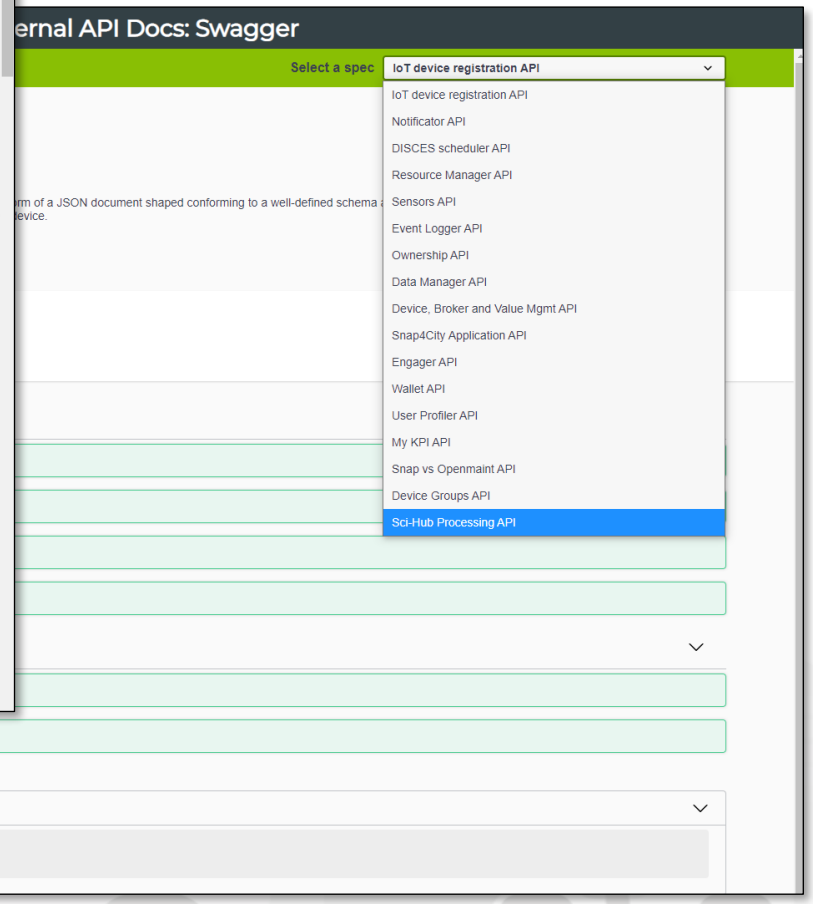
TOP

Smart City API: Internal and External



Internal and External Smart City API

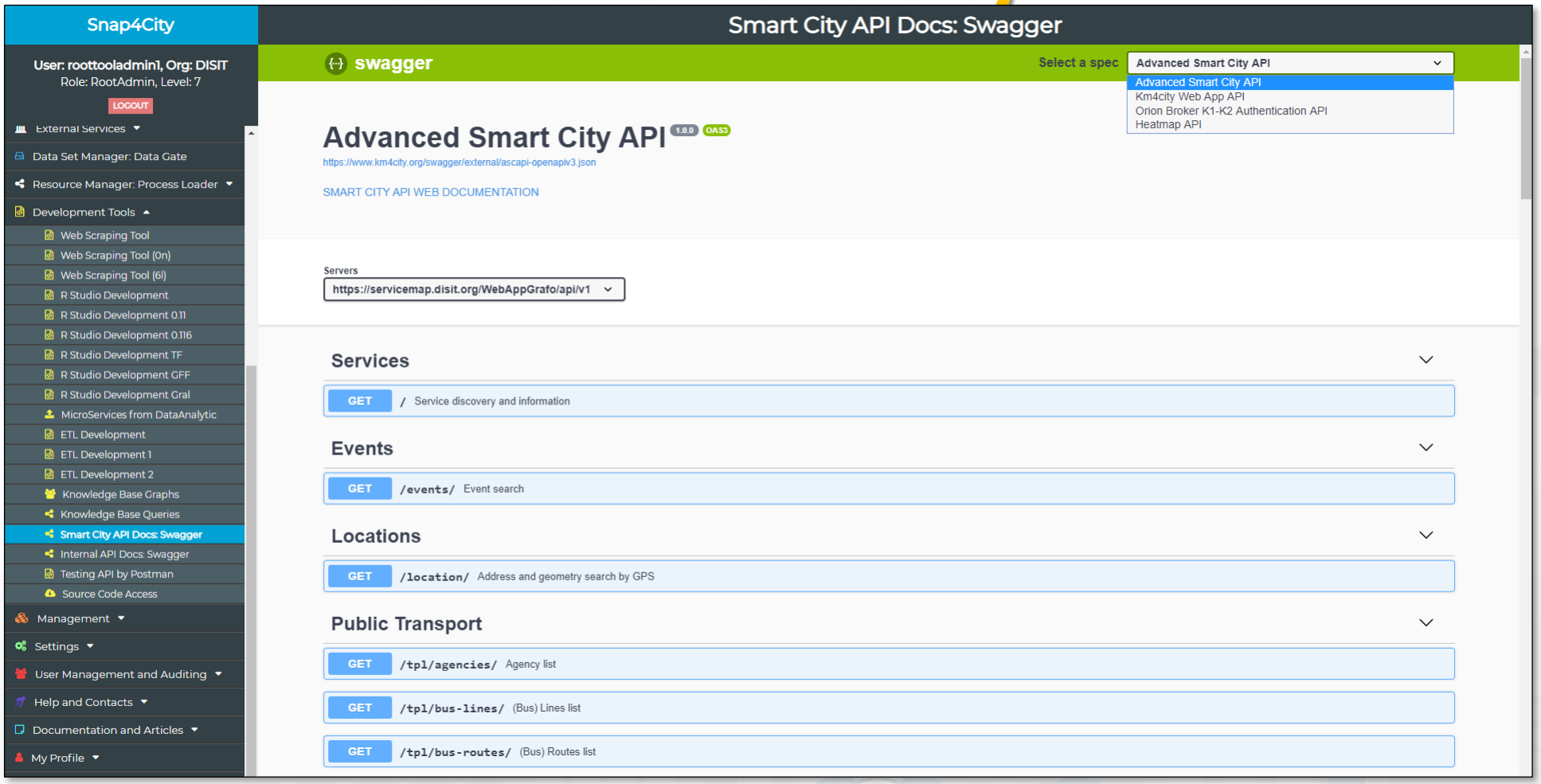




https://www.km4city.org/swagger/external/index.html

https://www.km4city.org/swagger/internal/index.html

External Smart City API

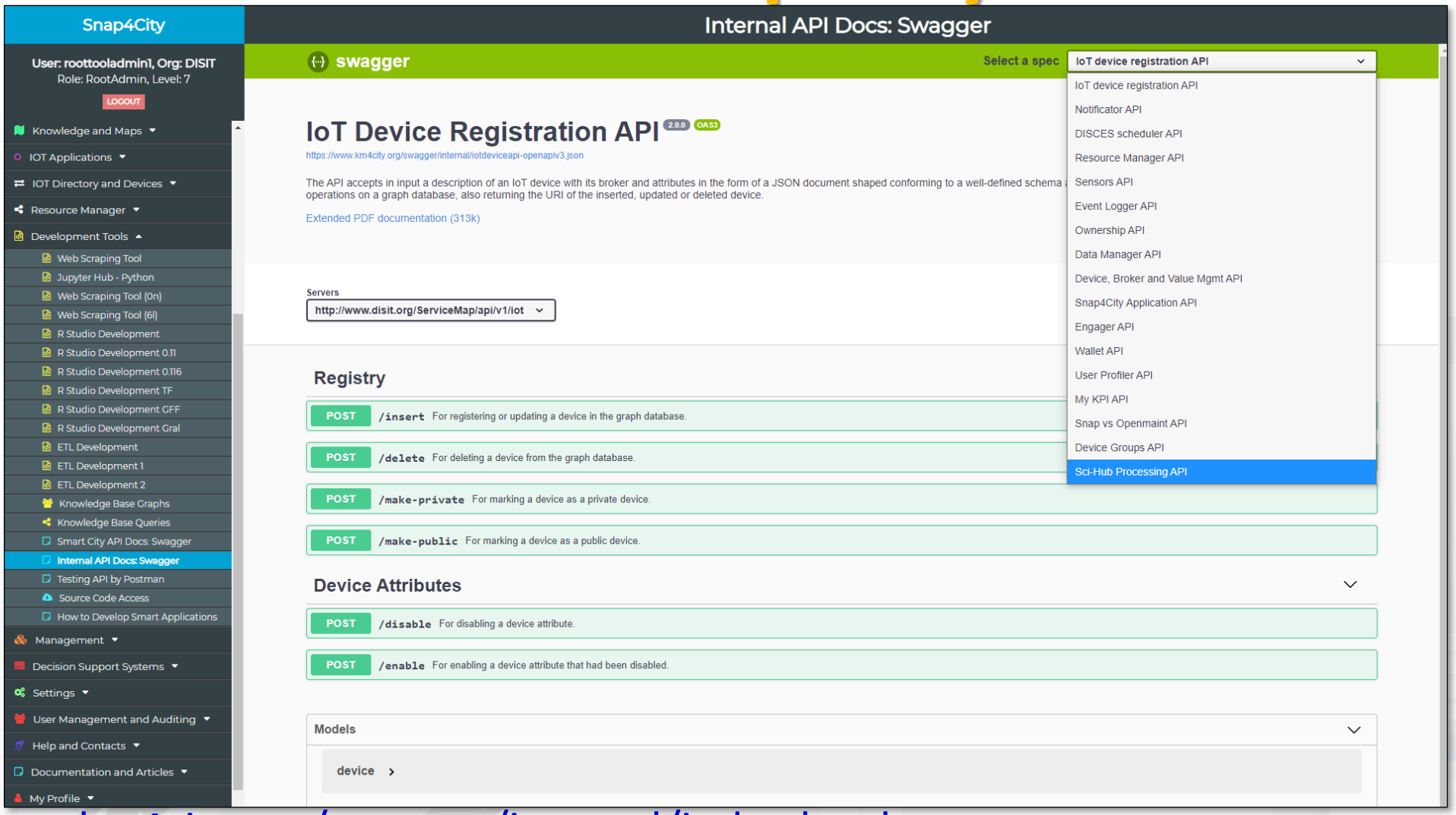


https://www.km4city.org/swagger/external/index.html

External Smart City API

- **Advanced Smart City API**
  - To access the Service Map resources and query
- **Km4city Web App API**
  - To exploit MicroApplications created as tools for Dashboards, totem, web Apps, etc.
- **Orion Broker K1-K2 Authentication**
  - To communicate with IOT Orion Brokers exploiting the Secure Filter of Snap4City.
- **Heatmap**
  - To save and access to HeatMaps of the Heatmap server

Internal Snap4City API



https://www.km4city.org/swagger/internal/index.html

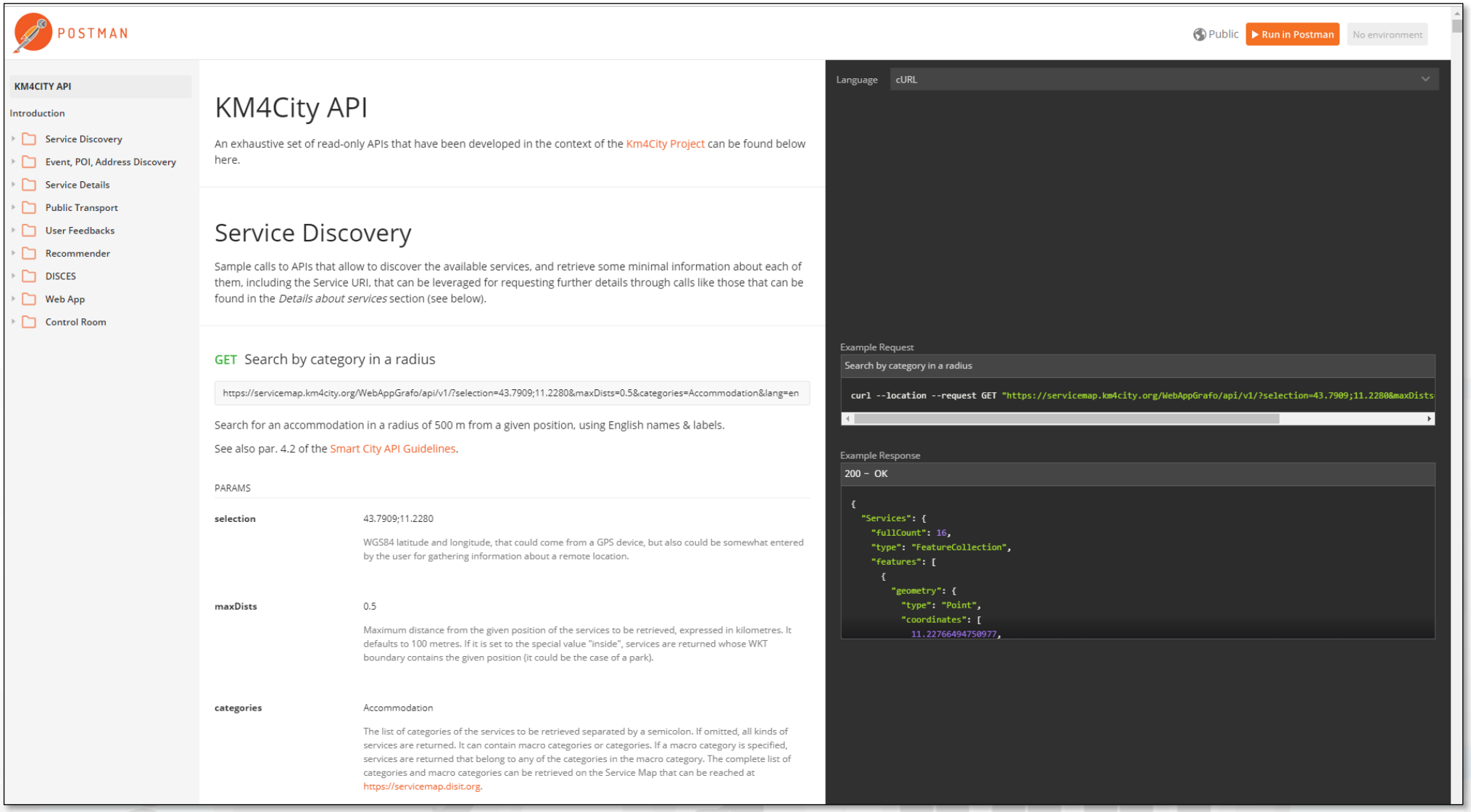
Internal Smart City API

- **IOT devices and tools API:**
  - **IoT device registration API**
    - API of the IOT Directory
  - **Sensors API**
    - API of the IOT Directory
  - **Device, Broker and Value Mgmt API**
    - API of the IOT Directory
- **Mobile App management**
  - **User Profiler API**
    - To manage the user profile for the Engager on Mobile Apps
  - **Engager API**
    - From the Engager to prepare engagements to the Mobile Apps
  - **Wallet API**
    - From the Engager to Wallet o the users of Mobile Apps and in general
  - **Snap4City Application API**

Internal Smart City API

- **Resources and entities**
  - **Snap4City Application API**
    - To manage IOT Apps
  - **My KPI API**
    - To manage MyKPI, MyPOI, POI, etc.
  - **Data Manager API**
  - **Resource Manager API**
    - To manage resources on the market place
  - **Ownership API**
    - To manage ownerships and delegations
  - **Device Groups API**
    - To manage ownerships and delegations
- **Notificator API**
- **DISCES scheduler API**
- **Event Logger API**
- **Snap vs OpenMaint API**
  - Integration with the workflow management and ticketing
- **SCI-HUB Processing API**
  - To activate data download and heatmap production from Copernicus satellite services

Postman



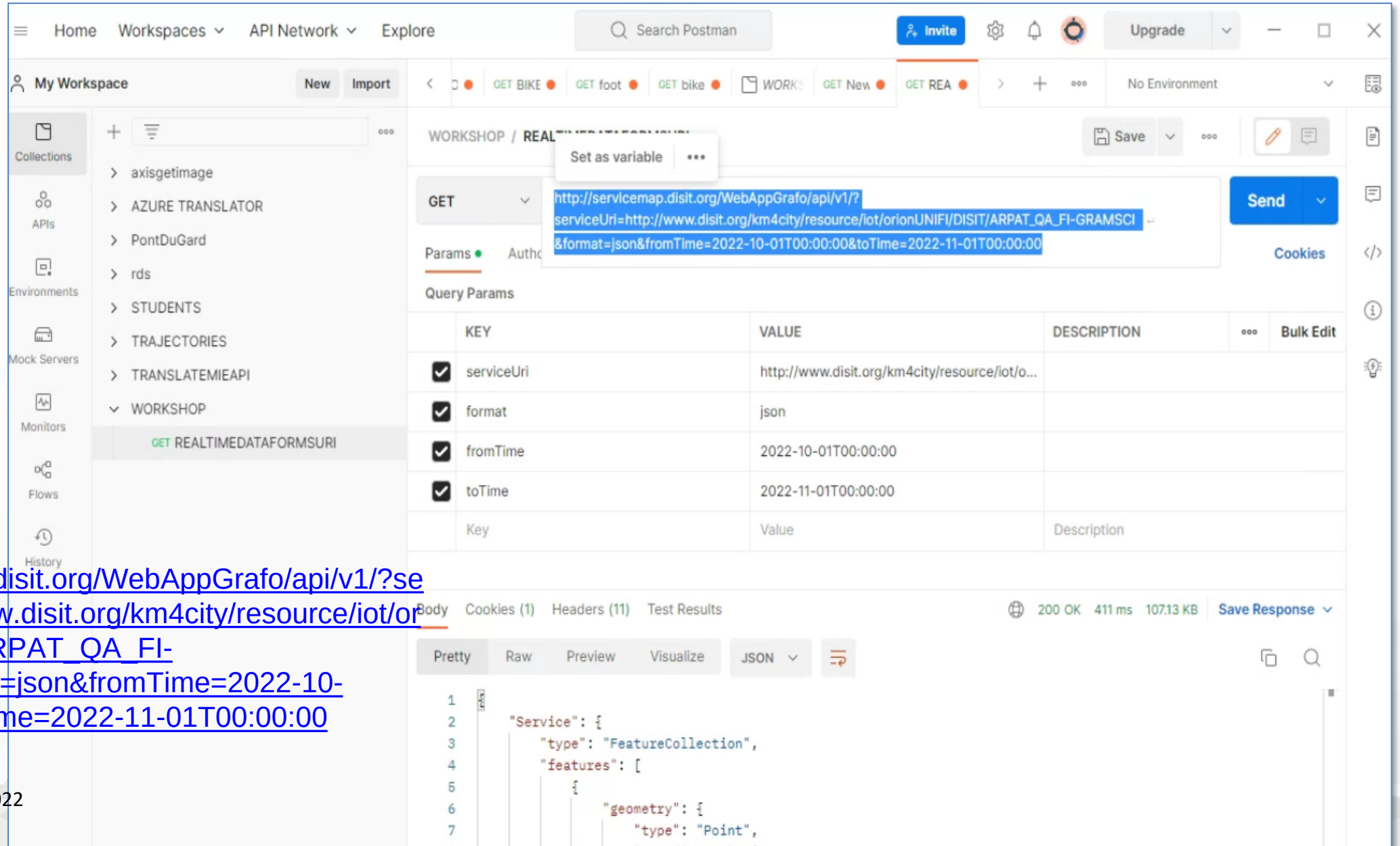
https://documenter.getpostman.com/view/4177198/km4city-api/RW83QsX5?version=latest

Exposing Services

- **Advanced Smart City API which can be confined into a single Smart City installation or Federated as well as for Super Service Map**
  - https://www.km4city.org/swagger/external/index.html
- **Federated Multiple Snap4City** Knowledge Bases. This allows the creation of mobile applications that may move from multiple cities and area accessing data and making queries transparently. This solution is presently in place among the Knowledge Bases of: Antwerp/Helsinki, Tuscany/Firenze, Sardegna, etc. The resulting Service is called Super Service Map and it is integrated in the Smart City API. For example, via:
  - https://www.disit.org/superservicemap/api/v1
- **Federated Open Data Portals** via DataGate/CKAN that presently presents now more than 13800 data sets linked for the cities of Helsinki and Antwerp.
  - https://datagate.snap4city.org/organization
  - Federation, Harvesting interface is: https://datagate.snap4city.org/harvest
- **WFS service of Snap4City** on top of Federated Smart City API or simple Smart City API of a single ServiceMap (smart City installation). This solution permits to GIS applications and platforms (such as ArcGIS OnLine ESRI, ArcGIS Enterprise ESRI, ArcGIS Map/pro Desktop, QGIS, GeoServer, etc.) to access at Snap4City data. For Example, via:
  - https://www.disit.org/superservicemap/api/v1/wfs
  - https://www.disit.org/superservicemap/api/v1/wfs?service=WFS&request=GetCapabilities&version=2.0.0
- **WMS service of Snap4City** for publishing **maps and heatmaps**, provided by an installed GeoServer third party open source tool. For example, via:
  - https://wmsserver.snap4city.org/geoserver/Snap4City/wms
  - https://www.km4city.org/swagger/external/index.html?urls.primaryName=Heatmap%20API

Test the API

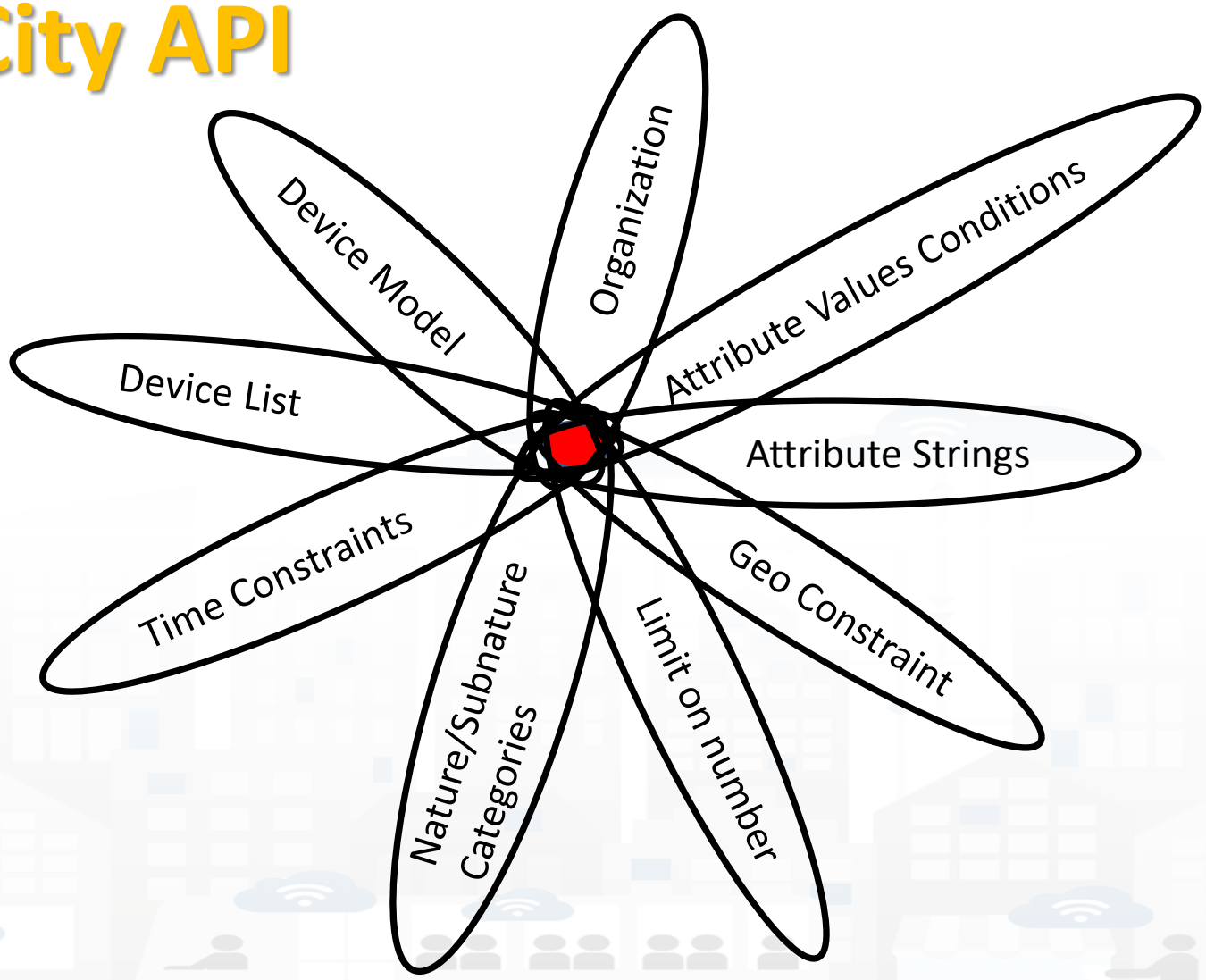
POSTMAN



http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/ARPAT_QA_FI-GRAMSCI&format=json&fromTime=2022-10-01T00:00:00&toTime=2022-11-01T00:00:00

Selection on Smart City API

- Combining different filters for selecting entities from Smart City APIs
- ***Be care***: filtering too much may lead to empty set ☺



How to Get the «Query» used in More Options (2a)

- **REST CALL by category → JSON (Options in RED), they are REST ASCAPI calls**
  - **Requesting a category, so that to see all Services of the same category (subNature)**
    - http://svealand.snap4city.org/ServiceMap/api/v1/?selection=59.581458578537955;16.71183586120606;59.62875017053684;16.875171661376957&categories=Street_light&maxResults=100&format=json
      - Please note that in the MoreOption dashboard the GPS area is neglected
    - https://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.64471;11.005751;43.89471;11.505751&categories=Green_areas&maxResults=200&format=json
      - Please note that in the MoreOption dashboard the GPS area is neglected
    - Custom PINS note: "selection" coordinates are used for collecting attributes in custom PINS. Other options such as "maxDists" cannot be used in custom PIN. All parameters can be used in other cases.
    - Different KB links are identified by their ASCAPI links: svealand.snap4city.org, servicemap.disit.org, ….
  - **Requests to SuperServiceMap for the network of Federated KBs** by using /api/…….

    Without prefixed KB to obtain merged results from more KBs. For example as:
    - /api/v1/?categories=Air_quality_monitoring_station&format=json
    - Please note that the direct links to the superservicemap can be of the form:
      - https://www.disit.org/superservicemap/api/v1/? ……

How to Get the «Query» used in More Options (2b)

- **REST CALL by ServiceURI → JSON (ServiceURI in RED), they are ASCAPI calls**
  - **Requesting single Service**
    - https://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/ARPAT_QA_FI-BOBOLI&format=json
    - https://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/ARPAT_QA_FI-MOSSE_SV&format=json
      - Different KBs links are identified by their ASCAPI links: svealand.snap4city.org, servicemap.disit.org,
  - **Requesting all IoT Devices that have been produced by the same Model**
    - **https://www.disit.org/superservicemap/api/v1?selection=59.36535064975547;13.457822799682619;59.39031474260852;13.566999435424806&model=SmartLightCapelon**&format=json
      - Please note that in this case the call is performed on the superservicemap, you can change to go directly on the right KB
      - You can specific both category and model to be more precise and focused.
    - **https://www.disit.org/superservicemap/api/v1/?selection=36.8092847020594;12.216796875000002;42.71473218539458;32.03613281250001&categories=Travel_information&format=json&fullCount=false&maxResults=500&model=DOMESTICMOVEMENTS2013-2018_1620304406**
      - In this case, we have a double filtering for model and for categories, plus other constraints
      - Please note that in the MoreOption dashboard the GPS area is neglected

How to Get the «Query» used in More Options (2c)

- **Requesting get data single device (view on map, if format HTML and not JSON)**
  Request to see the single device:
  - https://svealand.snap4city.org/ServiceMap/api/v1/?serviceUri=http://www.disit.org/km4city/resource/iot/orionCAPELON-UNIFI/CAPELON/5C0272FFFE894AF7&format=json&fromTime=3-day
  - **With ServerURI:** http://www.disit.org/km4city/resource/iot/orionCAPELON-UNIFI/CAPELON/5C0272FFFE894AF7
  - **From KB:** https://svealand.snap4city.org

- **REST CALL into the query of the more options for GIS services**
  - **Requests from GIS servers** by using WFS API and services
    - https://os.cittametropolitana.fi.it/imobi-ds/trafficsensors/1758EB9B47A3904E05673BFEEE61AB1776C4C0050E7BD7FB195EA33CC56767BE/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=TrafficSensors

Query by value

Queries can be complex *by geo-area, by cathegory, by IoT Device Model, a list of ServiceURI (all the same kind), with filters by value on specific Variables (numeric, and textual in AND), QUERY:*

- https://www.snap4city.org/superservicemap/api/v1/iot-search/?selection=43.77;11.2&maxDists=700.2&model=CarPark

- https://www.snap4city.org/superservicemap/api/v1/iot-search/?selection=42.014990;10.217347;43.7768;11.2515&model=metrotrafficsensor&**valueFilters=vehicleFlow>0.5;vehicleFlow<300**

- https://www.snap4city.org/superservicemap/api/v1/iot-search/?selection=43.77;11.2&maxDists=200.2&model=metrotrafficsensor&valueFilters=vehicleFlow>10;vehicleFlow<400&serviceUri=http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO1;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO10;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO11;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO13;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO14;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO15;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO16;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO17;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO18;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO19;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO2;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO20;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO21;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO22;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO23;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO24;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO25;http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/METRO26

How to Get the «Query» used in More Options (3)

- **ServiceMap (specific KB) and Query service**
  - The Query performed is saved and can be recalled with a QueryID, valid for that specific KB, and not accessible via SuperServiceMap / Federated KB
  - The QueryID is communicated via email
  - Specific REST Call with HTML is also provided to change the Query in server associated with the QueryID received

- **Query ID (only Read and Read/Write of the query)**
  - https://servicemap.disit.org/WebAppGrafo/api/v1/?queryId=1c8111893d40a2bb07a2078ffe299ced&format=json
  - Cannot be used for Custom PINs.
  - **Cannot be used to get data via ServiceMap since the Query ID is KB based**

Special Commands in «Query» of More Options (4)

- **Commands for Special Tool**:
  - **Traffic Flow** tool: https://firenzetraffic.km4city.org/trafficRTDetails/roads/read.php
  - **Scenario** tool: /scenario/
  - **Whatif** tool: /whatif/
- **Heatmaps**, see Data Analytic part of the training for the several versions which can be used:
  - https://wmsserver.snap4city.org/geoserver/Snap4City/wms?service=WMS&layers=PM2_5Average24HourFlorence
  - https://wmsserver.snap4city.org/geoserver/Snap4City/wms?service=WMS&layers=denseNO2_Firenze_IDW
  - WMSServer that is a GeoServer may be different for different installations of Snap4City

Time Series Data Access

- Time Series are attached to Devices which are identified by ServiceURI
- To **Access at the Time Series** (also called real time data) you can:
  1. From IoT App use the block «service info dev» **In this case, you automatically access to your private and delegated data. You do not need to perform the authentication since it is performed directly from the microservice IoT App context, both on cloud and on edge**
  2. From Python/Rstudio, Web and Mobile App, you can call Smart City API, see in this section and in ***Part 7 of the course***.
  3. Retrieve data from IoT App and pass them to Python/Rstudio as presented in other sections. This approach is viable for small amount of data, such as some thousands. For larger amount of data or to be more efficient we suggest to use case (2) which is a direct access to the Smart City API.

Private Device Data Retrieval

- We'll use the cloud installation of jupyterhub
- https://www.snap4city.org/650

Not All The Device in Snap4City are public...

for some you'll need an access token to the private **IoT Device** of that authenticated user ¶

so let's get the username and password

[1]:
```
### in the config.py file that i've created are stored the user and password for the snap4city authentication
# snap4cityauth = dict(
#     user = 'user name of snap4city',
#     psw = 'the password of the user',
#     clid= '<client id depending on the App kind>' has to be obtained from Snap4City organization by sending an email to snap4city@disit.org.
# )
import config
utente = config.snap4cityauth['user']
password = config.snap4cityauth['psw']
client_id = config.snap4cityauth['clid']
```

Private Device Data Retrieval

next let's get the auth token ¶

2]:
```
import requests
import json
url = "https://www.snap4city.org/auth/realms/master/protocol/openid-connect/token/"
data = {"client_id": client_id,"grant_type":"password","username":utente,"password":password}
r=requests.post(url, data)
print(r.status_code, r.reason)
responseToken=json.loads(r.text)
refreshToken = responseToken["refresh_token"]
print("access_token : {}... expires in {}s, token_type: {}".format(responseToken['access_token'][:20],responseToken['expires_in'],responseToken['token_type'] ))

#to update the token using the refresh_token
url = "https://www.snap4city.org/auth/realms/master/protocol/openid-connect/token/"
data = {"client_id": client_id,"grant_type":"refresh_token","scope":"openid profile","refresh_token":refreshToken}
r=requests.post(url, data)
print("updating token using the refresh token ",r.status_code, r.reason)
responseToken=json.loads(r.text)
```

```
200 OK
access_token : eyJhbGciOiJSUzI1NiIs... expires in 1500s, token_type: bearer
updating token using the refresh token  200 OK
```

Private Device Data Retrieval

so now you can access the private iot device data...

```
]: auth_token=responseToken['access_token']
hed = {'Authorization': 'Bearer ' + auth_token}

url = "https://www.snap4city.org/superservicemap/api/v1?serviceUri=http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/118907.682_485819.390-Plastic&accessToke

response = requests.get(url, headers=hed)
if response.status_code == 200: # ok
    print(json.loads(response.text))
```

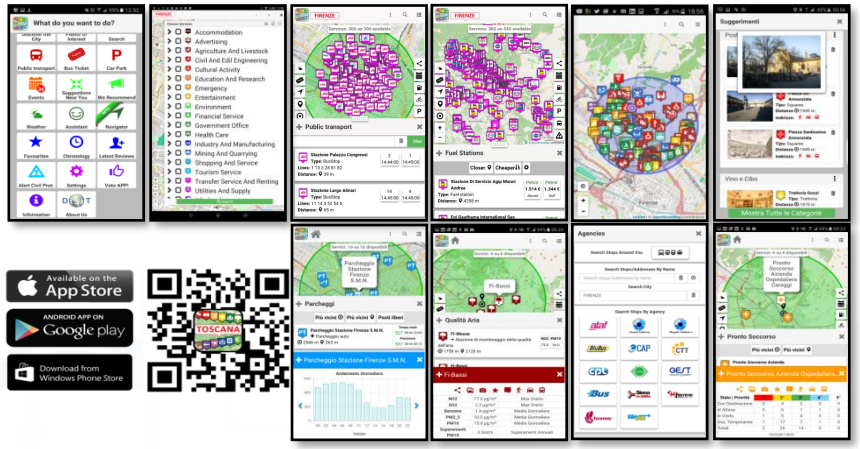

```
{'Service': {'features': [{'geometry': {'coordinates': [4.857379, 52.359085], 'type': 'Point'}, 'properties': {'address': '', 'avgStars': 0, 'brokerName': 'orionUNIF
I', 'cap': '', 'city': '', 'civic': '', 'comments': [], 'description': 'Plastic', 'email': '', 'fax': '', 'format': 'json', 'frequencySec': '600', 'isMobile': '', 'li
nkDBpedia': [], 'macaddress': '', 'maintenanceUrl': '', 'maxCapacity': '5', 'minCapacity': '', 'model': 'AmsterdamPlasticContainer', 'multimedia': '', 'name': '11890
7.682_485819.390-Plastic', 'nature': 'Environment', 'organization': 'DISIT', 'ownership': '', 'phone': '', 'photoOrigs': [], 'photoThumbs': [], 'photos': [], 'produce
r': 'Amsterdam city', 'protocol': 'ngsi', 'province': '', 'realtimeAttributes': {'dateObserved': {'attr_type': 'DeviceAttribute', 'data_type': 'string', 'different_va
lues': '0', 'value_bounds': 'unspecified', 'value_refresh_rate': '300', 'value_type': 'timestamp', 'value_unit': 'timestamp'}, 'weight': {'attr_type': 'DeviceAttribut
e', 'data_type': 'float', 'different_values': '0', 'value_bounds': 'unspecified', 'value_refresh_rate': '300', 'value_type': 'weight', 'value_unit': 'Kg'}}, 'serviceT
ype': 'Environment_Waste_container', 'serviceUri': 'http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/118907.682_485819.390-Plastic', 'starsCount': 0, 'subna
ture': 'Waste_container', 'typeLabel': 'Waste container', 'website': '', 'wktGeometry': ''}, 'type': 'Feature'}], 'type': 'FeatureCollection'}, 'realtime': {'head':
{'vars': ['measuredTime', 'dateObserved', 'weight']}, 'results': {'bindings': [{'dateObserved': {'value': '2022-01-14T09:52:09.000Z'}, 'measuredTime': {'value': '2022
-01-14T10:52:09.000+01:00'}, 'weight': {'value': '120'}}]}}}
```

TOP

FORGING & MANAGING FLEXIBLE MOBILE APPS, Web Apps and MicroApplications

Web and Mobile App Developers, to generate

Mobile Apps



Web App HTML5

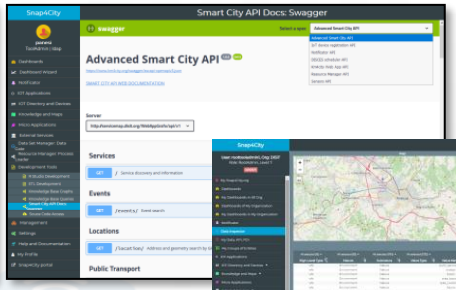
Embed into Web pages

City User

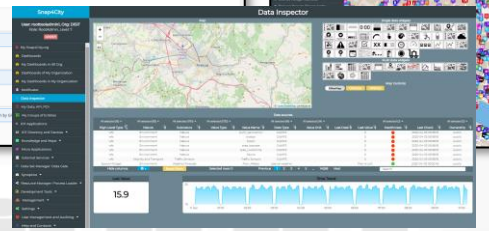
Advanced Smart City API

Mobile Application Monitoring Administrator

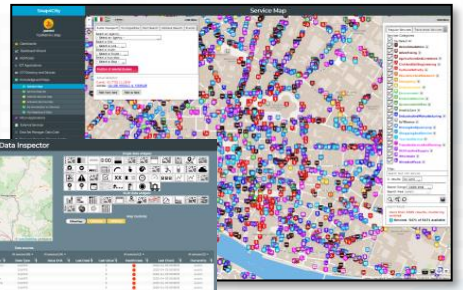
Km4City Open Source examples dev. tool kit



Swagger



DataInspector



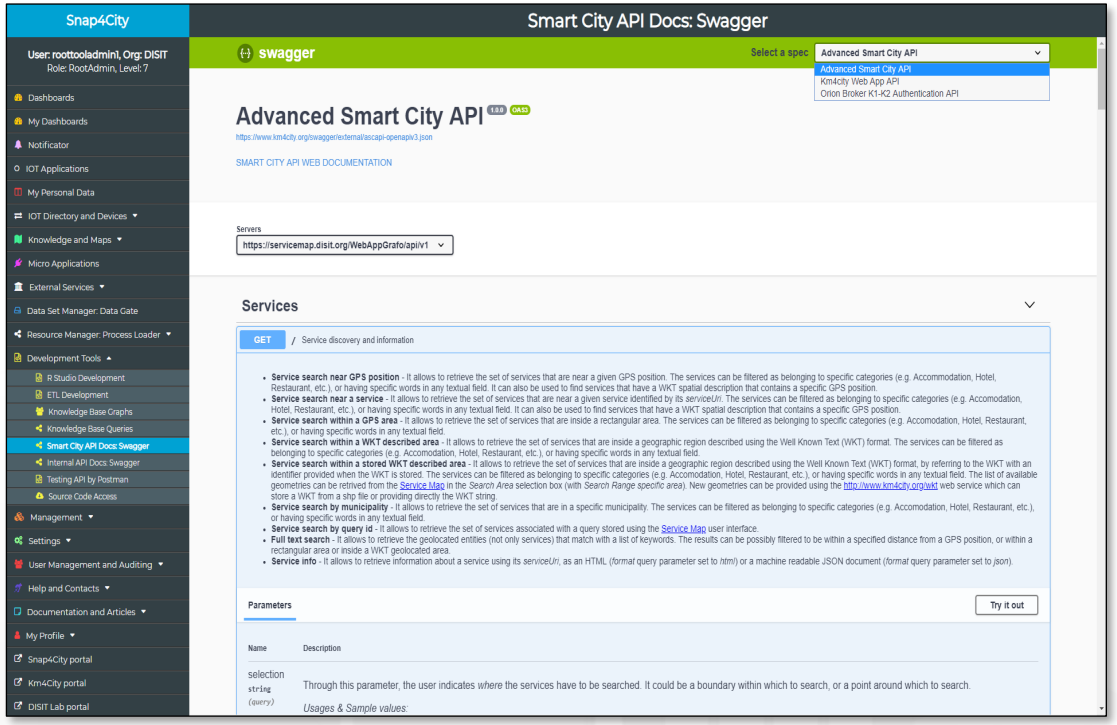
ServiceMap

Developer

Advanced SmartCity API

- Search data: by text, near, along, etc.
  - Resolving text to GPS and formal city nodes model
- Empowering city users: contributions, suggestions, forum discussions, etc.
- Events: Entertainment, critical and mobility
- Public and Private Mobility & Transport, and predictions
- POIs, Cultural and Touristic info
- Health services and predictions
- Environmental information, heatmaps; values
- Profiled Suggestions to City Users
- Traffic flow reconstruction
- Personal Assistant: PAVAL
- User Engagement: goal experiences, and assessment
- *Sharing knowledge among cities → see Knowledge base Management*

Swagger

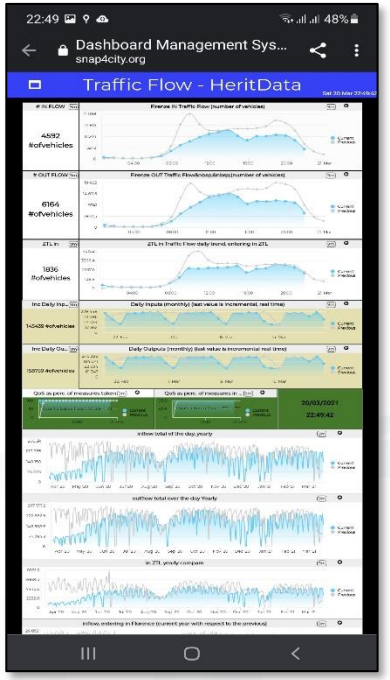


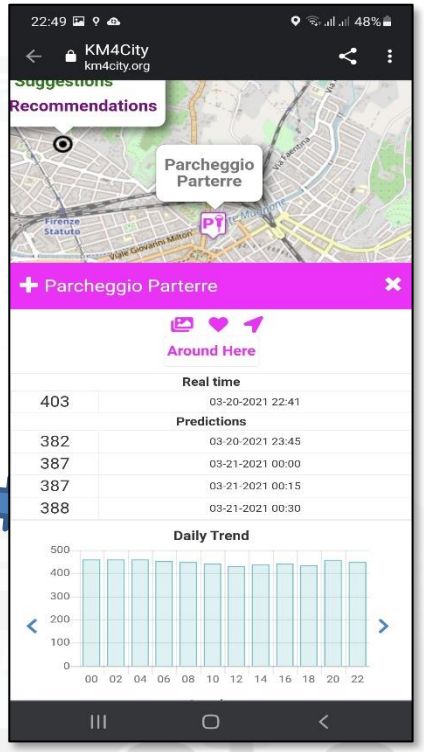
Mobile Apps exploiting our tech/Smart City API

- **Firenze Dove Cosa, Km4City**:
  - Android: https://play.google.com/store/apps/details?id=org.disit.siiMobile&hl=en&gl=US
  - Apple iOS: https://apps.apple.com/it/app/firenze-dove-cosa-km4city/id1028356115
- **Toscana dove cosa, Km4City**
  - Android: https://play.google.com/store/apps/details?id=org.disit.toscana&hl=en&gl=US
  - Apple iOS: https://apps.apple.com/it/app/toscana-dove-cosa-km4city/id1064554200
- **Antwerp in a Snap**
  - Android: https://play.google.com/store/apps/details?id=org.disit.snap4city.mobileApp.antwerp&hl=en&gl=US
  - Apple iOS: https://apps.apple.com/it/app/antwerp-in-a-snap/id1467737363
- **Helsinki in a Snap**
  - Android: https://play.google.com/store/apps/details?id=org.disit.snap4city.mobileApp.helsinki&hl=en&gl=US
  - Apple iOS: https://apps.apple.com/it/app/helsinki-in-a-snap/id1466970280
- **Tuscany in a Snap**
  - Android: https://play.google.com/store/apps/details?id=org.disit.snap4city.mobileApp.tuscany&hl=en&gl=US
  - Apple iOS: https://apps.apple.com/us/app/toscana-in-a-snap/id1471094480
- **Snap4Pisa: Pisa in a Snap:**
  - Android: https://snap4city.aedit.it/drupal//sites/default/files/snap.apk
- **Weee Life Mobile App:**
  - Android: https://play.google.com/store/apps/details?id=org.disit.lifeweee&hl=en&gl=US
  - Apple iOS: https://apps.apple.com/it/app/life-weee/id1470224854
- **Comune Facile Android:**
  - Android: https://www.e015.regione.lombardia.it/site/app-detail?id=131
  - Comune Facile IOS: https://www.e015.regione.lombardia.it/site/app-detail?id=130
- **POS by EPSON:** https://www.e015.regione.lombardia.it/site/app-detail?id=58
- **SnapBot:** https://www.snap4city.org/684

Florence Dashboard

https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MzAwNA==



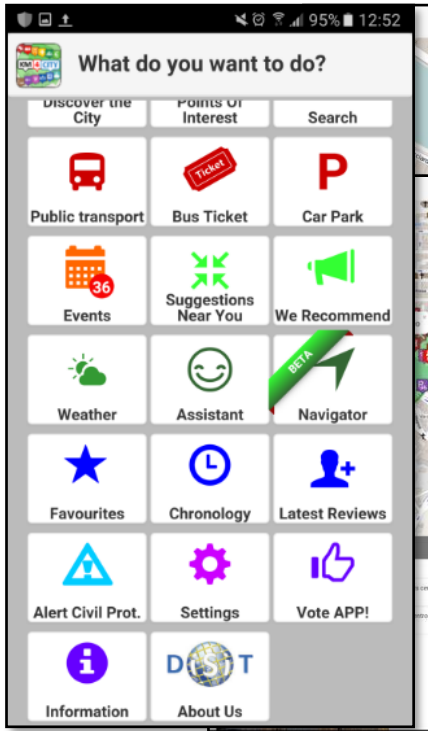


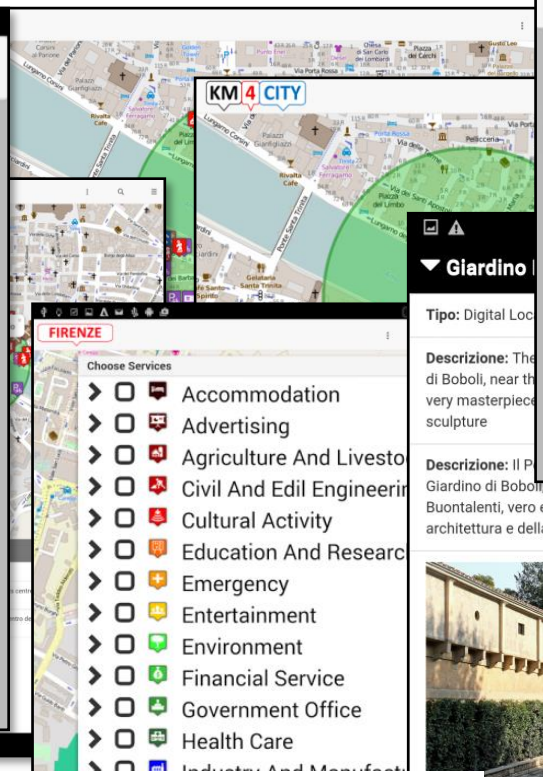
Web and Mobile App with Open Development Kit

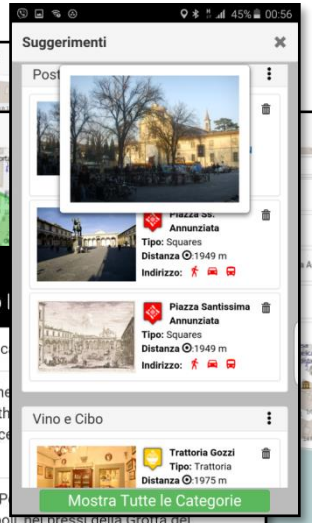
UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB

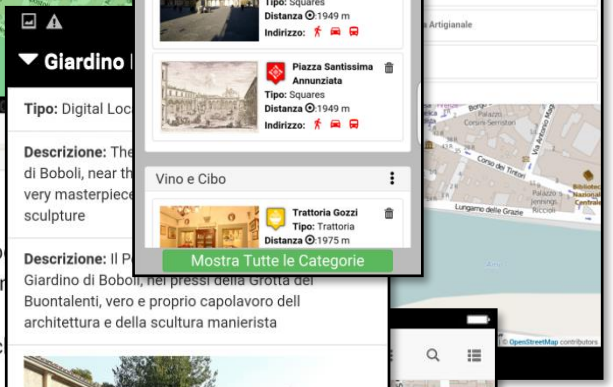
Mobile Apps

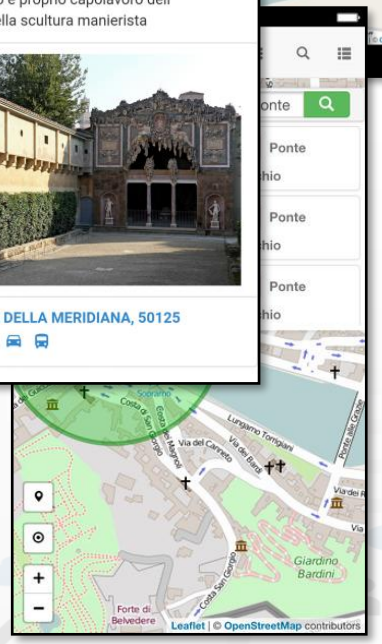
SNAP4CITY | KM 4 CITY

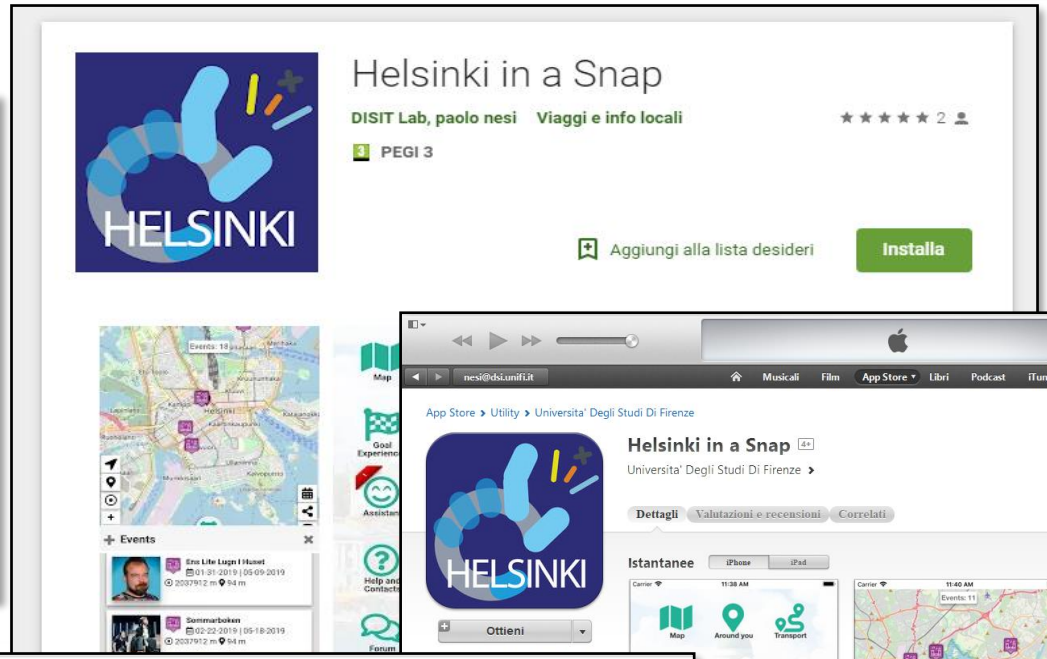












DISPONIBILE SU Google play

Scarica da App Store

Scarica da Windows Store

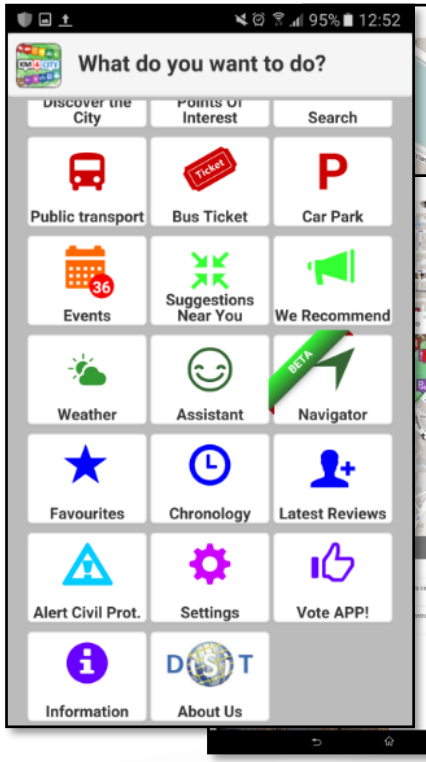
SNAP4CITY

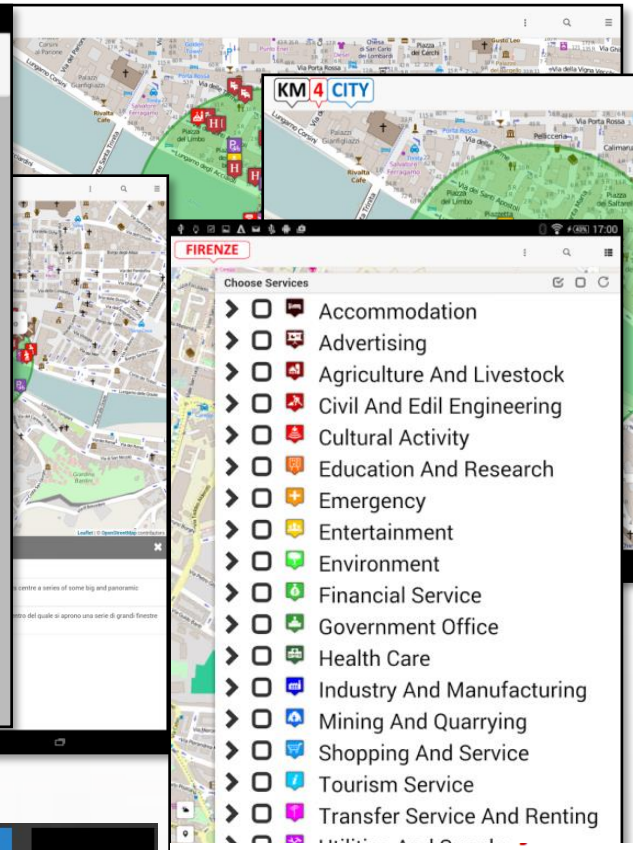
UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB

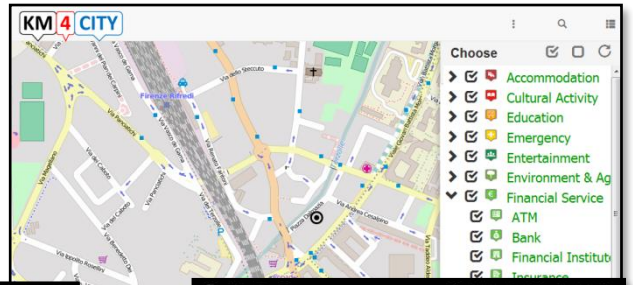
Km4CityMobile App

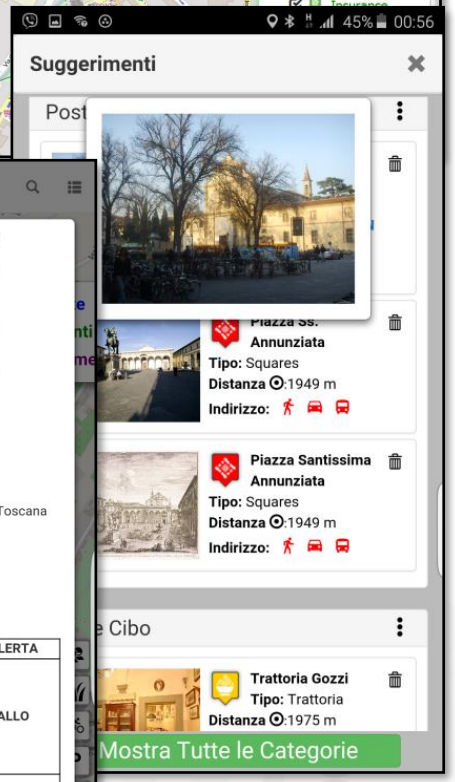
SNAP4CITY

http://www.km4city.org









DISPONIBILE SU Google play

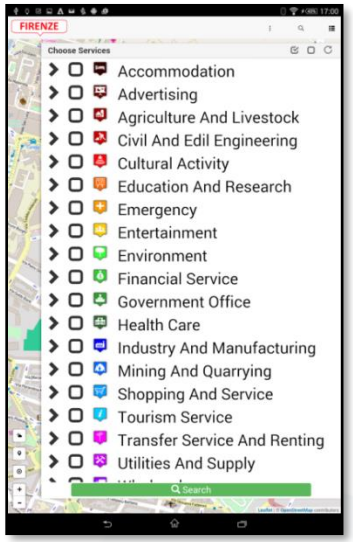
Scarica da App Store

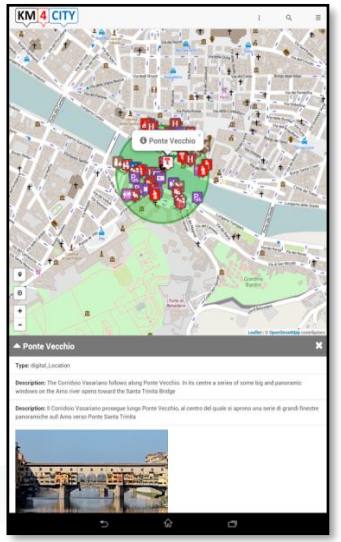
Scarica da Windows Store

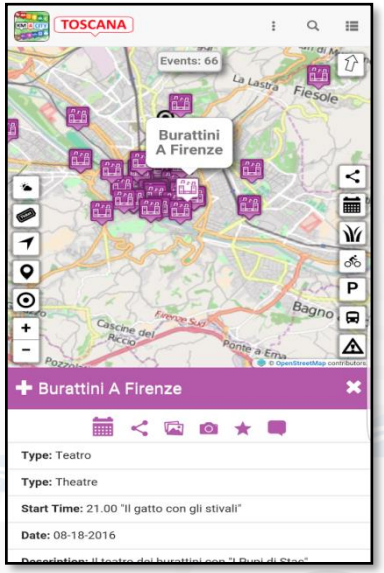
Sii-Mobility

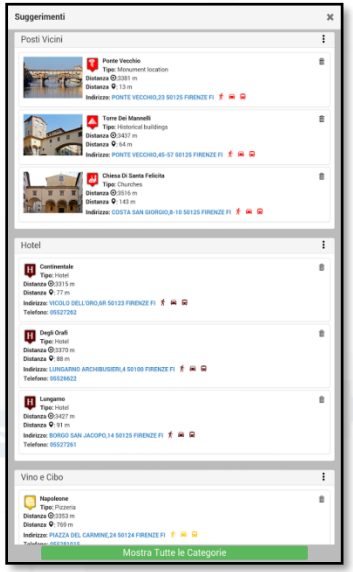
Km4City APP, features

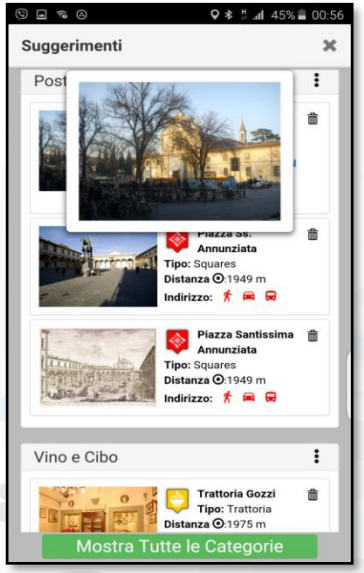
- **5 languages:** IT, EN, SP, DE, FR
- **Profiles** city users: citizens, commuter, student, tourist, operator, etc.
- **Profiled Menu** per POI
  - adaptive
- **Main Menu**: dynamic, and personalized
- **Search Text**
- **Search per POI**
  - Near to you, near to a point, a line, ...
- **Other search**
  - Close to you, events green areas, public transport, tickets , Cycling, parking, …
  - Etc.
- **POI**
  - Preferred, Social icon
  - Ranking, Comments, Images

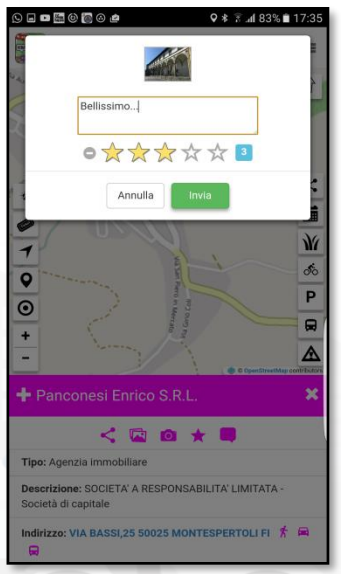












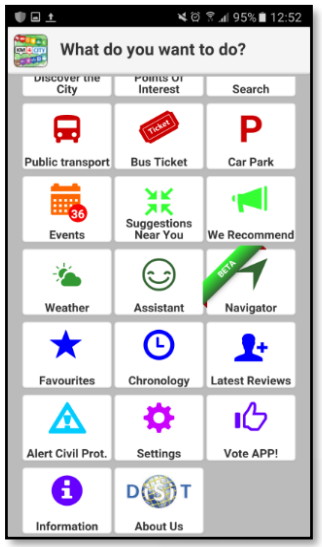
Available on the App Store

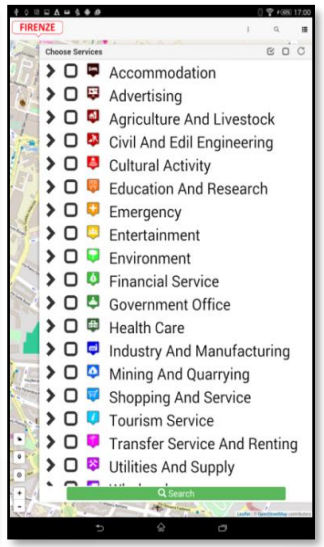
ANDROID APP ON Google play

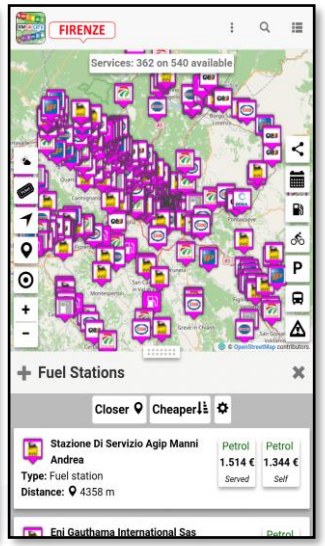
Download from Windows Phone Store

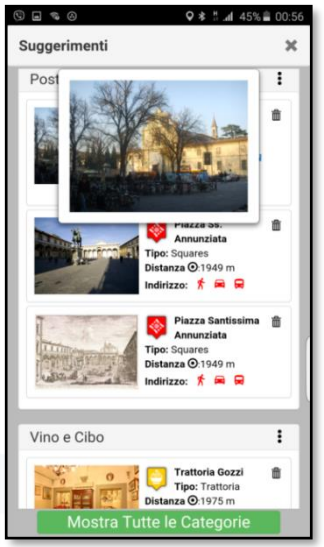
Km4City APP

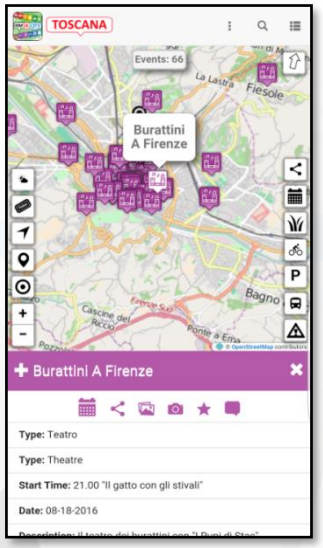
- Smart Parking, in Tuscany
- Smart First Aid in Tuscany
- Smart Public Transportation in Tuscany
- Smart Fuel pricing in Tuscany
- Bike Sharing in Pisa
- Weather condition in Tuscany
- Environmental data
- Pollution and Pollination in Tuscany
- Traffic Sensors in Tuscany
- Smart Routing in Tuscany
- Smart Transportation in Florence
  - Events, traffic, …
- Entertainment Events in Florence

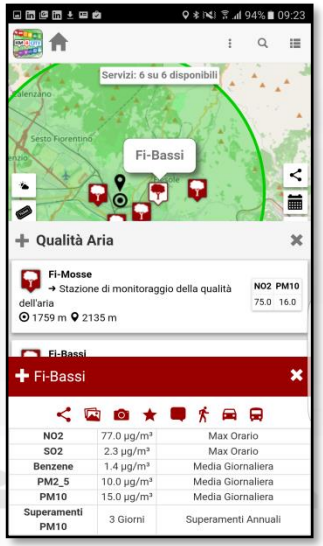


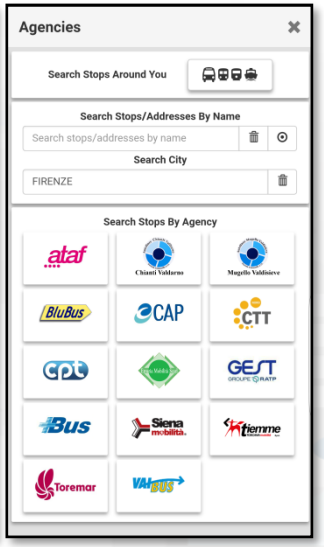












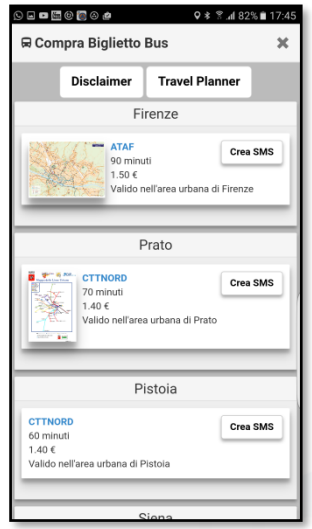
Available on the App Store

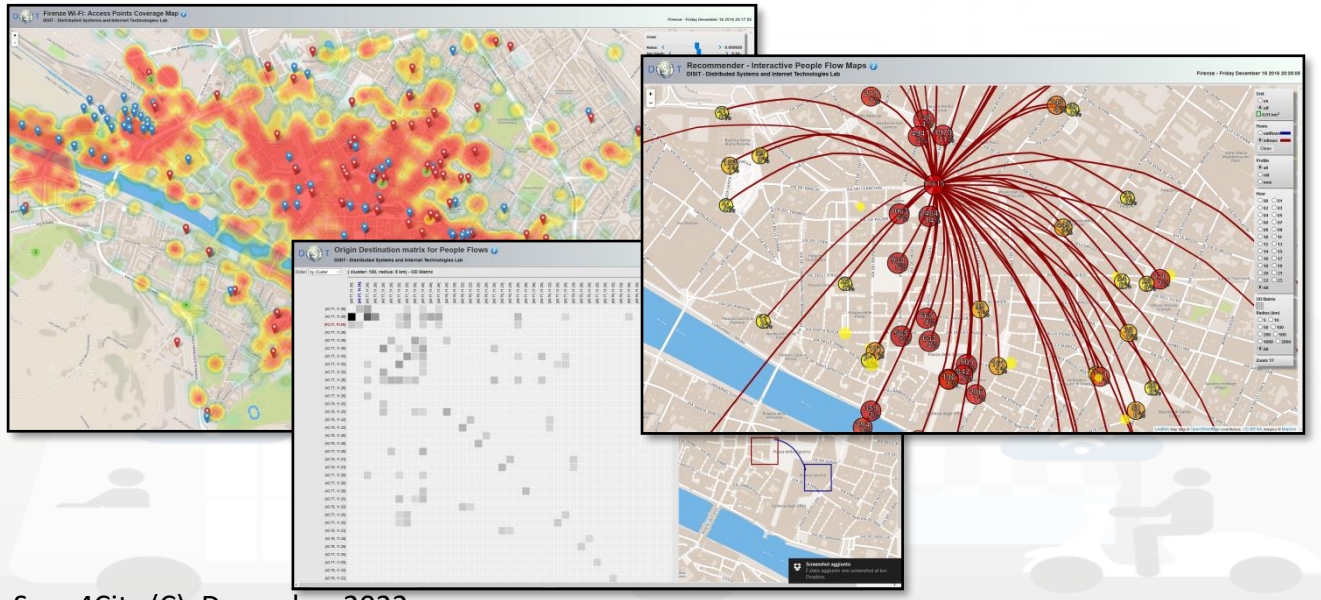
ANDROID APP ON Google play

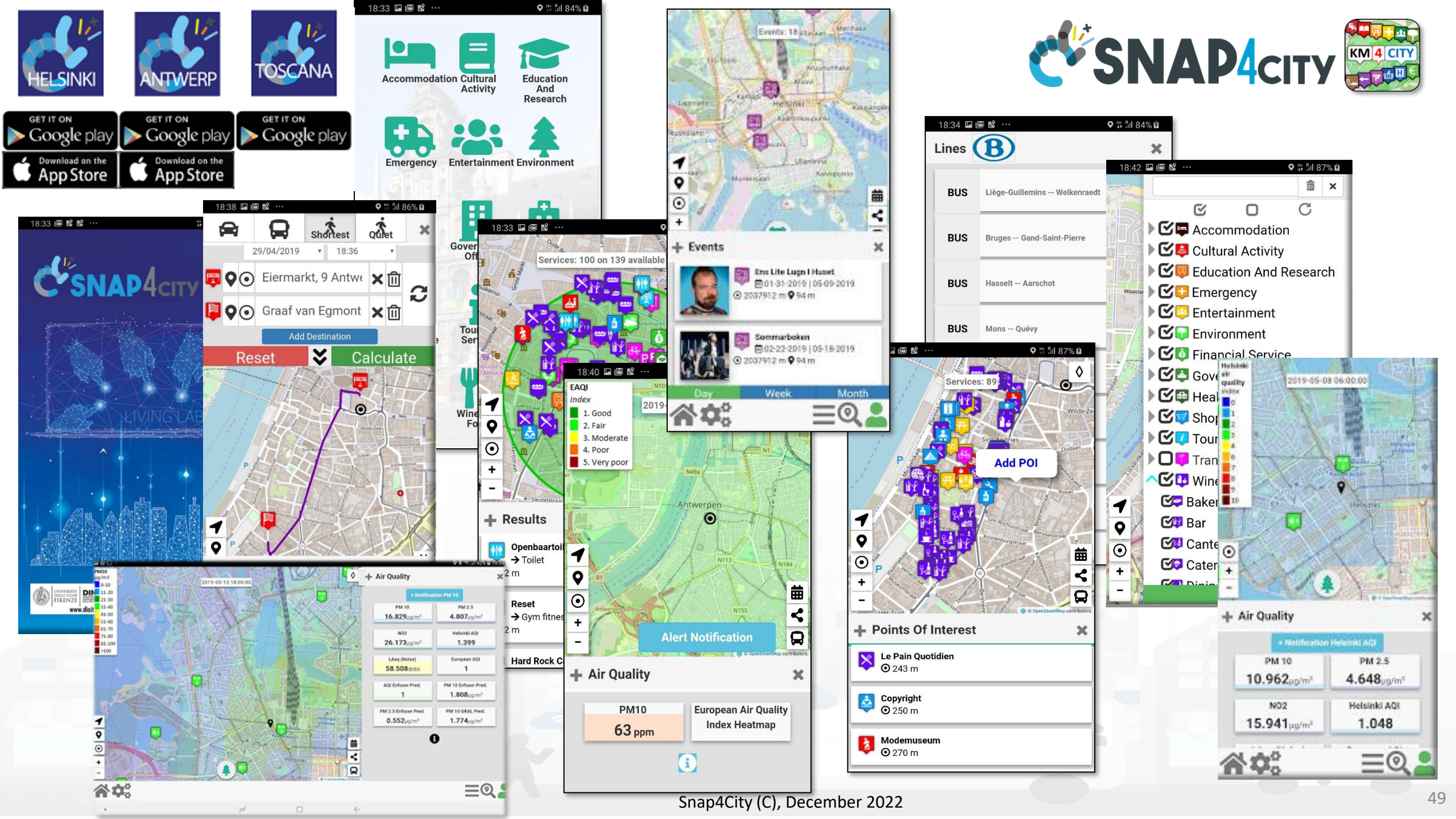
Download from Windows Phone Store

Km4City APP, features 3/3

- **Navigation 3D**
- **Ticketing for busses**
- **App used are tool for city assessment**
  - Wi-Fi status
  - iBeacon status
  - User behavior analysis
    - GPS movements kinds
    - OD matrix
    - International flows






HELSINKI
ANTWERP
TOSCANA
GET IT ON Google play
Download on the App Store
SNAP4CITY
KM 4 CITY
Accommodation
Cultural Activity
Education And Research
Emergency
Entertainment
Environment
Shortest
Quiet
Add Destination
Reset
Calculate
Events
Day
Week
Month
Services: 100 on 139 available
EAQI
Index
1. Good
2. Fair
3. Moderate
4. Poor
5. Very poor
Results
Antwerpen
Alert Notification
Air Quality
PM10
63 ppm
European Air Quality Index Heatmap
Lines
BUS
Liège-Guillemins -- Welkenraedt
Bruges -- Gand-Saint-Pierre
Hasselt -- Aarschot
Mons -- Quévy
Add POI
Points Of Interest
Le Pain Quotidien
243 m
Copyright
250 m
Modemuseum
270 m
Financial Service
Bar
Air Quality
PM 10
10.962
PM 2.5
4.648
NO2
15.941
Helsinki AQI
1.048

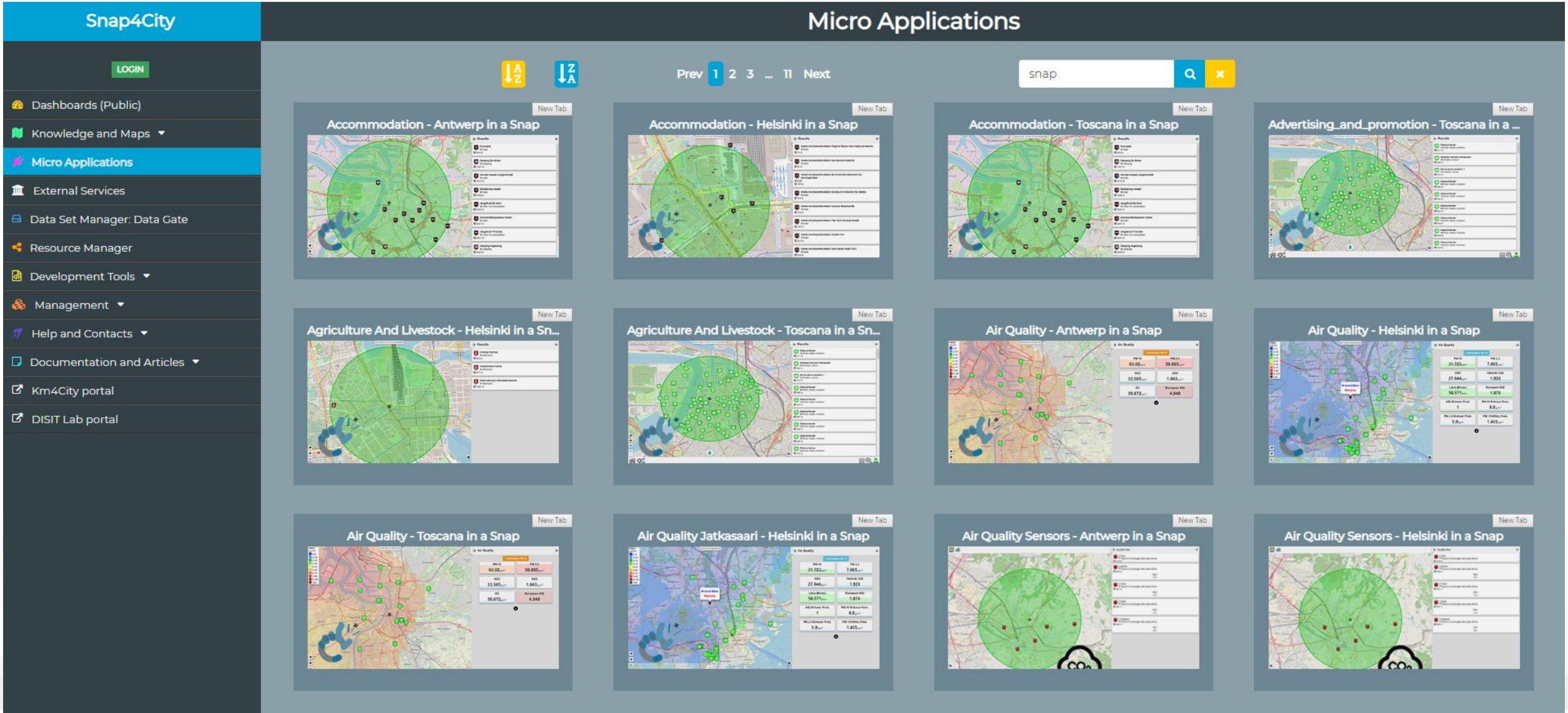
Mobile App Features

- **Discovery** POI/services
- **Search**: POI, streets, suggestions
- **Mobility and transport**: Pub/priv, routing, car position, time table, park, sharing, tickets, etc.
- **Environment and Weather**: values, sensors, heatmaps, notifications
- **Assistant, Forum, Developer Assistant**
- **Goal Experiences** (Engagement)
- **Personal** data, activities, POI, tracking, IOT App, Dashboards, etc.
- **Events**: entertainment, critical
- **Sharing** position and trajectories with friends
- **Monitoring** city and personal Dashboards
- **Personalized for Operators and Developers full control of their applications on cloud**

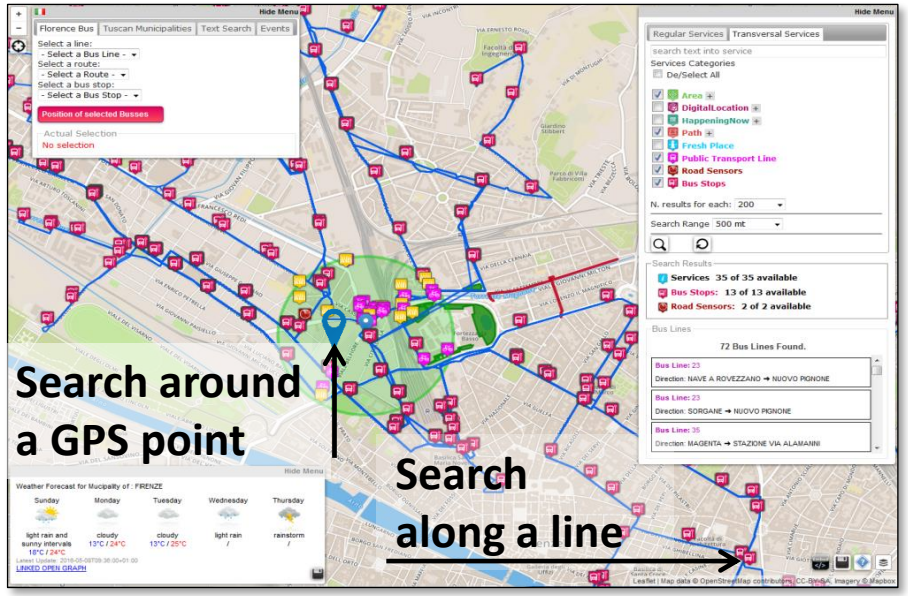
UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB

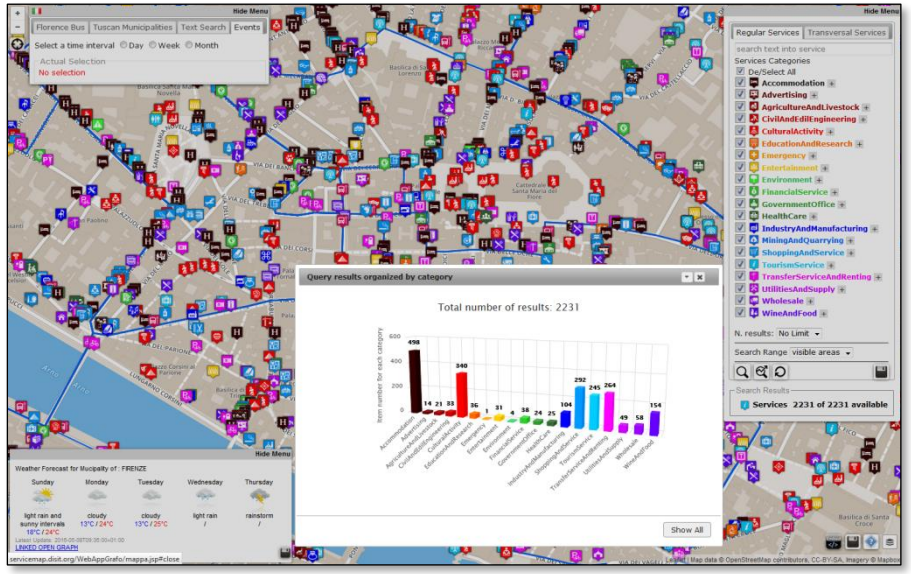
SNAP4CITY | KM4CITY

MicroApplications



ServiceMap Dev Tool (knowledge & Map tool)



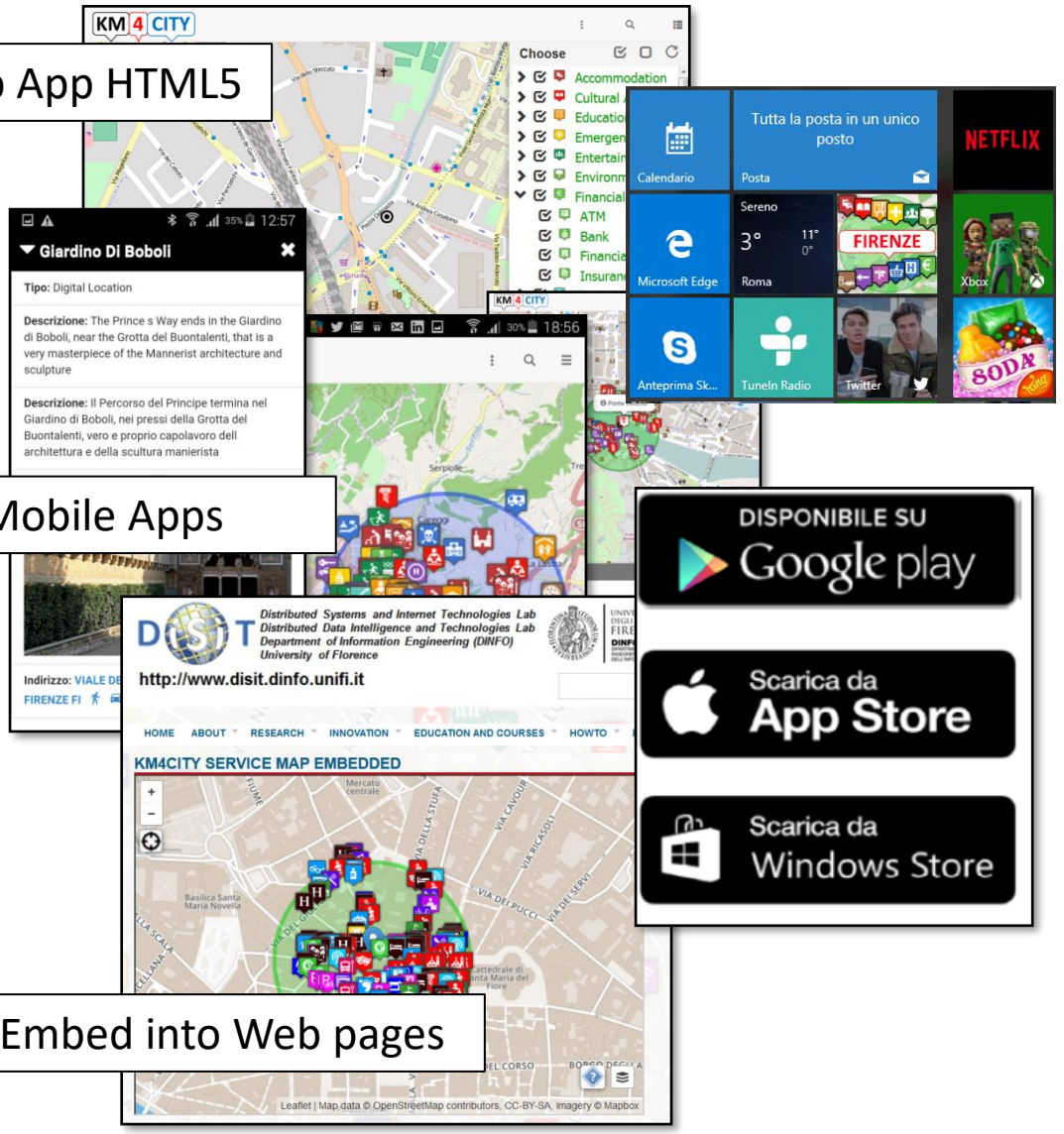


Smart City API call generation

Web App HTML5

Mobile Apps

Embed into Web pages



Advanced Smart City API

- based on Km4City engine on the back office and much more
  - Documented: https://www.disit.org/6597
- **ServiceMap tool** is used to visually generate/request:
  - REST Calls to exploit the Smart City APIs in web and mobile applications. The examples of REST calls are sent by email.
  - views which can be embedded in web pages
- **Documentation**:
  - **TC5.15 - Snap4City Smart City API Collection and overview, real time**
  - **ServiceMap and ServiceMap3D, Knowledge Model, Km4City Ontology**
  - **Knowledge Base Graphs and Queries: browsing and queries into the KB**

Understanding how City Users are using the City Services

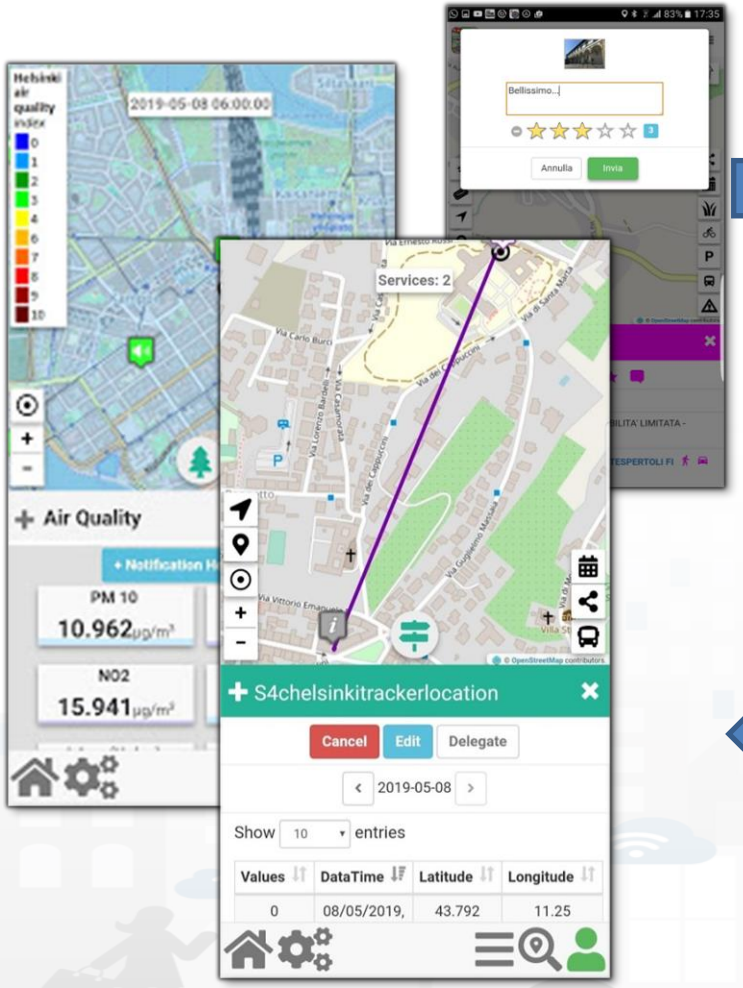
The App is a Bidirectional Device

- GPS Positions
- Selections on menus
- Views of POI
- Access to Dashboards
- searched information
- Routing
- Ranks, votes
- Comments
- Images
- Subscriptions to notifications
- ….

Produced information

- Accepted ?
- Performed ?
- …

Users



Derived information

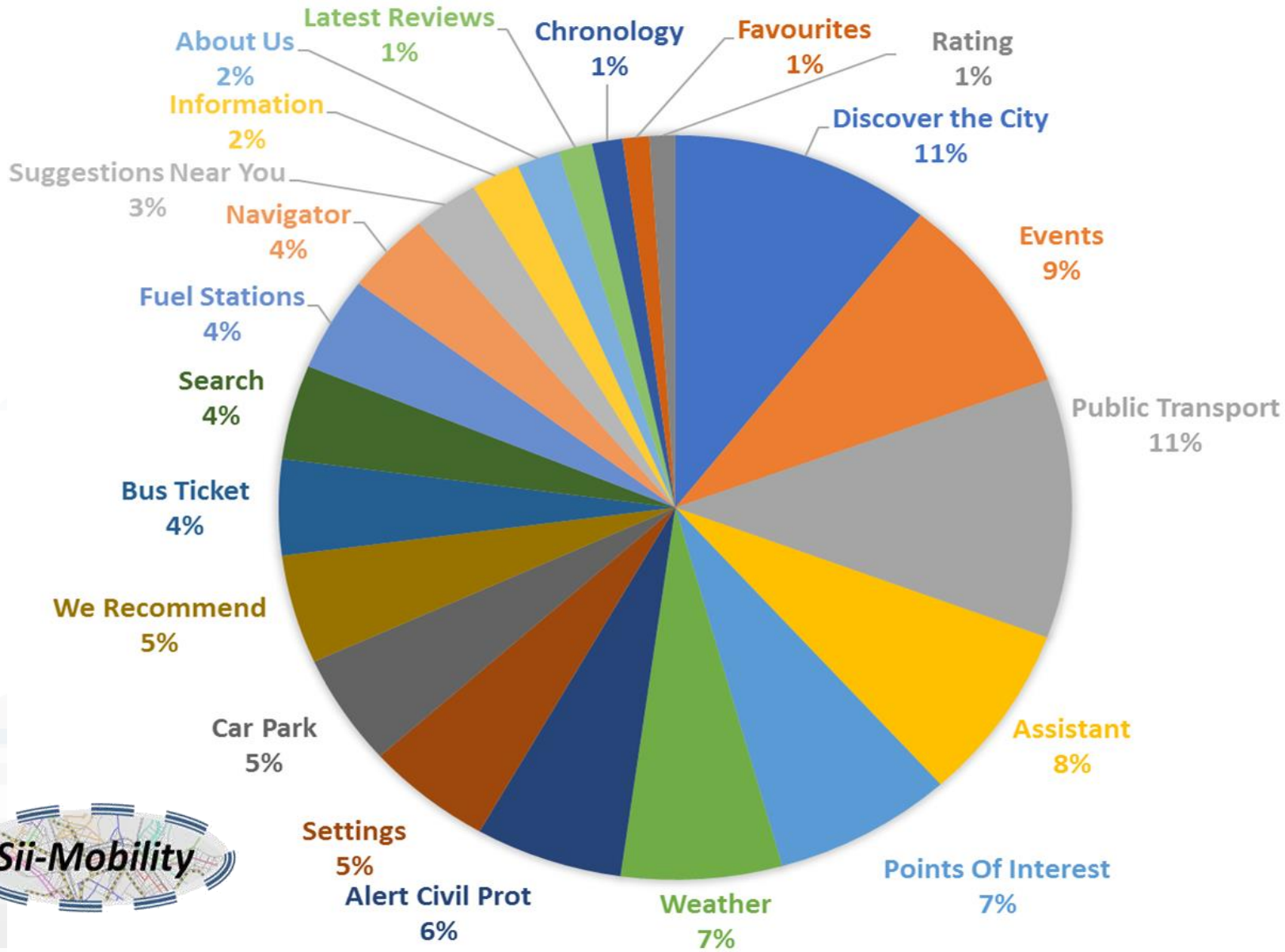
- Trajectories
- Hot Places by click and by move
- Origin destination matrices
- Most interested topics
- Most interested POI
- Delegation and relationships
- Accesses to Dashboards
- **Cumulated Scores from Actions**
- Requested information
- Routing performed
- …..

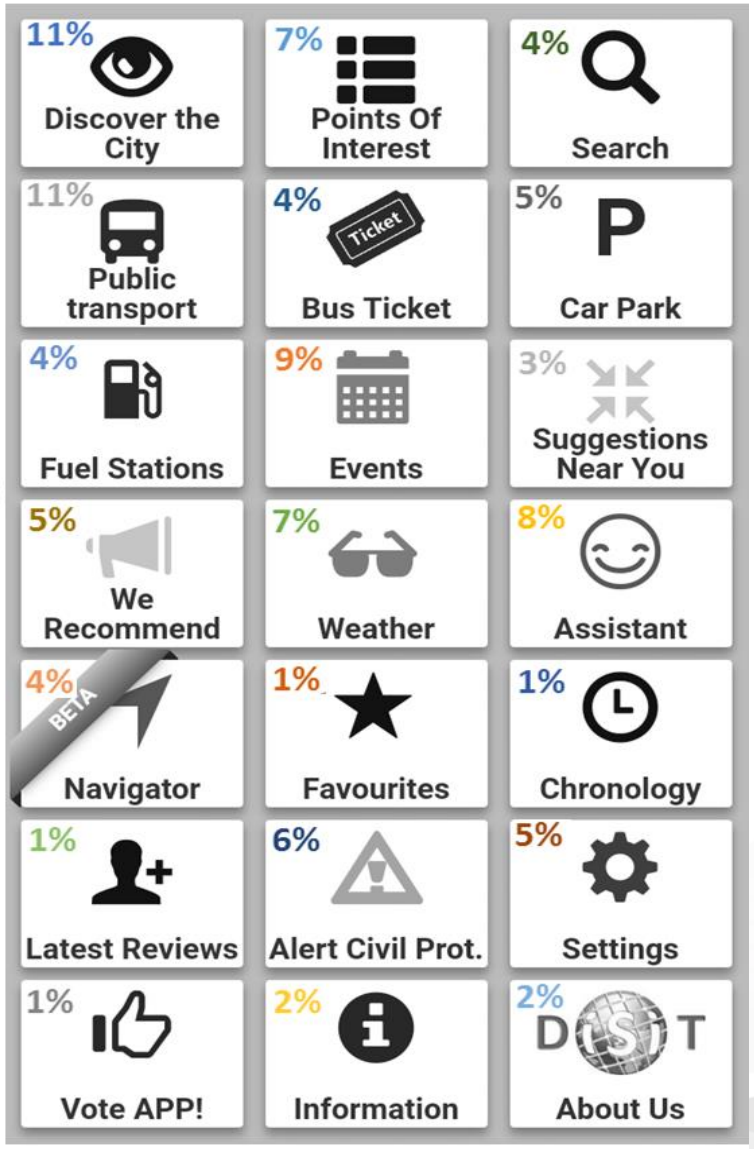
Produced information

- Suggestions
- Engagements
- Notifications
- …

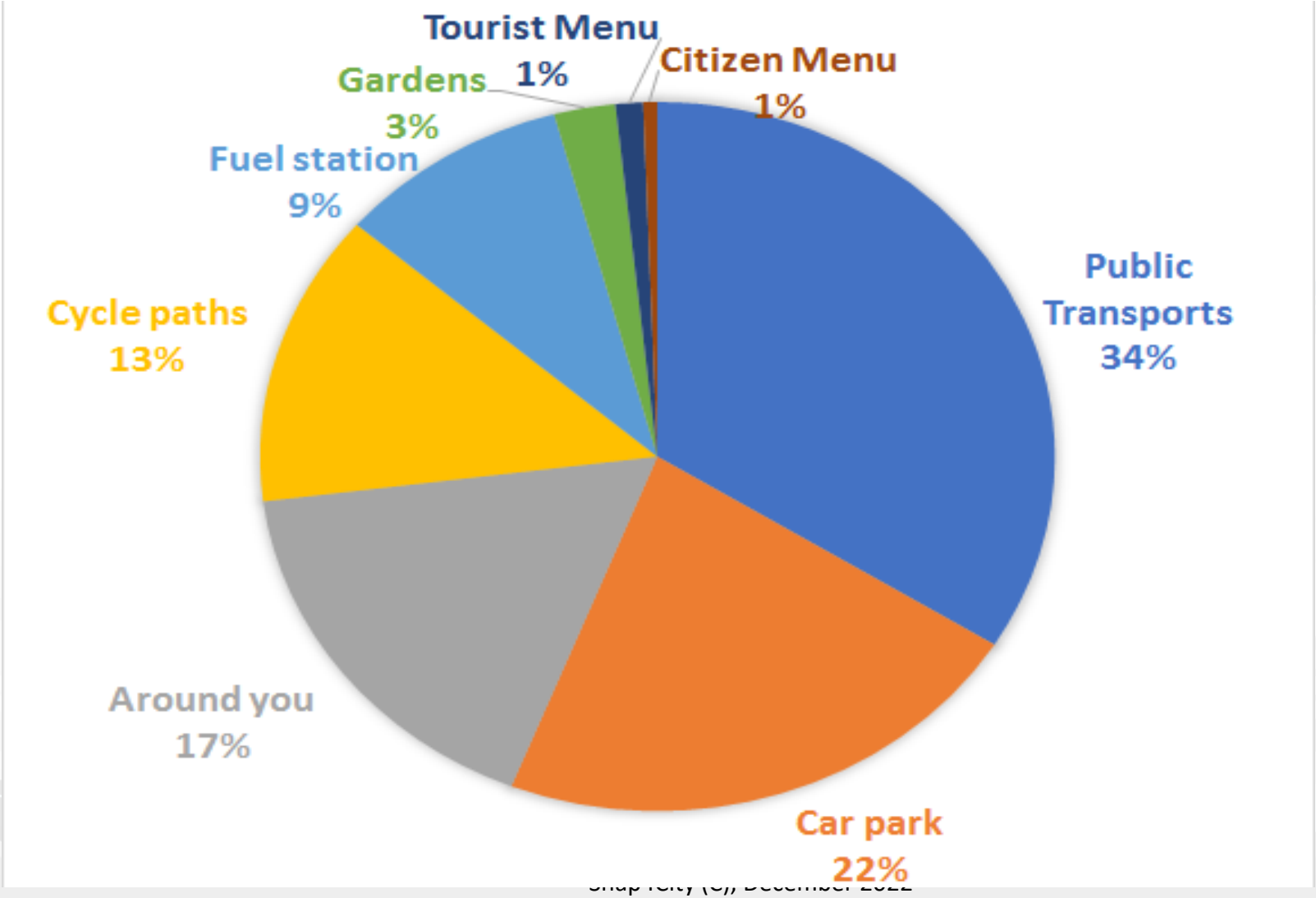
System

Users' preferences

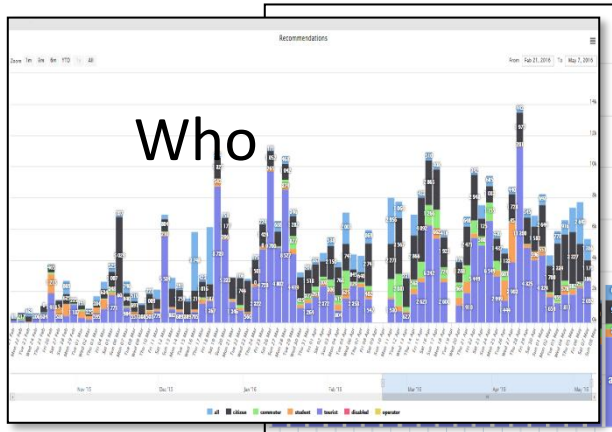


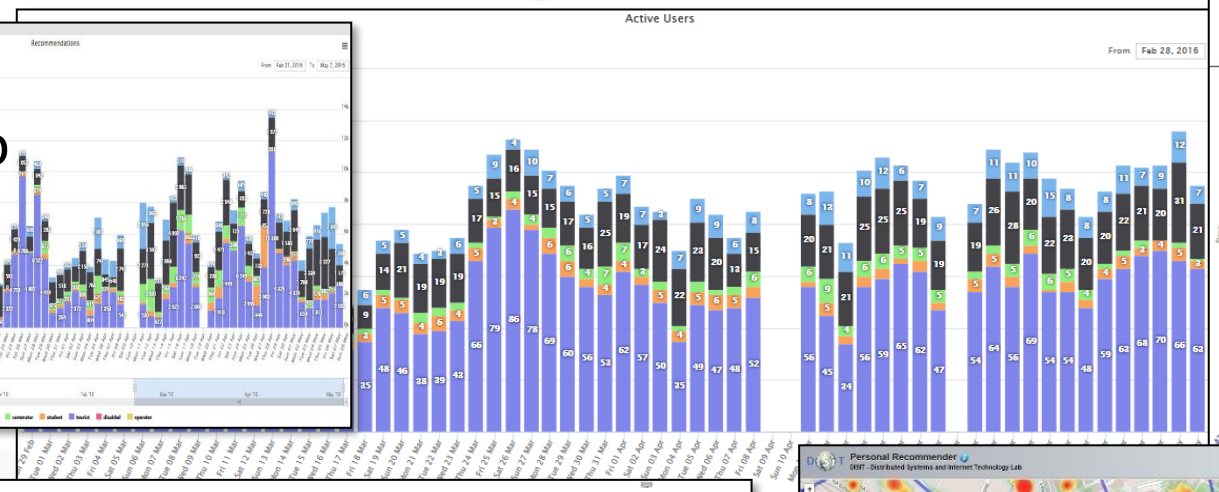


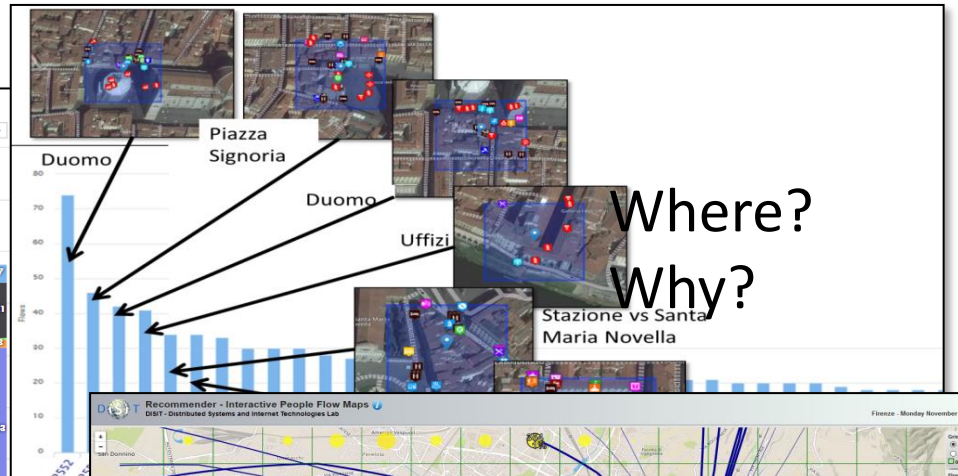
Preferred Users' Cathegories

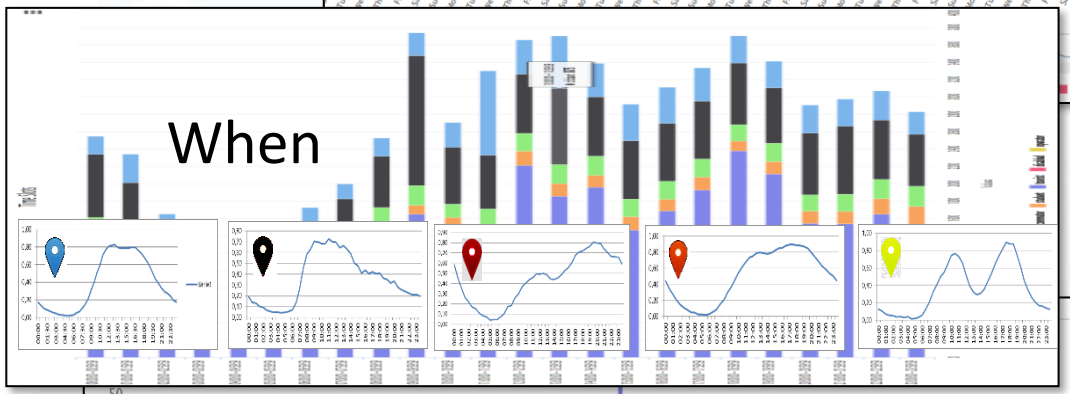


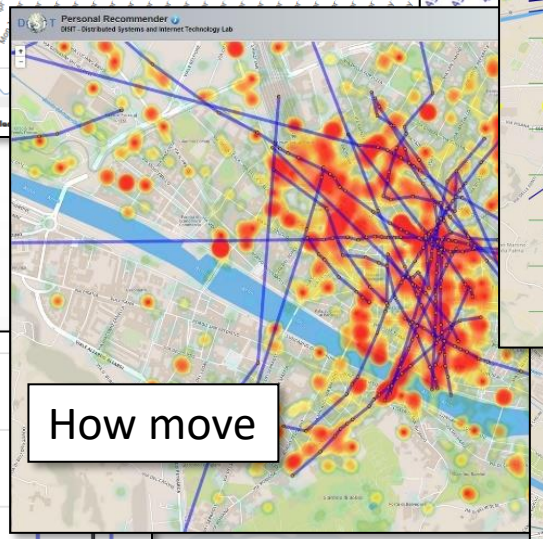
User Behavior Analyser for Collective Profiling

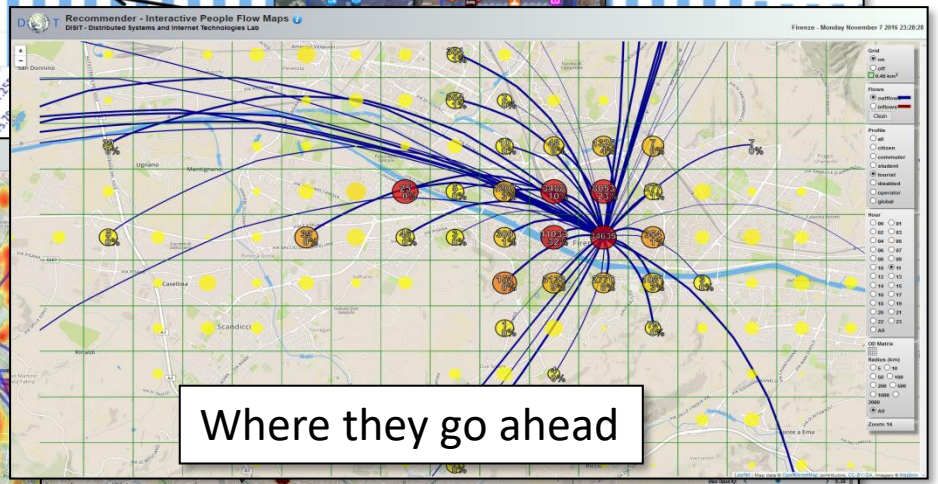


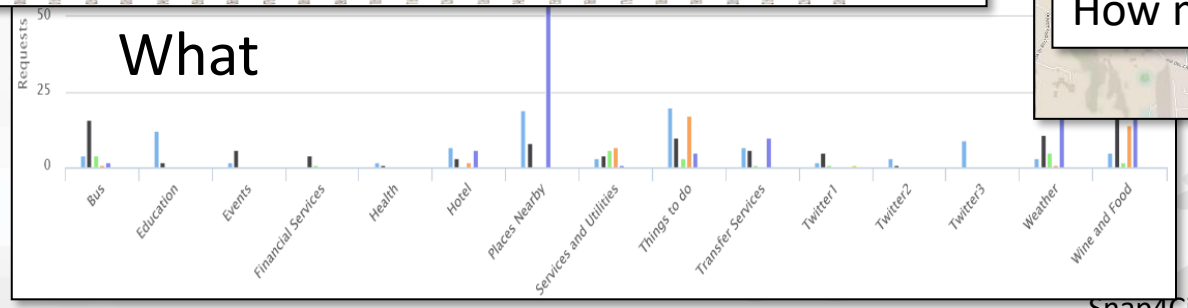


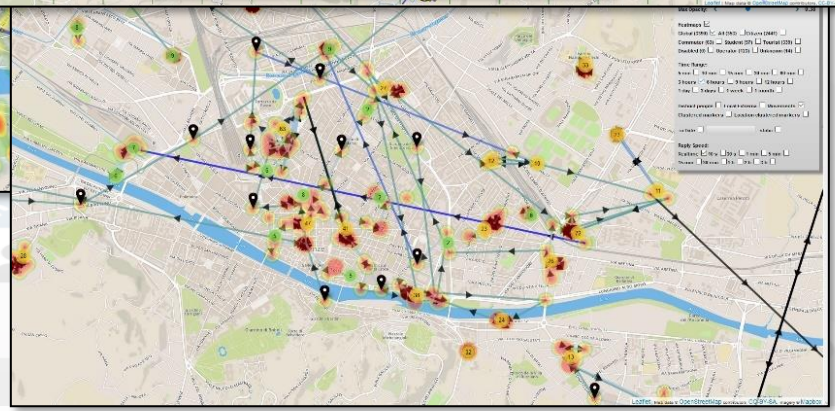




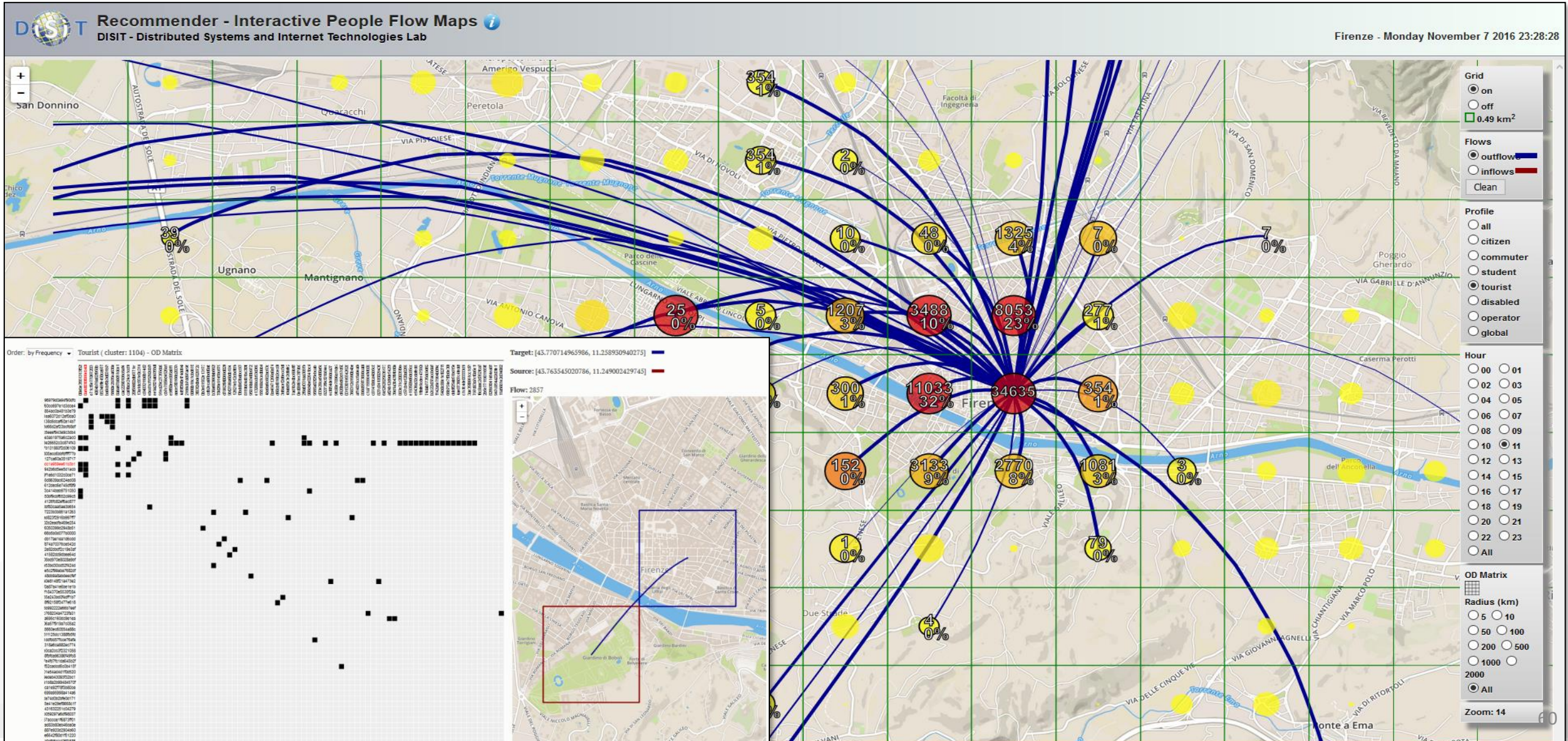




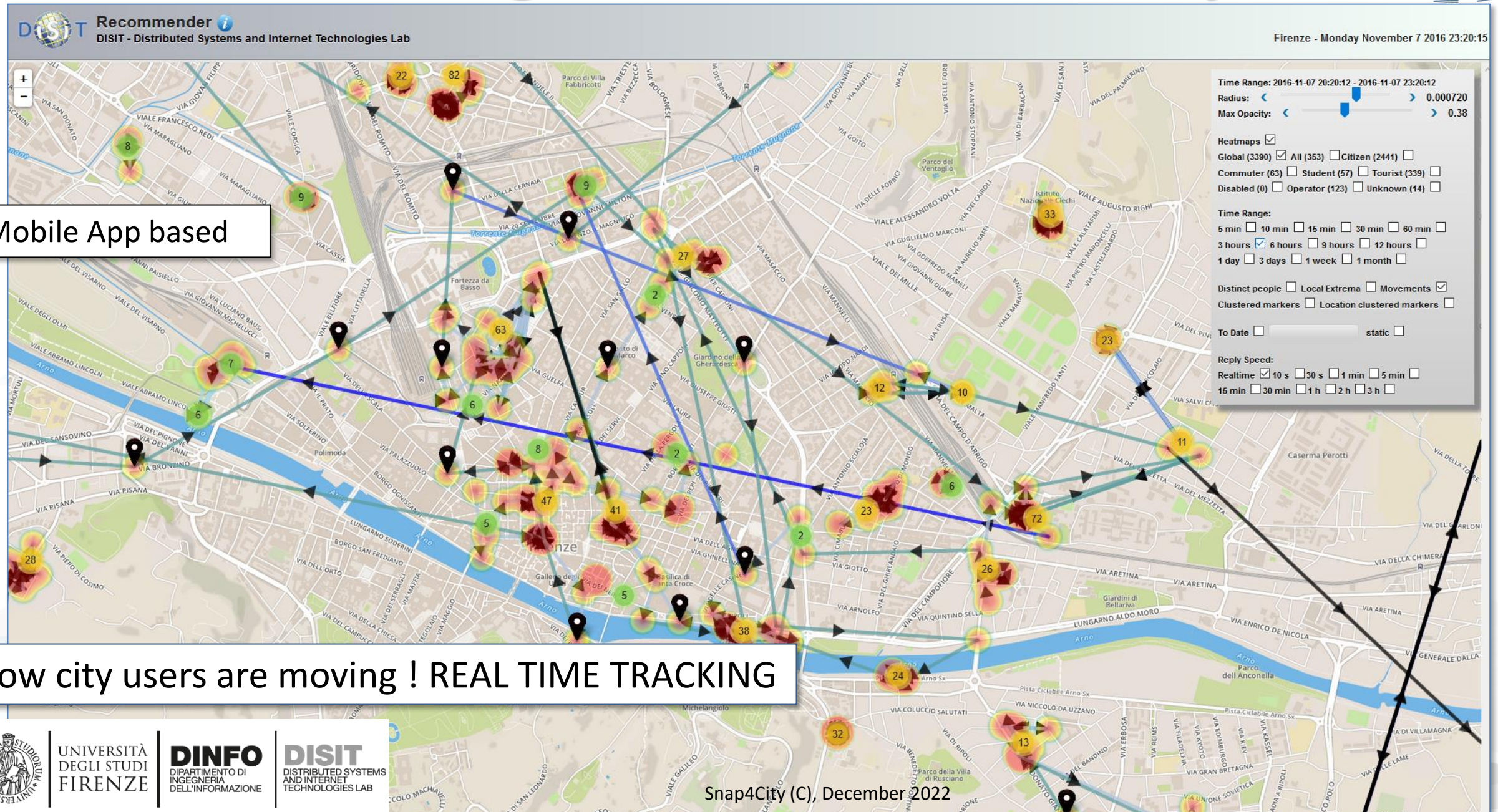




Scalable OD Matrix



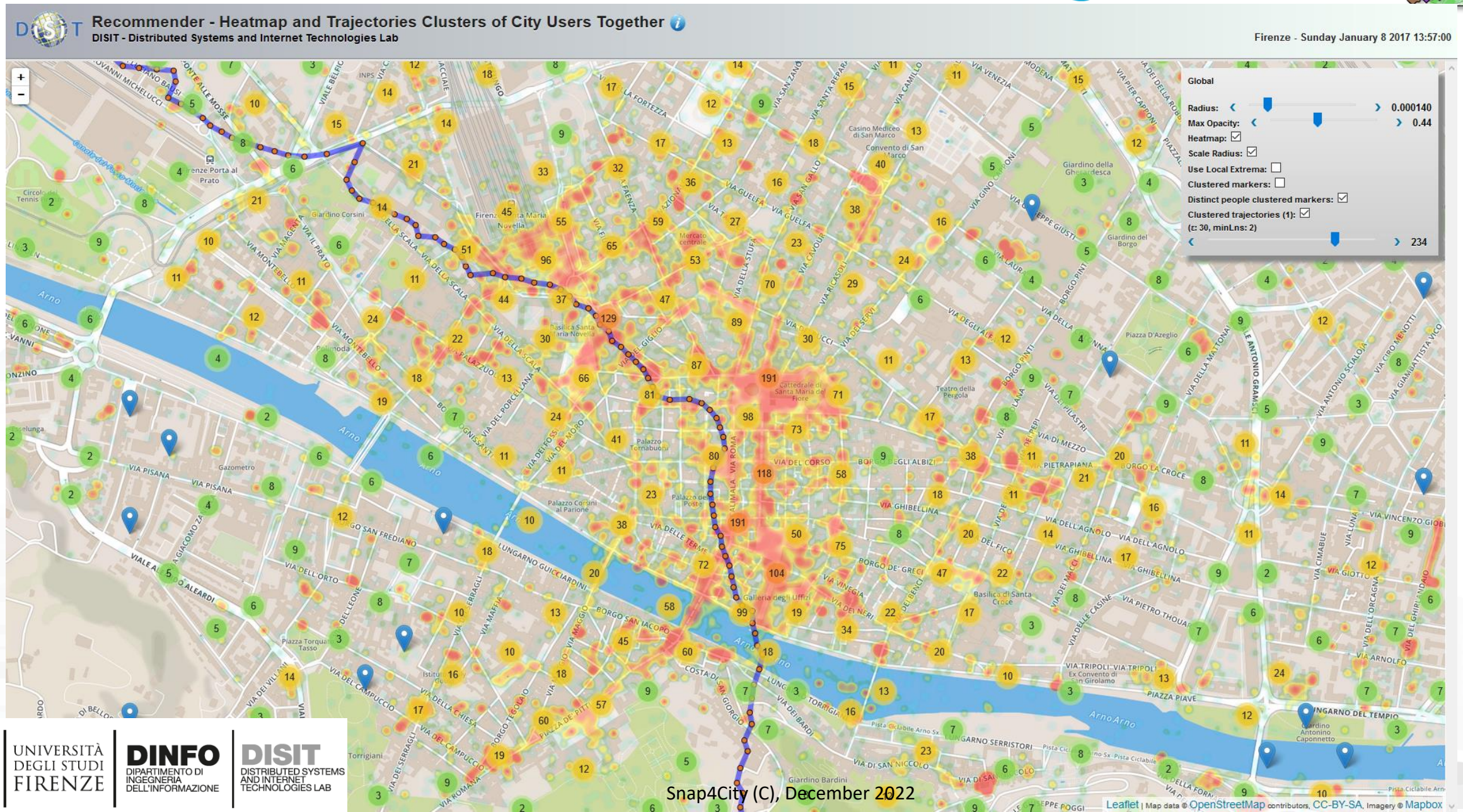
Real Time Traking: User Behaviour Analysis



Mobile App based

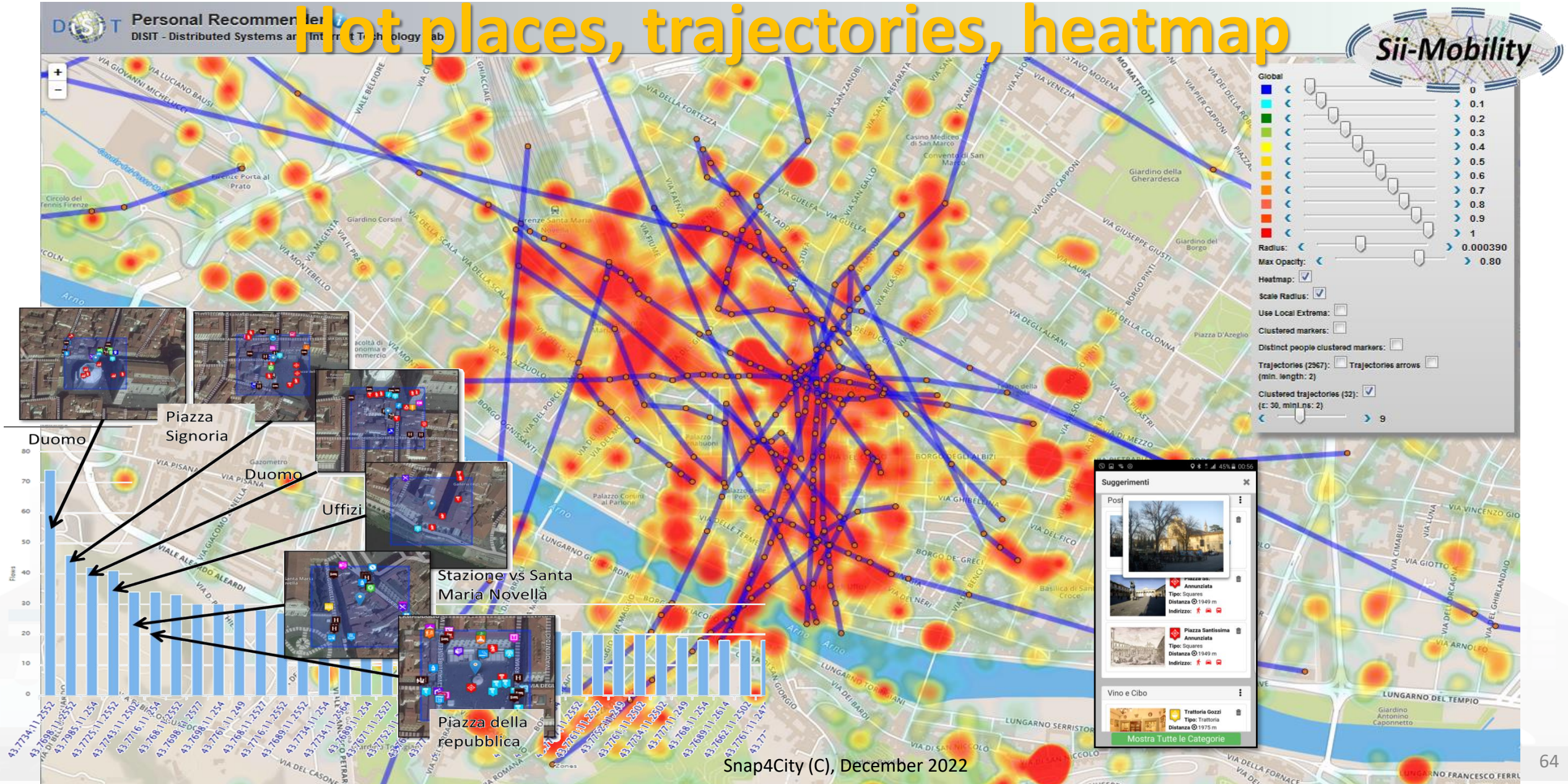
How city users are moving ! REAL TIME TRACKING

HeatMaps: Users as Sensors



User Behaviour Analyser

Hot places, trajectories, heatmap



Understanding City User Behaviour

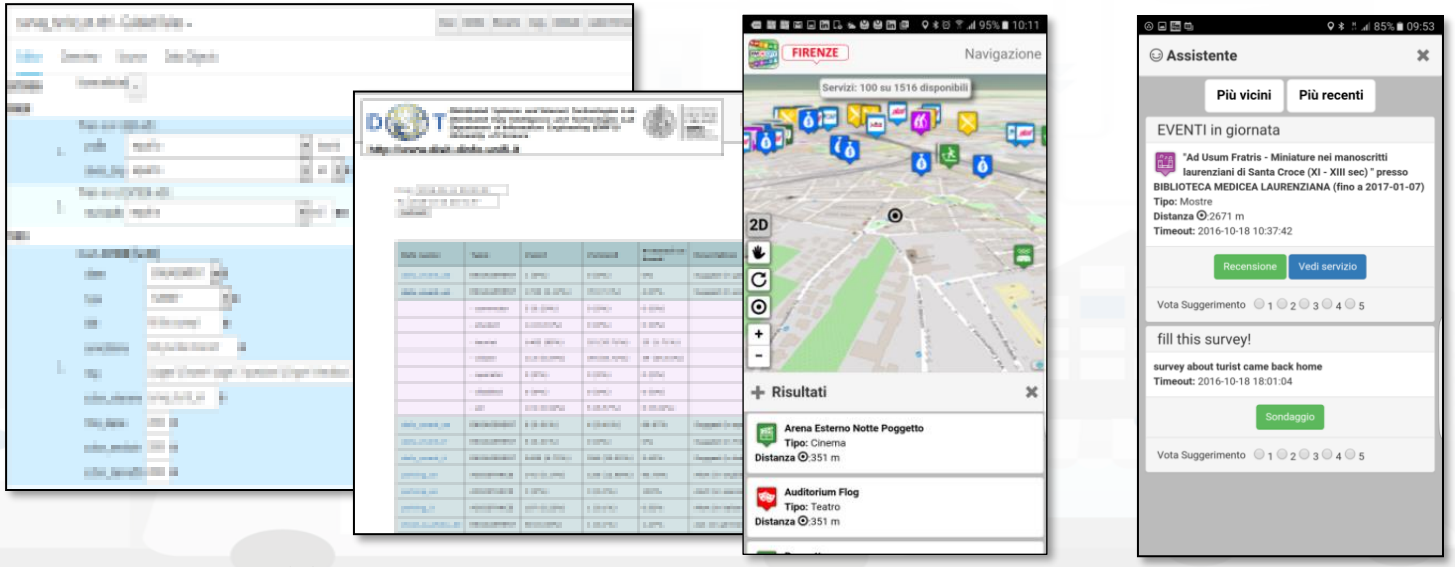
- **Mobile Applications** can send data via Advanced Smart City API to collect data about the city usage by the city users via a signed consent
  - See Mobile and Web App: Toscana in a Snap, Helsinki in a Snap, Antwerp in a Snap.
- **City User behavior analysis** includes production of:
  - suggestions, trajectories, hot places/heatmaps, etc.
  - origin destination matrices
  - data for the city user engagement
  - Etc.

https://www.snap4city.org/drupal/node/489

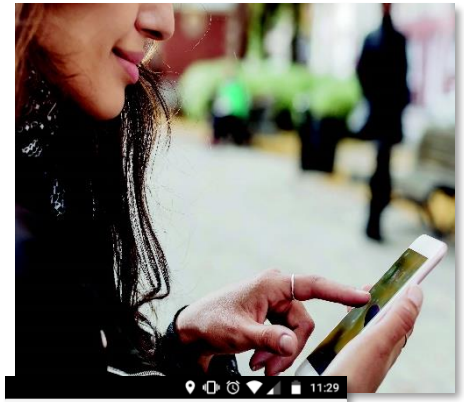
Engaging City Users Towards Virtuous Participated Attitude

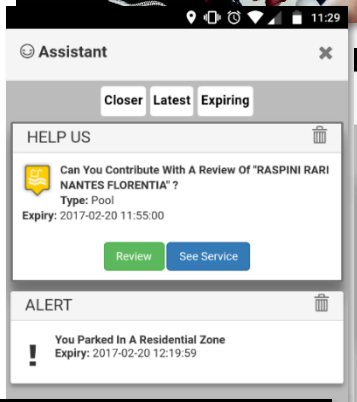
Profiled Engagements to City Users

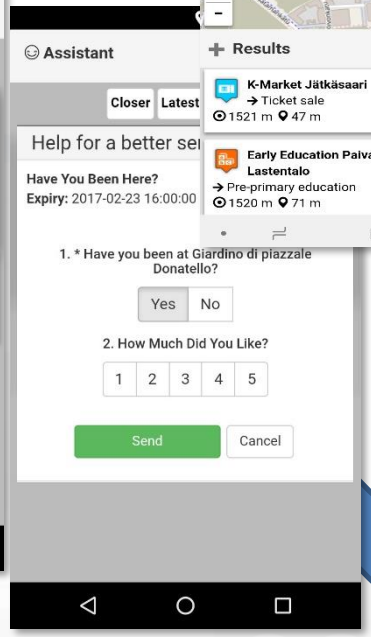
- The users are profiled to learn habits:
  - Personal POI, paths, Mobility habits
- Information and engagements sent to the users are programmed according to the context and user behavior to:
  - Stimulate virtuous habits
  - More sustainable habits
  - More healthy habits, etc.
  - Get feedbacks
  - Provide bonus and prices, …..
  - Send alerts, ….

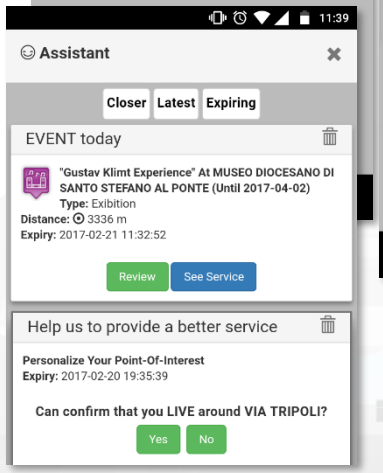


Users' Engagement

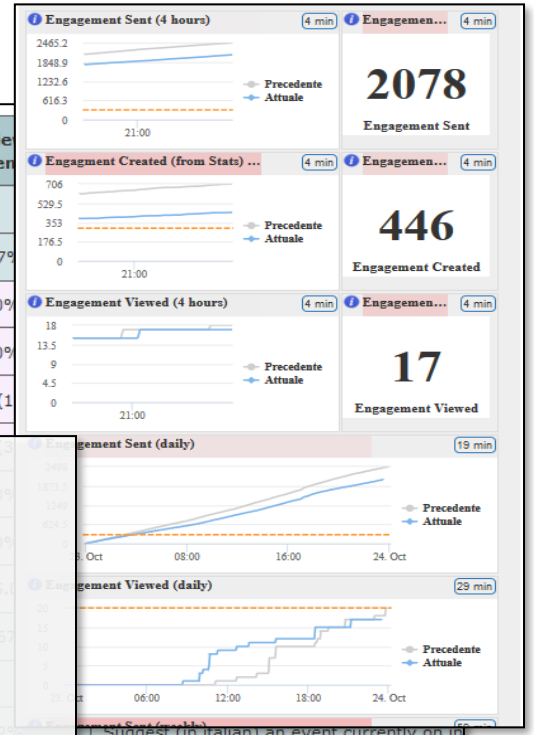








Rule name	Type	#sent	#viewed	#vie #sen
daily_event_de	ENGAGEMENT	1 (0%)	0 (0%)	0%
daily_event_en	ENGAGEMENT	1720 (2.12%)	70 (7.1%)	4.07%
	- commuter	5 (0.29%)	0 (0%)	0 (0%
	- student	14 (0.81%)	0 (0%)	0 (0%
	- tourist	1462 (85%)	25 (35.71%)	25 (1



Suggest (in italian) an event currently on in
Alert (in english) if the user parked in a res
Alert (in spanish) if the user parked in a res
Alert (in italian) if the user parked in a resid
Ask (in german) a contribution for a nearby

Inform
- Air Quality forecast is not very nice
- You have parked out of your residential parking zone
- The Road cleaning is this night
- The waste in S.Andreas Road is full

Engage
- Provide a comment, a score, etc.

Stimulate / recommend
- Events in the city, services you may be interested, etc..

Provide Bonus, rewards if needed
- you get a bonus since you parked here
- We suggest: leave the car out of the city, this bonus can be used to by a bus ticket

Rules

User context

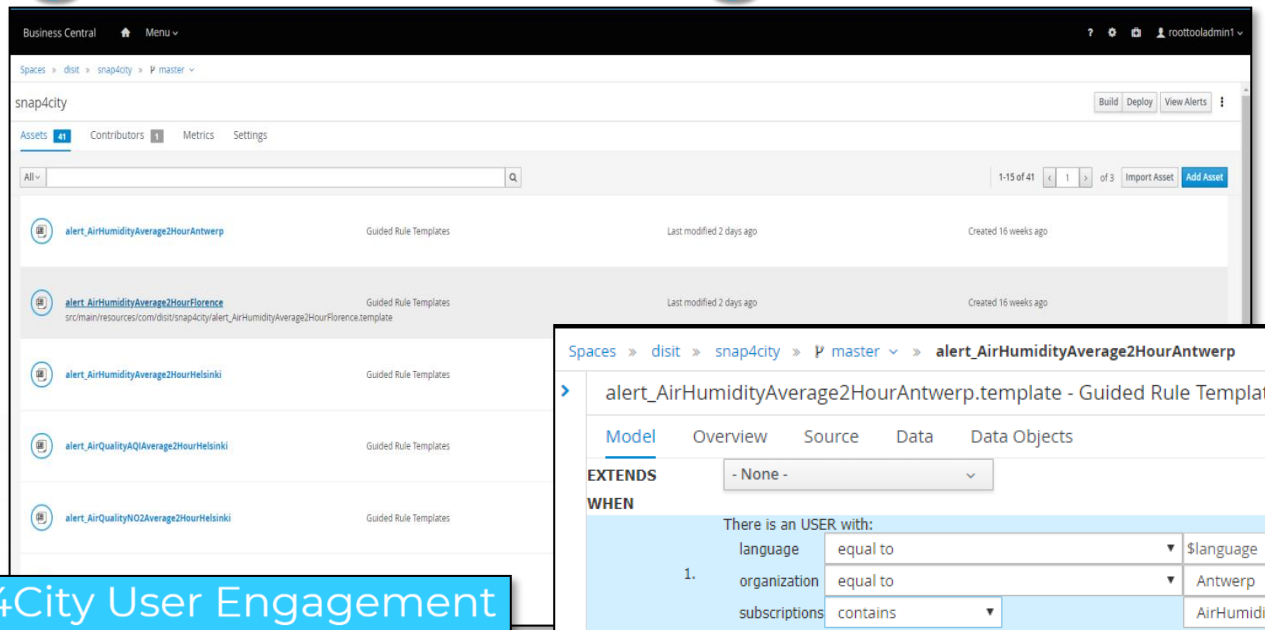
City context

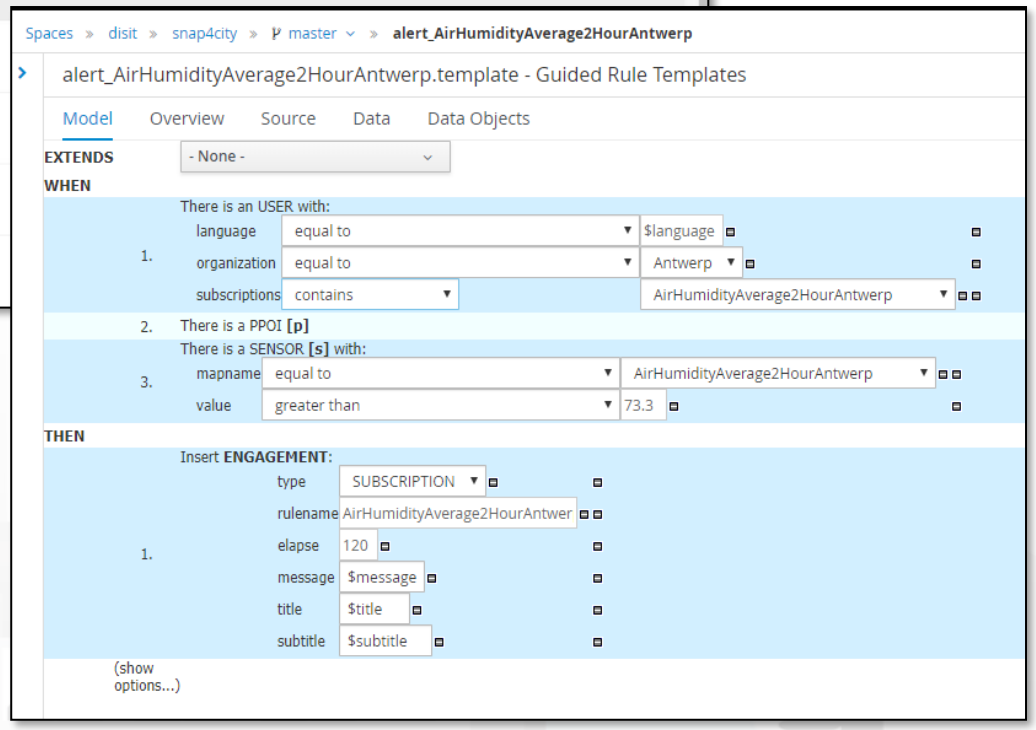
Engaging City Users

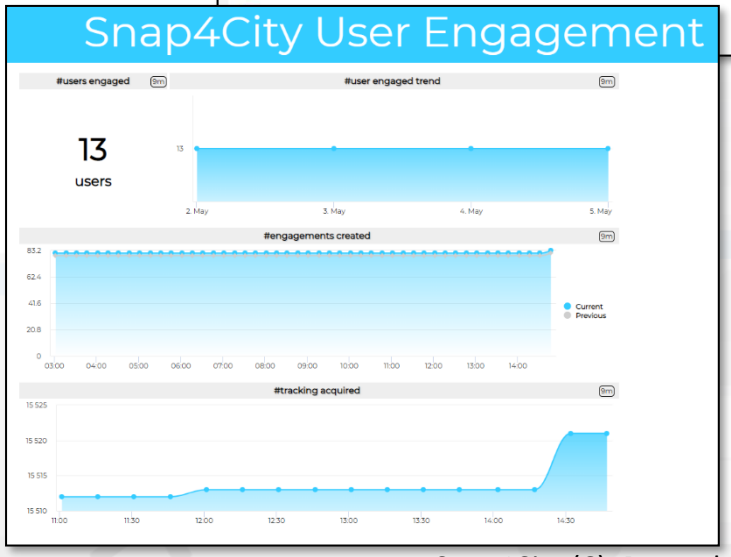
- **Mobile Applications** can use Advanced Smart City API to collect data about the city usage by the city users via a signed consent
- It can be used for sending engagements to them such as to:
  - **Inform**
    - You have parked out of your residential parking zone
    - The Road cleaning is this night
    - The waste in S.Andreas Road is full
  - **Engage**
    - Please Provide a comment, a score, etc.
  - **Stimulate / recommend**
    - Events in the city, services you may be interested, etc..
  - **Provide Bonus**
    - Since you have parked here you can get 1 Bonus
    - We suggest you to leave the car out of the city, this bonus can be used to buy a bus ticket

Engagement Manager

- Definition of Rules for campaigns
- Monitoring and follow-up for each City
- Segmented for user kind and interest







Campaing on Sustainable Mobility

Sii smart.
Sii-Mobility!
Scarica, viaggia, vinci!

Dal 15 aprile al 15 luglio scegliere il trasporto pubblico ti premia!
Scarica l'app "Toscana dove, cosa", guadagna punti viaggiando in autobus e vinci tanti fantastici premi.
Per maggiori informazioni visita il sito info.sii-mobility.org
TOSCANA
ataf
BUSITALIA
GRUPPO FERROVIE DELLO STATO ITALIANE
MIUR
UNIVERSITÀ DEGLI STUDI FIRENZE
DINFO
DISIT

Rules for Rewards

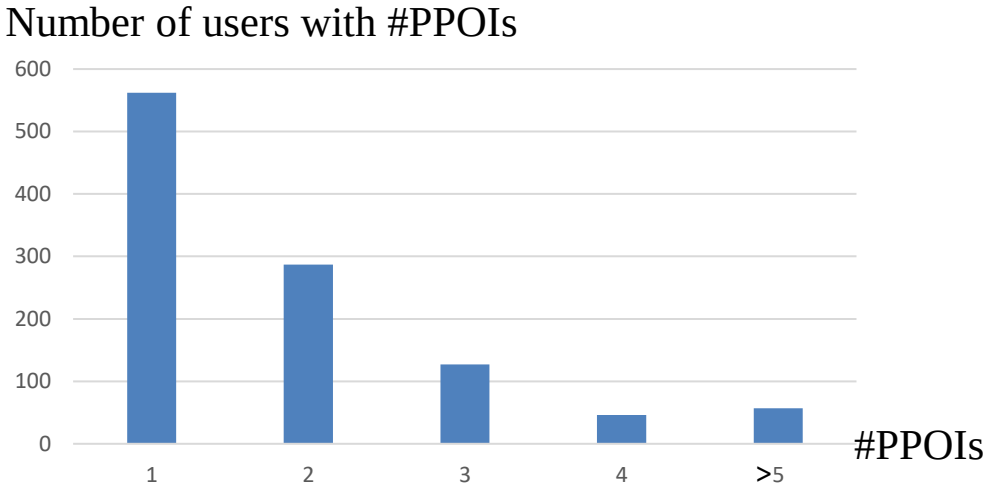
- **ASSISTANCE**
- If public transport is detected after bus line suggestion on trajectory usually made on private transport → 10points
  - Why don't you take the bus line 4 in Piazza Marconi to reach your workplace? You save money, you respect the environment and you will be stress free for not worry about parking!
- Once a day, if public transport is detected after suggestion on an alternative bus line availability →3points
  - Why don't you take the bus line 4 that stop just 50 meters far from you? You save money, you respect the environment and you will be stress free for the traffic jam!
- If public transport is detected for at least 30(?) minutes a day → 1point

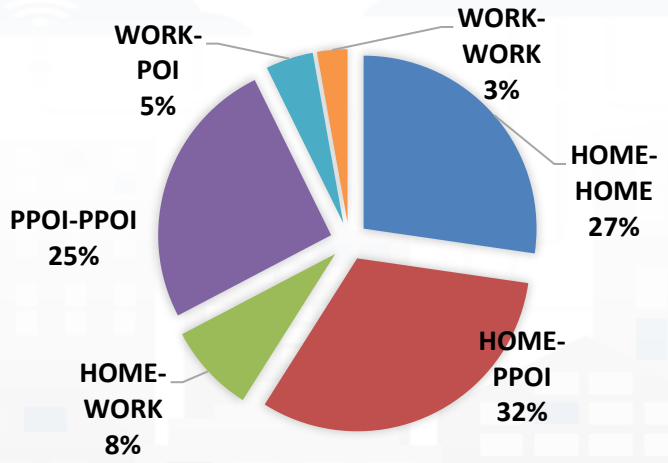
- **ENGAGEMENT**
- Survey on commuter and their preferred way of mobility → 1point
  - How many minutes you usually commute to go to work? How do you rate the service?
- Feedback on public transport →1point
  - Which current public transport are you using? Are the service in line with your expectation?
- Comments/Photo/Rate or survey on POI (public transport) →1point
- Survey on use of the App after N days or for tourist coming home → 1point
- Feedback on PPOI or mobility → 1point

Current Numbers

From 1° September 2016

- Detected 2108 PPOIs on 1080 users
  - 437 HOME
  - 285 WORK
  - 34 SCHOOL
  - 1350 EXTRA
- 130 PPOIs are feedbacked
- 460 survey responses



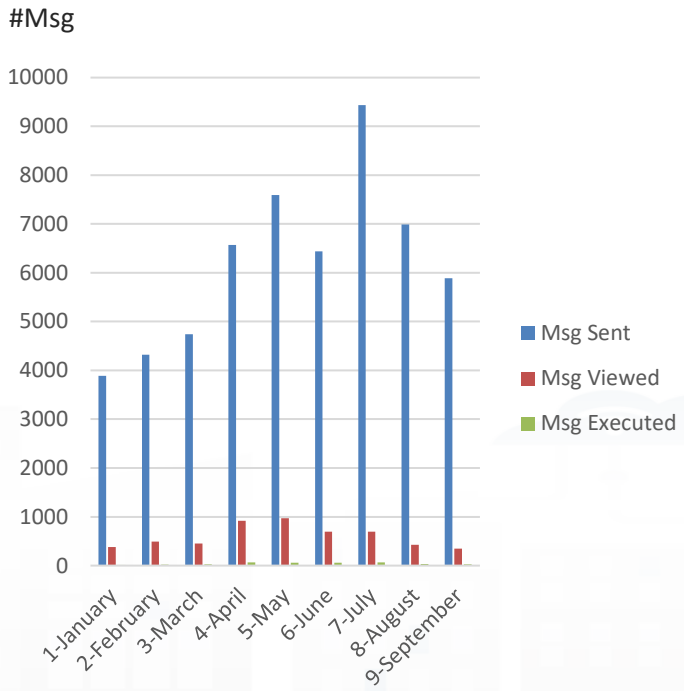


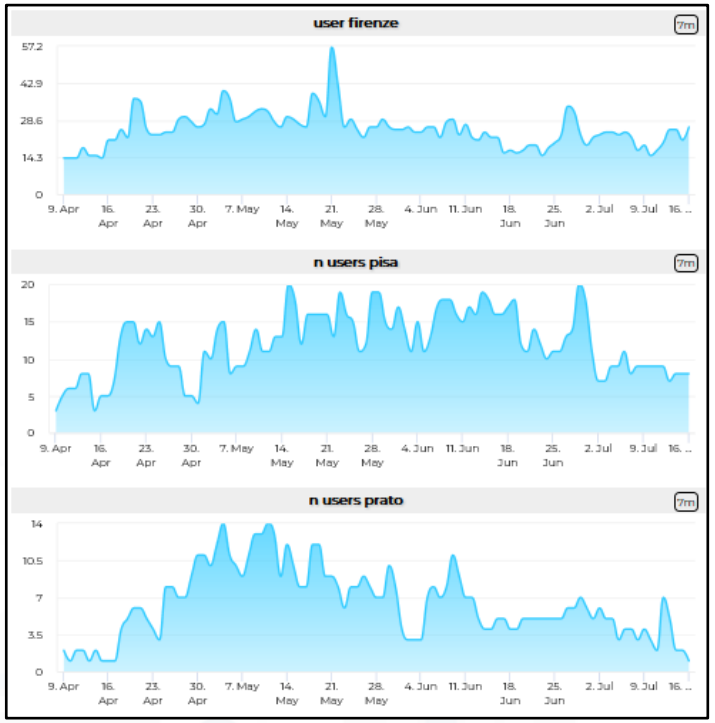
From 1° August 2017

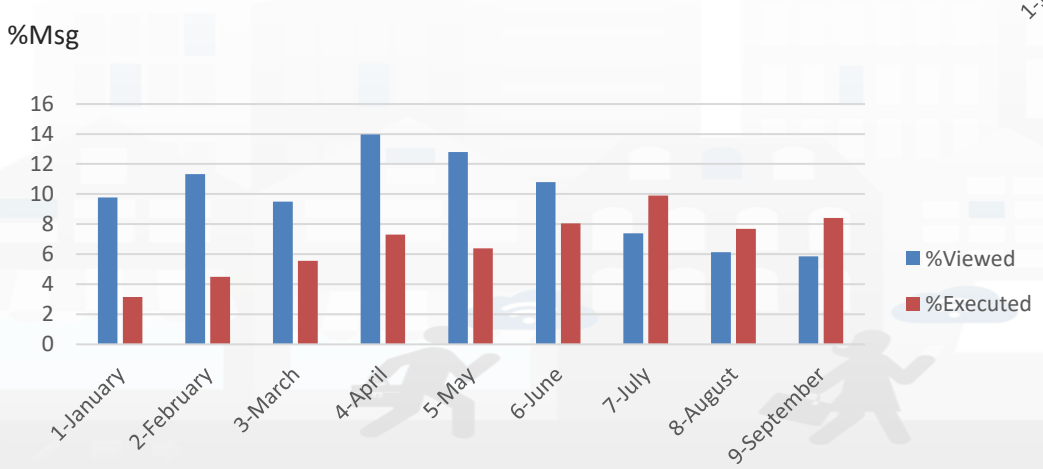
- Built 524 Markov Networks about user's trajectories

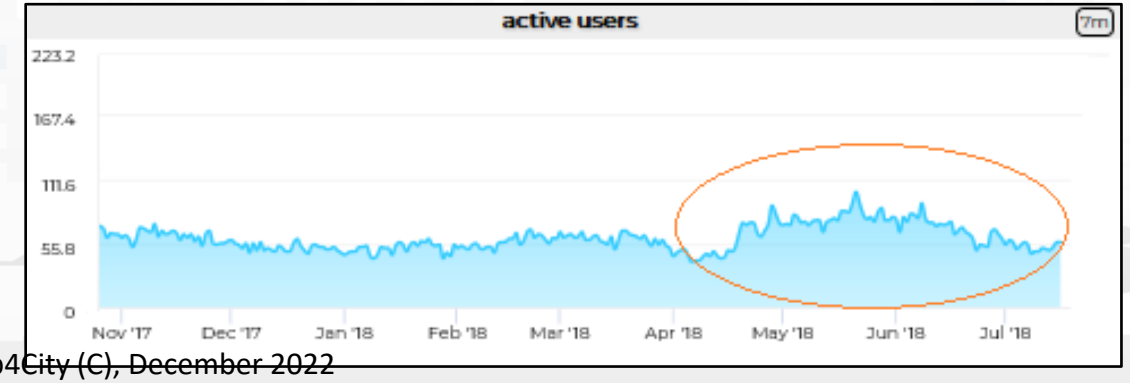
Validation of user Engagement

Months	Msg Sent	Msg Viewed	Msg Executed
1-January	3888	380	12
2-February	4319	489	22
3-March	4739	450	25
4-April	6567	918	67
5-May	7594	972	61
6-June	6437	695	55
7-July	9432	697	69
8-August	6988	429	73
9-September	5885	345	49
Total	55849	5375	433







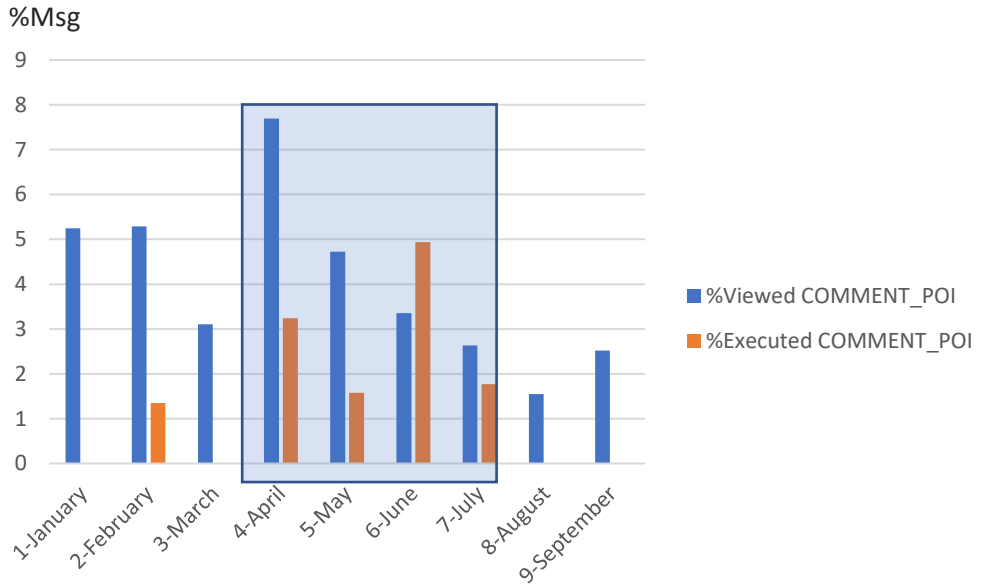


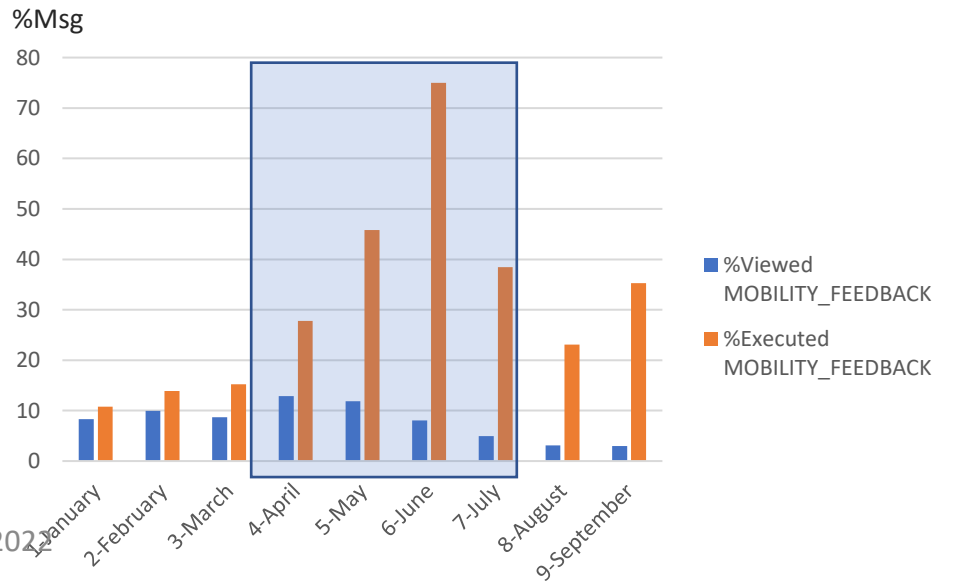
User Behaviour Analysis

VALIDATION

- During the PILOT new rules has been added (30 on a total of 80) and mostly all of them are still online
- COMMENT_POI: requires more user interaction and not very contextualized (POI proximity) → higher rate of sent, lower rate on execution
- MOBILITY_FEEDBACK: requires less user iteration and very contextualized (user in MOBILITY) → normal rate of sent, high rate on execution

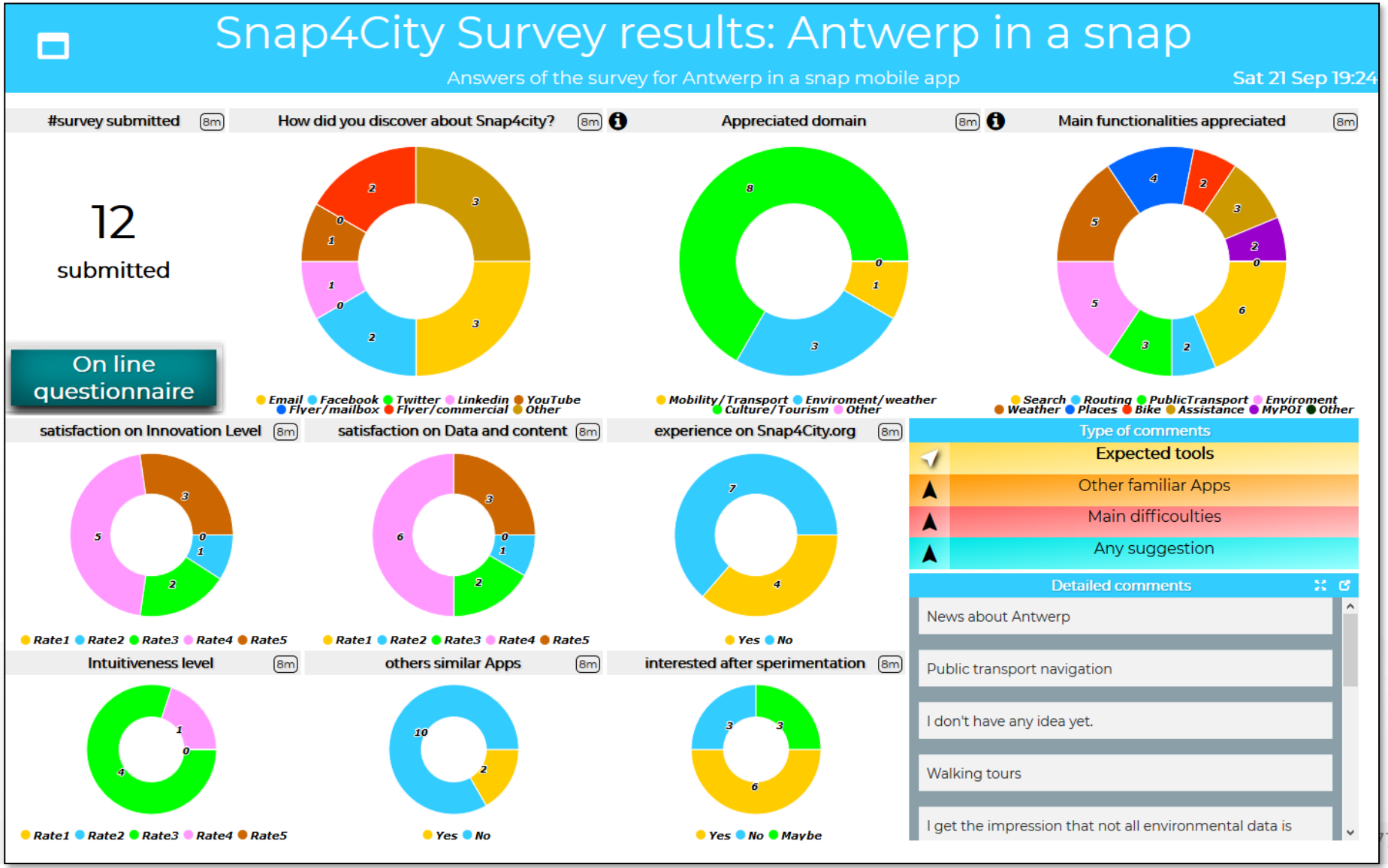
	Msg Sent	Msg Viewed	Msg Executed
COMMENT_POI	**21632**	**804**	**15**
MOBILITY_FEEDBACK	**5378**	**371**	**94**





https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTc2OQ==

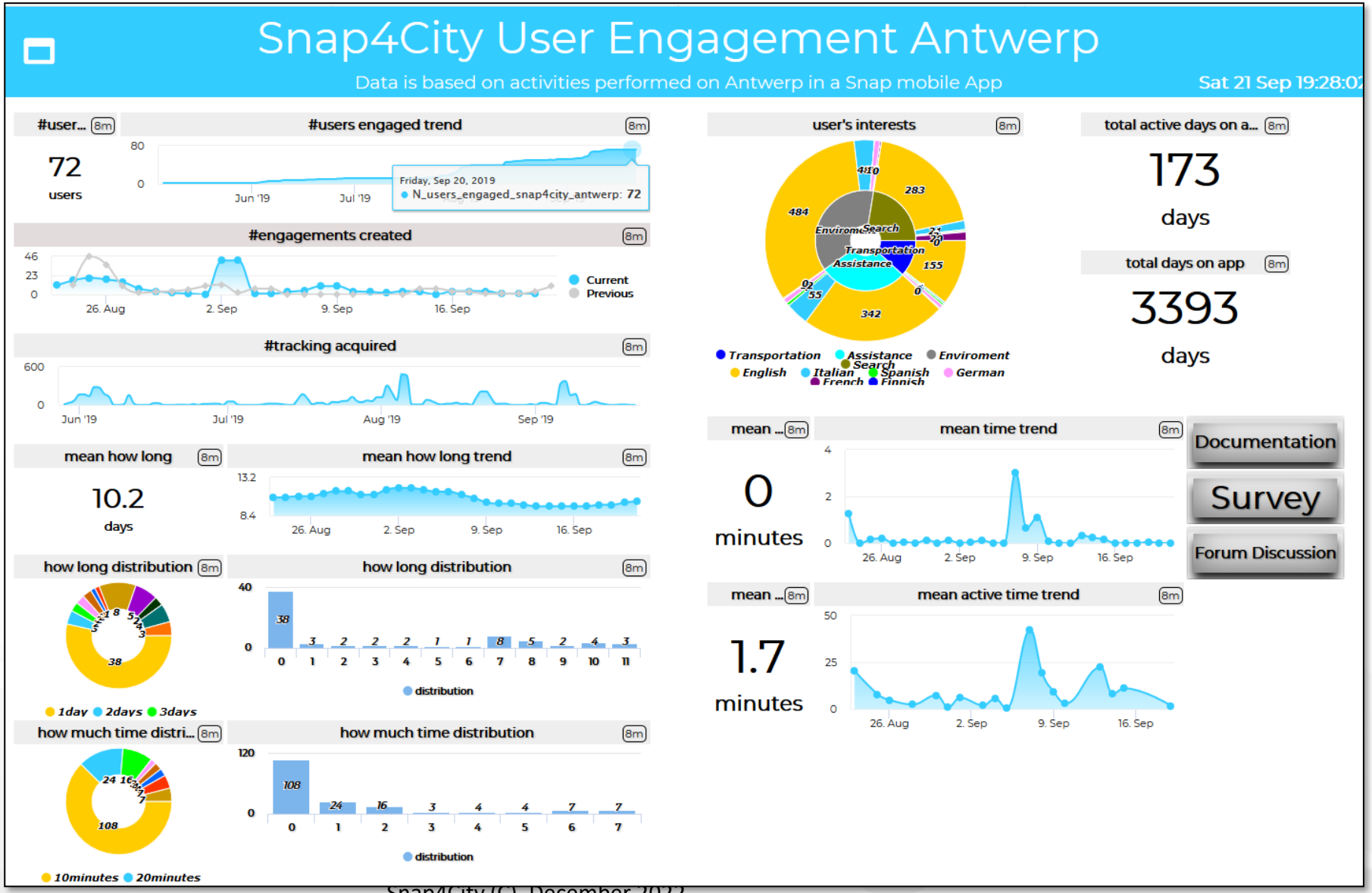
Dashboard created to monitor in real time the answers to the survey provided on the Mobile App directly by the Engagement tool



https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTc1OQ==

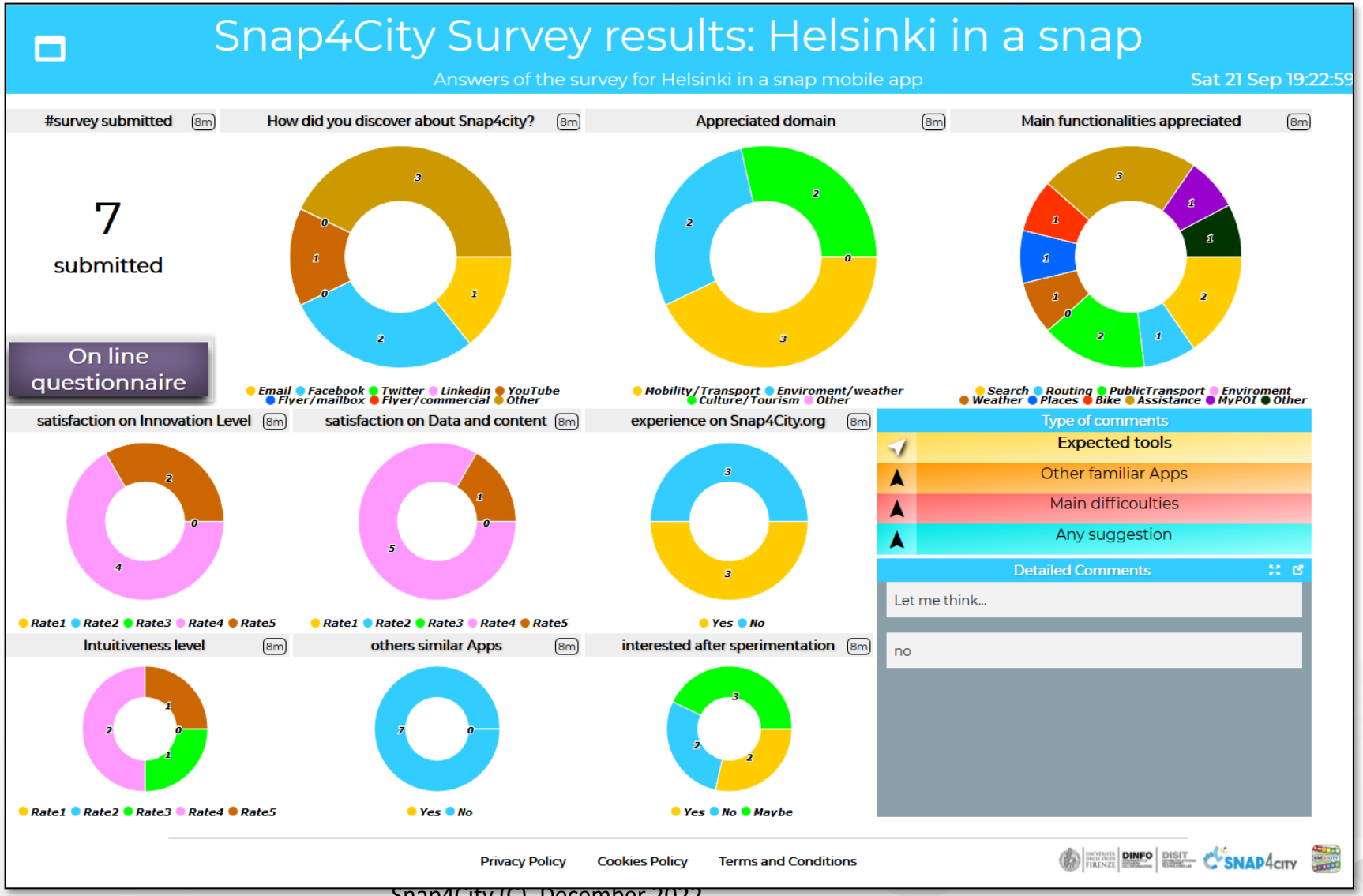
Dashboard monitoring the Mobile App:

- Collecting the clicks
- Describing the community of users in terms of the profile aspects
- Measuring the time spend, and topics of interest of the users, etc.



https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTc2OA==

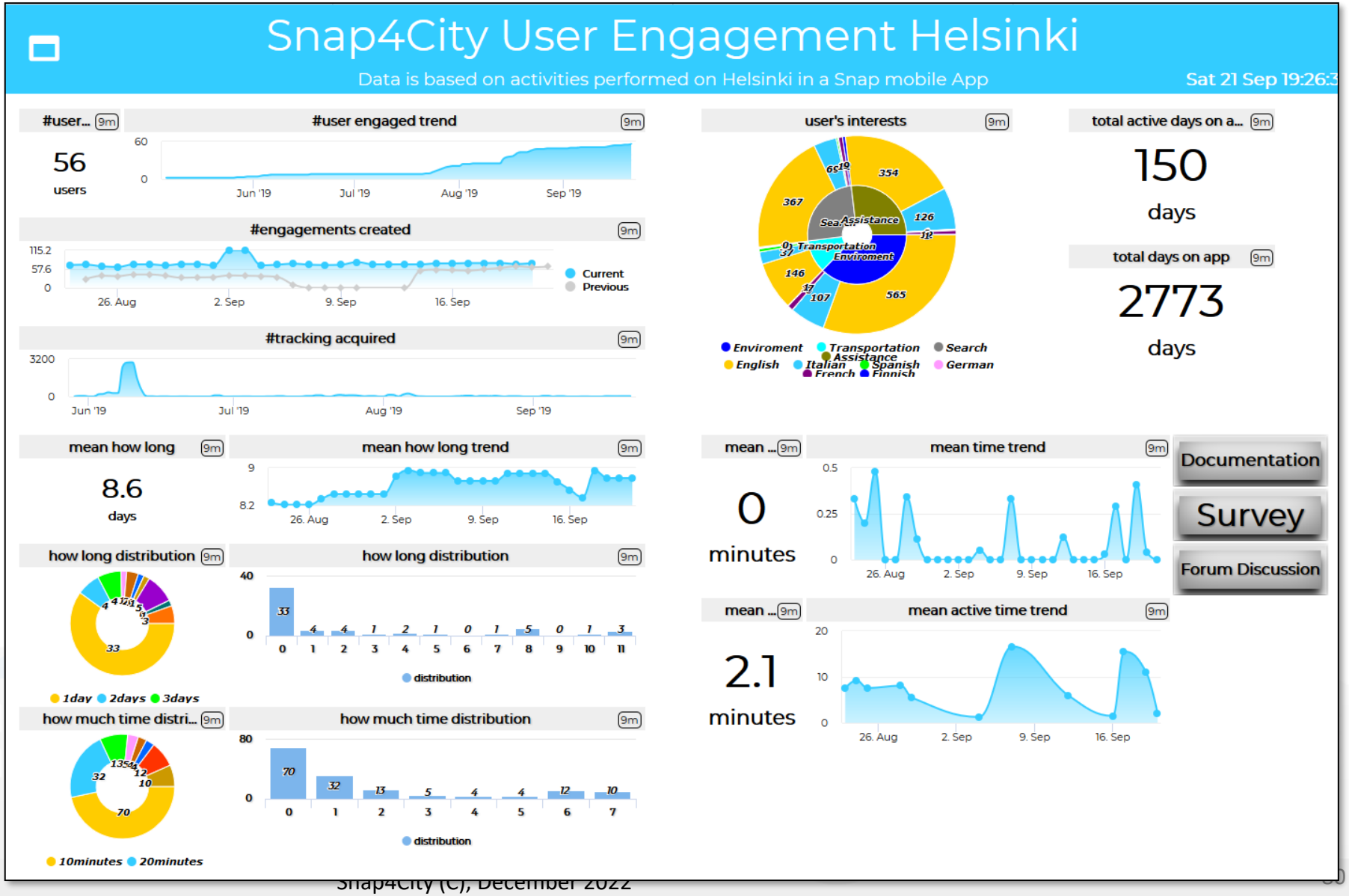
Dashboard created to monitor in real time the answers to the survey provided on the Mobile App directly by the Engagement tool



https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTc1OA==

Dashboard monitoring the Mobile App:

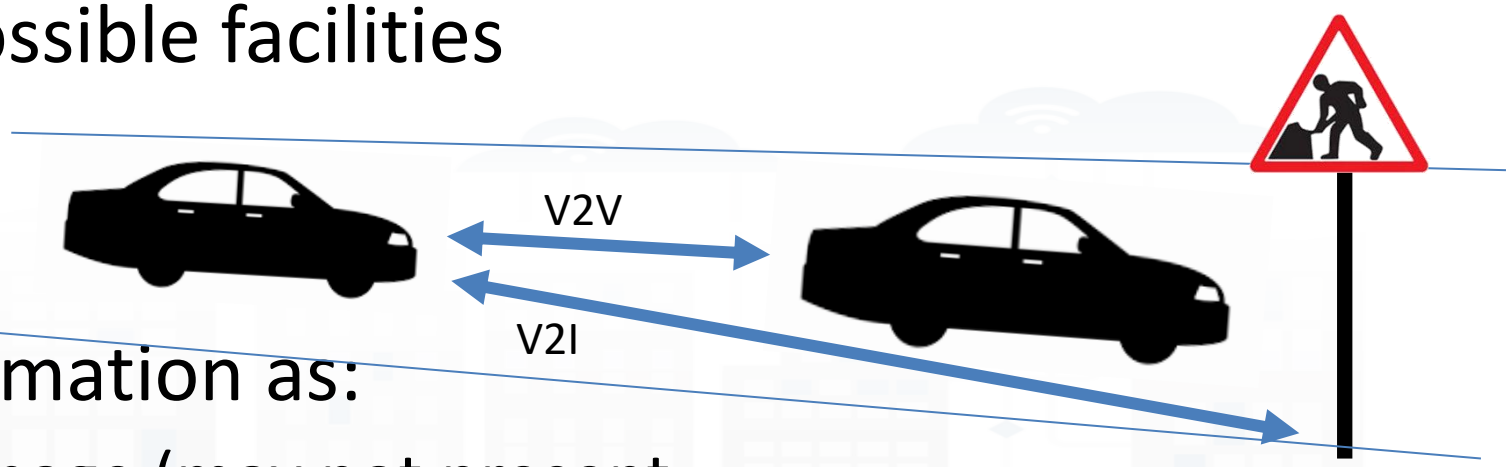
- Collecting the clicks
- Describing the community of users in terms of the profile aspects
- Measuring the time spend, and topics of interest of the users, etc.



Connected Drive

Main Concept of Connected Drive

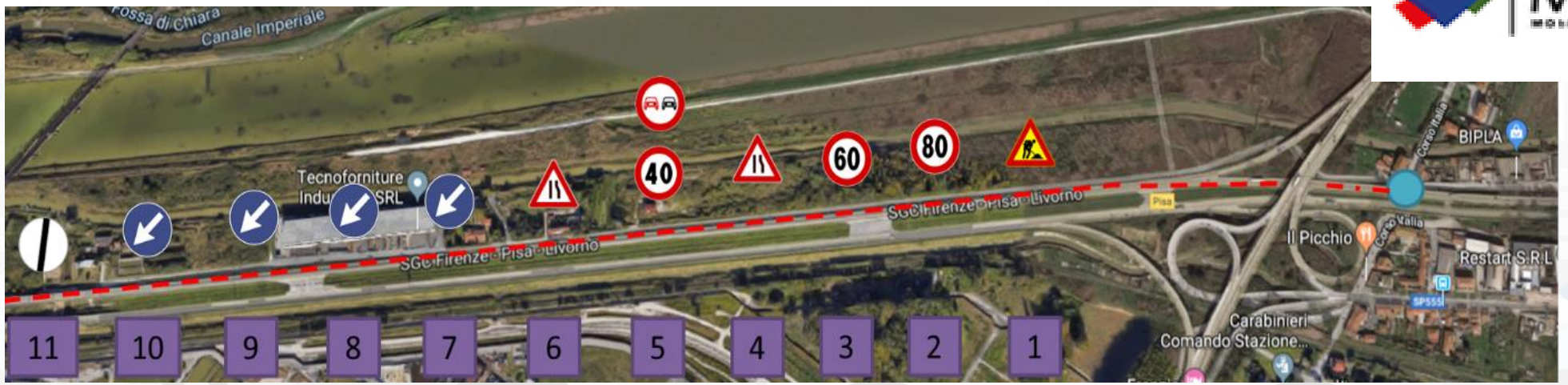
- Different kinds of communications may arrive on the vehicles on board devices
- Mobile Phones can be a possible facilities



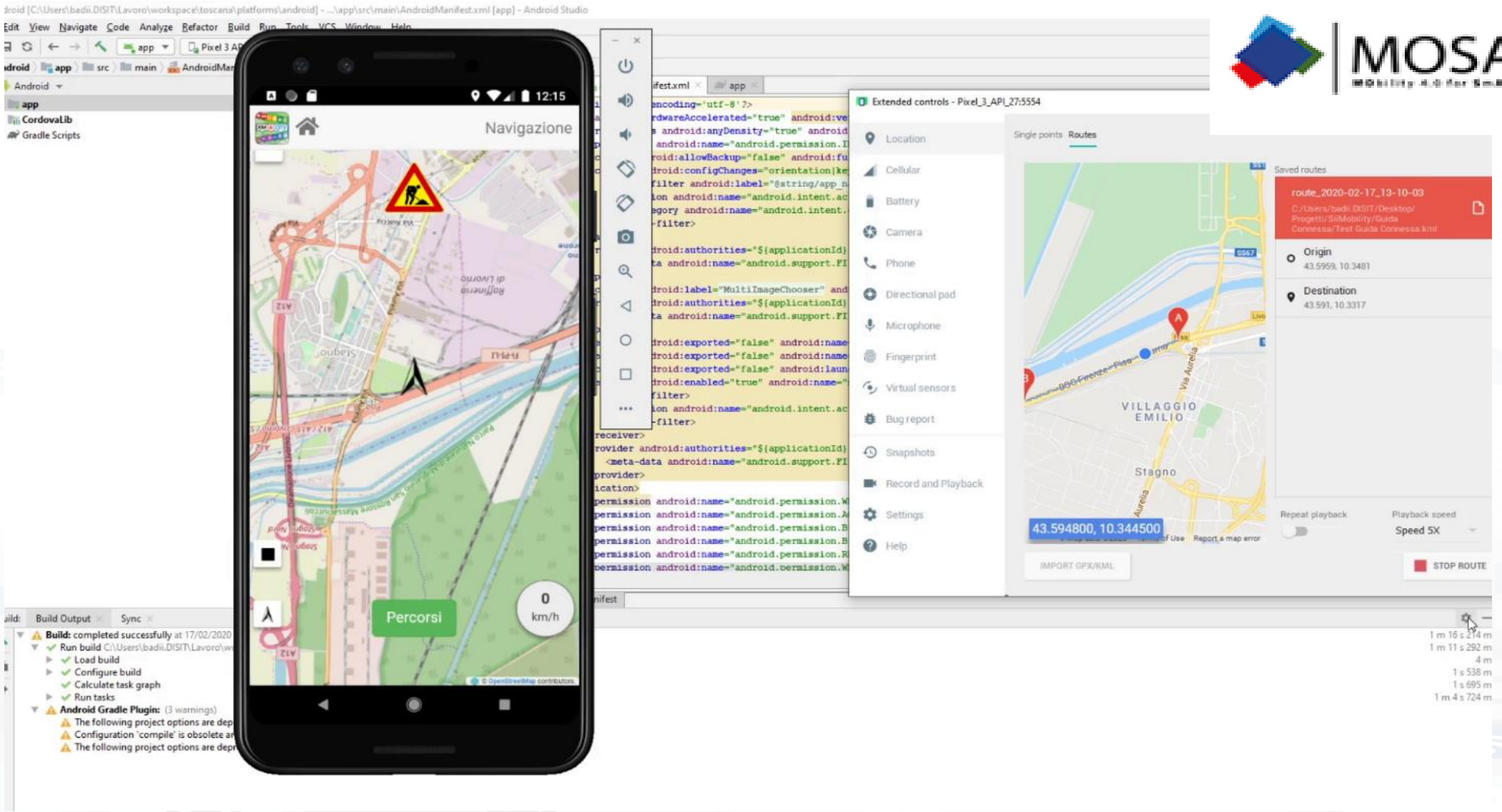
- Geolocated Real time Information as:
  - Alerting, dynamic digital signage (may not present physically on the road)
  - Supporting autonomous driving vehicles

Experimentation on «Toscana Dove Cosa»

- Mobile App supporting connected Drive V2I connections:
  - https://play.google.com/store/apps/details?id=org.disit.toscana&hl=it
  - https://apps.apple.com/it/app/toscana-where-what-km4city/id1064554200
  - For the MOSAIC project and pilot in Tuscany
- The mobile App has a Navigator which includes now the acquisition of connected drive messages



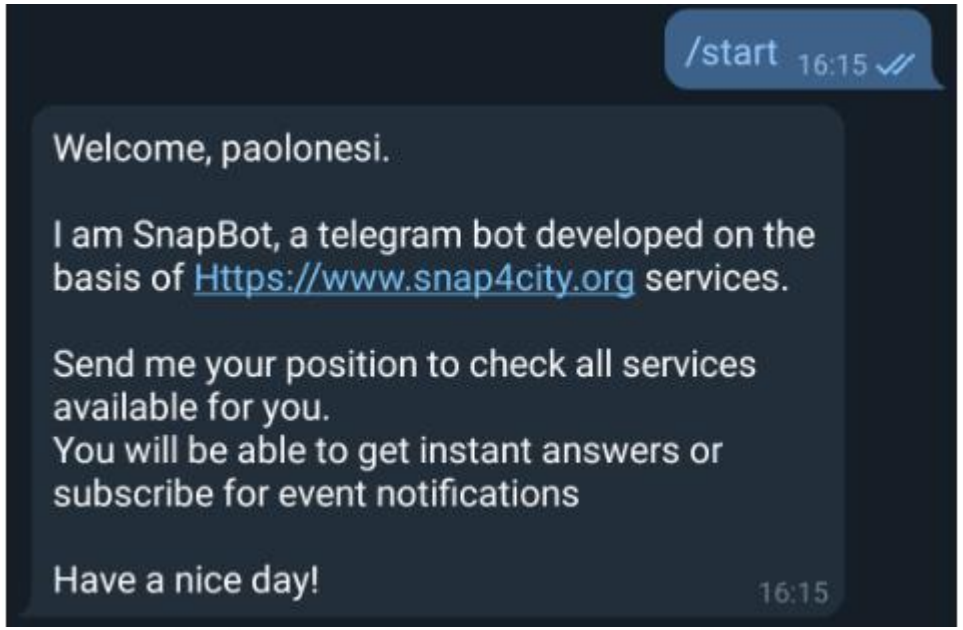
Scenario

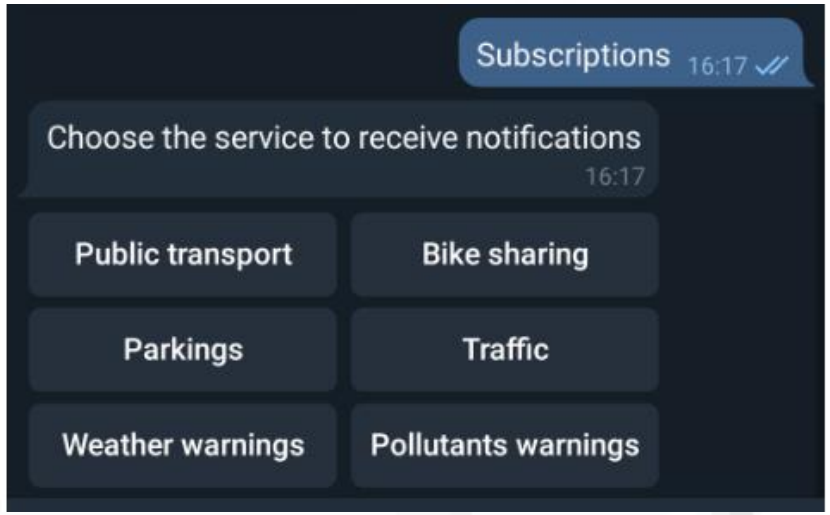

Navigazione
Percorsi
0
km/h
Extended controls - Pixel_3_API_27:5554
Location
Cellular
Battery
Camera
Phone
Directional pad
Microphone
Fingerprint
Virtual sensors
Bug report
Snapshots
Record and Playback
Settings
Help
Single points
Routes
Saved routes
route_2020-02-17_13-10-03
Origin
43.5959, 10.3481
Destination
43.591, 10.3317
VILLAGGIO EMILIO
Stagno
43.594800, 10.344500
Repeat playback
Playback speed
Speed 5X
IMPORT GPX/KML
STOP ROUTE
Build Output
Sync
Build: completed successfully at 17/02/2020
Run build
Load build
Configure build
Calculate task graph
Run tasks
Android Gradle Plugin: (3 warnings)

Integration with Telegram: SnapBot solution

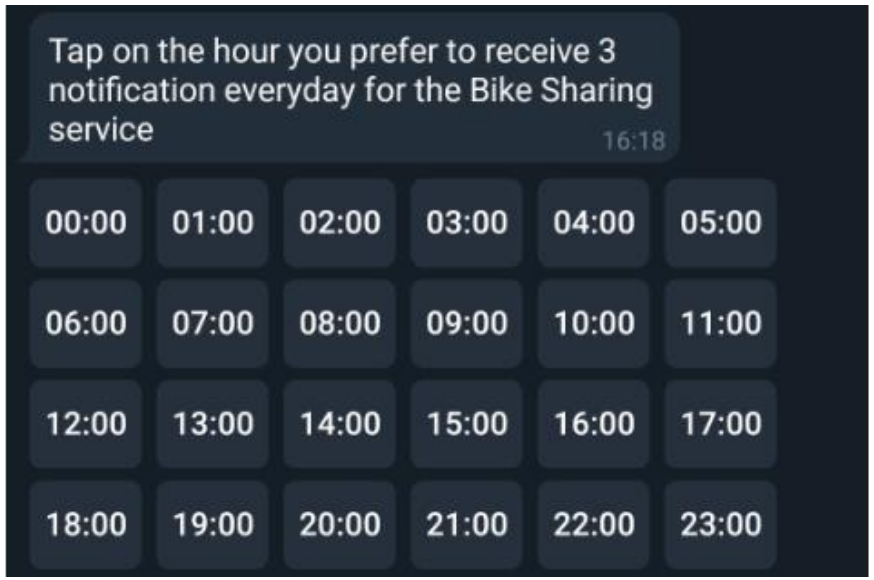
SnapBot

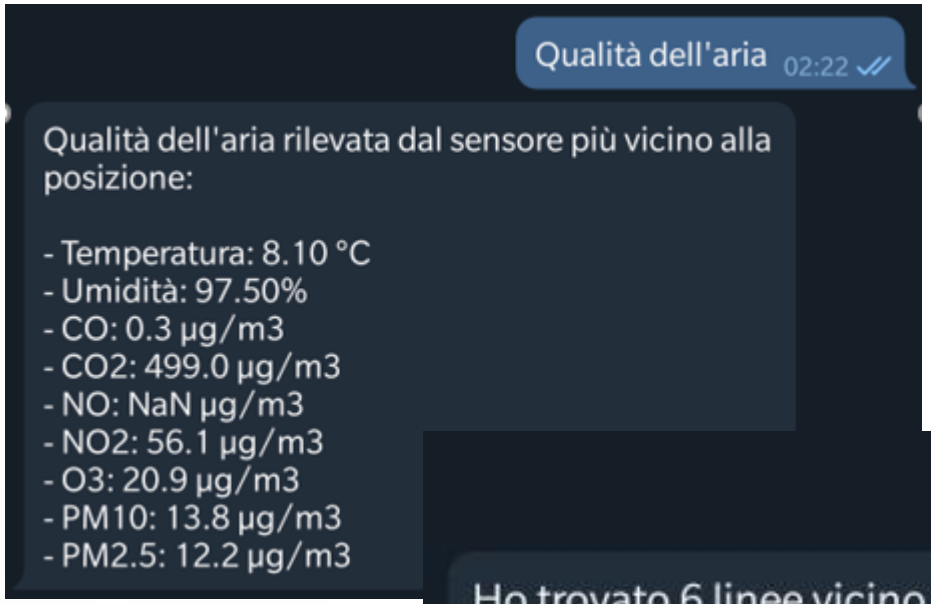
- provides real time smart city services to **Telegram** users, geolocalized, when you like, what you like
- active on Tuscany in all provinces and cities according to the data accessible on Https://www.snap4city.org
- Services on
  - Public Transport (more than 10 different operators),
  - bike sharing, parking lots,
  - traffic flow, weather warnings,
  - Air quality, pollutant,
  - find your location, etc.

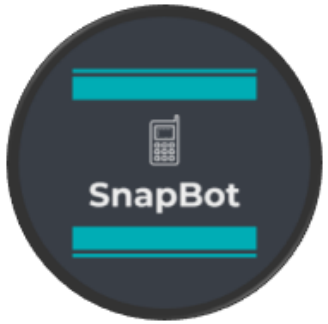


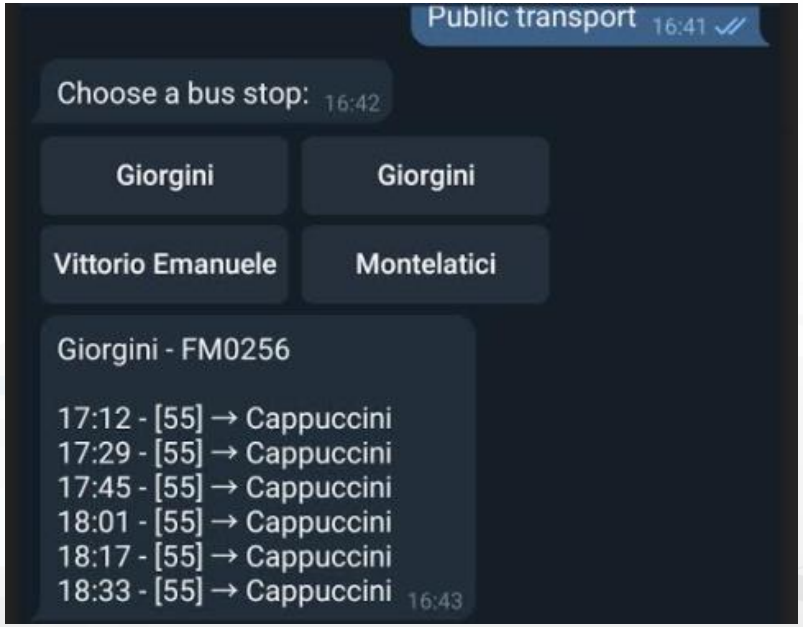


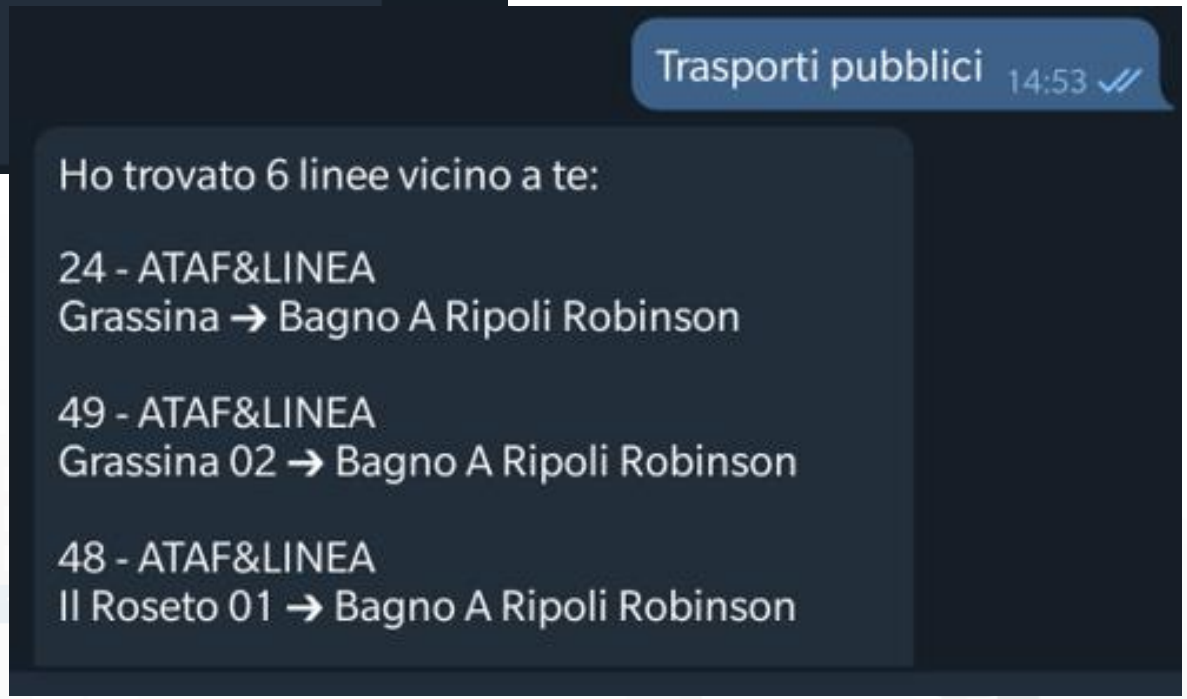
SnapBot


Tap on the hour you prefer to receive 3 notification everyday for the Bike Sharing service
16:18
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00

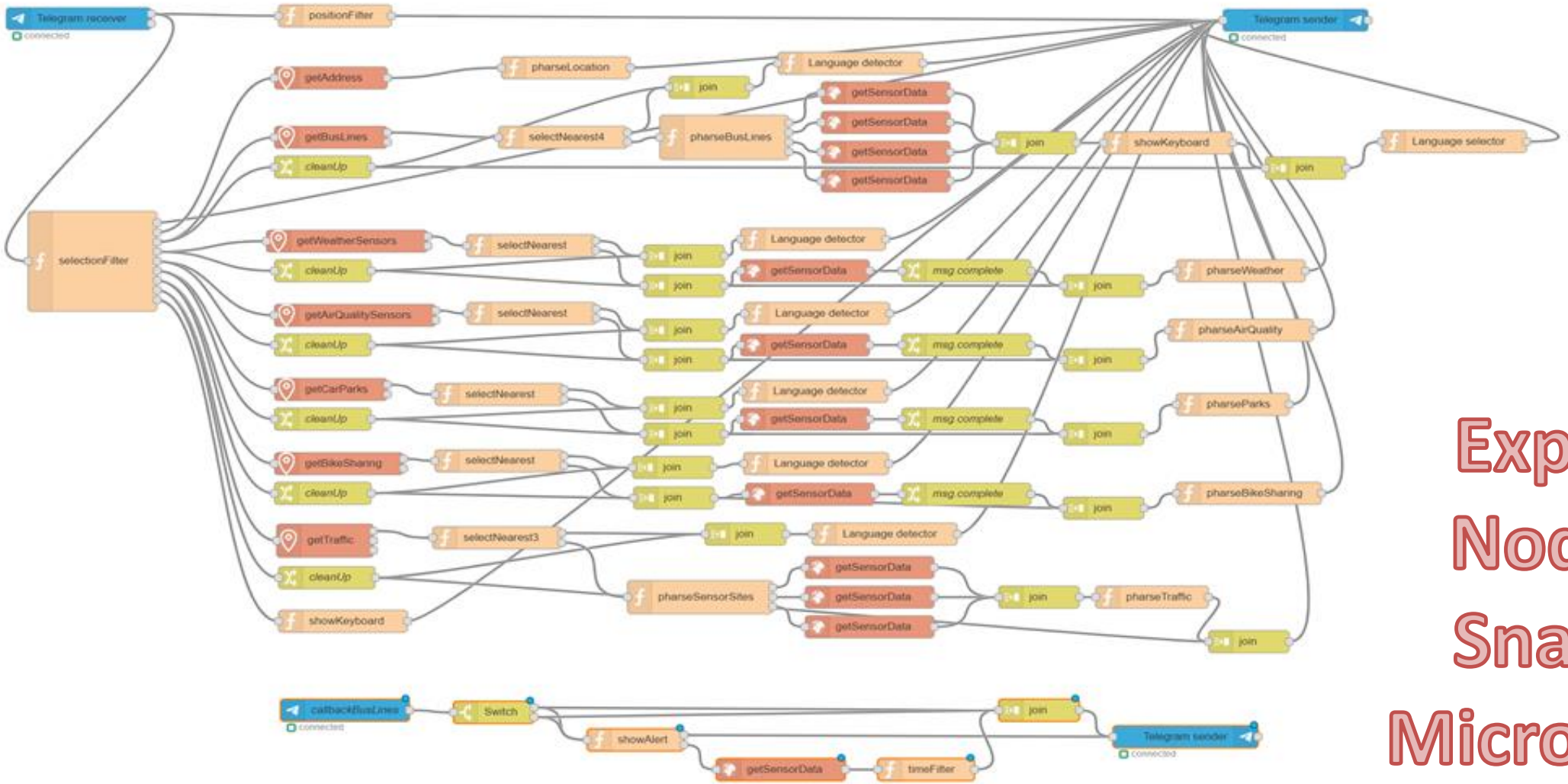

Qualità dell'aria 02:22
Qualità dell'aria rilevata dal sensore più vicino alla posizione:
- Temperatura: 8.10 °C
- Umidità: 97.50%
- CO: 0.3 µg/m3
- CO2: 499.0 µg/m3
- NO: NaN µg/m3
- NO2: 56.1 µg/m3
- O3: 20.9 µg/m3
- PM10: 13.8 µg/m3
- PM2.5: 12.2 µg/m3


SnapBot


Public transport 16:41
Choose a bus stop: 16:42
Giorgini
Giorgini
Vittorio Emanuele
Montelatici
Giorgini - FM0256
17:12 - [55] → Cappuccini
17:29 - [55] → Cappuccini
17:45 - [55] → Cappuccini
18:01 - [55] → Cappuccini
18:17 - [55] → Cappuccini
18:33 - [55] → Cappuccini 16:43


Trasporti pubblici 14:53
Ho trovato 6 linee vicino a te:
24 - ATAF&LINEA
Grassina → Bagno A Ripoli Robinson
49 - ATAF&LINEA
Grassina 02 → Bagno A Ripoli Robinson
48 - ATAF&LINEA
Il Roseto 01 → Bagno A Ripoli Robinson

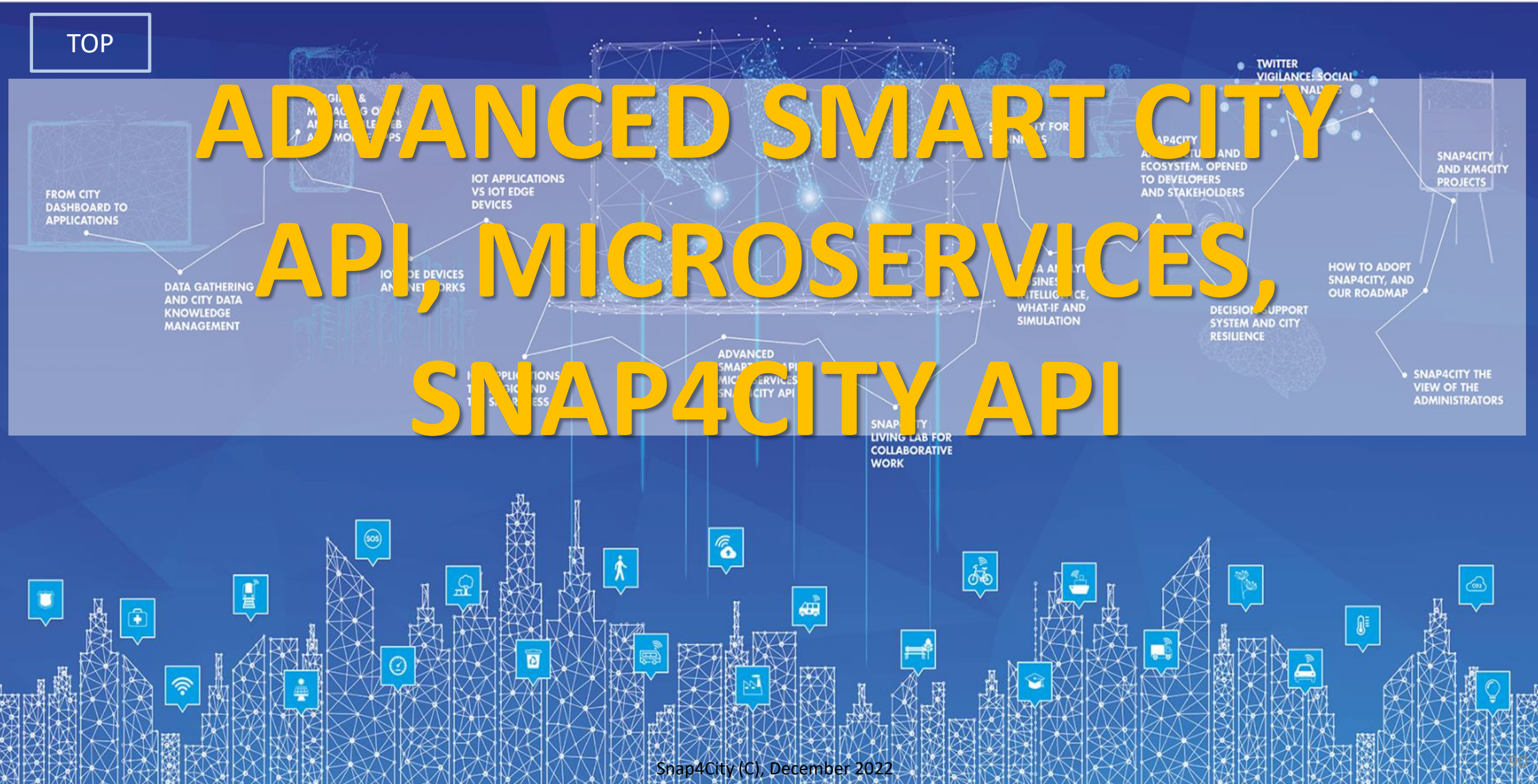
IOT App of SnapBot: OneShot Services



Exploiting Node-RED Snap4City MicroServices

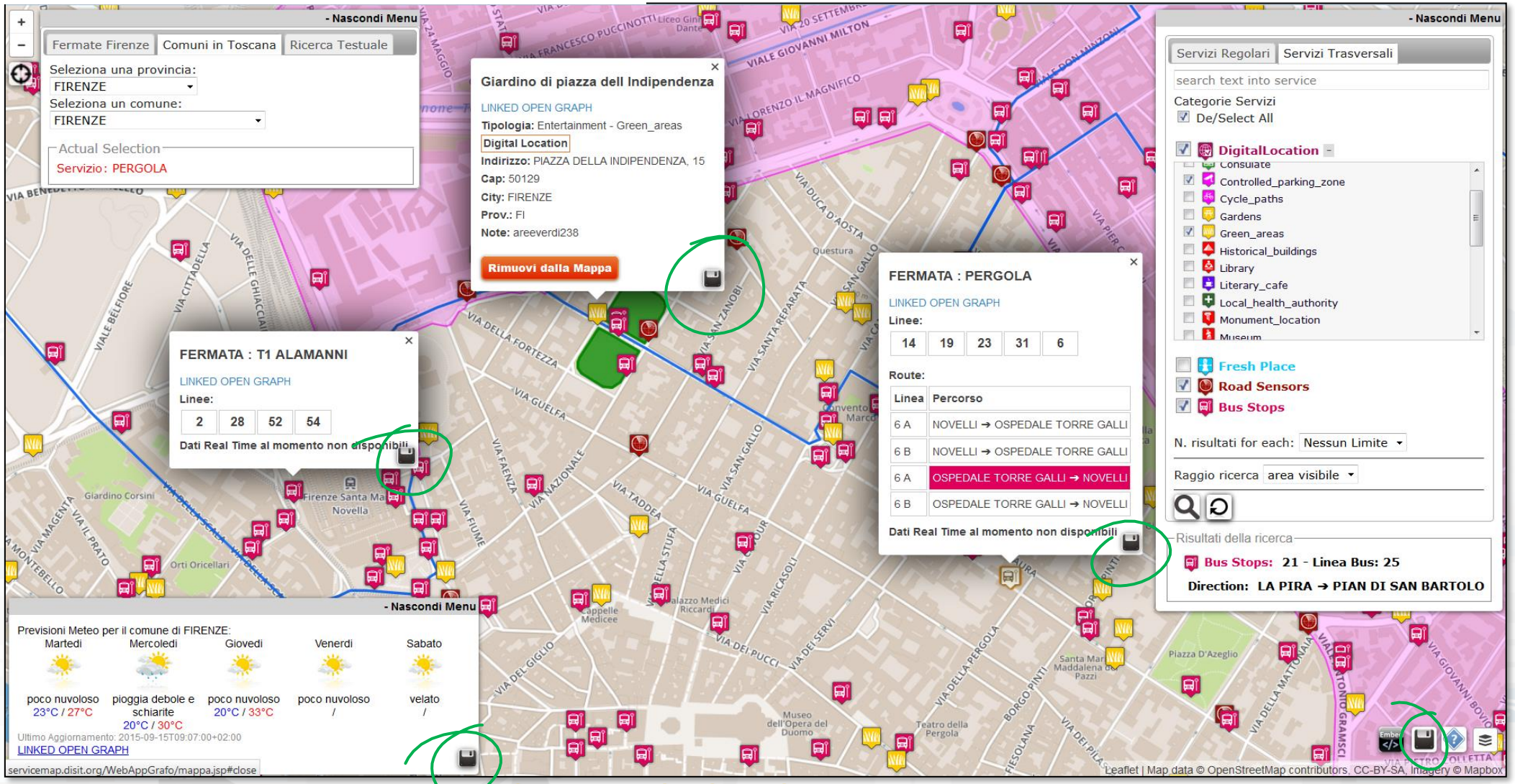
TOP

ADVANCED SMART CITY API, MICROSERVICES, SNAP4CITY API



Some structures from Km4City model

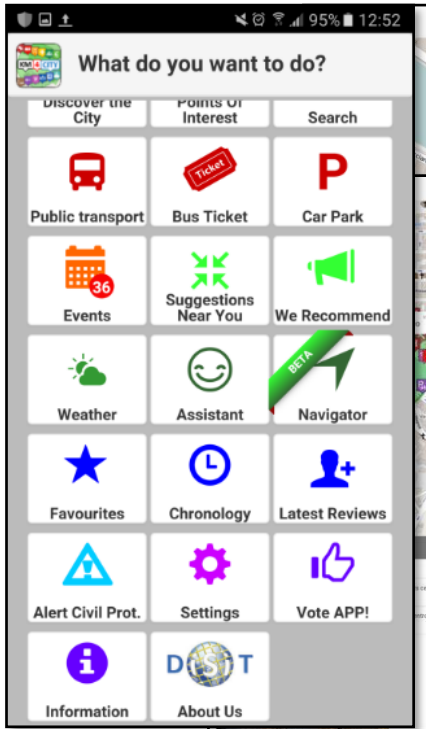
ServiceMap: https://servicemap.km4city.org

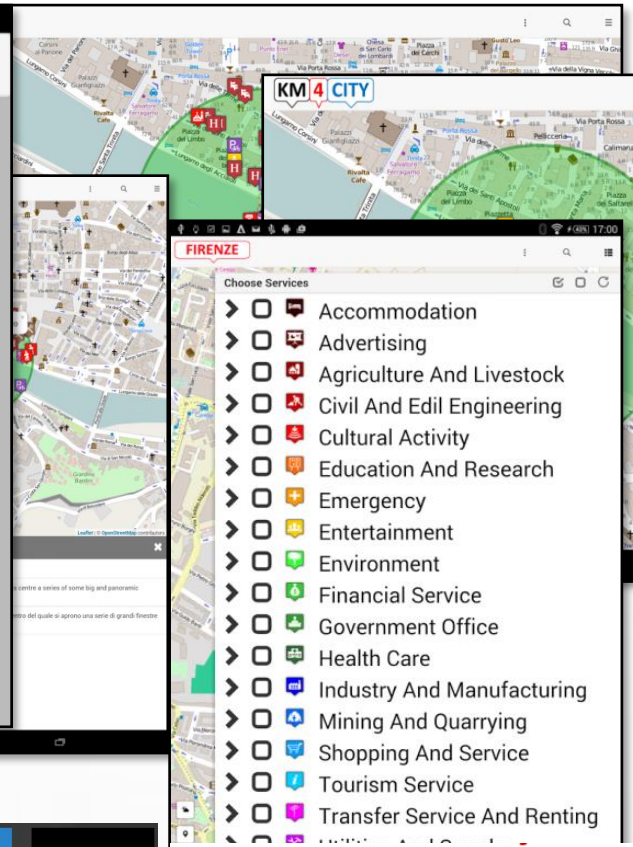


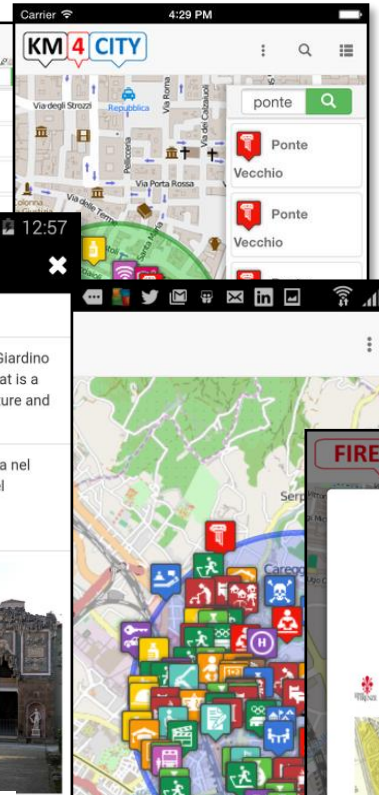
Areas, Bus lines, bike lanes, tram, RTZ, etc.

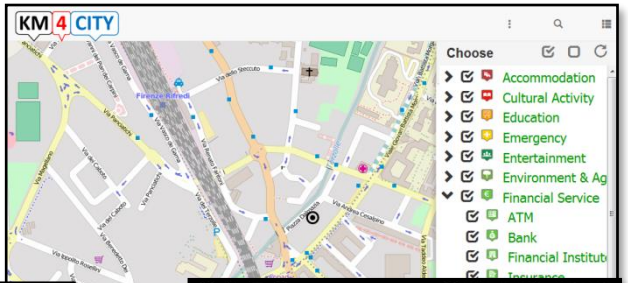
Km4CityMobile App

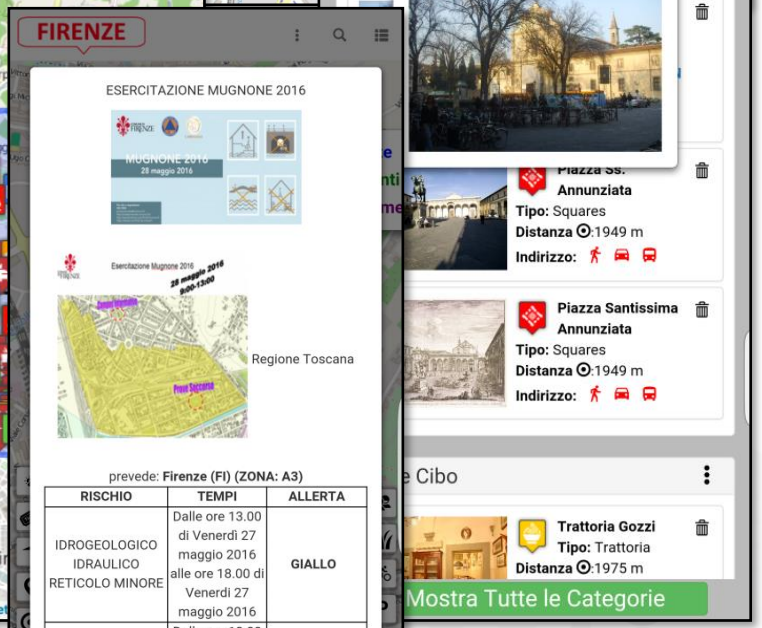
http://www.km4city.org











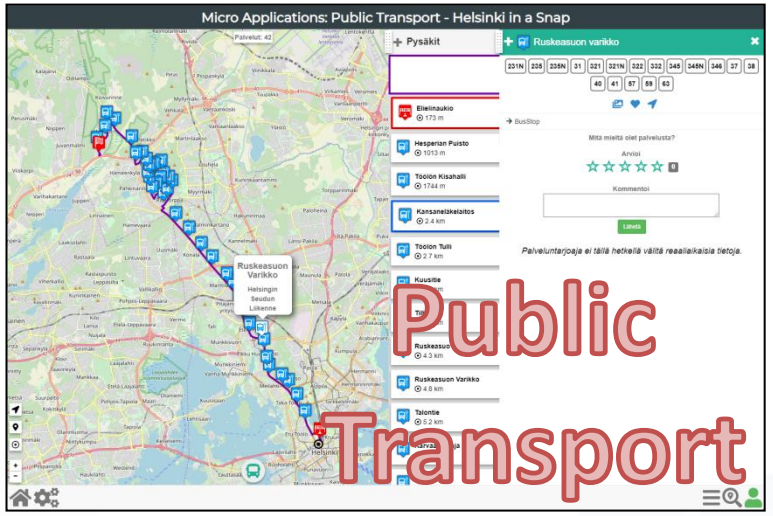
DISPONIBILE SU Google play

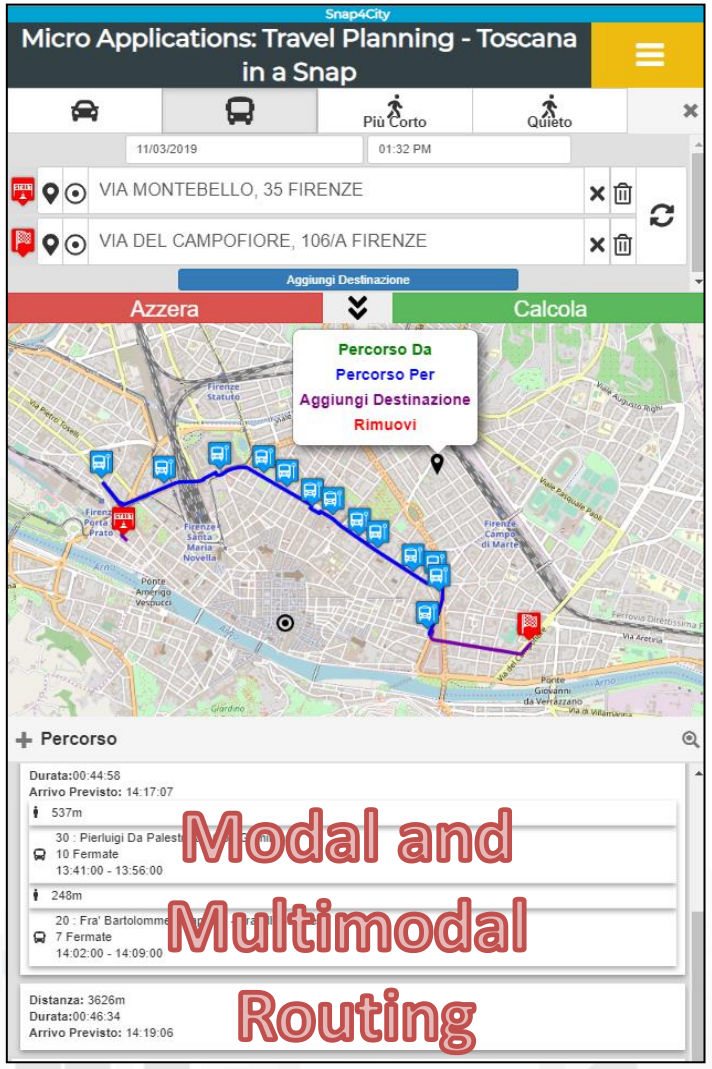
Scarica da App Store

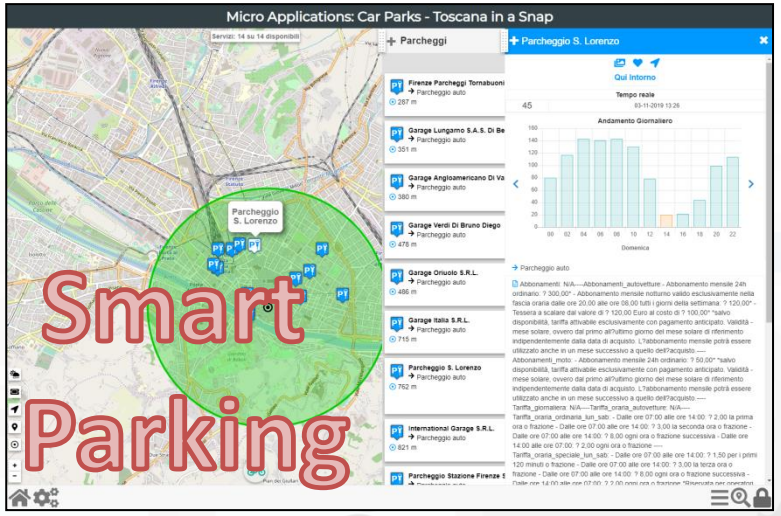
Scarica da Windows Store

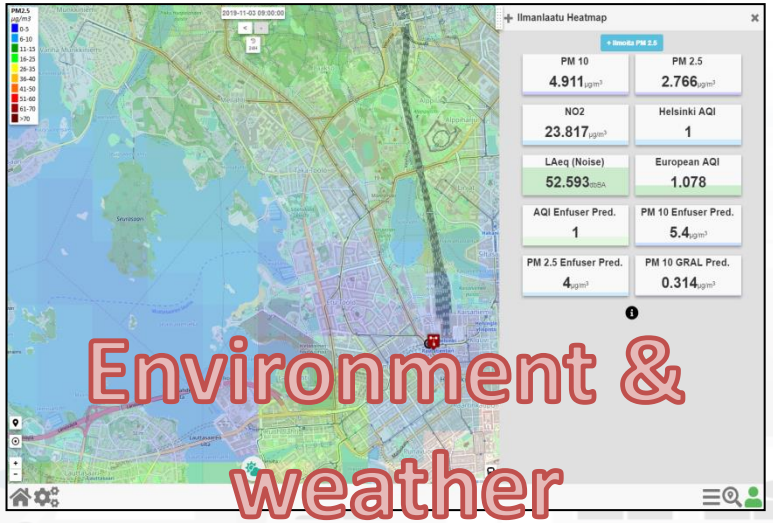
Sii-Mobility

MIcroApplications









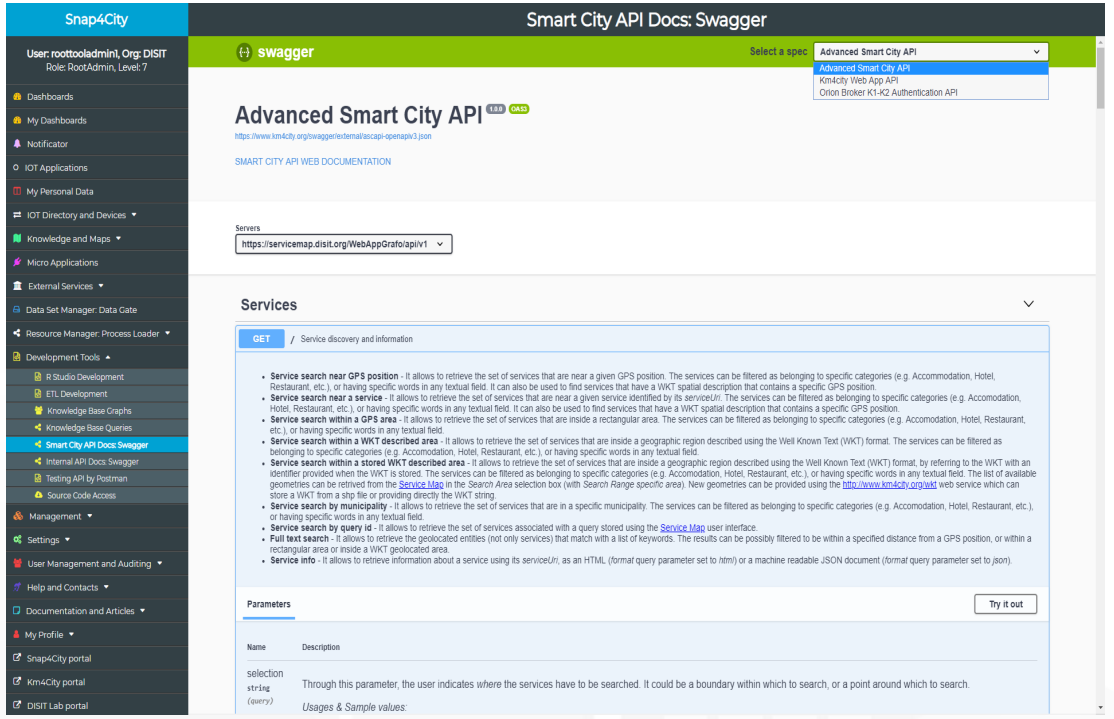
Advanced Devevelopment Kit features

- **Exploiting Km4City Advanced Smart City API**
  - Open Source: GitHub
  - Multiplatform: exploiting Apache Cordova Framework
  - Active since 2015
  - Adopted by a community of several Projects, Cities and SME
- **Respecting user privacy:**
  - Anonymous usage vs Authenticated usage (OAuth, email, ...)
- **Modular & Dynamic:**
  - Loading new modules from the WEB, and/or creating App by modular approach
- **Personalization and Profiling:**
  - Personalized menu, proposed POI for search
- **Reaching City Users:**
  - Alerting and notifications by location, by user behaviour

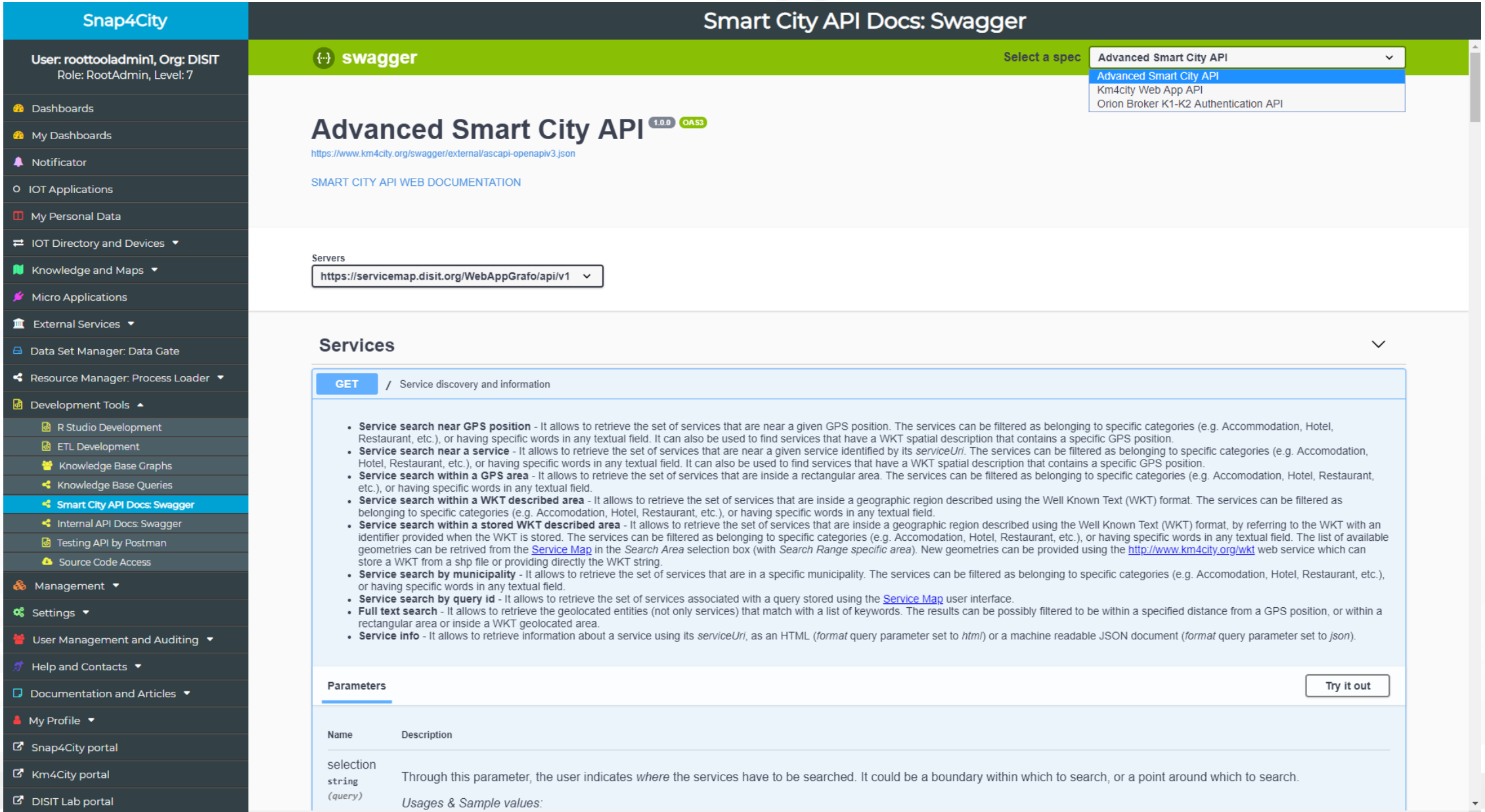
Advanced SmartCity API

- Search data: by text, near, along, etc.
  - Resolving text to GPS and formal city nodes model
- Empowering city users: contributions, suggestions, forum discussions, etc.
- Events: Entertainment, critical and mobility
- Public and Private Mobility & Transport, and predictions
- POIs, Cultural and Touristic info
- Health services and predictions
- Environmental information, heatmaps; values
- Profiled Suggestions to City Users
- Traffic flow reconstruction
- Personal Assistant: PAVAL
- User Engagement: goal experiences, and assessment
- *Sharing knowledge among cities → see Knowledge base Management*

Swagger

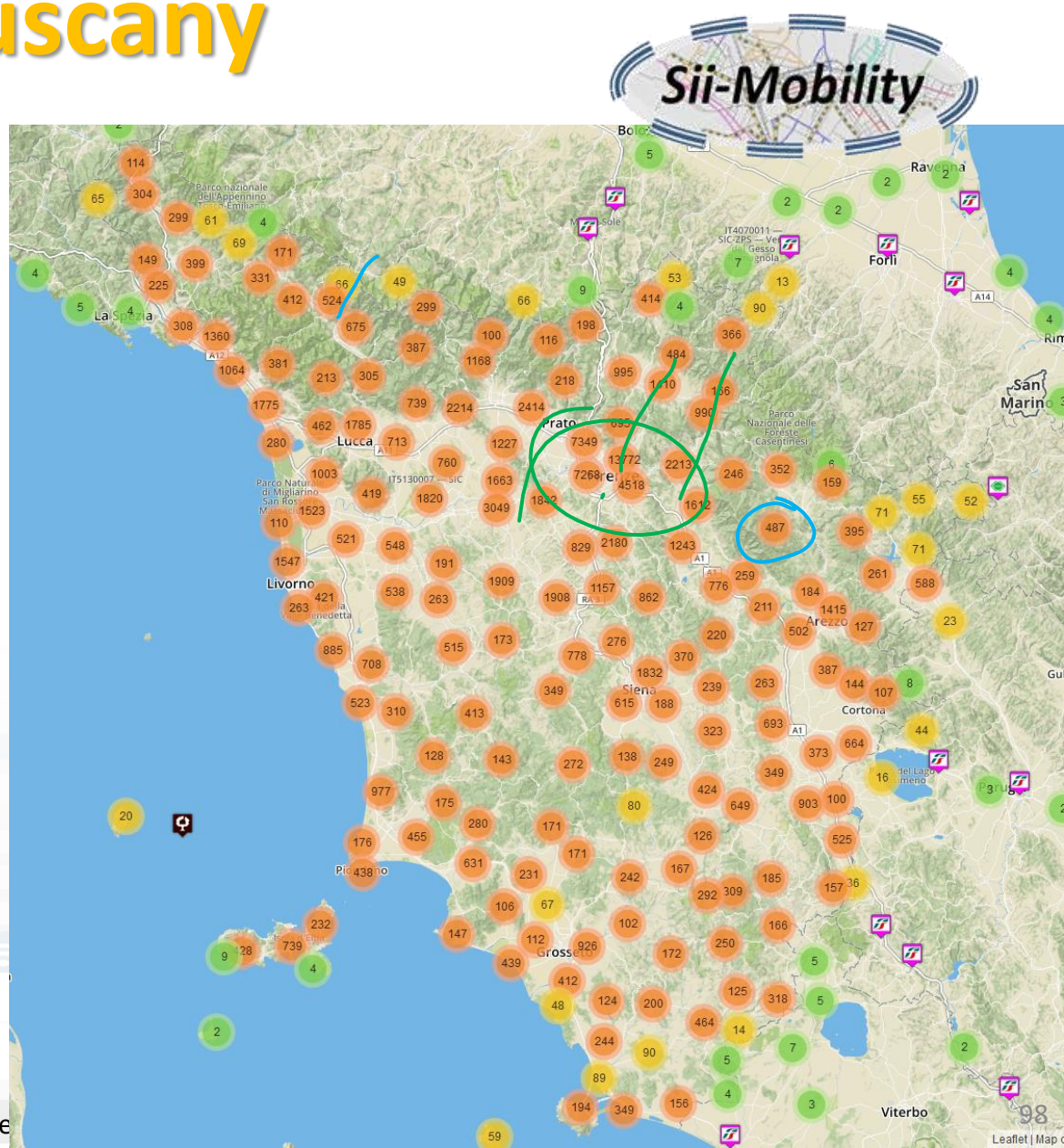


Swagger


Snap4City
Smart City API Docs: Swagger
User: roottooladmin1, Org: DISIT
Role: RootAdmin, Level: 7
Dashboards
My Dashboards
Notificator
IOT Applications
My Personal Data
IOT Directory and Devices
Knowledge and Maps
Micro Applications
External Services
Data Set Manager: Data Gate
Resource Manager: Process Loader
Development Tools
R Studio Development
ETL Development
Knowledge Base Graphs
Knowledge Base Queries
Smart City API Docs: Swagger
Internal API Docs: Swagger
Testing API by Postman
Source Code Access
Management
Settings
User Management and Auditing
Help and Contacts
Documentation and Articles
My Profile
Snap4City portal
Km4City portal
DISIT Lab portal
swagger
Select a spec
Advanced Smart City API
Advanced Smart City API
Km4city Web App API
Orion Broker K1-K2 Authentication API
Advanced Smart City API
1.0.0
OAS3
https://www.km4city.org/swagger/external/ascapi-openapiv3.json
SMART CITY API WEB DOCUMENTATION
Servers
https://servicemap.disit.org/WebAppGrafo/api/v1
Services
GET
/ Service discovery and information
Service search near GPS position - It allows to retrieve the set of services that are near a given GPS position. The services can be filtered as belonging to specific categories (e.g. Accommodation, Hotel, Restaurant, etc.), or having specific words in any textual field. It can also be used to find services that have a WKT spatial description that contains a specific GPS position.
Service search near a service - It allows to retrieve the set of services that are near a given service identified by its serviceUri. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant, etc.), or having specific words in any textual field. It can also be used to find services that have a WKT spatial description that contains a specific GPS position.
Service search within a GPS area - It allows to retrieve the set of services that are inside a rectangular area. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant, etc.), or having specific words in any textual field.
Service search within a WKT described area - It allows to retrieve the set of services that are inside a geographic region described using the Well Known Text (WKT) format. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant, etc.), or having specific words in any textual field.
Service search within a stored WKT described area - It allows to retrieve the set of services that are inside a geographic region described using the Well Known Text (WKT) format, by referring to the WKT with an identifier provided when the WKT is stored. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant, etc.), or having specific words in any textual field. The list of available geometries can be retrived from the Service Map in the Search Area selection box (with Search Range specific area). New geometries can be provided using the http://www.km4city.org/wkt web service which can store a WKT from a shp file or providing directly the WKT string.
Service search by municipality - It allows to retrieve the set of services that are in a specific municipality. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant, etc.), or having specific words in any textual field.
Service search by query id - It allows to retrieve the set of services associated with a query stored using the Service Map user interface.
Full text search - It allows to retrieve the geolocated entities (not only services) that match with a list of keywords. The results can be possibly filtered to be within a specified distance from a GPS position, or within a rectangular area or inside a WKT geolocated area.
Service info - It allows to retrieve information about a service using its serviceUri, as an HTML (format query parameter set to html) or a machine readable JSON document (format query parameter set to json).
Parameters
Try it out
Name
Description
selection
string
(query)
Through this parameter, the user indicates where the services have to be searched. It could be a boundary within which to search, or a point around which to search.
Usages & Sample values:

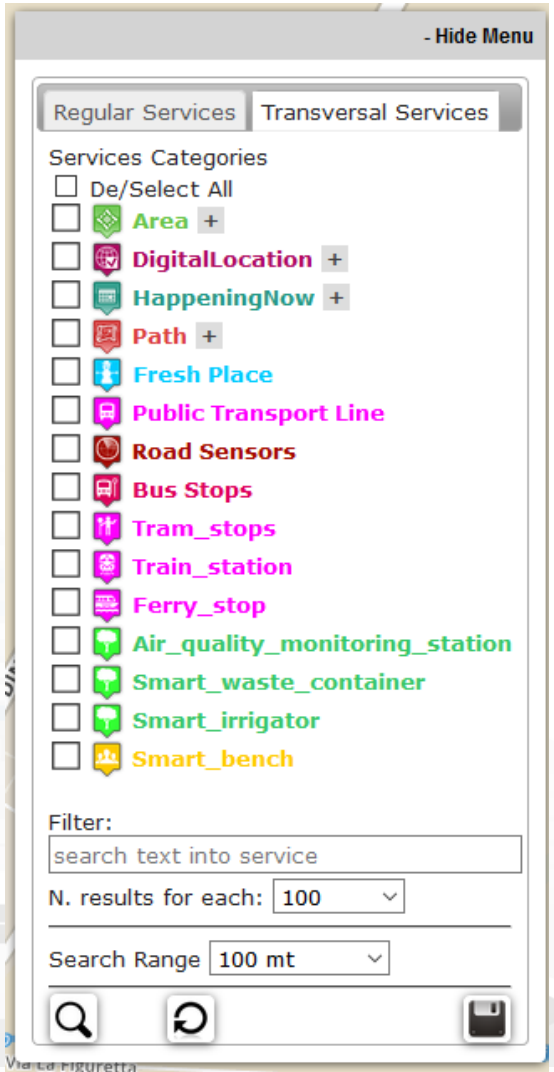
Thematic Data Domain Tuscany

- **Street and geoinformation of the territory and details for routing, navigation, …**
- **GeoResolution, Environmental data**
- **Mobility and Transport**: public and private, public transport, parking status, fuel stations prices, traffic sensors, etc.
- **Culture and Tourism**: POI, churches, museum, schools, university, theatres, events in Florence
- **Environmental**: pollution real time, weather forecast, etc.
  - Environmental data geo resolution
- **Social Media**: twitter data
- **Health**: hospital, pharmacies, status of the first aid triage in major hospitals, …
- **Alarms**: civil protection alerts, hot areas, …



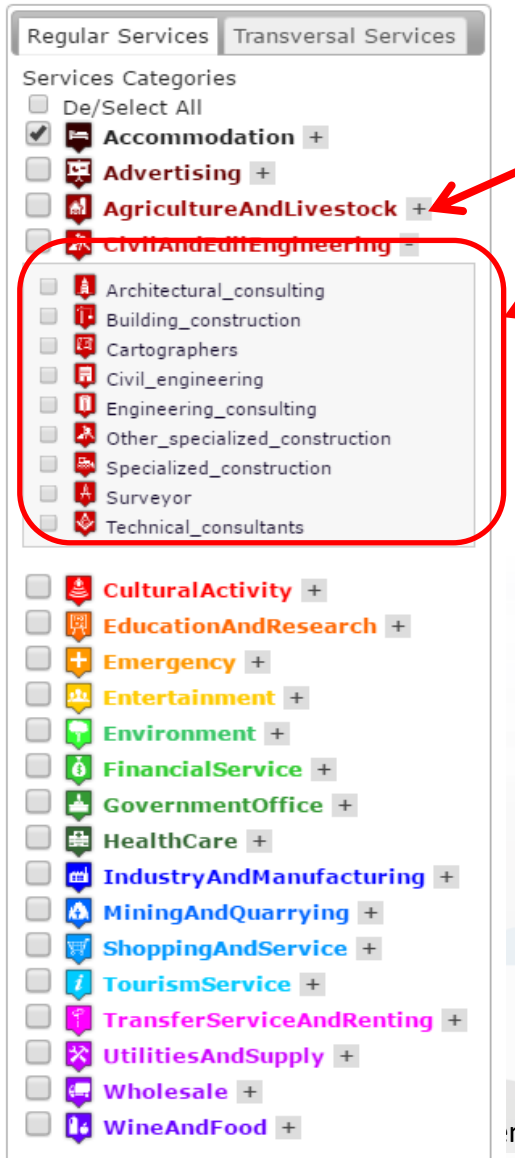
Access to Point of Interest information, POI

- **POI**: point of interest
- **type**: macro and subcategories
- **Position**: GPS, address, telephone, fax, email, URL, …
- **Description**: textual, multilingual, with images, …
- **Link** to dbPedia, Linked Open Data
- **Links to other services**
- **Real time data if any:** sensors data, timeline, events, prices, opening time, rules of access, status of services, status of queue, etc..
- *See transversal services on ServiceMap*
  - *Regular and in test platform*



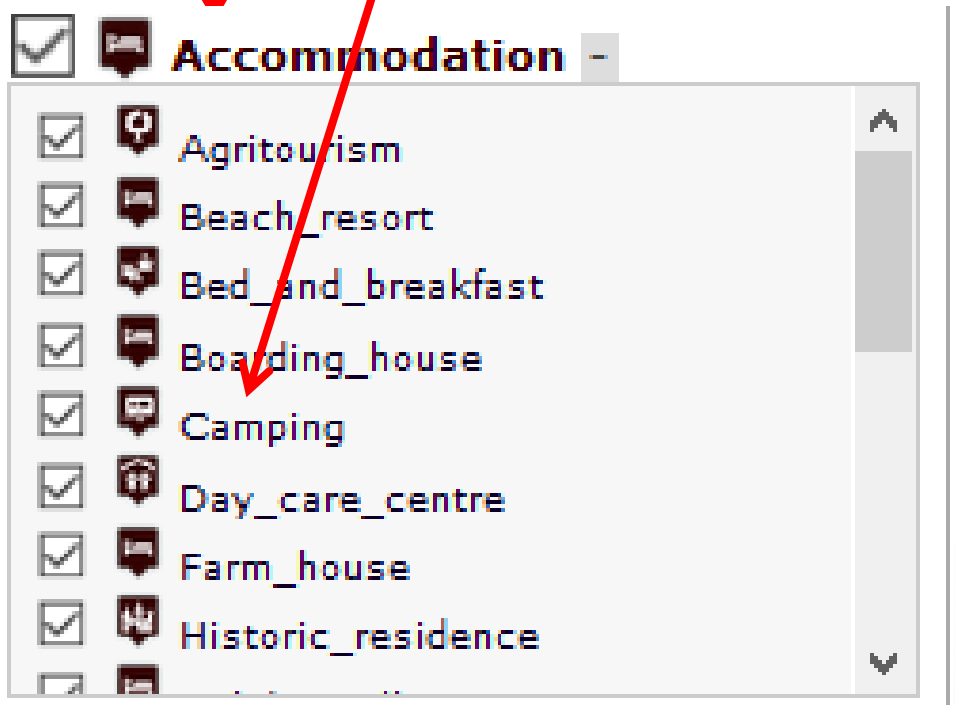
Concepts of Services: Macro and subcathegory

A SKOS area into the Km4CIty Ontology and Knowledge base for modeling POI and any element on map

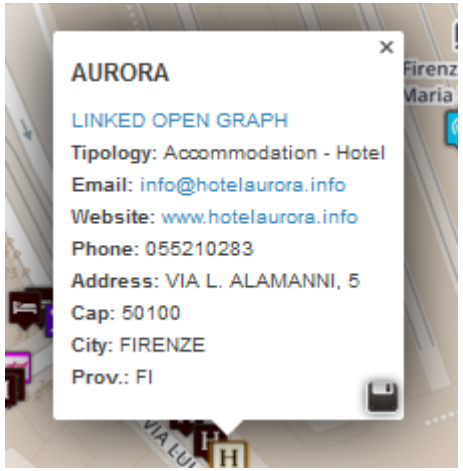


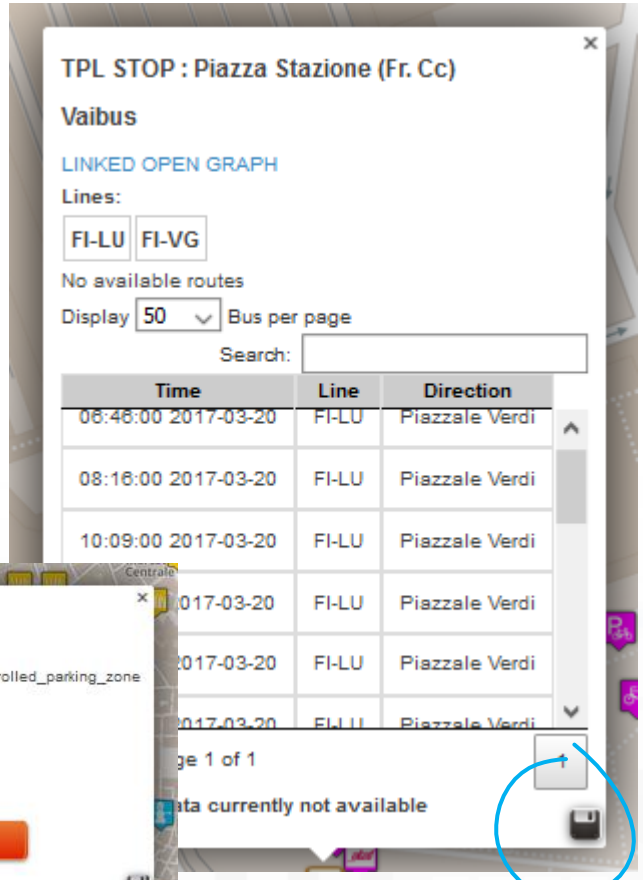
20 Service Macro Classes (nature)

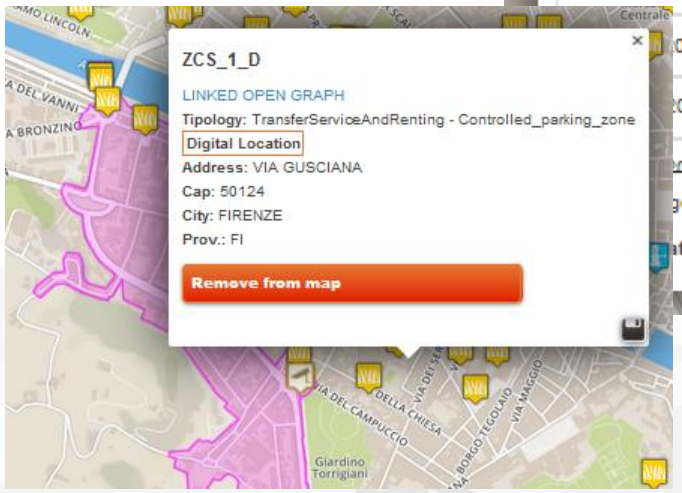
Service subClasses, (subnature)

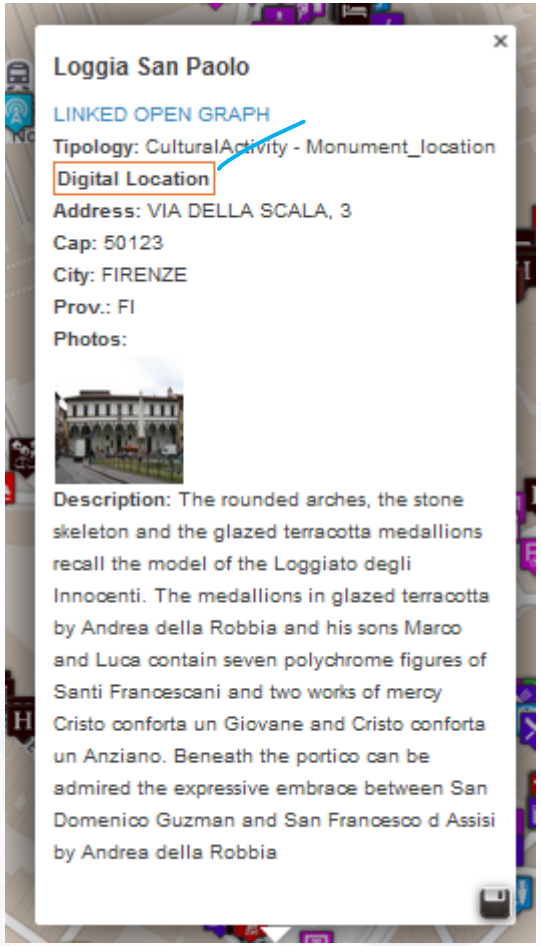


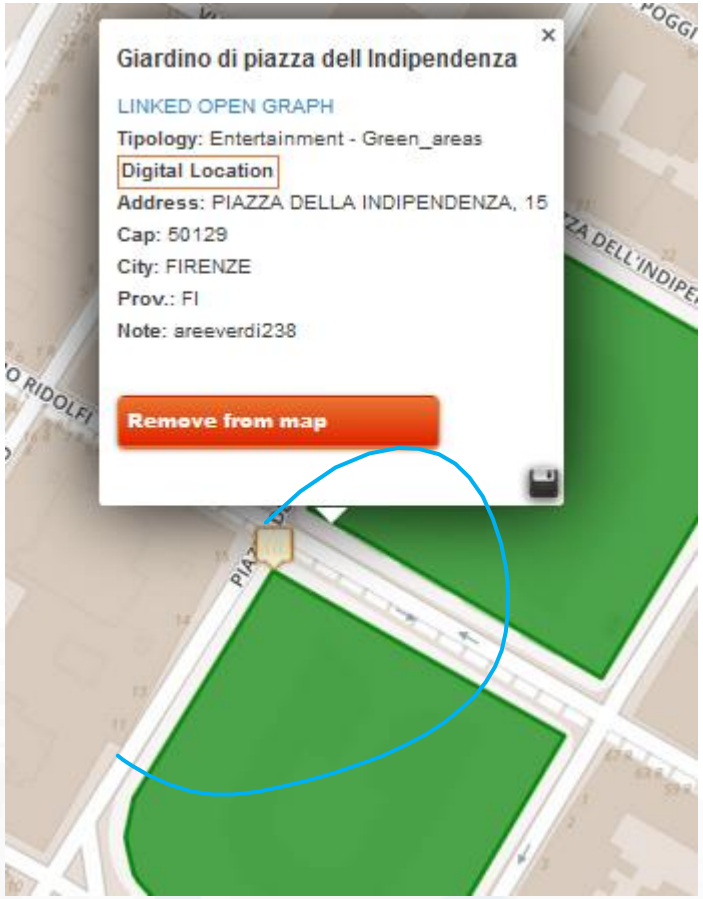
Service Information: different kinds of services











General Text Search Features

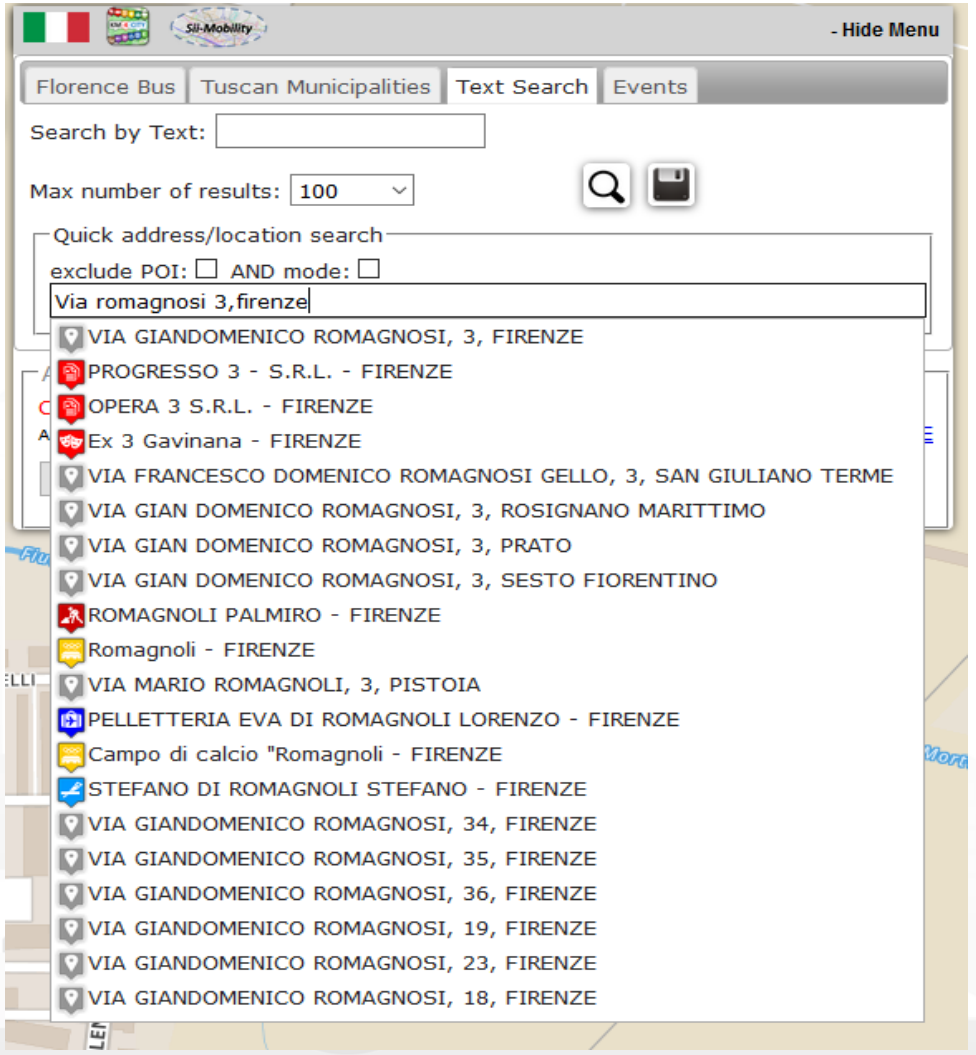
Search by text for POIs via:

- Full text: description, title, macro and category name
- Filtering by macro-cat and subcategory
- Filtering on distance and geometric shape

Search by text with assisted suggestion to get:

- Streets and civic numbers, or POI, locations

Geo resolution, from point to street; from civic to GPS, etc.

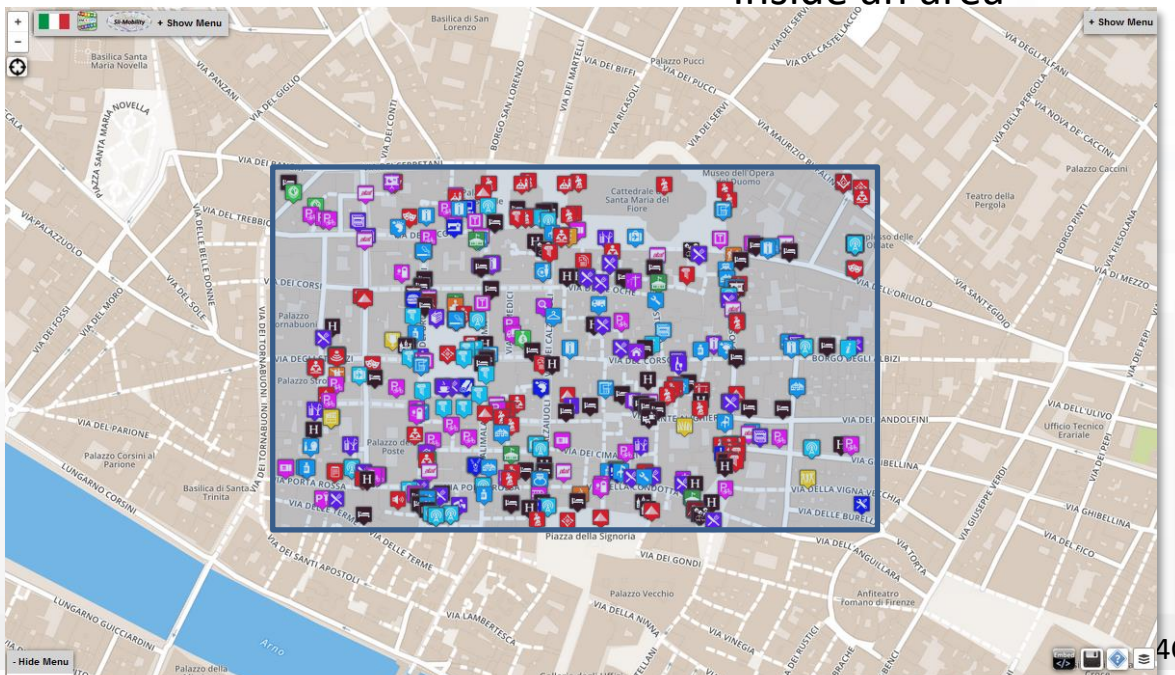


Search by Shape and Distance

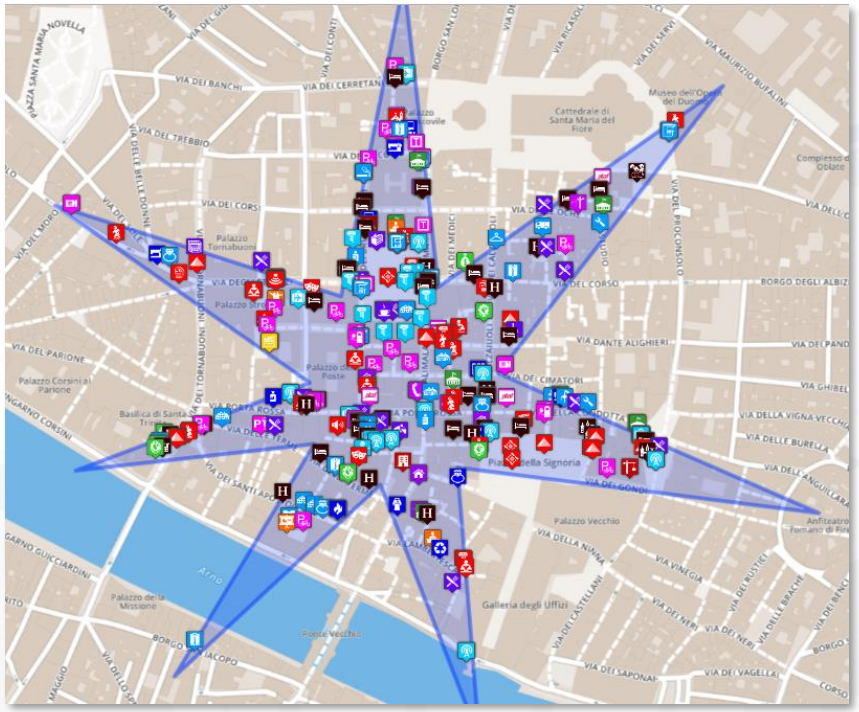
Around a point or POI

Each request or search in the Km4City model can be referred to a point and a ray, to an area, to a polyline

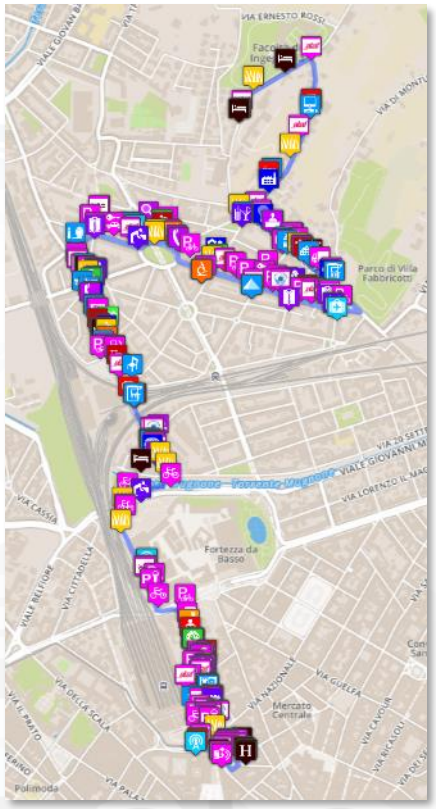
Inside an area



Inside a closed polyline



Along a polyline

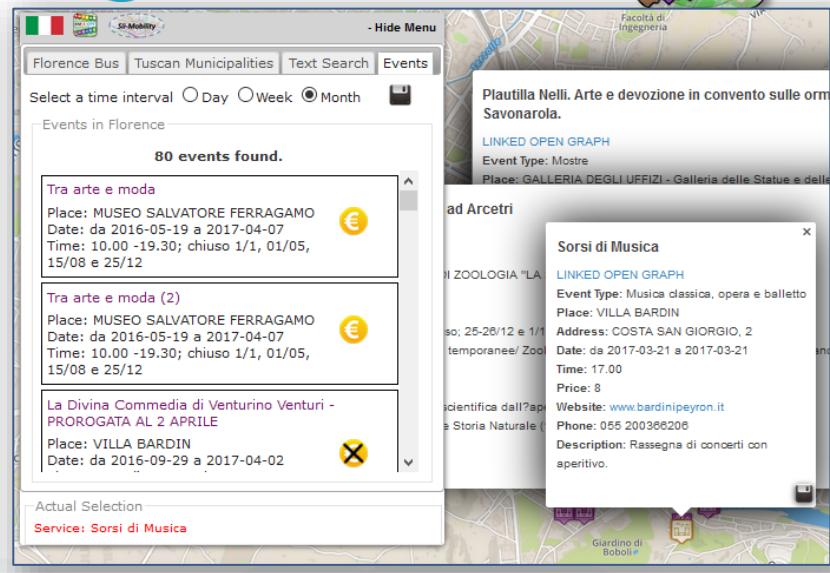


Empowering City Users

- Allow city users to
  - provide comments, images and scores associated with a certain Service (or place, via GPS), discussions on forums, etc.
  - Get list of last contributions of the same kind provided by other users
  - Save favorites
  - Share trajectories,
  - Save and Manage their own data, IOT data, etc.
- Contributions can be:
  - used as feedbacks
  - moderated by a back-office personnel
  - …
- Connection with powerful servers based on 311 standard it also possible

Access to Event information

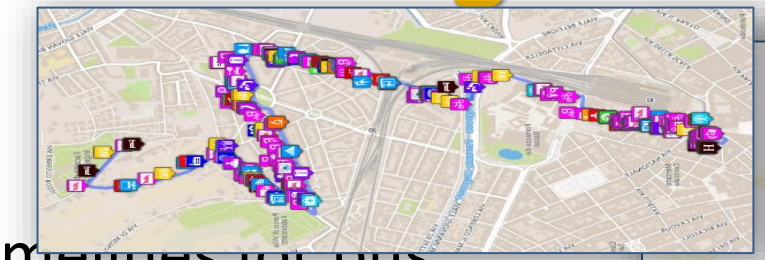
- **Getting Traffic Events: ESB, etc.**
- **Getting Critical Events: CAP standard**
- **Getting Police events**
- **Getting Entertainment Events** in the city
  - Theater, museum, show, sport, etc.
  - **Getting Event details**
    - Event kind, and thus ordering
    - in the day, week, and month
    - Location, and thus ordering, or selecting events per area, per residence
    - General information
    - Opening and cost (if any)
    - Etc.

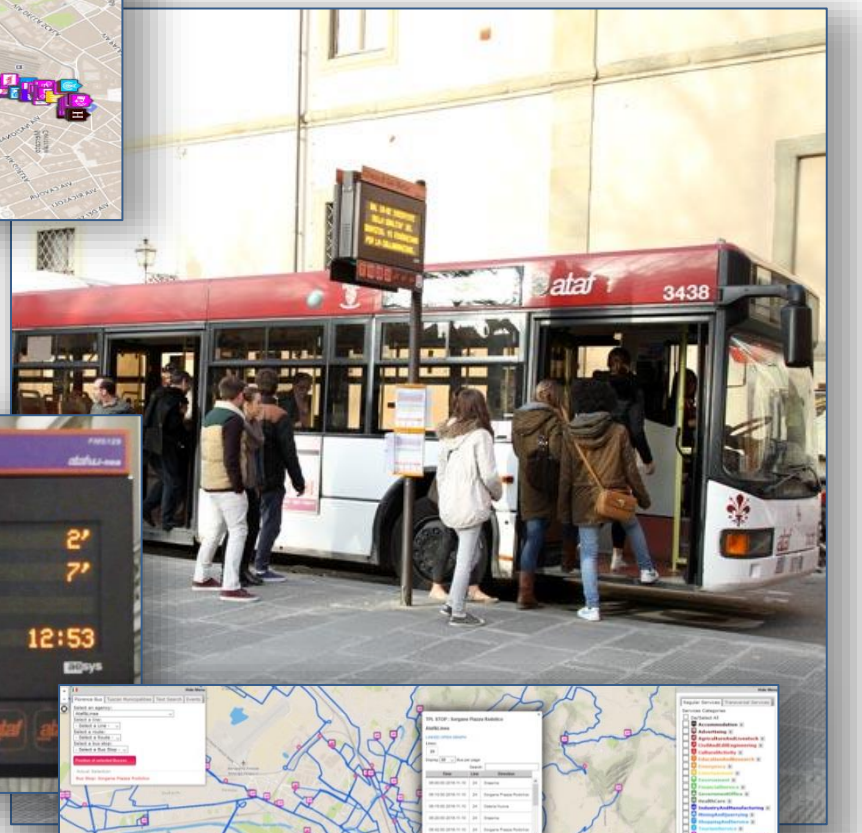


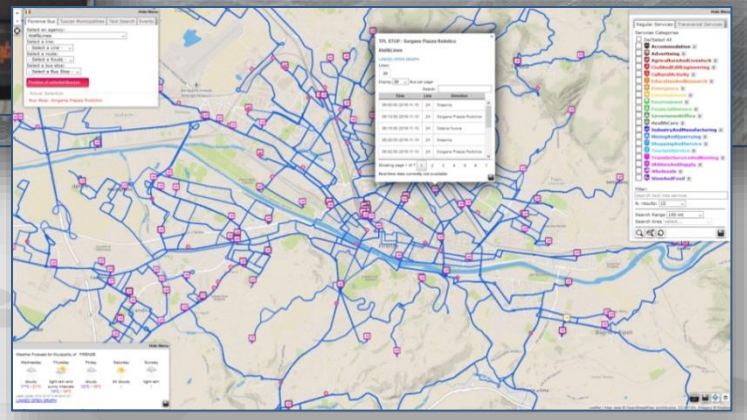
Supporting City Users in using Public Mobility

Public Transportation, PT

- Getting tickets
- Getting bus stops, lines, and timelines for bus, train and tramline (GTFS, ETL, ...)
- Getting Tunnel and Ferry Status
- Searching Services along a Pub. Transport line or closer to a stop
- Searching the closest bus stops
- searching for BUS stops via name
- real time delays of busses
- Modal/multimodal routing for Pub. Transport
- Tracking fleets, trajectories, etc.
- Get connected drive data



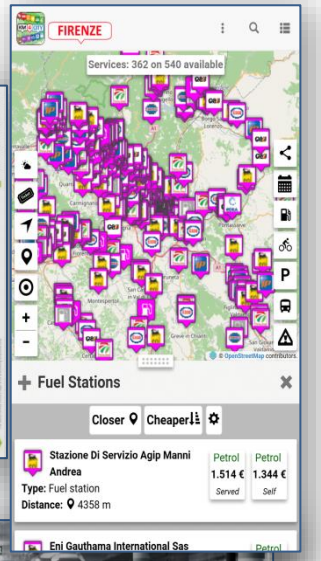


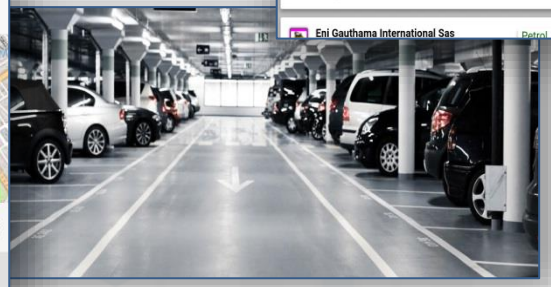


Supporting City Users using Private Mobility

Private Transport

- Parking status (DATEX II, …)
- Saving car park
- Getting **closer parking**
- **OBD2 data from your engine or fleet**
- **Getting parking forecast: short and long term**
- Getting **closer free space on parking**
- Getting **fuel stations** location and fuel product prices
- Getting bike sharing rack status
- Searching Services along a **path** or closer to a point or Service as Hotel, Restaurants, square, etc.
- Getting closer **cycling paths**
- Recharging stations: location and status
- Getting traffic information
- Heatmap where is safer to bike





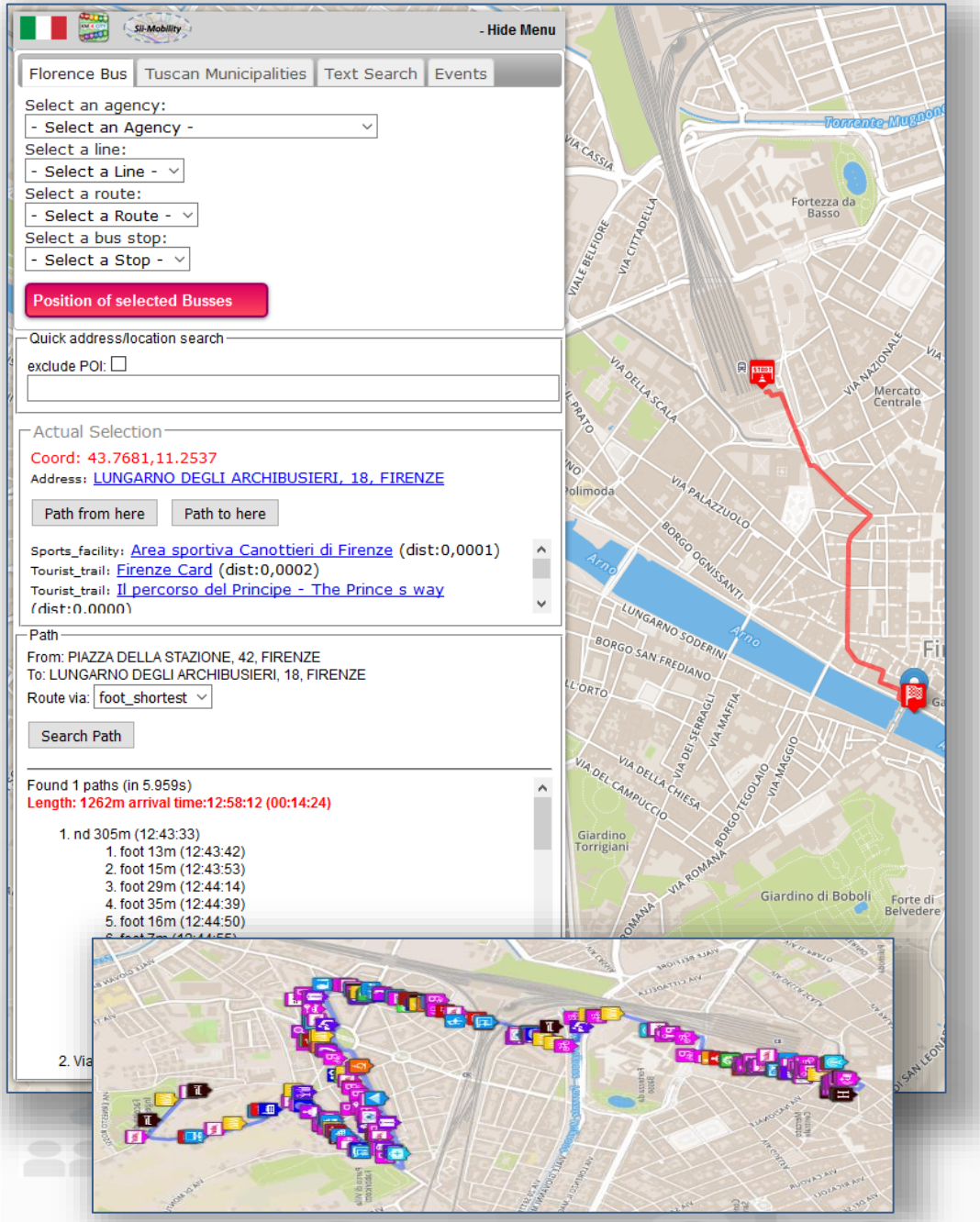
Private Mobility: routing and navigation paths

To get the path from two points/POIs:

- Shortest for pedestrian
- Quietest for pedestrian
- Shortest for private vehicles
- Multimodal with Public Transportation
- Constrained routing

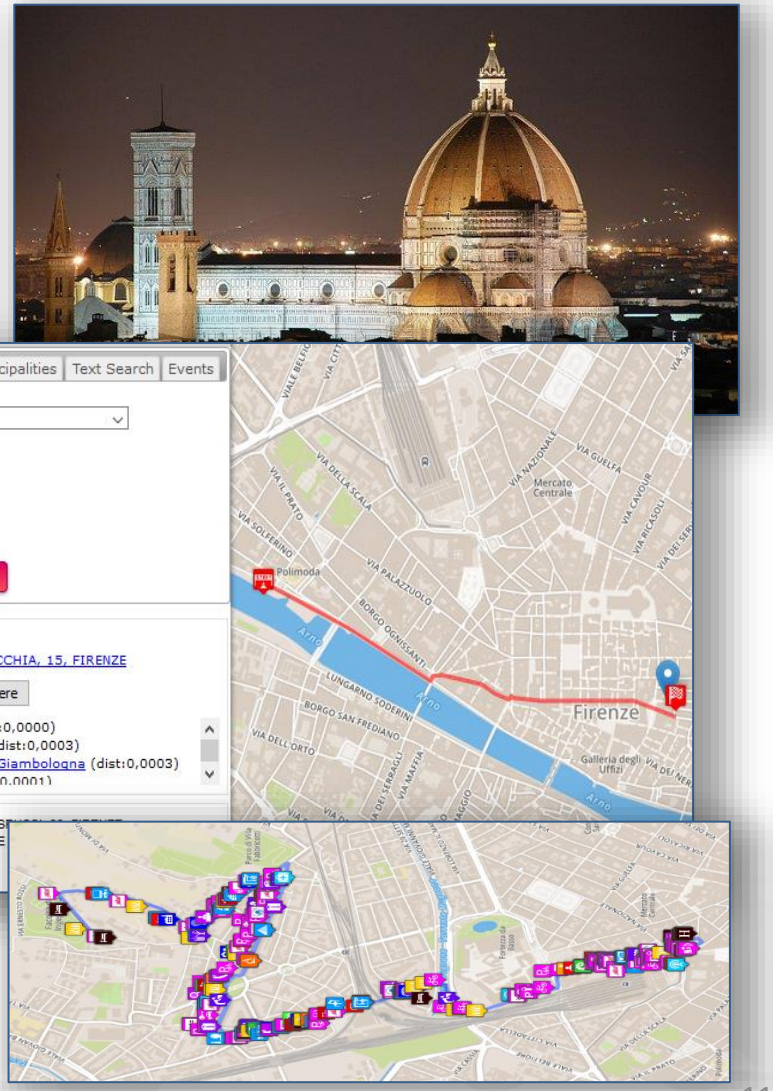
Search for POIs along the identified Path!

http://www.disit.org/ServiceMap



New Experience to access at Cultural and Touristic info

- Getting location and description of Point of Interests, POIs: culture and tourism first
  - Location, images, phone, URL, etc.
  - Get image, video, audio, …
- Search for POIs in areas and closer
- Get routing to reach location or POI by walking downtown
  - searching Services along the path
- Search for location, full text assisted
- Leave a score, take a picture, etc.

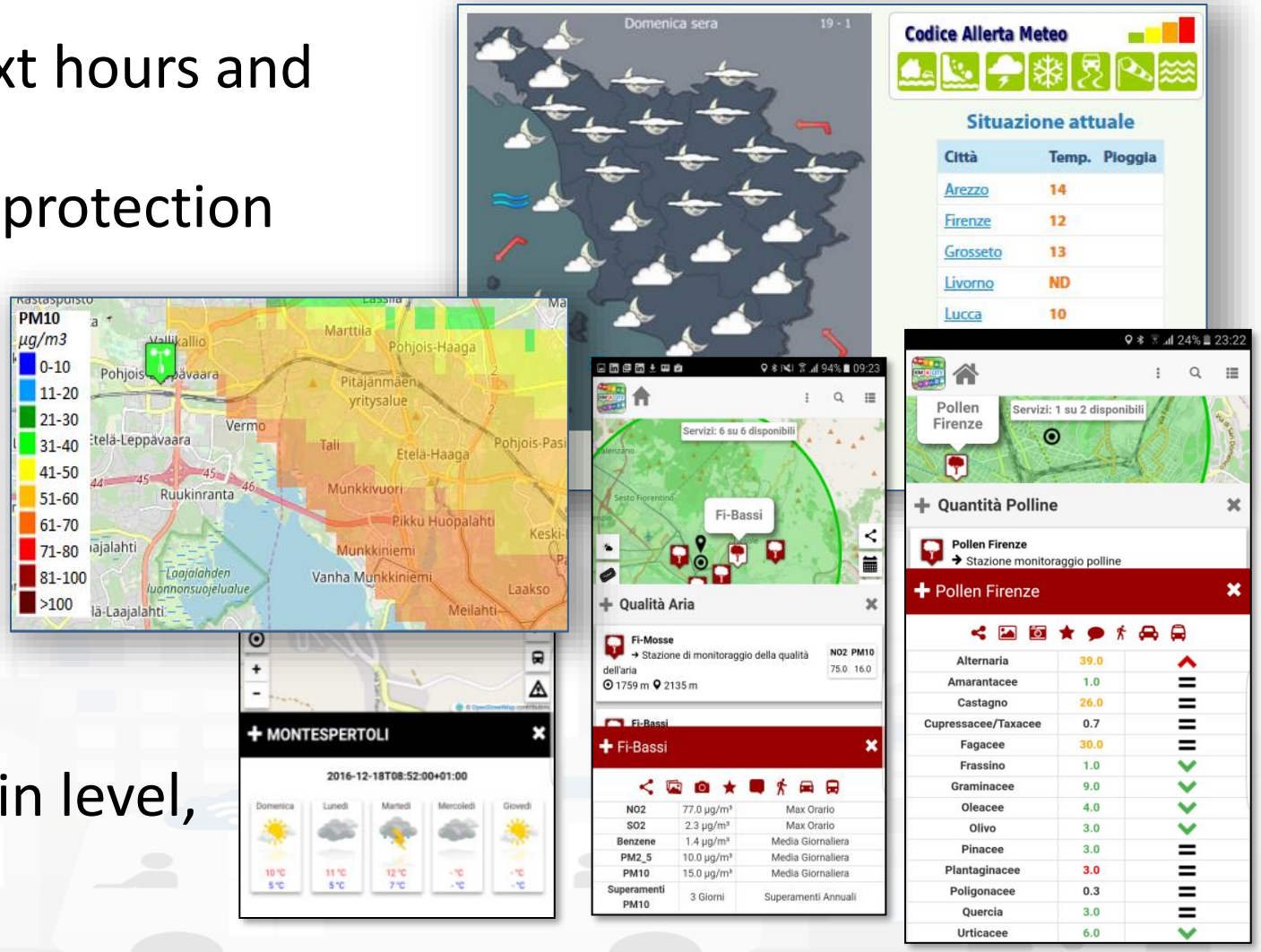


New way to access at health services

- Searching for pharmacies and hospitals
- Getting the closest hospital first aid locations and status
- Getting real time updated information about the first aid status of major hospitals (triage)

Access at Environmental information

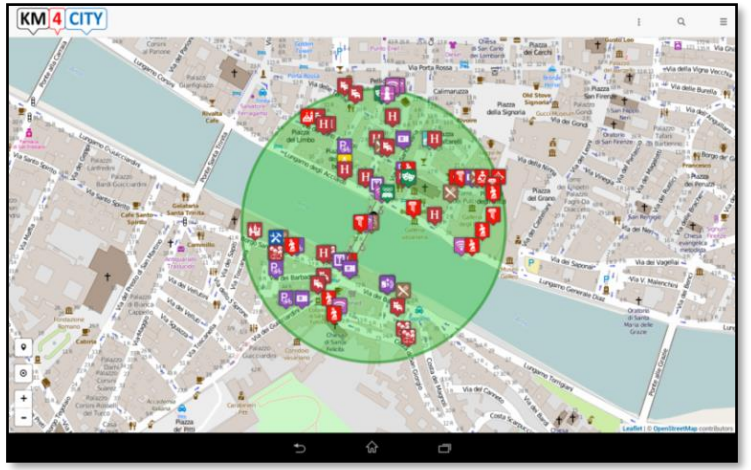
- Getting weather forecast for the next hours and days
- Getting alert information from Civil protection
- Getting air quality status
- Getting Air quality via heatmaps, heatmap animation
- Computing Air quality indexes
- Computing Air quality predictions
- Getting pollination status
- getting actual weather status: temperature, humidity, pressure, rain level,
- etc.

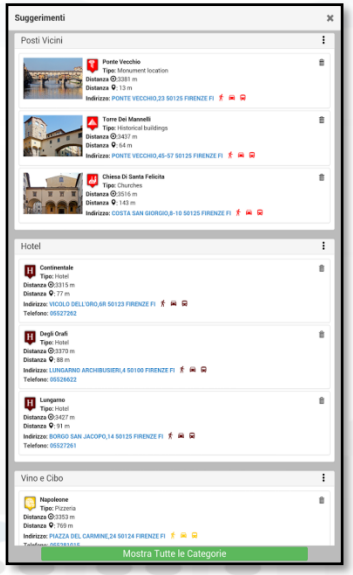


Profiled Suggestions to City Users

Personalized suggestions

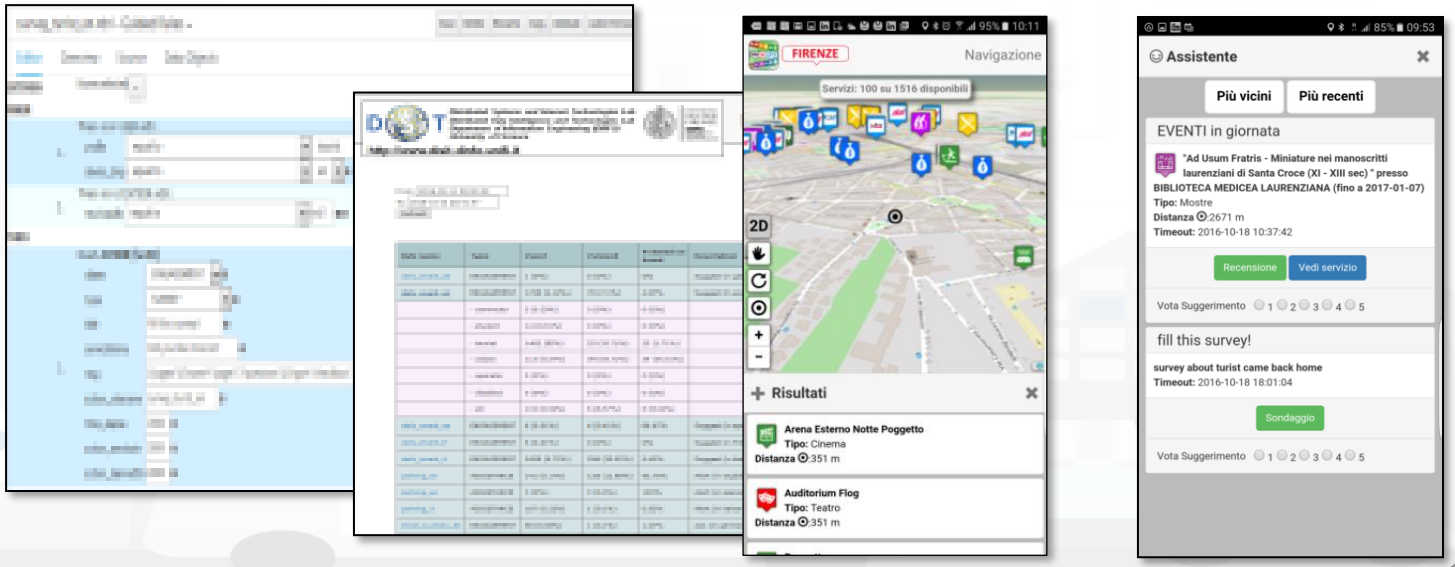
- The server provide suggestions in the user context (location and time) arranged in a number of categories
  - Culture, mobility, food and drink, etc.
  - Alerts: civil protection, city council, twitter data, etc.
- The city user may reject some of them, thus the suggestion engine learns about preferred topics and category





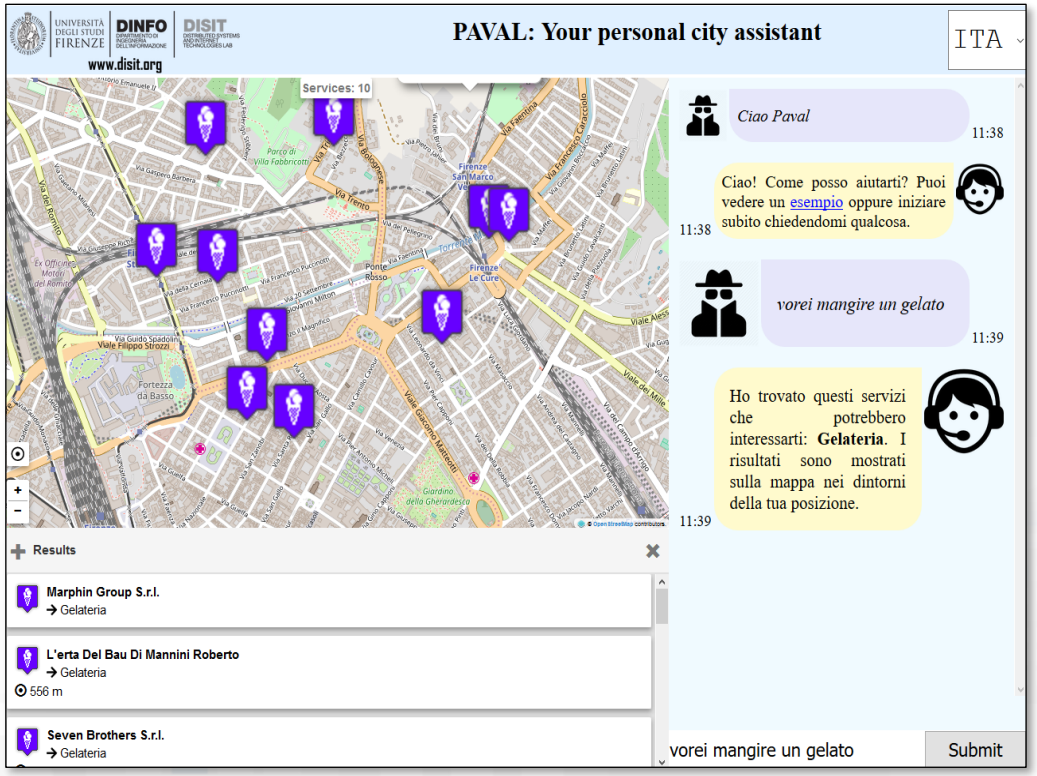
Profiled Engagements to City Users

- The user are profiled to learn habits:
  - Personal POI and paths
  - Mobility habits
- Information and engagements sent to the city users are programmed according to the user evolution to:
  - Stimulate virtuous habits
  - More sustainable habits
  - More healthy habits, etc.
  - Get feedbacks
  - Provide bonus and prices, …
  - Send alerts, …



PAVAL: Personal Assistant

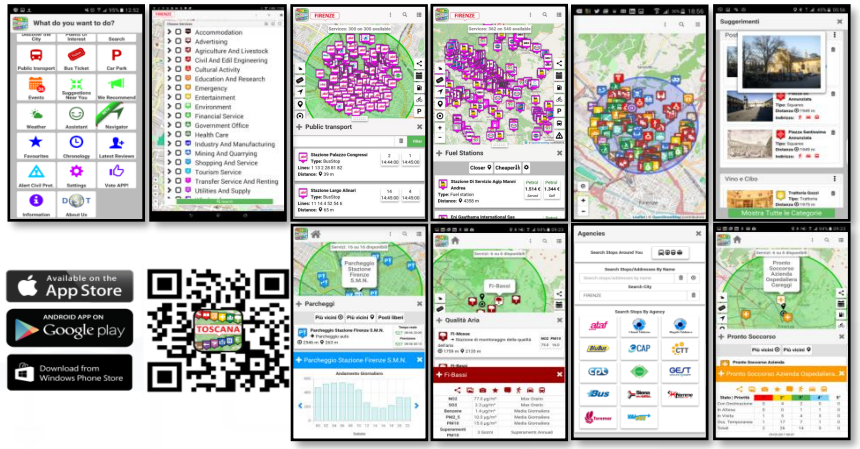
- **Your Personal Assistant** for navigating in the city
- Ask PAVAL to get help and information about the city services
- ITA, ENG
- Active on Florence and whole Tuscany
- Mobile and PC



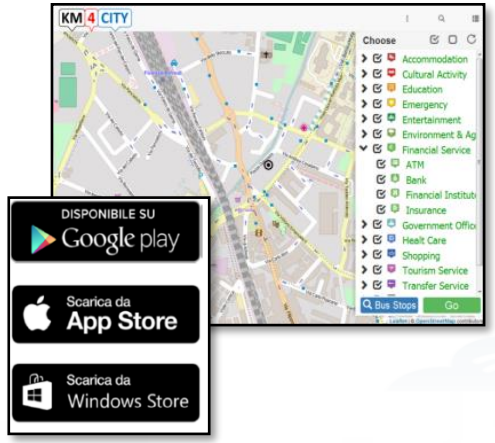
https://assistant.disit.org

Developing Web and Mobile Apps, MicroApps,..

Mobile Apps



Web App HTML5, MicroApplications



Embed into Web pages

City User

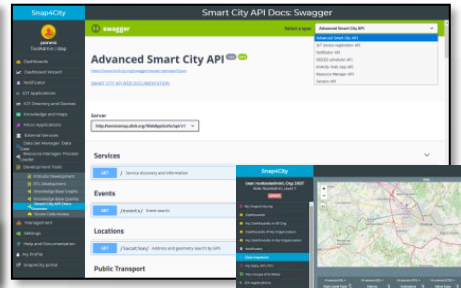
Advanced Smart City API

Mobile Application Monitoring Administrator

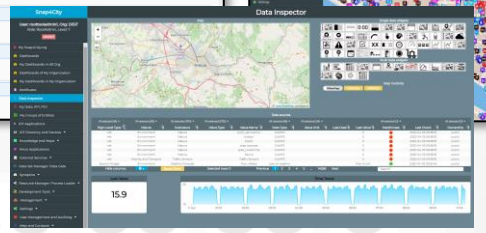
Knowledge Base, Km4City

GitHub

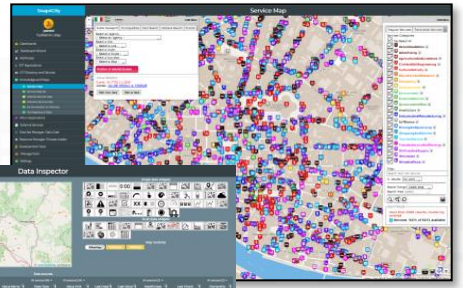
Snap/Km4City Open Source development tool kit



Swagger



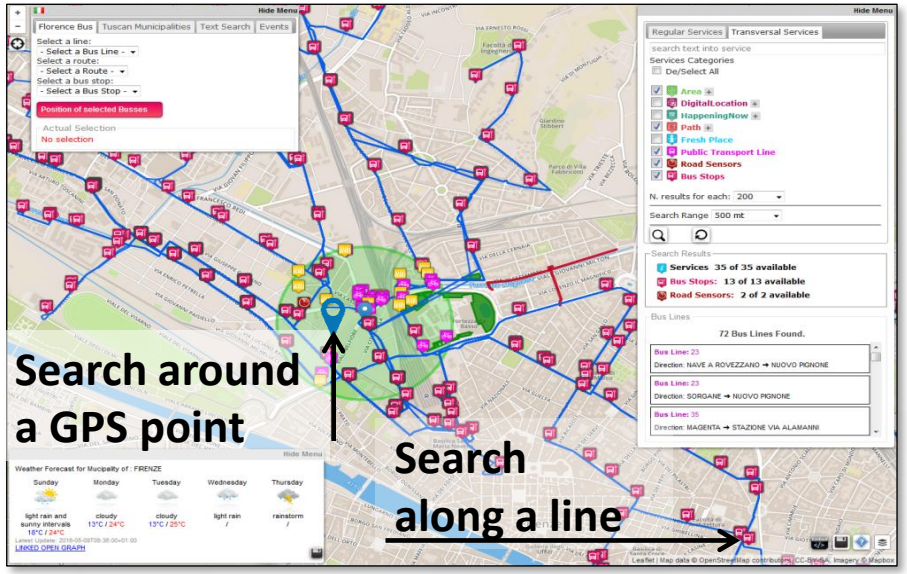
DataInspector



ServiceMap

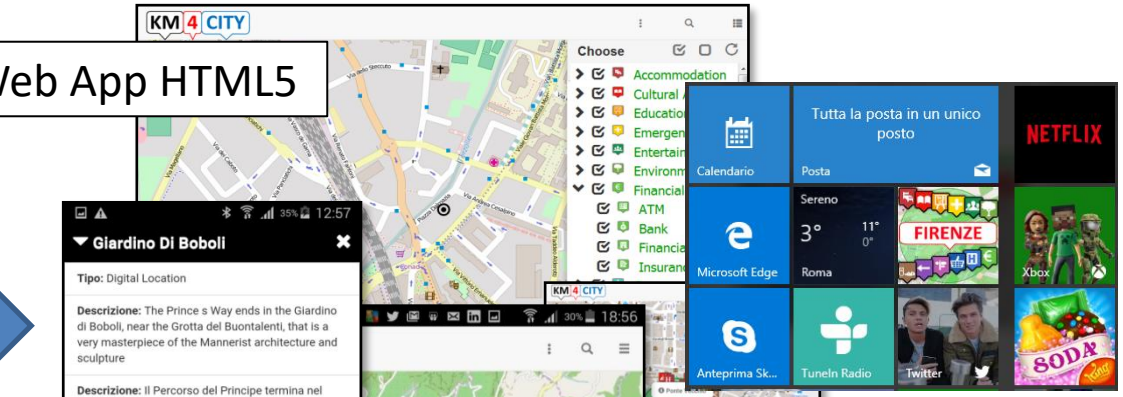
Developer

ServiceMap Dev Tool (knowledge & Map tool)



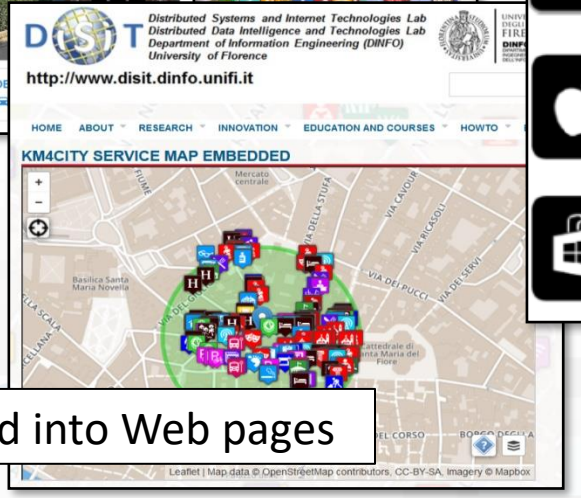
Smart City API call generation

Web App HTML5



Mobile Apps

Embed into Web pages



http://www.disit.org/6873

Smart City API

- based on Km4City engine on the back
- documented: https://www.snap4city.org/404
- **ServiceMap** tool to generate visually calls to exploit the Smart City API in web and mobile applications
- **Documentation and examples**:
  - **TC5.15 - Snap4City Smart City API Collection and overview, real time**
  - **ServiceMap and ServiceMap3D, Knowledge Model, Km4City Ontology**
  - **Knowledge Base Graphs and Queries: browsing and queries into the KB**
- **The Alternatives**:
  - just Dashboards directly exploiting data on graphics and/or
  - IOT Applications via Node-RED exploiting MicroServices also using the Smart City APIs

TOP

Federated Knowledge Bases and Smart City APIs

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

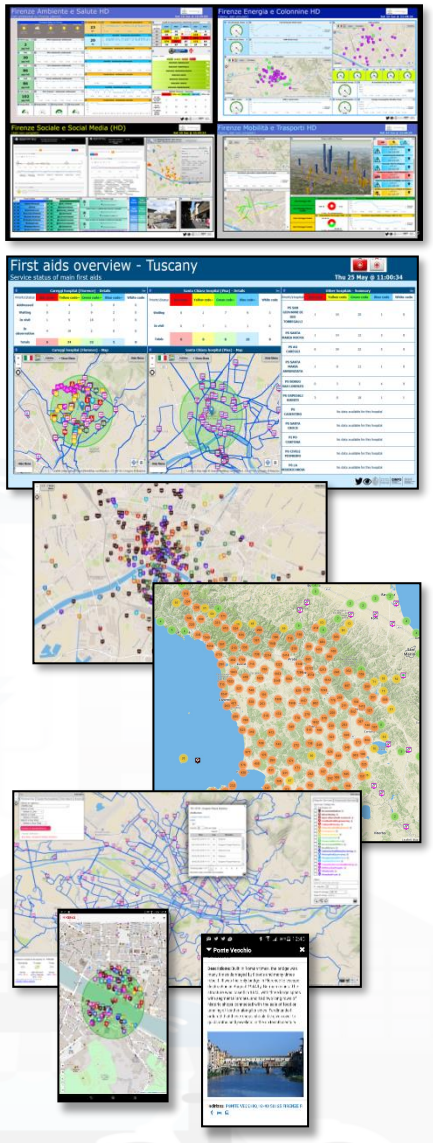
Km4City Federation

At different levels:

- Among cities/regions
- Among data providers
- Among Operators

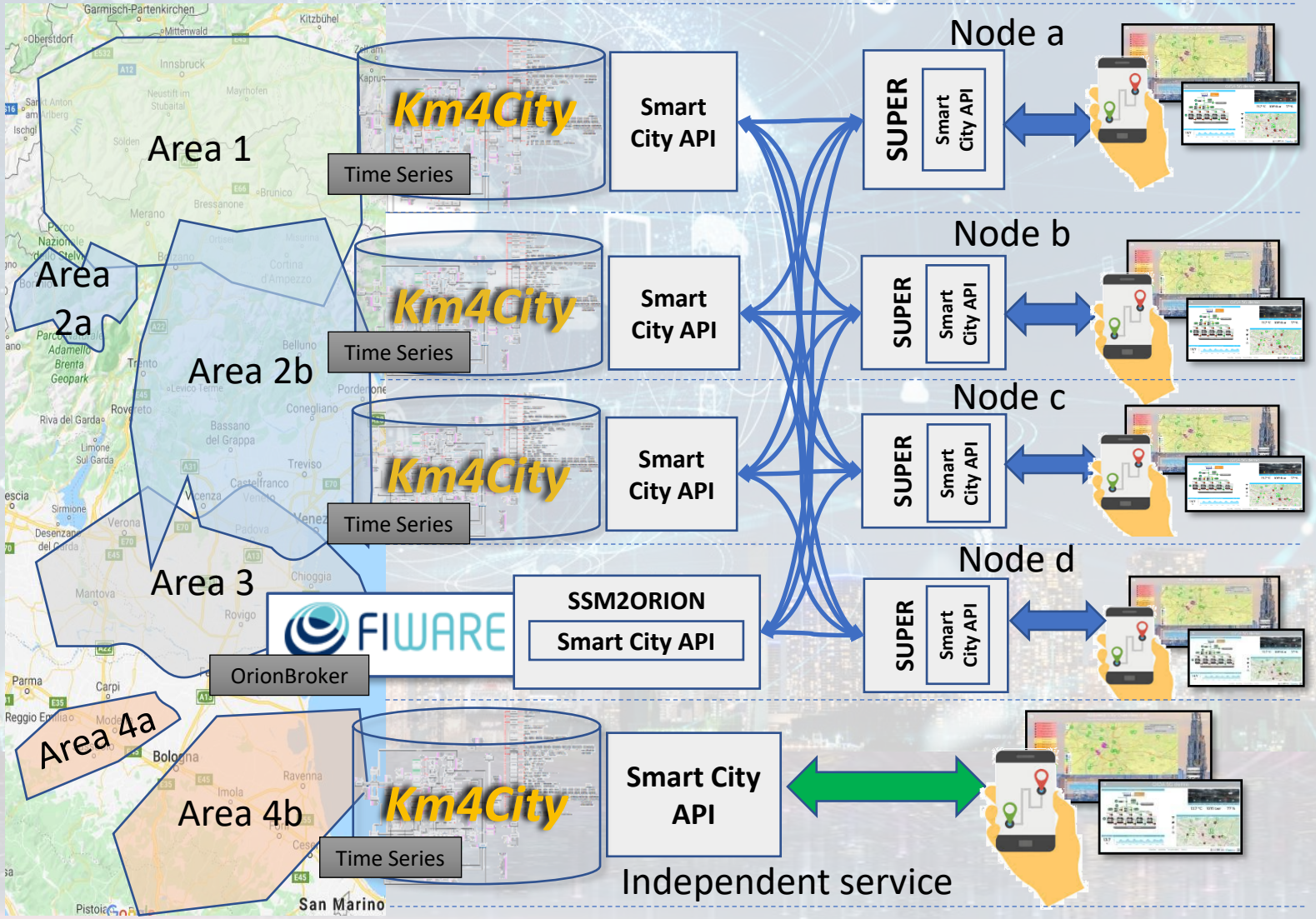
By Means of:

- Smart City API → Apps
- Km4City Smart City Ontology
- Dashboards/data analytics



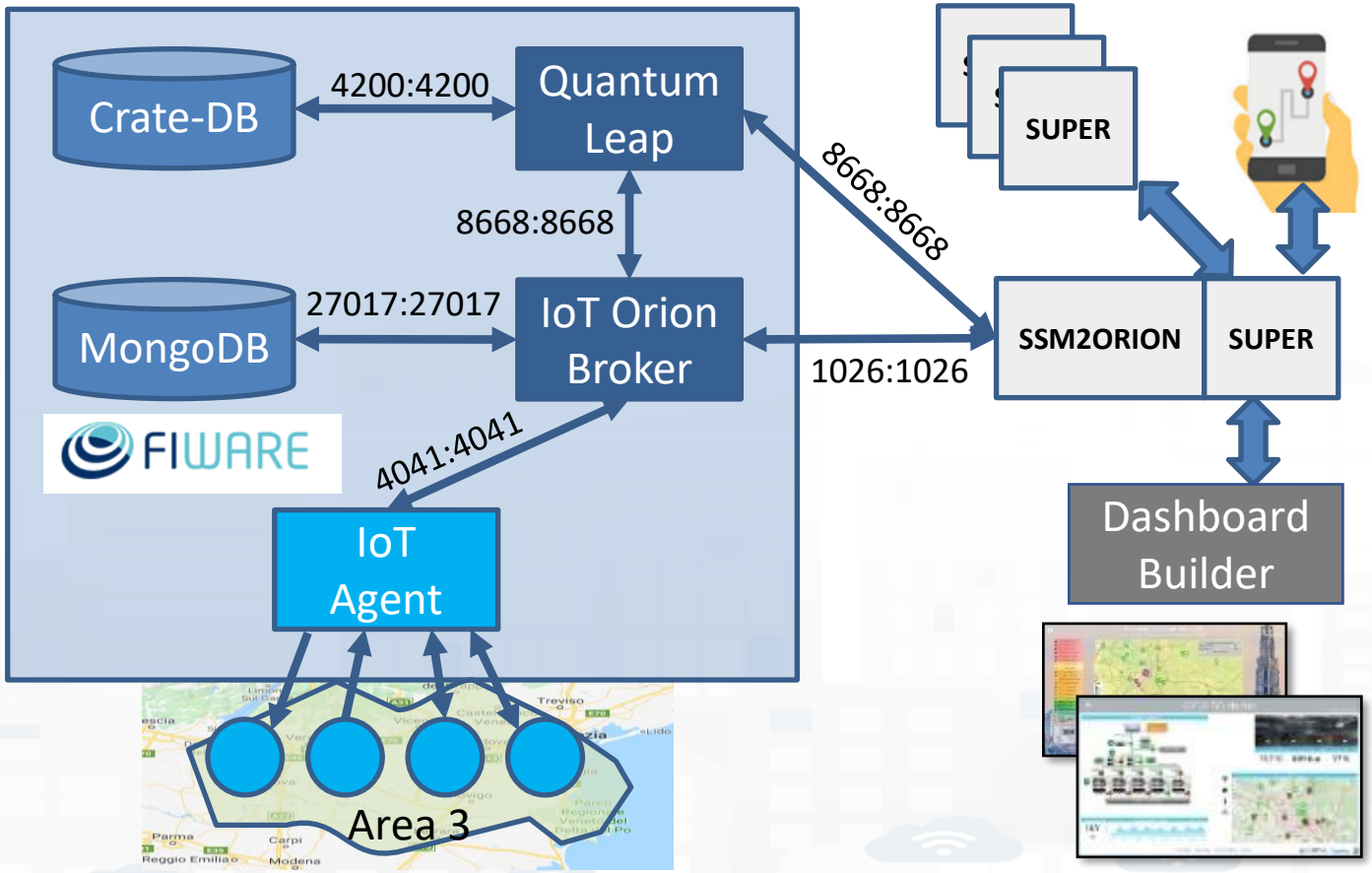
Federation of Smart City Services

Knowledge Base
Semantic Reasoners



- Km4City **Semantic Reasoner**
- **ServiceMap** interoperability
- **Seamless for multiple Mobile Apps**
- **Smart City API**
- **Super**:
  - distributed access and sharing services
  - Each city control its own data
  - Final user can pass from one city / area to another in seamless manner: without changing the mobile Apps

Federation of Snap4City vs IOT ORION Broker



- Super, Nodes and SSM2ORION presents the same Smart City APIs.
- The **network of Super** can be reconfigured dynamically
  - Multiple networks of Super can be realized as well
  - Distributed Searches via the Federation of Super are performed with o(1) complexity
  - Results from an API rest calls are provided in real time also when the size of the network is large
  - Dashboard widgets and Mobile Apps are enabled to use the Super
  - Clients can pass from one Super to another transparently: moving devices
- Nodes
  - do not need to permanently share data
  - data can be of any size, the data shared is typically public since users of different KB are different and not refer to the same LDAP/KeyCloak authentication/authorization service.
  - may have different number of services
  - Services can be based on KB as well as on Brokers
  - Services managed as HLT of: Sensors, Sensor-Actuators, POI.
    - Data of other HLTs are managed independently from the other SmartCity API such as: MyKPI, External Services, WFS GIS, Heatmaps, special tools, etc. etc.
- The solution support disjoined nodes, federation and independent services

Federated ServiceMap and Smart City API

To improve scalability, fault tolerance and federation among cities:

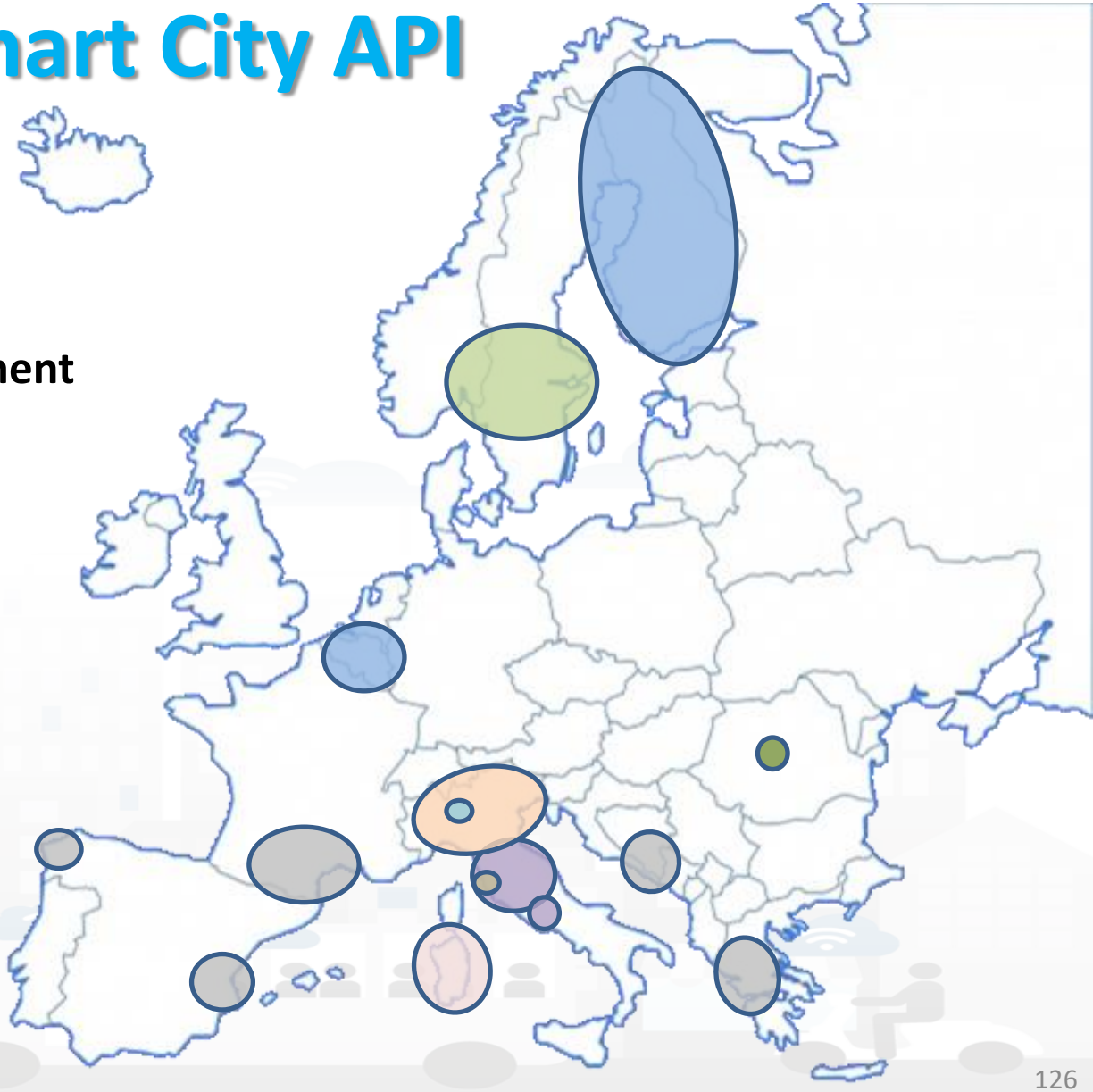
- **One entry point Smart City API for all zones**
- **Multiple Knowledge base See performance assessment**

At different levels:

- Among cities/regions
- Among data providers, Operators

By Means of:

- Smart City API → Apps
- Smart City Ontology
- Dashboards/data analytics
- Organization independent
- CKAN via harvesting



Exposing Services

- **Advanced Smart City API which can be confined into a single Smart City installation or Federated as well as for Super Service Map**
  - https://www.km4city.org/swagger/external/index.html
- **Federated Multiple Snap4City** Knowledge Bases. This allows the creation of mobile applications that may move from multiple cities and area accessing data and making queries transparently. This solution is presently in place among the Knowledge Bases of: Antwerp/Helsinki, Tuscany/Firenze, Sardegna, etc. The resulting Service is called Super Service Map and it is integrated in the Smart City API. For example, via:
  - https://www.disit.org/superservicemap/api/v1
- **Federated Open Data Portals** via DataGate/CKAN that presently presents now more than 13800 data sets linked for the cities of Helsinki and Antwerp.
  - https://datagate.snap4city.org/organization
  - Federation, Harvesting interface is: https://datagate.snap4city.org/harvest
- **WFS service of Snap4City** on top of Federated Smart City API or simple Smart City API of a single ServiceMap (smart City installation). This solution permits to GIS applications and platforms (such as ArcGIS OnLine ESRI, ArcGIS Enterprise ESRI, ArcGIS Map/pro Desktop, QGIS, GeoServer, etc.) to access at Snap4City data. For Example, via:
  - https://www.disit.org/superservicemap/api/v1/wfs
  - https://www.disit.org/superservicemap/api/v1/wfs?service=WFS&request=GetCapabilities&version=2.0.0
- **WMS service of Snap4City** for publishing **maps and heatmaps**, provided by an installed GeoServer third party open source tool. For example, via:
  - https://wmsserver.snap4city.org/geoserver/Snap4City/wms
  - https://www.km4city.org/swagger/external/index.html?urls.primaryName=Heatmap%20API

TOP

Web and Mobile App Development Kit

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**.

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}
   ]
};
```

Mustache JS

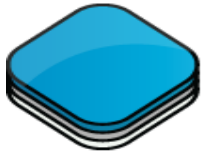
Template

```
<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



OpenLayers 3.0

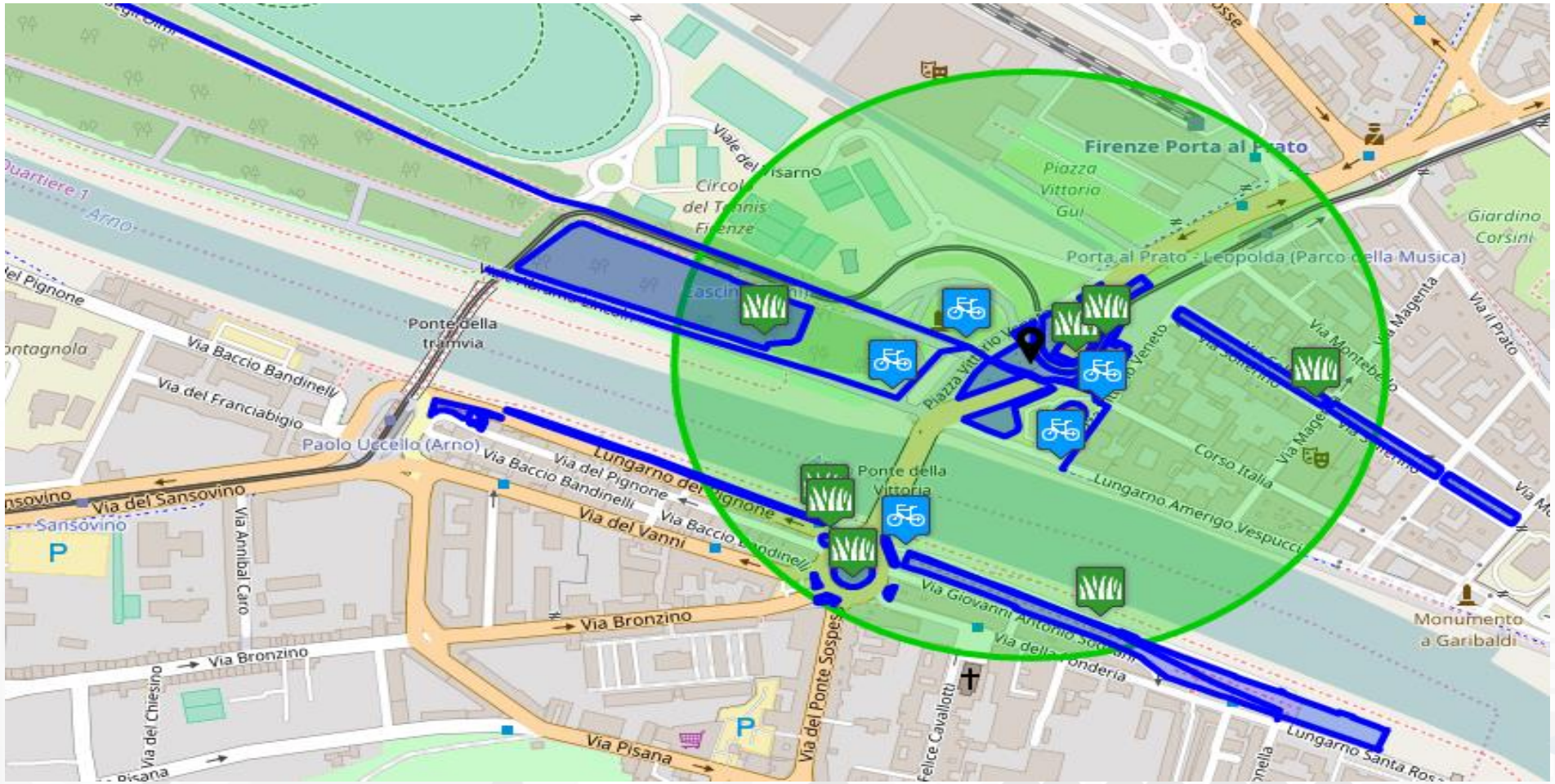
- **OpenLayers** is an open source **JavaScript library** for **displaying map data** in web browsers and can be used with a hybrid application developed with Cordova

- 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

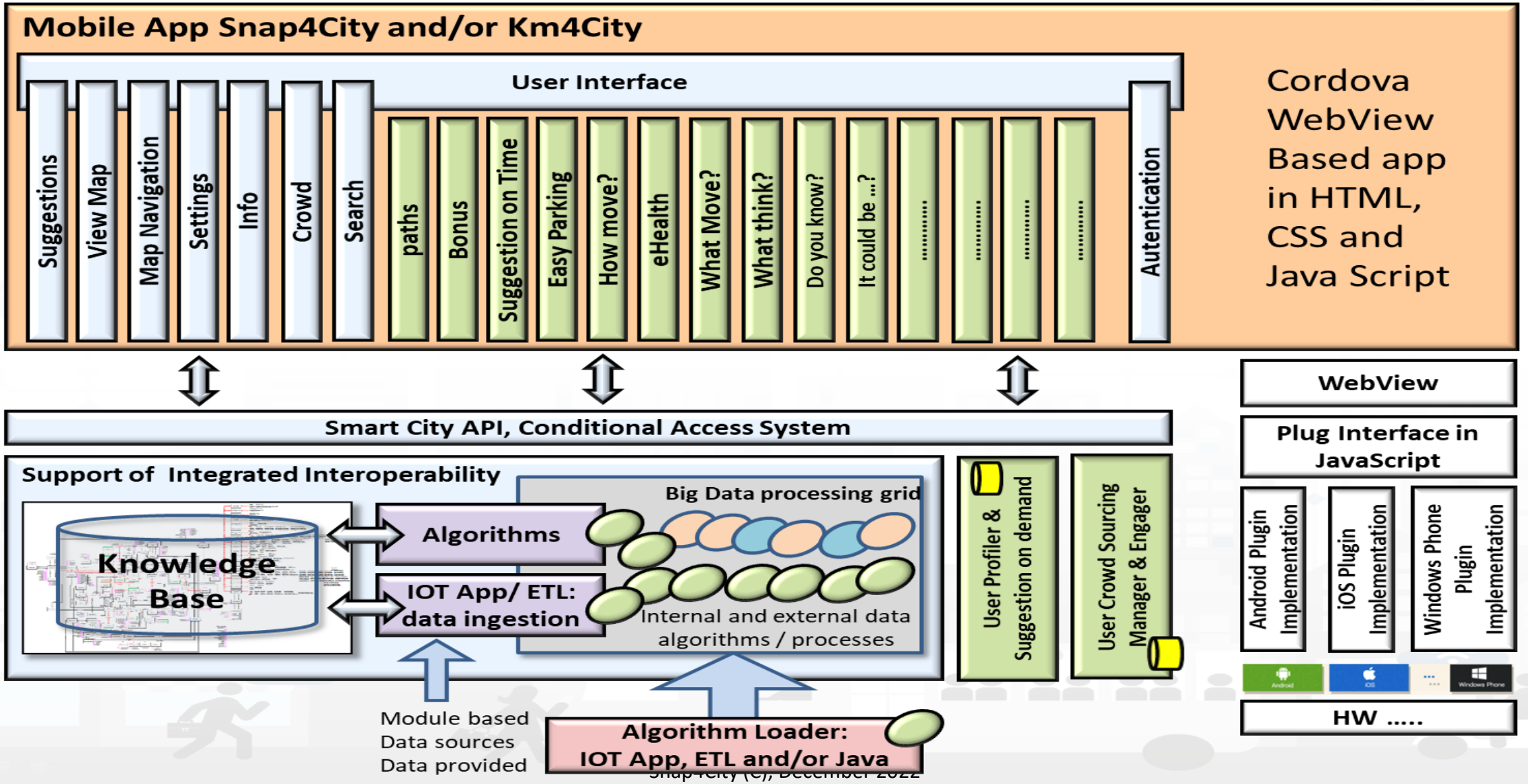
- In addition, OpenLayers 3.0 builds the map and objects added to it with a **canvas** renderer, which is **very efficient** when objects are **numerous and small** as the markers displayed for each search done with the app

- Documentation and downloads are available on the official website: **http://openlayers.org**

OpenLayers 3.0



General architecture of Mobile / Web App



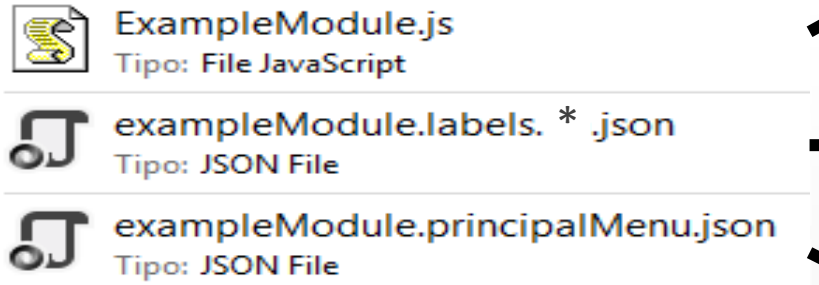
Create ParkingSearcher Module

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



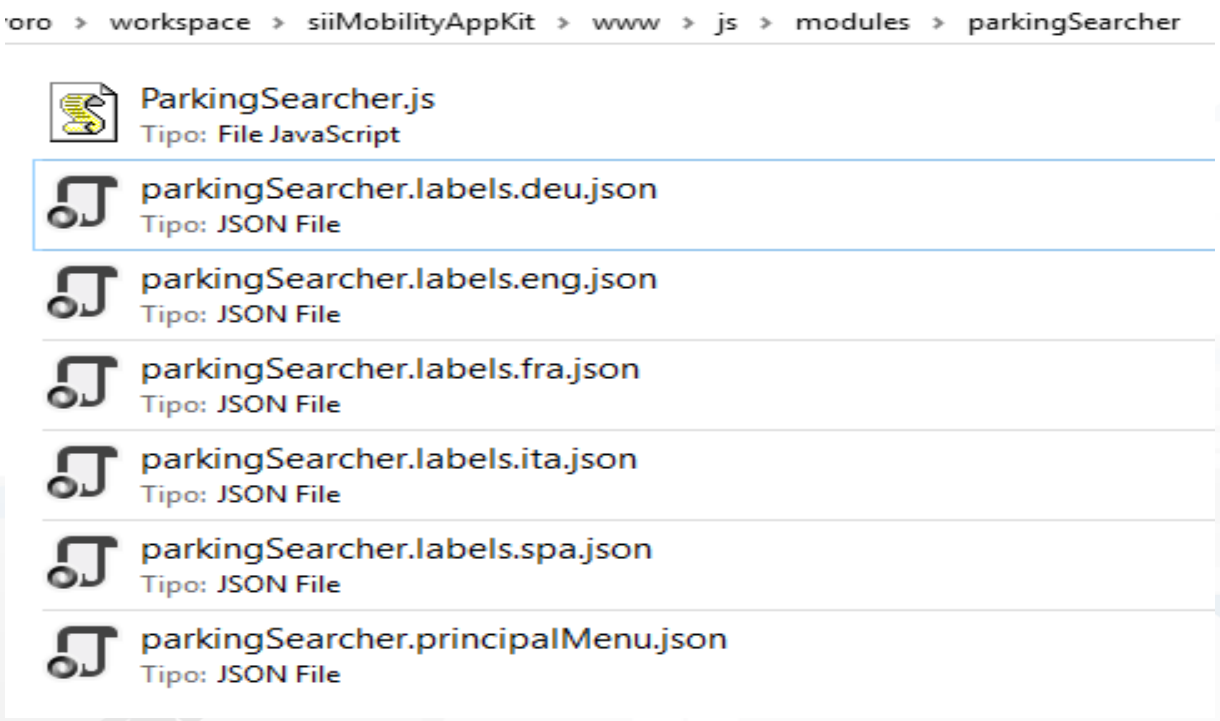
A **Javascript** file containing the **logic**

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**



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**

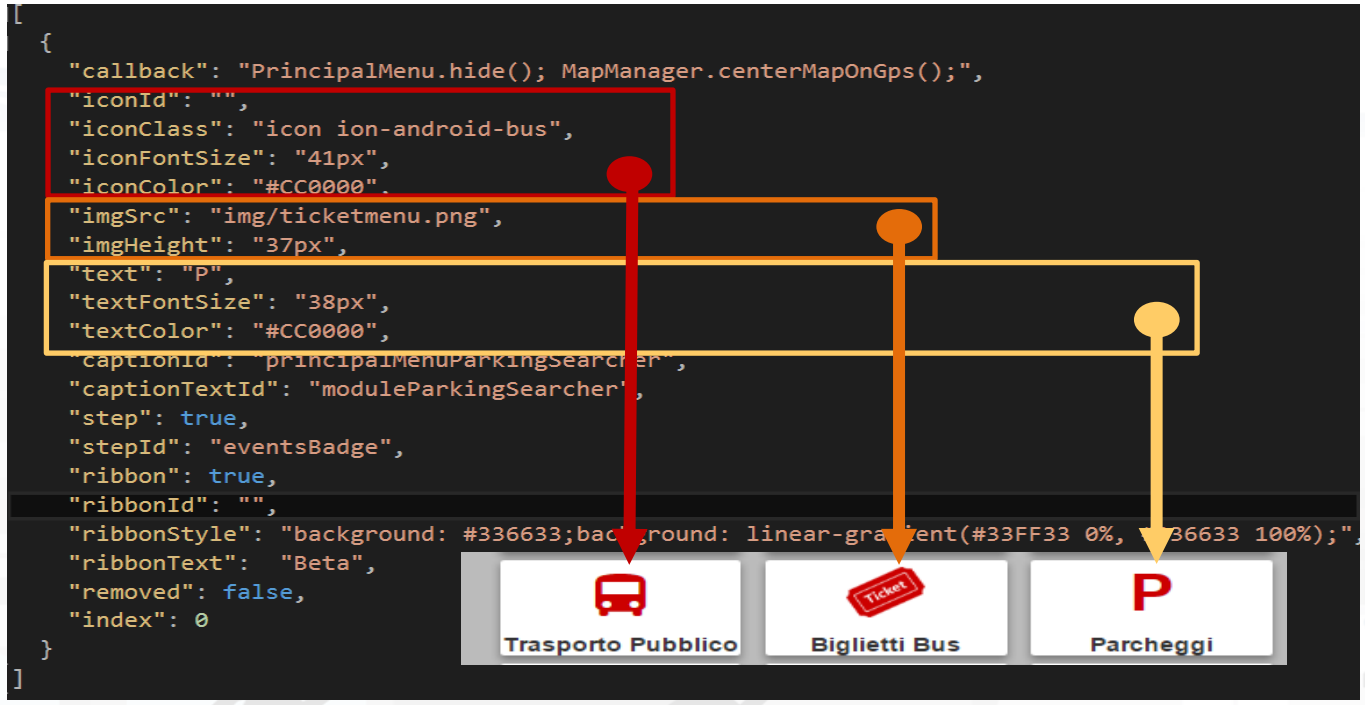
```
[
  {
    "callback": "PrincipalMenu.hide(); MapManager.centerMapOnGps();",
    "iconId": "",
    "iconClass": "icon ion-android-bus",
    "iconFontSize": "41px",
    "iconColor": "#CC0000",
    "imgSrc": "img/ticketmenu.png",
    "imgHeight": "37px",
    "text": "P",
    "textFontSize": "38px",
    "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**

ParkingSearcher in main menu

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



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
  }
]
```

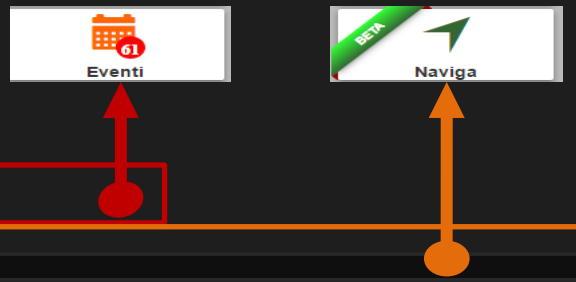
parkingSearcher.principalMenu.json

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

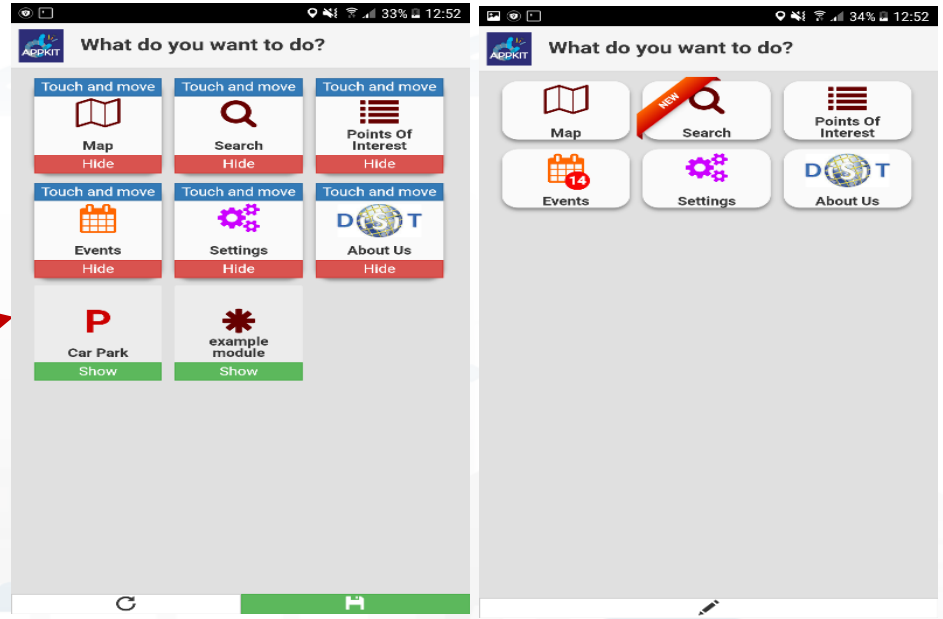
removed field is useful to allow the removal and the insertion of the buttons in the main menu by the user.

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

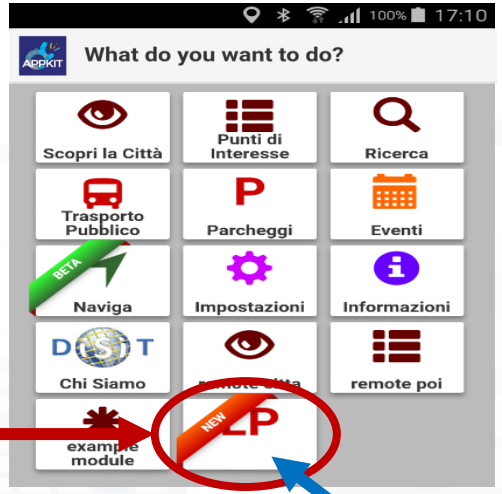
- Loading **new buttons modules** within the main menu, takes place by **comparing the captionId** field.
- If the menu already has a button with the **same captionId,** the first is **replaced** with the **new one**.
- To **remove** a button from the main menu (field **removed** hides it) add a **delete**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

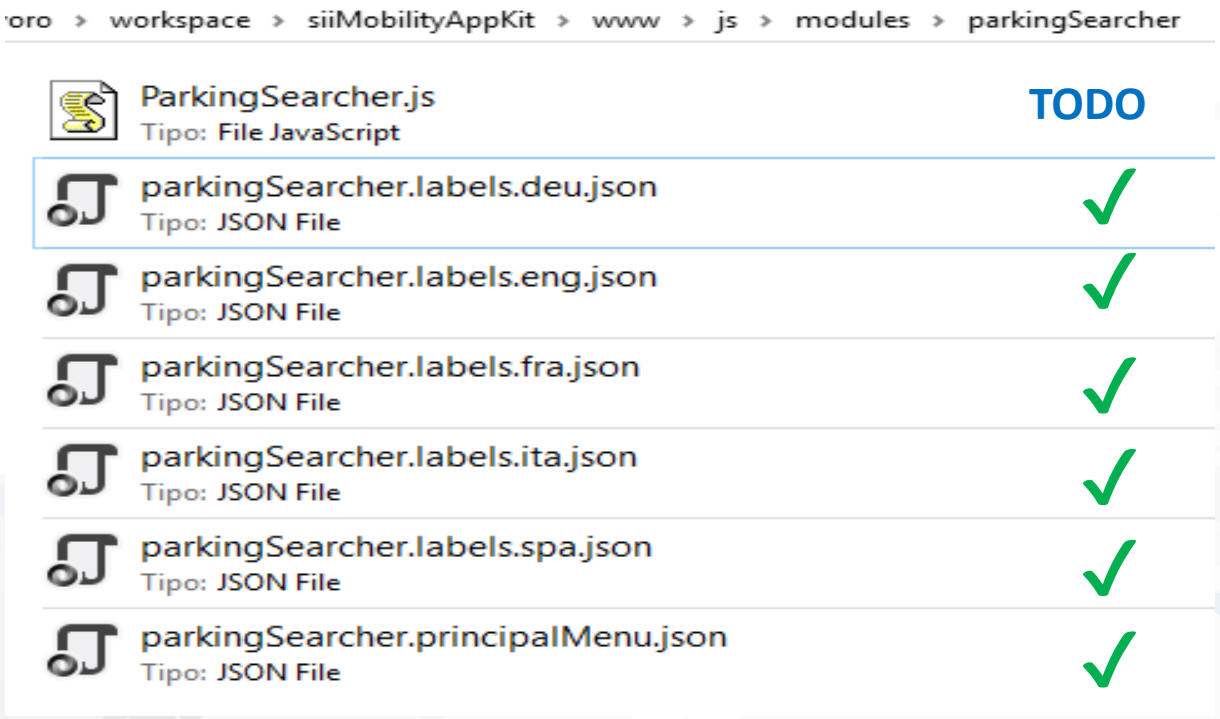
```
{
  "principalMenu": {
    "moduleParkingSearcher" : "Liste parkings"
  }
}
```

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



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**

```
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>
                $("#parkingMenuHeaderTit
                    Globalization.label .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

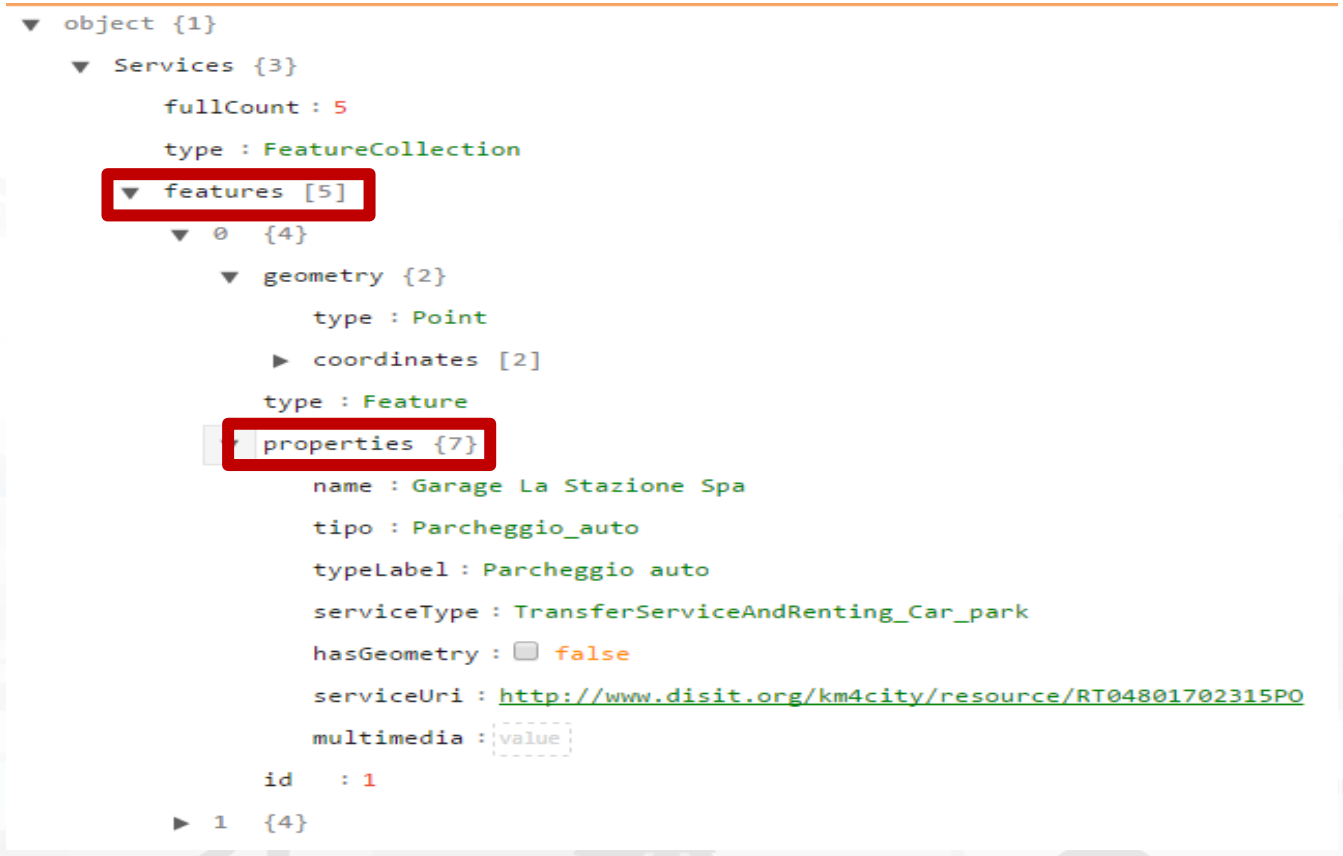
ParkingSearcher Called API

- 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



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

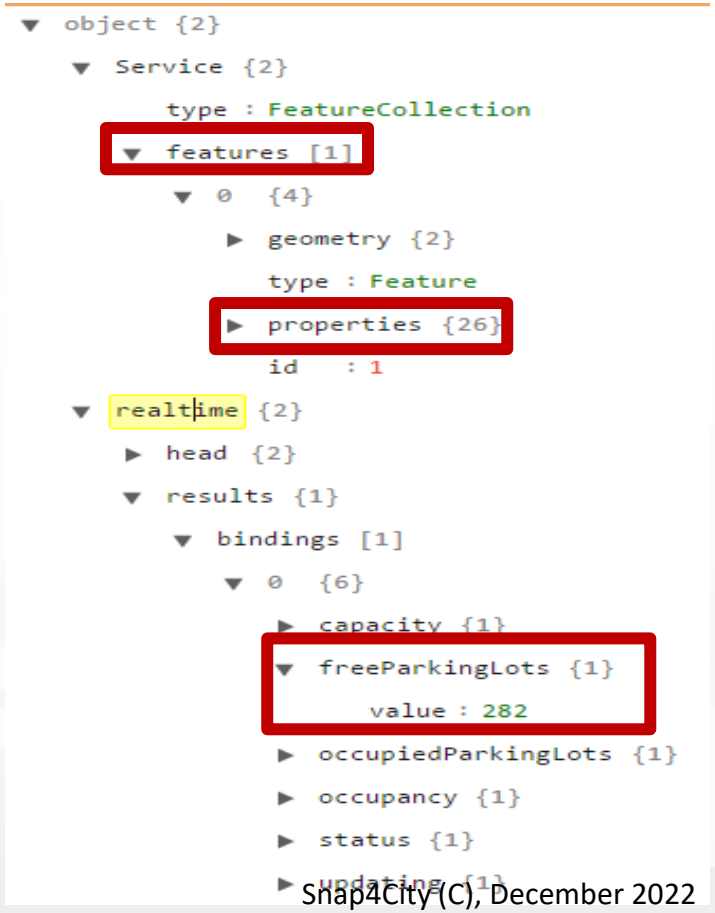
ParkingSearcher Called API

- 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**



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);
            }
        } else {
            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

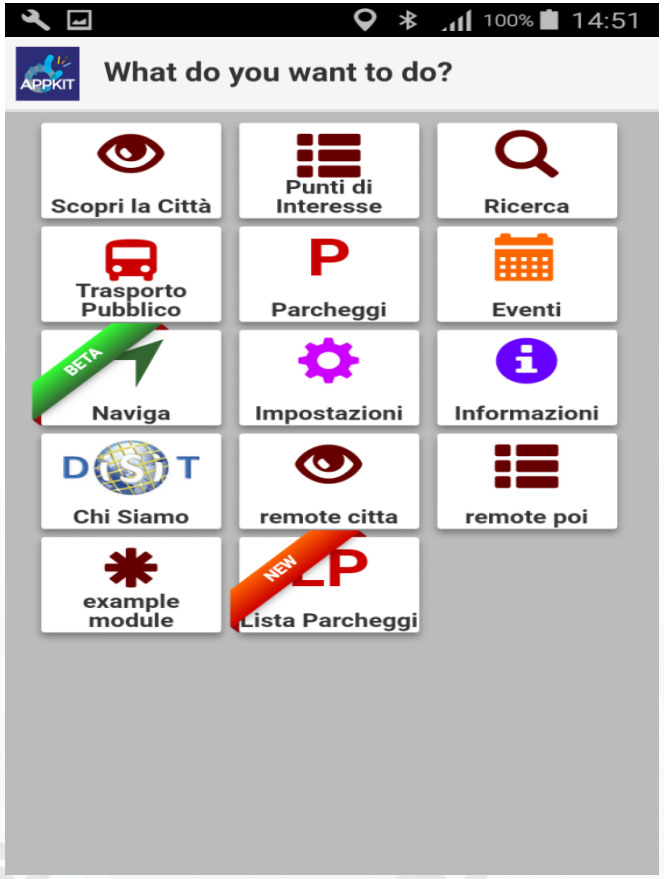
- Final version of the button with call to module logic

```
[
  {
    "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



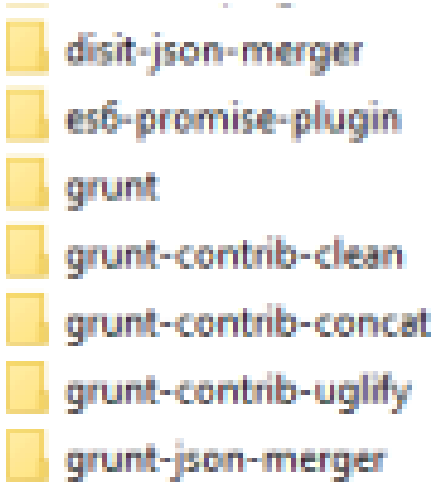
Gruntfile

For the creation of the app through the modules it is necessary to compile it with grunt before doing it with Cordova

https://gruntjs.com

Gruntfile

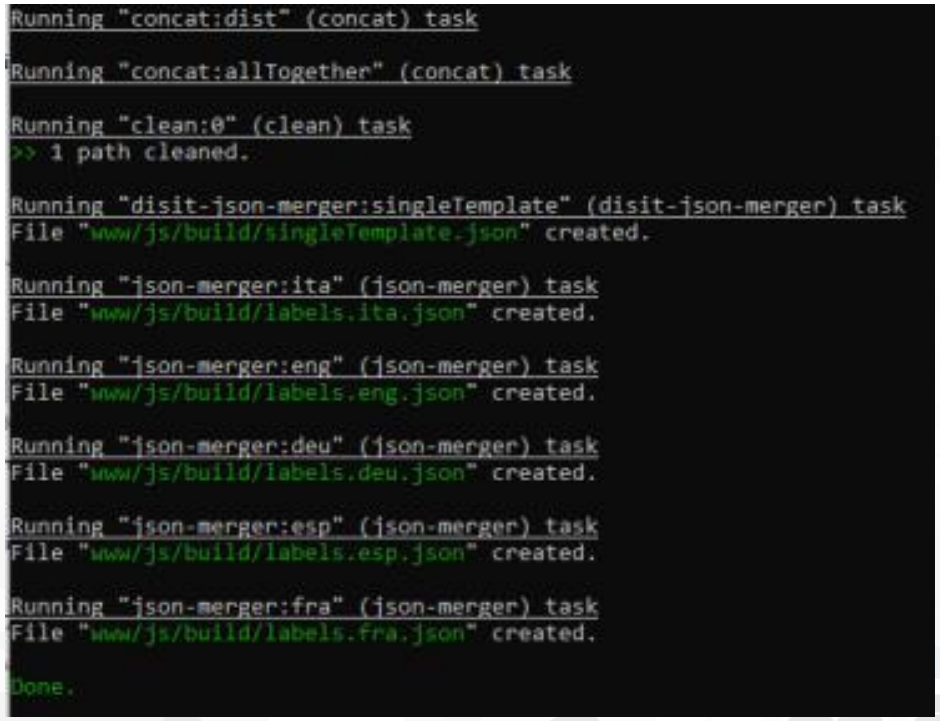
Inside the root folder of the Sii Mobility App Kit there is a gruntfile.js which will merge the js and json files as described inside it. Inside the root there is also a node_modules directory within which there must necessarily be these folders containing the plugins useful for merging files.



Gruntfile

To compile the files, the grunt command must be launched on the open terminal in the project root

If everything works properly the screen that should appear is as follows. If some package is missing it can be installed with the npm i packagename command. If disit-json-merger is missing download it from https://github.com/disit/siiMobilityAppKit/tree/master/node_modules/disit-json-merger



Gruntfile

If the grunt command was successful as in the image of the previous slide then the command Cordova build android can be launched and if all goes well you will have a screen like the following one

```
:compileDebugSources
:transformClassesWithDexBuilderForDebug
:transformDexArchiveWithExternalLibsDexMergerForDebug
:transformDexArchiveWithDexMergerForDebug
:transformNativeLibsWithMergeJniLibsForDebug
:transformResourcesWithMergeJavaResForDebug
:packageDebug
:assembleDebug
:cdvBuildDebug

BUILD SUCCESSFUL in 1m 54s
44 actionable tasks: 44 executed
Built the following apk(s):
        C:/Users/badii.DISIT/Lavoro/workspace/siiMobilityAppKit/
apk
```

Further readings

- TC5.16. Exploiting Smart City API for developing Mobile and Web Apps
- TC5.15. Snap4City Smart City API Collection and overview, real time
- TC5.17. Search on Services via Smart City API: MicroApplication, Exploiting Micro Applications in HTML5 based on Advanced Smart City API
- TC5.18. Snap4City API are documented in Swagger, and tested in Postman
- TC5.19. Using ServiceMap as a Tools for Developing web and mobile apps and micro applications

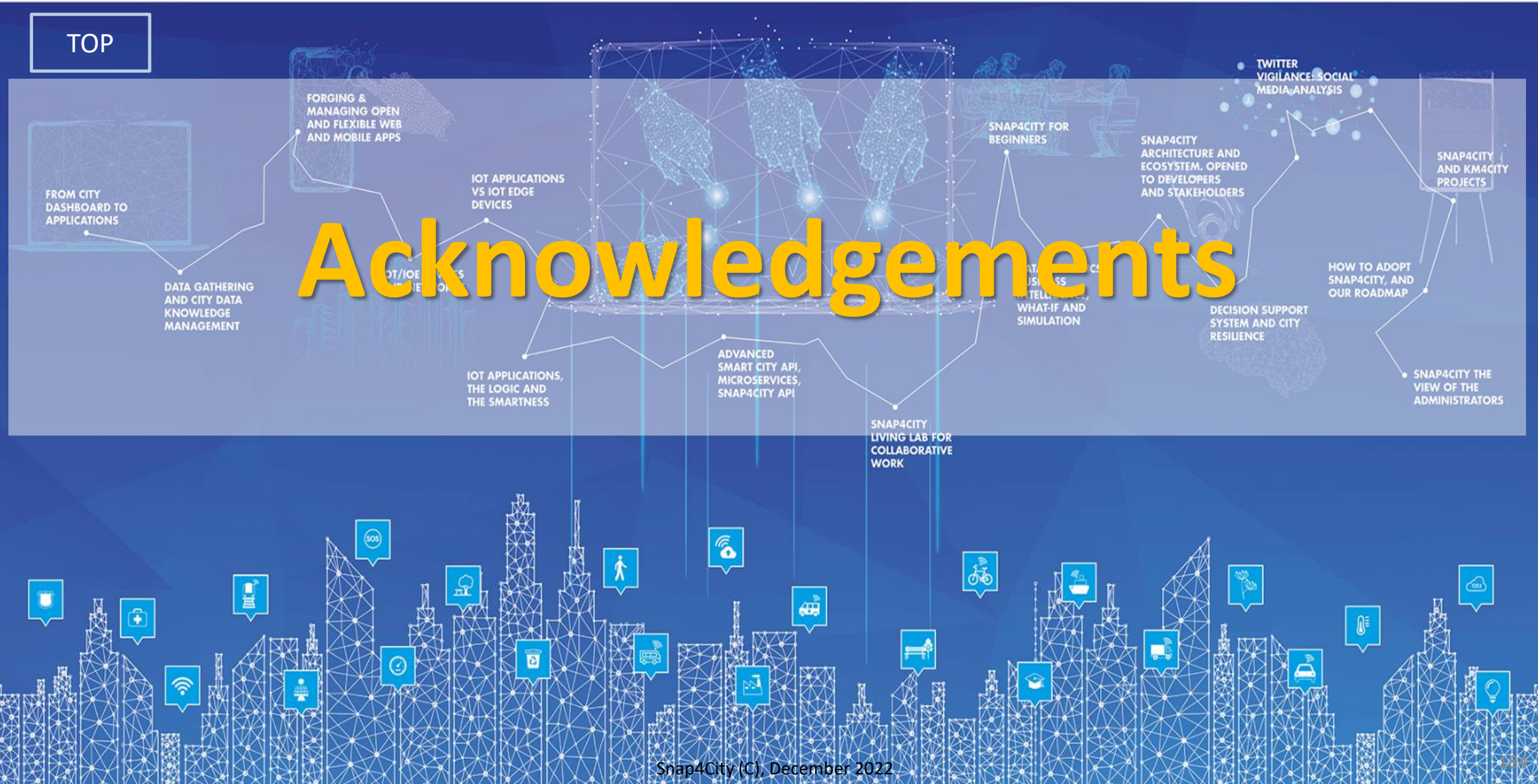
Useful links

- US1. Using City Dashboards
- US2. Using and Creating Snap4City Applications with Dashboards
- US3. Using and Creating Developer Dashboards, AMMA dashboard, and/or Resource Dashboards
- US4. Creating City Dashboards and related Event Monitoring and Actions
- US5. Discovering City Services Exploiting Knowledge Base via ServiceMap
- US6. Developing and using processes for data transformation
- US7. Data Analytics and related integration aspects
- US8. Using the Living Lab Support tools
- US9. Creating Snap4City IOT Applications, different formats, protocols, brokers, communications
- US10. Using and Managing the Scalable Snap4City Infrastructure
- US11. Using tools/services of a secure and privacy respectfully solution

Former Documentation

- **Documentation Smart City API**
  - http://www.disit.org/6991
- **Ontology and Km4City Tools:**
  - Http://www.km4city.org
  - http://www.disit.org/6506 Ontology and documentation
- **Snap4city is Open Source on GitHub as DISIT lab:**
  - https://github.com/disit
  - https://github.com/disit/snap4city (mobile App kit)

TOP



Acknowledgements

DISIT lab roadmap vs model and tools' usage

2013
Km4City Ontology 1.1

- Tuscany, Road Graph
- Mobility
- culture, tourism
- Events
- Parking
- Services
- Linked open graph

2014
- Weather Forecast
- Real Time Wi-Fi
- Entertainment
- LOD

- Twitter Vigilance
- Social Media Analytics, Sentiment Analysis

Km4City 1.4

2015
Km4City 1.5

- Resilience Decision Support
- Smart First Aid
- User Behaviour Analysis, predictions
- Risk Analysis

ckan, resolute

H2020 (2015-18)

2016
(2016-21) H2020 REPLICATE

Km4City 1.6.4

- Smart Energy
- Sustainable Mobility
- Control Room
- Dashboard

Km4City 1.6.2

Sii-Mobility (2016-21)

SII-MOBILITY SCN

- Infomobility
- Mobile App
- Routing
- Multimodality

FIWARE

Km4City 1.5

- Resilience Decision Support
- Smart First Aid
- User Behaviour Analysis, predictions
- Risk Analysis

2017
Sii-Mobility

- Origin-Destination and trajectories
- Traffic Reconstruction
- Offer Analysis
- OBU, smart devices

E0I5 digital ecosystem (Powered by EXPO MILANO 2015)

(2016-19) GHOST SIR

- Sardinia Region Smart City Strategies and plan

GREEN IMPACT C.A.P.A.C.I.T.Y.

GREEN IMPACT
POR FESR 2014-2020

- Industry 4.0
- Critical Plant
- Monitoring

- IOT/IOE, IOT App
- Living Lab
- Maker Support
- IOT Edge
- Smart City IOT
- GDPR,
- Privacy & Security

life weee (2017-20)

- Smart Waste

2018
MOSAiC MObility 4.0 for SmArt (i) City (2018-20)

- Mobility Demand / Offer Analytics and Strategy

Sii-Mobility

- User engagement
- Bike Sharing
- Data Analytics ++
- Social Predictions
- OBD2

IOT/IOE

Km4City 1.6.6

Node-RED

SNAP4CITY SELECT for Cities H2020 (2017-19)

2019
5G (2018-21)

- 5G tech
- Energy
- Industry 4.0
- Synoptics

TRAFAIR CEF (2018-21)

- Traffic and Mobility Impact on Pollution
- NOX predictions

EUROPEAN OPEN SCIENCE CLOUD

LAID Smart Bed

- Smart Health

GREEN FIELD PEAS Soda4.0 Optimization and Automated Decision System 4.0

- Industry 4.0

bee smart city

- Smart Lonato

Reverberi Enetec Gruppo MPES

SNAP4CITY KM4CITY

Winner of Select4Cities PCP

SNAP4CITY roadmap

SODA

2020

JRC EUROPEAN COMMISSION

Interreg Mediterranean HERIT-DATA

- Smart Tourism
- 6 Pilots
- Data Analytics
- Extended platform

Interreg MARITTIMO-IT FR-MARITIME MOBI MART

- Smart Mobility
- PISA, PUMS
- Living lab

smartGARDAlake

2021

JRC EUROPEAN COMMISSION

PC4City (2020-21)
Monitoring Terrain

Winner of Open Data Challenge of enel x

CAPELON

- Smart Light
- Sweden

Km4City 1.6.7

enel x
Contract

Smart Ambulance
(2021-22)

Enterprise
(2021-22)
Industry 4.0

Almafluida
Industry 4.0
(2021-22)

AMPERE (2021-22)
Industry 4.0

SYN-RG-AI
SmartCity

Smartea

ITALFERR GRUPPO FERROVIE DELLO STATO ITALIANE
Smart City

ASYMMETRICA URBAN TECHNOLOGY
Smart City

2022

AXIS COMMUNICATIONS
SmartCity

uni.systems
SmartCity

GRUPPO PRETTO
Industry 4.0

IMPETUS
Security and Risk

enel x
Contract

Filippi
Contract

ART-ER ATTRATTIVITÀ RICERCA TERRITORIO
Contract

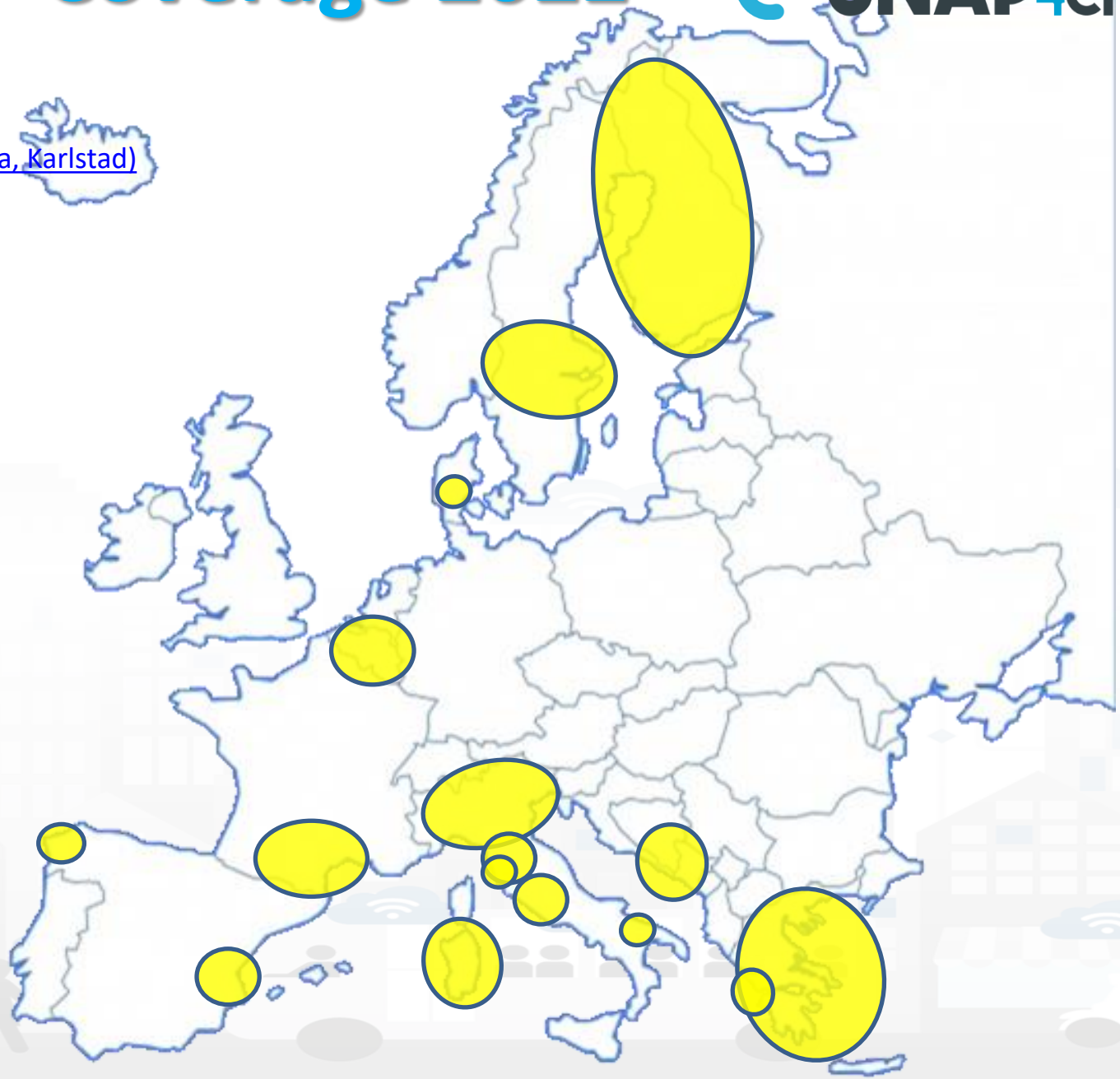
Some agreements

JRC EUROPEAN COMMISSION

Coverage 2022

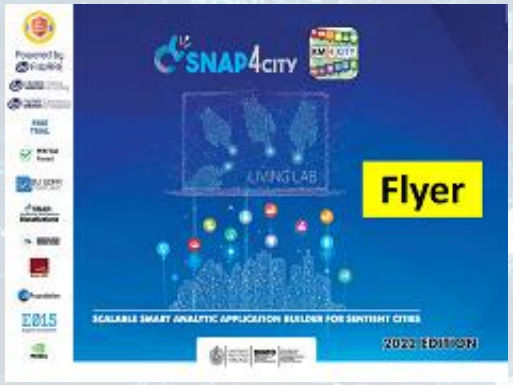
Main Organizations/areas

- Antwerp area (Be)
- Bologna (I)
- Capelon (Sweden: Västerås, Eskilstuna, Karlstad)
- DISIT demo (multiple)
- Dubrovnik, Croatia
- Firenze area (I)
- Garda Lake area (I)
- Greece (Gr)
- Helsinki area (Fin)
- Livorno area (I)
- Lonato del Garda (I)
- Modena (I)
- Mostar, Bosnia-Herzegovina
- Oslo & Padova (Impetus)
- Pisa area (I)
- Pistoia (I)
- Pont du Gard, Occitanie (Fr)
- Prato (I)
- Roma (I)
- Santiago de Compostela (S)
- Sardegna Region (I)
- Siena (I)
- SmartBed (multiple)
- Toscana Region (I), SM
- Valencia (S)
- Venezia area (I)
- WestGreece area (Gr)



2022 booklets

- Snap4City



https://www.snap4city.org/download/video/DPL_SNAP4CITY_2022-v02.pdf

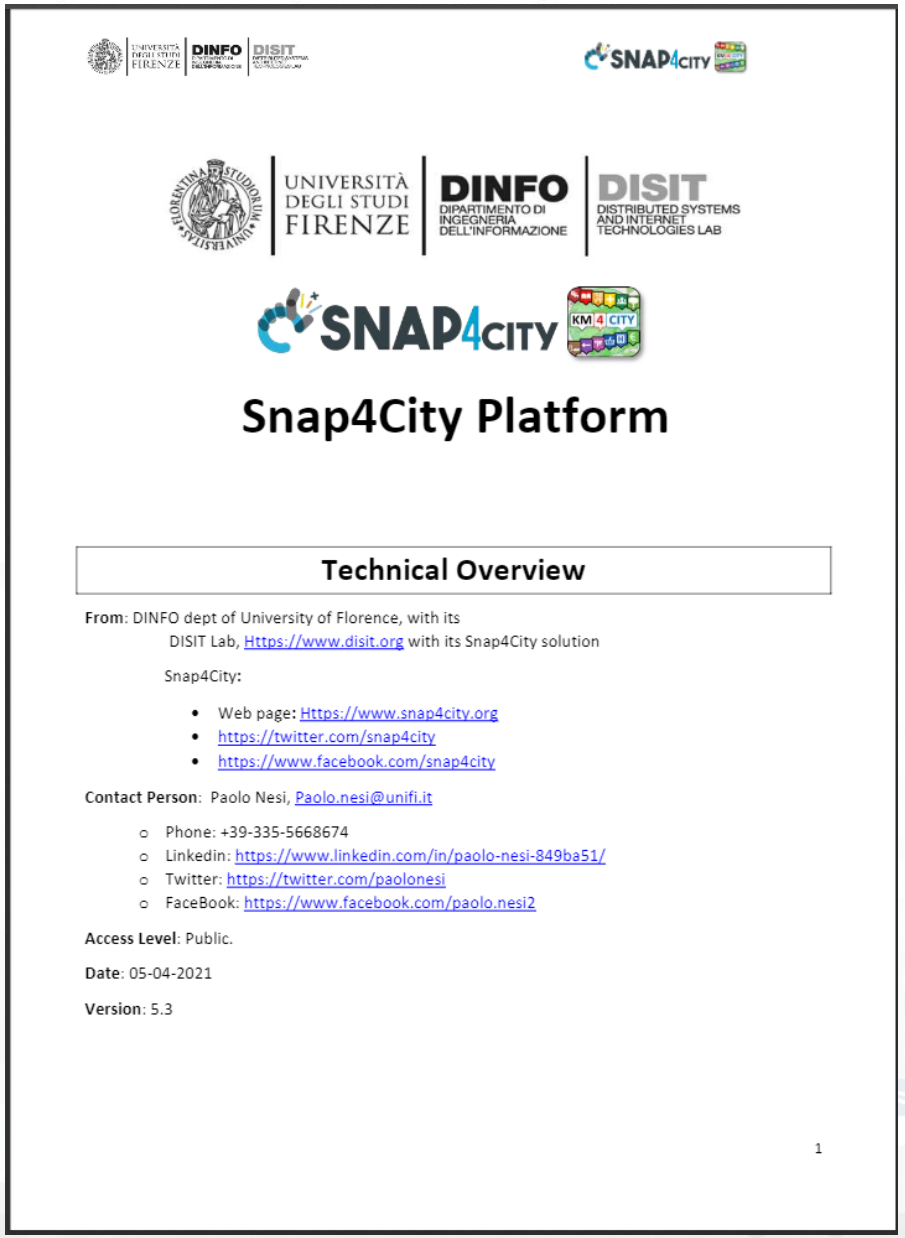
- Snap4Industry

https://www.snap4city.org/download/video/DPL_SNAP4INDUSTRY_2022-v03.pdf

- Solutions
- Data Analytics

https://www.snap4city.org/download/video/DPL_SNAP4SOLU.pdf

Overview



Snap4City Platform

Technical Overview

From: DINFO dept of University of Florence, with its
DISIT Lab, Https://www.disit.org with its Snap4City solution

Snap4City:

- Web page: Https://www.snap4city.org
- https://twitter.com/snap4city
- https://www.facebook.com/snap4city

Contact Person: Paolo Nesi, Paolo.nesi@unifi.it

- Phone: +39-335-5668674
- Linkedin: https://www.linkedin.com/in/paolo-nesi-849ba51/
- Twitter: https://twitter.com/paolonesi
- FaceBook: https://www.facebook.com/paolo.nesi2

Access Level: Public.

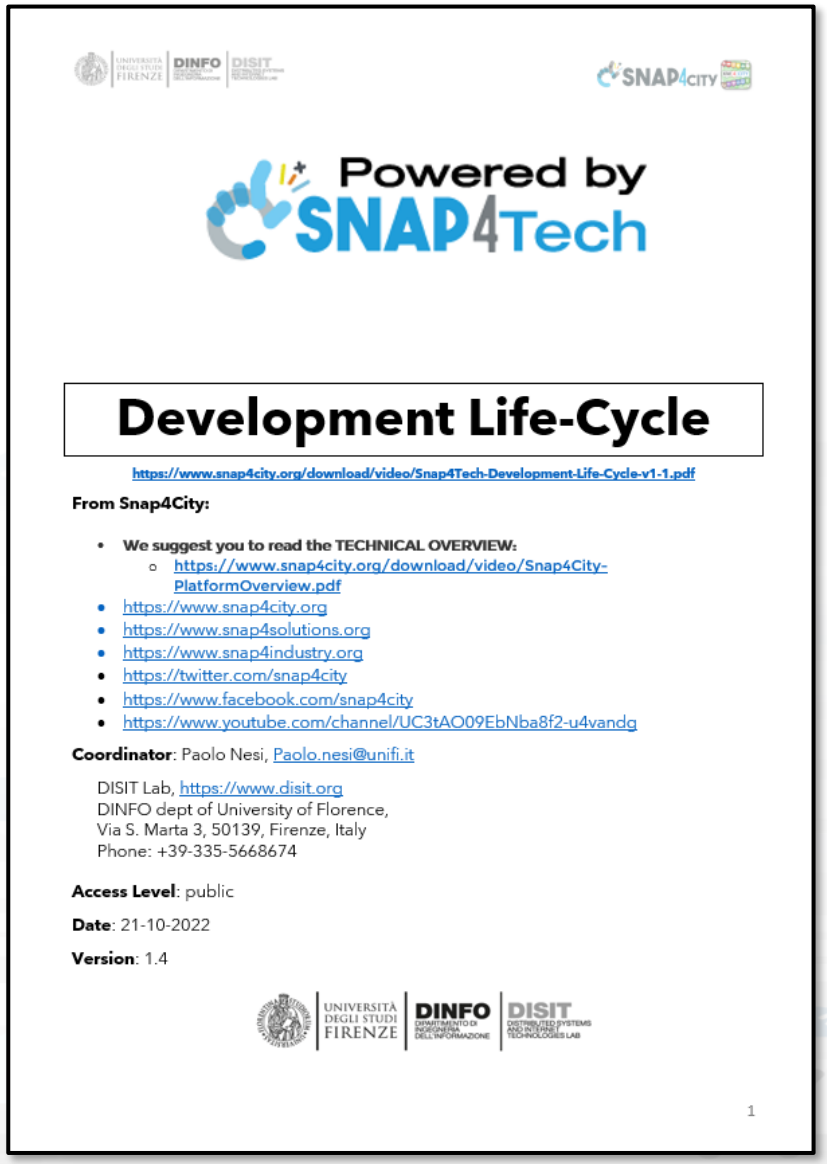
Date: 05-04-2021

Version: 5.3

1

- https://www.snap4city.org/drupal/sites/default/files/files/Snap4City-PlatformOverview.pdf

Development



Powered by SNAP4Tech

Development Life-Cycle

https://www.snap4city.org/download/video/Snap4Tech-Development-Life-Cycle-v1-1.pdf

From Snap4City:

- **We suggest you to read the TECHNICAL OVERVIEW:**
  - https://www.snap4city.org/download/video/Snap4City-PlatformOverview.pdf
- https://www.snap4city.org
- https://www.snap4solutions.org
- https://www.snap4industry.org
- https://twitter.com/snap4city
- https://www.facebook.com/snap4city
- https://www.youtube.com/channel/UC3tAO09EbNba8f2-u4vandg

Coordinator: Paolo Nesi, Paolo.nesi@unifi.it

DISIT Lab, https://www.disit.org
DINFO dept of University of Florence,
Via S. Marta 3, 50139, Firenze, Italy
Phone: +39-335-5668674

Access Level: public

Date: 21-10-2022

Version: 1.4

1

https://www.snap4city.org/download/video/Snap4Tech-Development-Life-Cycle.pdf

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

Main running instances (11/21)

- Sii-Mobility → mobility and transport, sustainability
- REPLICATE → ICT, smart City Control room, Energy, IOT
- RESOLUTE → Resilience, ICT, Big Data
- GHOST → Strategies, smart city
- TRAFAIR → Environment & transport
- MOSAIC → mobility and transport
- WEEE Life → Smart waste, environment
- Smart Garda Lake → Castelnuovo del Garda, SMARTEA
- 5G → Industry 4.0 vs SmartCity
- Green Impact → Industry 4.0, Chemical Plant, control and plan
- SmartBed (Laid) → smart health
- Green Field Peas (Soda) → Industry 4.0, Chemical plant
- MobiMart and PISA Agreement → data aggregation, mobility and transport, Living Lab
- Lonato del Garda → smart parking, environment
- Herit Data → tourism, culture and management
- ISPRA JRC → site management and services
- Capelon (Sweden) → smart light solutions
- PC4City → land slide monitoring and predictions
- Italmatic → industry 4.0 production control

Acknowledgements

- Thanks to the European Commission for founding. All slides reporting logo of **Snap4City** https://www.snap4city.org **of Select4Cities H2020** are representing tools and research founded by European Commission for the **Select4Cities** project. **Select4Cities** has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation Programme (grant agreement n° 688196)
- **TRAFAIR is a CEF** project. All slides reporting logo of TRAFAIR project are representing tools and research founded by the EC on CEF programme http://trafair.eu/
- 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 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 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 and research line of DISIT Lab exploited by a number of projects. Some of the innovative solutions and research issues developed into projects are also compliant and contributing to the Km4City approach and thus are released as open sources and are interoperable, scalable, modular, standard compliant, etc.*

TOP

SNAP4CITY

KM 4 CITY

Be smart in a SNAP!

SNAP4
Appliances and Dockers
Installations

CONTACT

DISIT Lab, DINFO: Department of Information Engineering
Università degli Studi di Firenze - School of Engineering

Via S. Marta, 3 - 50139 Firenze, ITALY
https://www.disit.org

www.snap4city.org

Email: snap4city@disit.org

Office: +39-055-2758-515 / 517
Cell: +39-335-566-86-74
Fax.: +39-055-2758570

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