Data Ingestion Processes and Tools

October 2020, Course
https://www.snap4city.org/577

SCALABLE SMART ANALYTIC APPLICATION BUILDER FOR SENTIENT CITIES
scalable Smart aNalytic APplication builder for sentient Cities: for Living Lab and co-working with Stakeholders

https://www.Snap4City.org

Data Ingestion Processes and Tools

October 2020, Course
https://www.snap4city.org/577

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

Snap4City (C), October 2020
## On Line Training Material (free of charge)

<table>
<thead>
<tr>
<th>what</th>
<th>1st part (*)</th>
<th>2nd part (*)</th>
<th>3rd part (*)</th>
<th>4th part (*)</th>
<th>5th part (*)</th>
<th>6th part (*)</th>
<th>7th part (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF</td>
<td>General</td>
<td>Dashboards</td>
<td>IOT App, IOT Network</td>
<td>Data Analytics</td>
<td>Data Ingestion processes</td>
<td>System and Deploy</td>
<td>Install</td>
</tr>
<tr>
<td>Interactive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>duration</td>
<td>2:55</td>
<td>3:16</td>
<td>3:41</td>
<td>2:00</td>
<td>2:48</td>
<td>2:35</td>
<td>1:47</td>
</tr>
</tbody>
</table>

[https://www.snap4city.org/577](https://www.snap4city.org/577)
General Overview of the full Course

• 1st part: General Overview
• 2nd part: Dashboards Creation and Management
• 3rd part: IOT Applications development, IOT Devices, IOT Networks
• 4th part: Data Analytics, in R Studio, in Python, how to Exploit and Manage Data Analytics in IOT Applications
• 5th part: Data Ingestion, Data Warehouse, Data Gate, IOT Device Data ingestion, IOT App for Data Ingestion, etc.
• 6th part: Snap4City Architecture, How To Install Snap4City
• 7th part: Smart city API (internal and external) Web and Mobile App development tool kit

A number of the training sections include excercitations

Updated versions on: https://www.snap4city.org/577
See also courses in ITALIANO: https://www.snap4city.org/485
• Solution Analysis and Design
• Data Ingestion Capabilities and Architecture
• Data Ingestion Strategy and Tools: (Data Inspector, DevDash, Knowledge Base)
• Upload Data Sets of POIs via IOT Applications (Map data, KB)
• Data Ingestion via IOT Brokers
  – IOT Directory, IOT Broker Registration, IOT Device Model, IOT Device Registration
• Real Rime Data Ingestion via IOT Applications
• Data Ingestion via API: External Services, using HTTP MicroService on IOT Applications
• Data Ingestion via IOT Applications towards MyKPI
• Data Ingestion via Web Scraping
• Data Streams from Smart City API, participatory
• Data Streams from Mobile Devices
• Data Streams from Dashboards
• GIS Data Import and Export
• Integration with CKAN, open data manager and portal
• Social Media data collection and exploitation
• Data Ingestion and Transformation via ETL processes (only for former versions of Snap4City)
• Acknowledgements
Retails

Data: Public and Private, Static and Real Time

Private: user movements, social media, crowd sources, commercial (retail)
Public: infomobility, traffic flow, TV cameras, flows, ambient, weather, statistic, accesses to LTZ, services, museums, point of interests, ...

Smart City Engine

Services & Suggestions
Transport, Mobility, Commercial (retail), Tourism, Cultural

API for SME
Pub. Admin: detection of critical conditions, improving services
Tune the service, reselling data and services, prediction

User profiling
Collective profiles
User segmentation

User Behavior
Crowd Sources
Dynamic ticketing, whispers to save time and money, geoloc information, offers, etc.

Personal Time Assistant

Challenges: Requests and Deductions
Public Admin.
Mobility Operators
Tourism Museums

Commercial: customers prediction and profiles, promotions via ads
Tune the service, prediction
Living Lab Accelerating

GO!

Collaborative Platform

City Operators

Resource Operators

agreements

networking

tutorials

documentation

Inhouse companies

Tech providers

workshops

Category Associations

Corporations

Advertisers

City Users

Advertisers

Large Industries

collaborations

subscription to applications

personal services

Manage Apps & Dashboards, User Engagement

Upload context Open Data

Connect IOT/IOE

Connect external Services

Data Ingestion and Analytic algorithms

Advanced Smart City API, MicroServices

Produce City IOT Applications & Dashboards

Produce Apps and Dashboards for City Users

Monitor City Platform

Promote Applications & Dashboards

City Operators

Early Adopters

Start-ups

partnerships

Licensing, Gold services

Help desk

Case Studies

Research groups

Rule of Thumb

Hackathons

GO!

Go!
Solution Analysis and Design
Snap4City Innovation Matrix and Process

see Course Part 6
Data vs Smart Services enabling on Snap4City

- Public Transportation and mobility activated services in some where with Snap4City
  - Smart parking (parking locations and real time parking data) ... predictions
  - Smart Fuel pricing (fuel station locations and real time prices)
  - Routing (detailed GIS information, text indexing of streets, POI, etc.)
    - Quite routing, perfect shopping, etc. etc. (more data in needed....)
  - multimodal routing (detailed GIS information, Public transport time schedule)
  - Info traffic (traffic flow sensors, real time Traffic events, their localization, etc.)
  - Dense info traffic (traffic flow sensors and traffic flow reconstruction algorithm)
  - Car/Bike/Scooter Sharing (position and availability of Cars/Bikes, Scooters) ... predictions
  - Smart Biking (cycling paths, environmental data) ... predictions
  - E-vehicles (position, status of recharging stations, ...) ... predictions vs booking
  - Smart river crossing (position and status of Underpass, Ferry) ... prediction
  - Quality of Public Transport (actual time of arrival at the bus stops, wrt planned time schedule)
  - Early Warning vs Resilience (combination of several data including mobility, events, Social to perform early warning...)

Snap4City (C), October 2020
Data vs Smart Services enabling on Snap4City

- Social and Users Behaviour
  - Smart First Aid
  - search for POI and public transport services
  - Social Media Monitoring and acting
  - Information to Tourists
  - Early Warning, prediction of audience
  - Improvement of services for Tourists

- Weather and environment, quality of life
  - Weather forecast/condition
  - Air quality Pollution
  - Pollination
  - Alerting on Air quality for multiple parameters
  - Information Heatmaps for weather and air quality
  - Air quality indexes, and forecast

(Location of First AID, real time status of triage)
(POI geolocalized, spatial queries, along paths)
(Identif. of dysfunction, quality of service perceived)
(Entertainment Events)
(Twitter data, social media)
(people flow, usage of services)
(Origin Destination Matrices, trajectories, heatmaps)
(People Monitoring, via App, Wifi, PAX Counter)
(Twitter Data, social mea,....)

(Weather forecast)
(pollution sensors, PM10, PM2.5, NOX, etc.)
(Pollination sensors)
(Prediction of parameters time slots, notification)
(air quality sensors, heatmaps, prediction)
(......................)
Data Ingestion Capabilities and Architecture
Development Life Cycle
Smart City Services

Analysis & Design
- Analysis
- Design
- Data Discovery
- Data Ingestion

Data Analytics
- Data Analytics Development
- IOT App Development
- Special Tool Development
- Dashboard Development

Deployment
- Deploy
- Testing
- Publication Production

Snap4City (C), October 2020
SNAP4CITY

URBAN PLATFORM: SMART CITY IOT AS A SERVICE AND ON PREMISE

IOT APPLICATIONS - INSTANT APPS
- Data Driven Applications
- Real Time Processing
- Batch Processing
- Any Protocol & Format

DASHBOARDS & APPLICATIONS
- Control Room
- Situation Room
- Operator Dashboards
- Business Intelligence
- What-If Analysis
- Decision Support
- Simulations
- Risk Analysis
- Resilience Analysis

MOBILE & WEB APPLICATIONS
- Development Kit
- Suggestions
- Mobile Apps
- Monitoring Panels
- Platform Utilities
- Ready to use Smart Applications

MICROSERVICES & ADVANCED SMART CITY API
- Open IoT Devices
- IoT Edge
- IoT Gateway
- Pax Counters
- IoT Buttons

LIVING LAB - DEV TOOLS - COWORKING
- Test Cases, Scenarios, Videos, Hackathons
- Open Sources, Community of Cities
- Training Tutorials, Community Management

BIG DATA - DATA ANALYTICS
- Predictions
- Anomaly Detection
- What-If Analysis
- Traffic Flow Reconstruction
- Origin-Destination Matrices
- Social Media Analysis
- Offer vs Demand Analysis
- Environmental Data Analysis and Predictions
- Real Time Heatmaps
- Routing
- Alerting
- Early Warning
- Personal and Virtual Assistants
- Smart Solutions
- Smart Sharing
- Participatory

DATA ANALYTICS TOOLS - MICRO-APPLICATIONS
- IoT Directory
- Service Map
- Resource Manager
- Data Gate
- R Studio
- ETL

KM4CITY DATA AGGREGATE KNOWLEDGE BASE - EXPERT SYSTEM OF THE CITY – BIG DATA STORE

IOT MNG - DATA MNG - DATA INSPECTOR – PROCESS MNG – USER ENGAGEMENT – GDPR MNG ...

OIS
CITY UTILITIES
OPEN DATA
LEGACY & EXTERNAL SERVICES
PERSONAL DATA
IOT / IOC
BROKERS
KPI
INDUSTRY 4.0
SOCIAL MEDIA
• **Open Data:**
  – Data gate, federation of Open Data Portals
  – ETL processes (PULL)
  – IOT Application processes

• **IOT Networks:**
  – IOT Application processes, data driven or PULL
  – IOT Brokers (Push) → IOT Shadow

• **Web Pages:**
  – Web scraping, crawling processes

• **Social media: Twitter, Facebook...**
  – Twitter Vigilance, IOT App

• **Mobile Apps**
  – Smart City API

• **Files upload: CSV, Excel, etc.**
  – IOT Applications, ETL

• **REST API, WS, FTP, LD, LOD, etc.**
  – IOT Applications, ETL

• **Data base accesses**
  – GIS: WFS, WMS
  – ETL, IOT Application

---

**Any kind of data and flows**

- **IOT Broker**
- **Web Scraping**
- **GIS data, Maps, ...**
- **API, External Services**
- **Rest Call ......MS**
- **External Data Stores**
- **LD, LOD**

---

**SNAP4City (C), October 2020**
Standards and Interoperability


https://www.snap4city.org/65
Snap4City vs Formats

• Snap4City is capable to ingest and work with any format:
  – Data exchange: JSON, GeoJSON, XML, HTML, HTML5, DATEX, GTFS, binary, etc.
  – GIS formats: WMF, WFS, heatmaps, ....
  – Table: CSV, XLSX, XLS, database, ...
  – archive file formats: zip, rar, 7z, tgz, ...
  – image formats: png, gif, tiff, ico, jpg, ...
  – video formats: mp4, avi, mov, ...

• Search the format you need to cope on the search box of Snap4City portal!: Snap4City Supported Protocols, adding new protocols
Snap4City: Builder of Sentient Cities Solutions

Dashboards with data driven IOT Applications enforcing intelligence

IOT and data World

My IOT Devices

IOT Applications

Big Data Analytics, Artificial Intelligence

Dashboards and Apps
Snap4City Services also on IOT Edge!!!

IOT Networks
- IOT Brokers
- IOT Gateways
- IOT Edge Devices
- IOT Devices

IOT Applications
- Big Data Analytics, Artificial Intelligence
- Dashboards and Apps

Snap4City (C), October 2020
Snap4City: IOT Directory and data/device Discovery

IOT Networks
- IOT Gateways
- IOT Brokers
- IOT Devices

IOT Edge Devices
- Knowledge Base
- Semantic Reasoners
- (IOT) Discovery
- IOT Directory

IOT Applications
- Dashboards and Apps

Big Data Analytics, Artificial Intelligence

Snap4City (C), October 2020
Further readings

- **HOW TO: create a Dashboard** in Snap4City
- **HOW TO: add a device to the Snap4City Platform**
- **HOW TO: add data sources to the Snap4City Platform**
- **HOW TO: define privacy rules for personal data, produced by the end-users own device**
- **HOW TO: Develop Smart Applications, Snap4City development Life Cycle**
- **HOW TO: HLT vs Ingestion, and HLT vs Widgets**
- **HOW TO: Develop an IOT Application for Data Ingestion**
- **HOW TO: Upload data into Knowledge Base, ServiceMap (triple upload)**
- **HOW TO: Create as set of Devices with BulkProcessing**
- **HOW TO: Create an IOT Device Model**
- **HOW TO: Create an IOT Device Instance from IOT Directory tool**
- **HOW TO: save / export a MyKPI data into a CSV file**
- **HOW TO: produce heatmaps, custom heatmaps on any data**
Further readings

• HOW TO: add a device to the Snap4City Platform
• HOW TO: add data sources to the Snap4City Platform
• HOW TO: add IOT Device data source from external broker to the platform.
• TC9.13: How to upload a local file into your IOT Application
• TC9.16 Web Scraping to get data from web pages
• TC6.3. Creating ETL processes for automated data ingestion and data transformation
• TC9.2. Managing heterogeneous File Ingestion, protocols, formats via IOT applications, and open standards
• TC6.1. Managing DataSets via DataGate: ingest, search, download, upload, annotate, share
• TC6.2. Search on DataGate for Data Sets
• TC2.25. Registering external MicroService calling RestCall services, using it on IOT applications

In Yellow alternative & legacy solutions
Data Gathering and Knowledge Management

- Data ingestion can be performed by using multiple tools:
  - We suggest:
    - IOT App for static and real time data and flow, from IOT Brokers/Devices
    - DataGate for Static Data, upload them as files, or collected from other CKAN
    - WebScraper for scraping data from Web Pages, when authorized!
    - ETL for static / periodic data in PULL
  - See how to test cases:
    - HOW TO: add data sources to the Snap4City Platform
    - HOW TO: define privacy rules for personal data, produced by the end-users own device
    - US6. Developing and using processes for data transformation
    - TC6.1 - Managing DataSets via DataGate: ingest, search, download, upload, annotate, share
    - TC6.3 - Creating ETL processes for automated data ingestion and data transformation
    - TC6.5 - Managing Heterogeneous File Ingestion via ETL processes
    - TC6.9 - ETL processes for multiprotocol and format data ingestion, see on GITHUB for library
    - TC9.2 - Managing heterogeneous File Ingestion, protocols, formats via IOT applications, and open standards
Data Ingestion
Architecture and Kinds
Snap4City Architecture

Data Sources, External Services
Data Sources, Brokers, External Services
PULL Data

Data Ingestion, aggregation, regularization, reconcile:
NIFI, IOT App

Back-End
Knowledge base
Semantic reasoners
Indexing and aggregating
Elastic search
Search and Query, Smart City API
Facet, semantic search
Data Analytics, Simulations
R, Tensor Flow, Python, ….
IOT Applications, Business Logic
Node-RED + Snap4City MicroServices

Front-End
Rendering Acting, Acting, Widgets, MicroApps
User interface, Interactive Dashboard, Drill down, maps, heatmaps

Inform, announce, Act!, warning, alarms, What-IF, …

Snap4City (C), August 2020
Smart City Functional Architecture

**Back-End**
- Knowledge base
  - Semantic reasoners
- Indexing and aggregating
  - Elastic search
- Search and Query, Smart City API
  - Facet, semantic search
- Data Analytics, Simulations
  - R, Tensor Flow, Python, ...
- IOT Applications, Business Logic
  - Node-RED + Snap4City MicroServices

**Data Sources, External Services**
- PULL Data
  - NIFI, IOT App

**Front-End**
- Rendering
  - Acting, Widgets, MicroApps
- User interface
- Interactive Dashboard
- Drill down, maps, heatmaps

**Inform, announce, Act!, warning, alarms, What-IF, ..**

**Transport systems**
- Mobility, parking

**Public Services**
- Govern, events, ...

**Sensors, IOT Cameras, Wi-Fi**
- Environment, Water, energy

**Shops, services, operators**

**Social Media**
- Crawler and Manager

**Smart City**

- Functional Architecture

Snap4City (C), October 2020
Modalities and Strategies for data ingestion

- **Road Graphs**: from GIS, and/or OSM (see Snap4City tool for that), ...
- **Data** of any format via any protocol
- **Structured and non-structured** data (tables and free text, mixt)
- **Static data**/metadata descriptors ingested as: a file or manually via user interface:
  - IOT App from file: https://iot-app.snap4city.org/nodered/nr9xtwc/simple
  - IOT Directory from user interface, ingesting them as Sensors
  - IOT Directory from file, ingesting them as a set of Sensors
  - Manually as MyPOIs which have to be transformed in POIs by the Administrator
  - Data Gate, a module of CKAN, allows to process files as POI, Sensors, etc.
  - ETL Process from file which can be used for ingesting any kind of file
- **Simple data and complex data** as: GIS data, heatmaps, tracks, etc.
- **Real Time Event Driven data** can be ingested by:
  - Automatically by NIFI for the Data Shadow construction, or
  - Producing an IOT App, a WebScraping process, a special tool, an ETL, etc.
Static vs Dynamic Information

• Most of the valuable data are Real Time/Dynamic data, based on Static info.
• For example, data coming from Smart Park or a Smart Light solution
  – They are composed of a set of Devices,
    • the Collection of Devices has in addition common information such as: licensing, gateway access, gateway credentials, description of provider, time of update, etc.
    • each Device has Static information such as:
      – data structure: device ID, Current consumption, temperature, ……
      – GPS coordinates… (if the devices is not a moving one)
      – Classification: nature, subnature, Healthiness criteria, MTTF, etc.
    • Each Device produces a data package/message according to minimum data Structure as Dynamic information, in PUSH towards the Gateway.
  – The Gateway in turn can be:
    • a sort of Broker sending data to other consumer in Push or
    • can provide also data in PULL when requested by some consumer.
• This means that data fully describing the situation is composed by Static and Dynamic aspects that have to described into the platform and reconnected when one access to the device…
What happen to data into the platforms

• **Static** information has to be collected in advance
  – For each **collection of devices**, for each **device**

• **Dynamic** information has to connected to Static one at the ingestion time

• **PLUS**: when data enter into the platform a wide number of information is added/created such as:
  – Internal technical identification, modalities to access to the information
  – Historical data storage
  – Eventual derived data: date and time of ingestion, average, quality level, etc.
  – Rights to access at the data: who can access and to do what?
  – Visual representation of data, may be an image of the Device how it has been installed and where, by who, telephone number for maintenance, etc.
  – Process ID used to ingest the data, possible IOT Applications that are using the data, ...
Snap4City Architecture vs Data Ingestion

Data Sources, External Services
- PULL Data

Data Sources, Brokers, External Services
- Data Driven, Real Time

Data Ingestion, aggregation, regularization, reconcile:
- NIFI, IOT App

Knowledge base
- Semantic reasoners

Indexing and aggregating
- Elastic search

Search and Query
- Smart City API
- Facet, semantic search

Data Analytics, Simulations
- R, Tensor Flow, Python, ...

IOT Applications, Business Logic
- Node-RED + Snap4City MicroServices

Inform, announce, Act!, warning, alarms, What-IF, ...

Front-End
- Rendering Acting, Computing, Widgets, MicroApps
- User interface, Interactive Dashboard, Drill down, maps, heatmaps

Back-End

Snap4City (C), July 2020
Data are **not so Simple as one can imagine**

- Data to be managed into the Smart City IOT are not so simple as one may imagine, and not limited to take into account only IOT Devices.
- **THUS**, a large number of data **TYPES and sources** have also to be addressed:
  - E.g.: external services, heatmaps, trajectories, maps, OD matrices, actuators, personal data, KPI, API descriptors, special widgets, events, predictions, Tweets, posts, GIS, mobile devices, etc.
  - With their complexity of managing data, licensing, etc...
- **THEY** are called **High Level Types:** how and which tool / process can cope with them into the Smart City Platform?... See next!
  - Thus a unified model is needed.
- In 2\textsuperscript{nd} part we have shown how to show specific types of data
### All of them can be shown on Dashboards, what about manipulate them!!!!

<table>
<thead>
<tr>
<th>HLT, High Level Types++</th>
<th>GPS</th>
<th>Static</th>
<th>Dynamic</th>
<th>Single</th>
<th>Time Series</th>
<th>Trajectory</th>
<th>HTTP</th>
<th>How to ingest/change/manage/see</th>
</tr>
</thead>
<tbody>
<tr>
<td>POI (Point of Interest)</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>DataGate, ETL, IOT App, API</td>
</tr>
<tr>
<td>MyPOI data</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Dashboard, IOT App, UserInterface, API</td>
</tr>
<tr>
<td><strong>Sensor data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Dashboard, IOT Directory, IOT App, UserInterface, API</td>
</tr>
<tr>
<td><strong>Sensor Actuator data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Dashboard, IOT App, UserInterface, API</td>
</tr>
<tr>
<td><strong>MyKPI data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Dashboard, IOT App, UserInterface, API</td>
</tr>
<tr>
<td><strong>Dashboard-IOT App (msg)</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Dashboard, IOT App, API</td>
</tr>
<tr>
<td><strong>Dashboard-IOT App real time</strong></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>WS</td>
<td></td>
<td></td>
<td></td>
<td>Dashboard, IOT App, API WS</td>
</tr>
<tr>
<td><strong>Synoptics data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Dashboard, IOT App, UserInterface</td>
</tr>
<tr>
<td>My Personal Data</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Dashboard, IOT App, UserInterface, API</td>
</tr>
<tr>
<td><strong>KPI (metrics) data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Dashboard, IOT App, API, SQL calls x Metrics</td>
</tr>
<tr>
<td>Special Widget (complex)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Dashboard, ETL, special, IOT App, API</td>
</tr>
<tr>
<td>Complex Event (msg)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Dashboard, ETL, special, IOT App, API</td>
</tr>
<tr>
<td>WFS/WMS (GIS data)</td>
<td>Yes</td>
<td>[yes]</td>
<td>[yes]</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Dashboard, GIS tools, or GeoServer, IOT App</td>
</tr>
<tr>
<td>GTFS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>ETL, special [IOT App], MicroApplications</td>
</tr>
<tr>
<td>OD Matrices</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Special tools, MicroApplications</td>
</tr>
</tbody>
</table>
POI: Point of Interest vs Classification

- A POI is defined as an element of a set (collection) and with general info:
  - Nature: ........
  - Subnature: ....

- Specific info for each POI
  - Location: lat, lon
  - A set of Attributes
    - www, email, opening time, phone, cap, address, city, etc.
  - Eventually a link to data

Nature
- Accommodation
- Advertising
- AgricultureAndLivestock
- CivilAndEdilEngineering
- CulturalActivity
- EducationAndResearch
- Emergency
- Entertainment
- Environment
- FinancialService
- GovernmentOffice
- HealthCare
- IndustryAndManufacturing
- IoTDevice
- MiningAndQuarrying
- ShoppingAndService
- TourismService
- TransferServiceAndRenting
- UtilitiesAndSupply
- Wholesale
- WineAndFood

SubNature

Snap4City (C), October 2020
MyKPI and Sensors: examples

- **HLT**: MyKPI
- **Nature**: Industry and manufacturing
- **Subnature**: Chemical
- **Value Type**: Density percentage
- **Value Name**: CloroParaffine
- **Data Type**: float mykpi
- **Value Unit**: %
- **Last Date**: 2019-02-25
- **Last Value**: 87.0
- **Healthiness**: ●
- **Last Check**: 2020-04-03 10:28:12
- **Ownership**: private
- **Organization**: Firenze

- **HLT**: Sensor
- **Nature**: From IOT Device to KB
- **Subnature**: IOT Sensor
- **Value Type**: Battery Level
- **Value name**: Irrigator fioriera Gag
- **Data Type**: float
- **Value Unit**: V
- **Last Date**: 2020-04-01 12:59:00
- **Last Value**: 5.18
- **Healthiness**: ●
- **Last Check**: 2020-04-03 03:28:12
- **Ownership**: public
- **Organization**: Firenze
MyKPI vs IOT Device Sensor

- **MyKPI** can be grouped in a Groups of Values. They may have GPS changing

- **Sensors** can be also grouped, but born directly grouped into an **IOT Device** as defined into the IOT Directory. They have GPS static

- Please note that the "dateObserved" has to be in standard ISO to have Time Series

---

**IOT Device may be sending data as:**

```json
{   "id":"dev45",   "type":"misura",   "attributes": [   {   "name":"dateObserved",   "value":"2020-04-06T17:00:00.000Z",   "type":"time" },   {   "name":"temp 1",   "value":28.976,   "type":"float" },   {   "name":"Hum 1",   "value":35976,   "type":"float" },   ..........   {   "name":"kevi",   "value":721732,   "type":"integer" } ] }
```
Now, it is more clear about what we intend as:

---

**High Level Types**

<table>
<thead>
<tr>
<th>HLT, High Level Types++</th>
<th>GPS</th>
<th>Static</th>
<th>Dynamic</th>
<th>Single</th>
<th>Time Series</th>
<th>Trajectory</th>
<th>HTTP</th>
<th>How to ingest/change/manage</th>
</tr>
</thead>
<tbody>
<tr>
<td>API (Ext. Srv., any prot.)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ETL, Special, IOT App, ...</td>
</tr>
<tr>
<td>External Service (web pag)</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>ETL, Special, IOT App, Web Scraper, ...</td>
</tr>
<tr>
<td>MicroApplication (webapp)</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Dashboard, IOT App, API, FTP, ...</td>
</tr>
<tr>
<td>Heatmap matrix</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Maps, IOT App, MicroService, UserInterface, API</td>
</tr>
<tr>
<td>Synoptics (group)</td>
<td>Yes</td>
<td>Yes</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Dashboard, Special Tools, IOT App, API, ...</td>
</tr>
<tr>
<td>Special Tools (functional)</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>Yes</td>
<td></td>
<td></td>
<td>As MyPersonalData, ...</td>
</tr>
<tr>
<td>Typical Trends (not yet)</td>
<td>(yes)</td>
<td>Yes</td>
<td>(Yes)</td>
<td>(yes)</td>
<td>Yes</td>
<td></td>
<td></td>
<td>MicroApp, Special tools, (API), ...</td>
</tr>
</tbody>
</table>

**Non HLT**

| Traffic Flows (are coming) | (yes) | Yes    | Yes     | Yes    | Yes         | Yes        | Maps, Special tools, API, ...                                  |
| Color Maps                 | Yes    |        | Yes     |        |             |            | Maps, Tables, Special tool, User Interface, API               |
| GTFS (see Sensors, POI)    | Yes    | Yes    | Yes     | Yes    | Yes         |            | Maps, Special tools, API, ...                                  |
| Typical Trajectory (MyKPI) | Yes    | Yes    | Yes     | Yes    | Yes         |            | Maps, Special tools, API, ...                                  |

All of them can be shown on Dashboards, what about manipulate them!!!!

---

**How to Ingest**

Snap4City (C), October 2020
<table>
<thead>
<tr>
<th>High Level Types</th>
<th>Unified Data and Services Model/Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic</td>
<td></td>
</tr>
<tr>
<td>Nature</td>
<td></td>
</tr>
<tr>
<td>SubNature</td>
<td></td>
</tr>
<tr>
<td>Value Type</td>
<td></td>
</tr>
<tr>
<td>Value Name</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td></td>
</tr>
<tr>
<td>Value Unit</td>
<td></td>
</tr>
<tr>
<td>Last Date/Time</td>
<td></td>
</tr>
<tr>
<td>Last Value</td>
<td></td>
</tr>
<tr>
<td>Healthiness</td>
<td></td>
</tr>
<tr>
<td>Last Check</td>
<td></td>
</tr>
<tr>
<td>Ownership</td>
<td></td>
</tr>
<tr>
<td>Status For Admin</td>
<td></td>
</tr>
<tr>
<td>Ownership</td>
<td></td>
</tr>
</tbody>
</table>

**Unified Data and Services Model/Classification**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Date/Time</td>
<td>Date when the last update was made</td>
<td>2020-04-03T10:00:00Z</td>
</tr>
<tr>
<td>Last Value</td>
<td>Value of the last update</td>
<td>4.972</td>
</tr>
<tr>
<td>Healthiness</td>
<td>Measure of the healthiness of the sensor</td>
<td>85</td>
</tr>
<tr>
<td>Status For Admin</td>
<td>Status of the sensor for administration</td>
<td>private</td>
</tr>
<tr>
<td>Ownership</td>
<td>Ownership of the sensor</td>
<td>DISIT</td>
</tr>
</tbody>
</table>
How to ingest/change/manage

- **Dashboard**: by creating a Dashboard that can act/change the values with actuators (2nd Day Slides)
- **API**: you can use the Smart City API to change / provide the values (see 3rd Day Slides)
- **IOT App**: by developing an IOT Application on Node-RED exploiting Snap4City MicroServices (4th Day Slide, and in part in these slides)
- **DataGate**: you can use the DataGate tool to ingest the Static data (mainly Open), and publish them
- **ETL**: by developing an ETL process, and put it in execution via DISCES
- **IOT Directory**: you can use the IOT Directory tool to change the parameters, and set up the ingestion process, via IOT Brokers, IOT Devices, IOT Edge.
- **Special**: by using a special tool for developing a process, or for creating SVG Synoptics
- **UserInterf**: there is a number of Tools with Graphic User Interface that you can use to change the values, see in the menu on the left.
- **Web Scraper**: by creating a Web Scraping process and exploiting the results into an IOT Application
- **As MyPersonalData**: they are substantially MyPersonalData
- **From third party tools**: they can be manipulated by using third party tools
IOT Applications

- Created on browser
- A huge number of Protocols
  https://www.snap4city.org/65
- Scheduled internally and managed as Container
- Largely diffused approach as Node-RED
- Large number of Snap4City MicroServices, well documented
- PUSH and PULL models
- Simple mechanism to add new Features
- Very diffused in IOT
- Static and Dynamic data models depending on IOT Broker capabilities
- Scalable on Cloud
- Also present in IOT Edge devices

ETL processes

- Created with Spoon editor on VM (on premise or via remote access to VM)
- A Large number of protocols
  https://www.snap4city.org/65
- Scheduled by DISCES in the back office
- Well known data warehouse model
- Well documented for the process
- Only PUSH models
- Complex mechanism to add a new functionality
- Very diffused in Data transformation
- Static and Dynamic data models well linked ..
# Data Ingestion Methods Comparison

<table>
<thead>
<tr>
<th></th>
<th>Datagate</th>
<th>ETL</th>
<th>IoTApp</th>
</tr>
</thead>
<tbody>
<tr>
<td>types of data managed</td>
<td>S</td>
<td>S, P</td>
<td>S, P, RT</td>
</tr>
<tr>
<td>Data protocol types managed</td>
<td>PULL</td>
<td>PULL</td>
<td>PULL and PUSH</td>
</tr>
<tr>
<td>Scheduling</td>
<td>external</td>
<td>external</td>
<td>internal</td>
</tr>
<tr>
<td>Flows to manage N instances of the same dataset</td>
<td>N</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>Users’ technical level</td>
<td>without</td>
<td>medium/high</td>
<td>low</td>
</tr>
<tr>
<td>Development time</td>
<td>1, 2 hours</td>
<td>1, 2 weeks</td>
<td>3, 4 days</td>
</tr>
<tr>
<td>Semantic (KM4City)</td>
<td>standard template</td>
<td>ad hoc (manual)</td>
<td>ad hoc (semi-automatic)</td>
</tr>
<tr>
<td>Developed number</td>
<td>1334 datasets</td>
<td>162</td>
<td>76</td>
</tr>
<tr>
<td>Mean number of blocks</td>
<td>0</td>
<td>120.333</td>
<td>27.67</td>
</tr>
<tr>
<td>Mean number of lines of code</td>
<td>0</td>
<td>275</td>
<td>229</td>
</tr>
</tbody>
</table>
Dictionary for Data Fields
Semantics and Technical Meaning
Unified Data and Services Model/Classification

Semantic Nature

• Exists a Dictionary for the 4 categories

• They are related each other and not all values are possible

• Right setting lead to right rendering on graphs and automated combinations and processing

• The Dictionary is used by many tools

Technical meaning

• Value Type

• Value Unit

SubNature

• SubNature

• SubNature

Value Unit

Snap4City (C), October 2020
### Two Examples

<table>
<thead>
<tr>
<th><strong>HLT: MyKPI</strong></th>
<th><strong>HLT: Sensor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature: Industry and manufacturing</td>
<td>Nature: From IOT Device to KB</td>
</tr>
<tr>
<td>Subnature: Chemical</td>
<td>Subnature: IOT Sensor</td>
</tr>
<tr>
<td>Value Type: Density percentage</td>
<td>Value Type: Battery Level</td>
</tr>
<tr>
<td>Value Unit: %</td>
<td>Value Unit: V</td>
</tr>
<tr>
<td>Data Type: float mykpi</td>
<td>Data Type: float</td>
</tr>
<tr>
<td>Value name: CloroParaffine</td>
<td>Value name: Irrigator fioriera Gag</td>
</tr>
<tr>
<td>Last Date: 2019-02-25</td>
<td>Last Date: 2020-04-01 12:59:00</td>
</tr>
<tr>
<td>Last Value: 87.0</td>
<td>Last Value: 5.18</td>
</tr>
<tr>
<td>Healthiness: [green]</td>
<td>Healthiness: [green]</td>
</tr>
<tr>
<td>Last Check: 2020-04-03 10:28:12</td>
<td>Last Check: 2020-04-03 03:28:12</td>
</tr>
<tr>
<td>Ownership: private</td>
<td>Ownership: public</td>
</tr>
<tr>
<td>Organization: Firenze</td>
<td>Organization: Firenze</td>
</tr>
</tbody>
</table>
Example of Energy and its Value Units

### Dictionary Editor for Data Fields

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Dictionary Type</th>
<th>Description</th>
<th>Parent Value Name</th>
<th>Child Value Name</th>
<th>Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>energy</td>
<td>value type</td>
<td>Energy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Energy, supply</td>
<td>substantide</td>
<td>Energy Supply</td>
<td>UtilitiesAndSupply</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kWh</td>
<td>value unit</td>
<td>Kilowatt per hour</td>
<td>energy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MWh</td>
<td>value unit</td>
<td>Megawatt per hour</td>
<td>energy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UtilitiesAndSupply</td>
<td>nature</td>
<td>Utilities and supply</td>
<td>Accommodation or Office</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wh</td>
<td>value unit</td>
<td>Watt per hour</td>
<td>energy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First << Prev 1 Next >> Last
Data Ingestion Strategy and Tools
Snap4City Architecture vs Data Ingestion

Data Sources, External Services
- PULL Data

Data Sources, Brokers, External Services
- Data Driven, Real Time

Data Ingestion, aggregation, regularization, reconcile:
- NIFI, IOT App

Knowledge base
- Semantic reasoners

Indexing and aggregating
- Elastic search

Search and Query
- Smart City API
  - Facet, semantic search

Data Analytics, Simulations
- R, TensorFlow, Python, ...

IOT Applications, Business Logic
- Node-RED + Snap4City MicroServices

Inform, announce, Act!, warning, alarms, What-IF, ...

Rendering
- Acting, Widgets, MicroApps
  - User interface, Interactive Dashboard, Drill down, maps, heatmaps

Snap4City (C), August 2020
Snap4City Data Ingestion Flow Diagram

Static Models

- IOT Broker Registration
- IOT Device Model Reg.
- IOT Device Registration

Dynamic Models

- Real Time data in Pull
  - Any gateway, server

IOT Orion Broker

Knowledge Base

Semantic Reasoners

Indexing and Aggregating

NIFI, Elastic Search

Information, File

IOT Apps

Snap4City Tools

IOT Device/Gateways
• **Static** (unified model for multidomain indexing)
  – Geodata, Open Data as POI, Data Bases, records, etc.
  – They change over time sporadically

• **Models** (Registration of IOT Device Models, IOT Devices, Brokers)
  – Registration of entities with their metadata and data structures
    • Brokers, Devices, structures of real time data, machine models for IOT Industry 4.0, sensor models/structure, etc.

• **Time Series** (dynamic data)
  – Any instance of IOT Devices over time and space with any kind of entity relationship each other and with other city entities
Checking data ingestion results

- Data Inspector
- ServiceMap, ASCAPI
  - LOG / LOD viewer
  - Super Service Map
- IOT Broker
- ACAPI: Swagger
- IOT Directory
- Data Inspector
- ServiceMap, ASCAPI
- DevDash (Kibana)
- Elastic Search

Some functionalities are limited to certain roles
• The **IOT Orion Broker** can be feed by means of
  – IOT **App** of Snap4City (to implement Agents and/or Adapters)
  – IOT **Agents** and/or **NGSI Adapters** by FiWare for different protocols
  – IOT brokers of any kind

• **IOT App, IOT Agents, Adapters and IOT Brokers** can
  – be on IOT Edge
  – be on other clouds and services
  – work on a large range of different protocols and kinds
  – have or not Snap4City tools and libraries installed
• The **Internal IOT Orion Brokers** at Snap4City are used as gate for data ingestion and actuations. Since they are
  – connected with the IOT Directory and discovery of the Knowledge Base to make easy the production of Dashboards by wizard, Data Inspector;
  – Synchronized automatically with NIFI/ElasticSearch for the Automated Data Shadow and Indexing
  – Ready to be used by IOT App to subscribe for creating even driven IOT Apps, on IOT Edge and Cloud, etc...
  – Compatible and harmonized with FiWare networks
• Direct Data Ingestion is also possible

• Data Storage: KB, and I&A, reported on right side can be acted via API REST Call
  – for direct feeding data into store and retrieval,
  – which can be exploited by:
    • IOT Applications
    • applications in Python, R Studio, Java
Data Inspector vs Data Processes Details

Some functionalities are limited to certain roles
Context and Problems

• Smart city context includes solutions that presents
  – **Data**: heterogeneous, large volume, several protocols, legacy data systems, several semantics, real time, multiple domains, etc..
  – **Processes**: several of different kinds, aperiodic, periodic, event driven, ....
  – **Users**: owners, responsibles, developers, final users, etc.
  – **Relationships**: among data, processes, users and mixt
  – **Non Functional Requirements**: security, GDPR, reliability, quality, scalability, etc.
    • Interoperability: legacy, protocols, modularity,

• **Data Ingestion**: the models and mechanism for data gathering
• **Data Inspection**: the model and solution to identify problems, and understand solution
Data Inspector (Digital Twin info) Major Submodels

- Digital Twin
  - Device and sensors data
  - Values
  - Healthiness criteria and values
    - Machine learning tools
  - Images and physical world
  - Licensing
  - Users

- Users
  - Defined the Data and Devices
  - Defined the processes
  - Create dashboards
  - Etc.

- Process Views
  - Device Management tool
  - Data ingestion processes
    - ETL, IOT Apps
  - Data storage access views
    - Index views
    - Relationships view
  - Data Analytics and Transformation
    - IOT App, R Studio, Python
  - Data Rendering Dashboards
    - Synoptics
  - Processes’ Developers
Data Inspector: all you need to know about data, data sources and ingestion processes
Data Inspector (open on your left side menu)

- Cross Filtering on the basis of:
  - MAP: pan and zoom → lock, center on GPS coordinates of the user
  - Data Source Classification:
    - faceted filtering
    - full text search
  - Click on data source to see it on map, and see the graphics representation, just to learn how widgeting it.
  - Full Text Search if you remember some desc...

- Selecting a Data Source on Map:
  on its Pin you can see:
  - Real time data
  - Time trend: 4 hours, 12 hours, 1 day, 1 week, 30 days.
  - Full status and description (only for Administrators)
• Click with the mouse on it

Knowledge Base view

Some functionalities are limited to certain roles
Some functionalities are limited to certain roles

- Specific values of selected
- Information of the values of the other sensors on the same device
- View Trends, marking problems, healthiness by point according to a Fuzzy model
- Marking problems for future machine learning processes (separate tool)
• Two different criteria
  – **H1**: at least an event in the last 24 hours
  – **H2**: machine learning for most of Sensors devices
Details regarding the IOT Ingestion process

- For IOT Device data
- IOT Broker details

Some functionalities are limited to certain roles
Details regarding the Ingestion process

- For ETL processes
- Scheduling details and status

Some functionalities are limited to certain roles
Some functionalities are limited to certain roles
**HLT: From Dashboard to IOT APP**

- Click with the mouse on it
HLT: Dashboard-IOT App (From Dashboard to IOT APP)

Snap4City (C), October 2020
HLT: Sensor-Actuator (From Dashboard to IOT Device)
Some functionalities are limited to certain roles

- Click with the mouse on it
The fields shown may be present or not depending on the HLT and on the information received.
DevDash based assessing Elastic Search Data via Kibana
Operator Business Analysis
Dashboards: DevDash, AMMA

- Dynamic Filtering, Adaptable, ...
- Full data details, drill down,...
- Synergic with Data Inspector which addresses data relationships, processing and information
Data Analyzer: DevDash

COUNT EVENTS

- Count over dates:
  - 2019-11-04: 1,000,000
  - 2019-11-05: 2,000,000
  - 2019-11-06: 3,000,000

EVENT COUNTS

- Count over time periods:
  - 00:00 - 01:00: 10,000
  - 01:00 - 02:00: 20,000
  - 02:00 - 03:00: 30,000

Facet Fields

- Organization:
  - Select...
- Kind:
  - Select...
- API:
  - Select...

Device Name

- temp_station03: 5.27%
- temp_station04: 5.29%
- Water_detector04: 5.95%
- Water_detector06: 5.94%
- Water_detector02: 5.94%
- DISIT: 70.77%
- think: 16.28%

Distribution Value Type
Set up of the Road Graph on Knowledge Base, performed with an open source tool
Snap4City Architecture vs Data Ingestion

Data Sources, External Services
- PULL Data
- Data Sources, Brokers, External Services
  - Data Driven, Real Time

Data Ingestion, aggregation, regularization, reconcile:
- NIFI, IOT App

Back-End
- Knowledge base
  - Semantic reasoners
- Indexing and aggregating
  - Elastic search
- Data Analytics, Simulations
  - R, Tensor Flow, Python, ...
- IOT Applications, Business Logic
  - Node-RED + Snap4City MicroServices

Search and Query, Smart City API
- Facet, semantic search

Front-End
- Rendering
  - Acting, Widgets, MicroApps
  - User interface, Interactive Dashboard, Drill down, maps, heatmaps

Inform, announce, Act!, warning, alarms, What-IF, ..
Km4City: Knowledge Base

- Multiple DOMAINS
- Geospatial reasoning
- Temporal reasoning
- Metadata
- Statistics
- Risk and Resilience
- Licensing
- Open and Private Data
- Static and Real time
- IOT/IOE

Ontology Documentation:
http://www.disit.org/6506
http://www.disit.org/6507
http://www.disit.org/5606
http://www.disit.org/6461

Schema: http://www.disit.org/km4city/schema
RDF version: http://www.disit.org/km4city.rdf
Set up of the Knowledge Base, KB

• The KB starts empty, it has to be initialized with the Road Graph(s) of interest.

• Road Graphs can be obtained from:
  – GIS of the municipalities, regional govern, etc.
  – Open Street Map, OSM
  – Etc.

• See this note on KM vs OSM: https://www.snap4city.org/397

• Snap4City provides a tool for feeding the KB with OSM
  – TC5.10- Open Street Map ingestion process
  – OSM2KM4C tool is included into KBSM, VM and Docker
    https://www.snap4city.org/471
  – Tool: https://github.com/disit/osm2km4c

• The load of city of 1 Million of inhabitants can be done in few hours.
Snap4city Data Ingestion Flow Diagram

- **Static or quasi Static Data:** POI, etc.
- **IOT Broker Registration:** sporadic
- **IOT Device Model Reg.**
- **IOT Device Registration:** periodic
- **Real Time data in Pull:** Any gateway, server
- **IOT Device/Gateway:** push
- **IOT Brokers:** push
- **IOT Devices:** push

**IOT Orion Broker**
- **IOT App:** push
- **IOT Orion Agent:** push

**IOT Directory**
- **Dynamic:** pull
- **(1c) static:** pull

**IOT Orion Broker**
- **IOT App:** push

**Knowledge Base**
- **Semantic Reasoners:** triples
- **Indexing and Aggregating:** NIFI, Elastic Search
- **OSM to Road Graph Setup**
- **SURI Link**

**IOT Apps**
- **Snap4City Tools**
- **Information, File**
- **IOT Device/Gateway**

Snap4City (C), October 2020
Linked Open Data

LOG: https://log.disit.org

Schema: http://www.disit.org/km4city/schema
RDF version: http://www.disit.org/km4city.rdf
Usage of the ServiceMap and Knowledge Base Browsing
Search by Shape and Distance

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

ServiceMap: https://servicemap.km4city.org

Snap4City (C), October 2020
Views of the Knowledge Base

- How pass from ServiceMap to Linked Open Graph, Linket Data view tool
Main Organizations/areas

- Antwerp area (Be)
- Capelon (Sweden)
- 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)
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
Multiple Knowledge Bases Km4City/ServiceMap

- A Mobile App may refer to one Smart City API Server (for Area 1) via SUPER and receive data from the Federated SUPERS (Area 2) if navigation, queries, etc. are leading to discover out of the addressed KB.
  - SUPER can be used for creating redundant and/or balanced distributed solutions for Federated KB. See Area 2, the two KB in the front.
  - Federated SUPER ServiceMap can have overlapped KB even totally.
  - A Mobile App can be developed to support multiple Smart City API servers, for balancing and
- The usage of Super (ServiceMap) is not mandatory so that separate services can be produced as well.
- SuperServiceMap and ServiceMap presents the same Smart City APIs.
Super Service Map

- ServiceMap do not need to permanently share data
- Distributed Searches are performed with $o(1)$ complexity
- Each ServiceMap can be of any size
- Each ServiceMap may have different number of services
- Services of SCAPI in ServiceMap Can be georeferenced or not
- Clients can pass from one SuperServiceMap to another transparently: moving devices
- The network of ServiceMap can be reconfigured dynamically
- Results from aa API rest call are provided in real time also when the size of the network is large
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
• **Km4City is the reference ontology for Snap4City**, It allows to:
  – keep connected city entities each other:
    • Semantic Index, reticular
    • Perform spatial, geographic, and temporal reasoning
  – Discover city entities and their relationships via IOT App and Smart City API:
    • IOT devices, IOT sensors, city elements, roads, services, Brokers, etc. etc.
  – Provide access via Advanced Smart City API
  – Federate other Km4City Knowledge Bases, the approach allows to scale geographically and create redundancies, improving performances

• **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
Tuscany, Florence, Pisa...
Upload Data Sets of POIs Via IOT App
IOT Applications

- IOT Edge App
- IOT Application
- Bib APP
- ChargingStations
- Deprecated - SIIMobilityControlRoom
- SamsungGalaxy54BarCode
- esercitazione
- iot-App

Owner: badii, panesi, pb3, owner: samolaudy, owner: tester1s, owner: tester2, owner: comundashas.
HOW TO learn creating IOT Applications

• Follow the IOT Application development tutorial from Snap4City
  – 3° Part
  – https://www.snap4city.org/577
IOT Applications Development

- MicroServices collections
- My IOT Applications
- IOT App. Editor
- ServiceMap Discovery
- Dashboard Collection, Editor and Wizard
- Knowledge Base, Km4City
- Resource Manager

Generating IOT App With Dashboard
Sharing/saving reusing IOT App

Snap4City (C), October 2020
Data Ingestion is mainly via IOT Applications
**Snap4city Data Ingestion Flow Diagram**

- **Static or quasi Static Data**: POI, etc.
- **IOT Broker Registration**
- **IOT Device Model Reg.**
- **IOT Device Registration**
- **Real Time data in Pull**
  - Any gateway, server (periodic)
- **IOT Orion Broker**
  - Pull
  - Push
- **IOT Directory**
  - Pull
  - Dynamic
- **IOT Orion Broker**
  - Pull
  - Push
- **IOT App**
  - Pull
- **Knowledge Base**
  - Semantic Reasoners
- **Indexing and Aggregating**
  - NIFI, Elastic Search
- **Snap4City Tools**
- **IOT Device/Gateway**
- **IOT Brokers**
- **IOT Devices**
- **Adapter Information, File**
- **Adapter Dynamic**
- **Adapter Automatic**

**SURI Link**

**Subscription Note**

**Real Time**
Static data

Pre-requirements:
• Knowledge Base main categories (Service Map or )

Register Static Data on Snap4City KB:
1. ‘Regularize’ your data putting them in csv file (possible fields available here: https://www.snap4city.org/589 )
2. Create your IoTApp starting from the sample available here:
   – Description: https://www.snap4city.org/596
3. Upload your csv containing N rows, each row is a Point of Interest
Regularization

1. Create your csv (with 33 fields in the header). Here after some of them (M = to be filled Mandatory):
   - id: Unique POI identifier (Recommended, if not present is created starting from the file name)
   - subCategory (M): KM4City category (ex: "Museum")
   - latitude (M): POI latitude (ex: "43.7936")
   - longitude (M): POI longitude (ex: "11.2604")
   - province: (ex.: "FI")
   - nameITA/nameENG: POI name
   - city: (ex.: "FIRENZE")
   - postalcode: (ex.: "50100")
   - streetAddress: (ex.: "via di Santa Marta")
   - civicNumber: (ex.: "3")
   - descriptionShortENG : (ex: "Exhibition...")
   - url: (ex: "https://www.snap4city.org")
   - ... others...here: [https://www.snap4city.org/589](https://www.snap4city.org/589)
IoTApp

Create your IoTApp:

- Login > IoT Application > 'Create New' button
- Wait some minutes
- Copy the Source code available here: https://www.snap4city.org/drupal/sites/default/files/files/iotapp-poi-ingestion.txt
- Import the code and save ('Deploy' button)
Modify your IoTApp:

- Open the ‘get simple block’ and search for your App id (ex: ‘nr8jero’)
- Put your mail and credentials on the ‘send email form ..’ block
- Go to the page ‘https://iot-app.snap4city.org/nodered/nr8jero/simple’ and upload your csv file
- Add your e-mail credential
IoTApp (2)

Modify your IoTApp:

- Send the file you receive to us, or copy in a file the triples available in the web page and send them us

- Reference people: snap4city@disit.org

---

File Snap4city_static_POI.csv uploaded!

Result sent to michela.paolucci@unifi.it

---

```xml
<http://www.disit.org/km4city/resource/Museum_FI_history> <http://www.w3.org/2003/01/geo/wgs84_pos#lat> "43.7938" .
<http://www.disit.org/km4city/resource/Museum_FI_history> <http://www.w3.org/2003/01/geo/wgs84_pos#geometry> "POLYGON (POINT (43.7938 11.2604) POINT (43.794 11.25987) POINT (43.7935 11.25987) POINT (43.7938 11.2604) POINT (43.7938 11.2604))" .

<http://www.disit.org/km4city/resource/Theatre_FI_xx> <http://www.w3.org/2003/01/geo/wgs84_pos#lat> "43.6987" .
<http://www.disit.org/km4city/resource/Theatre_FI_xx> <http://www.w3.org/2003/01/geo/wgs84_pos#geometry> "POINT (43.6987 11.2587)" .
```

---

How to load triples into the KB (Admin and premise)
Snap4city Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server

periodic

IOT Device

IOT Devices

IOT Brokers

pull

IOT App

sporadic

(1c) static

IOT Directory

IOT Orion Broker

IOT Orion Broker

IOT App

Adapter

Information, File

IOT Apps

Snap4City Tools

IOT Device/Gateway

subscription note

Knowledge Base
Semantic Reasoners

Indexing and Aggregating
NIFI, Elastic Search

SURI Link

Real Time

Information, File

IOT Apps

Snap4City Tools

IOT Device/Gateway
Load Triples

Pre-requisites:
- Available only for your IOT App on premise and for administrators

Load Static Data on Snap4City KB:
- Register Static Data on Snap4City KB (point 1 and 2 on the right)
- Connect the additional blocks present in the IOT App and save (‘Deploy’ button)
- Upload your csv

Register Static Data on Snap4City KB:
1. ‘Regularize’ your data (csv)
2. Create your IoTApp
3. Upload your csv

https://www.snap4city.org/596
Load Triples

- Additional Option Only for ONPREMISE Snap4City versions and for administrators
- In case you need to upload ONLY on the Knowledge Base a set of data containing static data and dynamic data all together:
  - Example: Cultural Events or weather predictions
- Use a predefined IoT App available here [https://www.snap4city.org/594](https://www.snap4city.org/594) (with the copy and past method, you can create your Mobile App)
- Add:
  - Inject block
  - Function block
  - Debug block
- The function block must contain ...
- Now you can see the triples to be updated in the Knowledge Base in the Debug Tab

Snap4City (C), October 2020
Data Ingestion of Time Series via IOT Brokers
Snap4City Architecture vs Data Ingestion

Data Sources, External Services
- PULL Data

Data Ingestion, aggregation, regularization, reconcile:
- NIFI, IOT App

Knowledge base
- Semantic reasoners

Indexing and aggregating
- Elastic search

Data Analytics, Simulations
- R, Tensor Flow, Python, ...

IOT Applications, Business Logic
- Node-RED + Snap4City MicroServices

Search and Query, Smart City API
- Facet, semantic search

Inform, announce, Act!, warning, alarms, What-IF, ..

Front-End
- Rendering
- Acting
- Widgets
- MicroApps
- User interface, Interactive Dashboard, Drill down, maps, heatmaps

Snap4City (C), August 2020
Snap4city Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server

periodic

IOT Orion Broker

IOT Directory

IOT Orion Broker

Knowledge Base

Semantic Reasoners

Indexing and Aggregating

NIFI, Elastic Search

Based on Brokers

Information, File

IOT Apps

Snap4City Tools

IOT Device/Gateways

IOT Devices

IOT Brokers

IOT App

Adapter

Automatic

push

push

push

push

push

pull

pull

pull

sporadic

(1c) static

Dynamic

subscription note

SURI Link

Real Time
The process

- IOT Network concepts, See Course 2020 Part 3
  - IOT Devices, IOT vs Dashboards, etc.

- IOT Broker registration if not performed, See Course 2020 Part 3
  - IOT Brokers can be Internal or External (managed by third party)
The process

- **IOT Device registration** on IOT Directory: concept
  - the IOT Directory performs the registration on KB and NIFI for automated Data Shadow storage

- **HOW TO: register IOT Device(s)** on IOT Directory:
  A. Single IOT Device **Manually from Zero** or by **IOT Device Model** (via user interface IOT Dir)
  B. in **Bulk** by uploading a File with IOT Devices via the IOT Directory user interface + CSV file
  C. Automatically via an **IOT App using special Block/MicroService having** an IOT Device Model
  D. Automatically by activating a **querying the IOT Broker** from the IOT Directory (for External)
IOT Directory
IOT Network Manager vs Final User

Network of IOT Brokers

IOT Broker

IOT Directory

Discovering

IOT Network Manager

Discovering

IOT Application

Dashboard Wizard

My IOT Device

Final user Manager

Register

Registering

Browsing

Knowledge Base, Km4City

Knowledge and Storage Data from the Field and City
Main Features of the Snap4City IOT Directory:

- **IOT Directory is a technology of Snap4City ONLY**
- **Register IOT Brokers**
  - Different kind of Brokers, different kinds of authentications and protocols
  - Registered IOT Orion Brokers can be queried for collecting their managed devices (typically for External IOT Brokers), so that those IOT Devices are registered
  - IOT Brokers/Gateways are registered on NIFI to send messages into the Data Shadow, automatically
- **Register IOT Devices**: singularly or at groups (in Bulk)
  - Registration can be custom or based on IOT Device Model
  - IOT Edge are registered as IOT Devices as well
  - Registered IOT Devices are saved into local DataBase and into the Knowledge Base
- **Provide support for security aspects**:  
  - Generation of Certificates, Keys, etc., according to the model
  - Collection of keys when IOT devices are on some IOT Gateway or Second Level IOT Broker.
- **Manage Ownership and Delegation** for
  - IOT Brokers, IOT Devices, IOT Device Values also called Sensors/actuators, IOT Device Models
## IOT Directory Features vs Users Roles

<table>
<thead>
<tr>
<th>Entities</th>
<th>what</th>
<th>By using IOT Directory and:</th>
<th>Manager</th>
<th>AreaManager</th>
<th>ToolAdmin/RootAdmin</th>
<th>IOT App microservices</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOT Sensor/Actuator</td>
<td>Browse, use</td>
<td>Several Tools</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Delegate</td>
<td>API, ..</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Discovery</td>
<td>KB, API, ..</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>IOT Devices</td>
<td>Browse, use</td>
<td>Several Tools</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Yes (use)</td>
</tr>
<tr>
<td></td>
<td>Create, change, delete</td>
<td>API, ..</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Register in Bulk</td>
<td>API, ..</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delegate, Change Owner</td>
<td>API, ..</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Discovery</td>
<td>KB, API, ..</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>IOT Device Models</td>
<td>Browse, Use</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Yes (use)</td>
</tr>
<tr>
<td></td>
<td>Create, change, delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delegate, change ownership</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOT Brokers</td>
<td>Browse, use</td>
<td>use</td>
<td></td>
<td></td>
<td></td>
<td>Yes (use)</td>
</tr>
<tr>
<td></td>
<td>Register/change/Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delegate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Periodic Update</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
IOT Broker Registration
Snap4city Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server

periodic

pull

sporadic

(1c) static

Dynamic

IOT Directory

sporadic

subscription note

Real Time

push

pull

IOT Orion Broker

IOT Orion Broker

Indexing and Aggregating
NIFI, Elastic Search

SURI Link

Knowledge Base
Semantic Reasoners

Registration triples

IOT Apps

Snap4City Tools

IOT Device/Gateways

Information, File

130

Snap4City (C), October 2020
# IOT Brokers

## Table of IOT Brokers

<table>
<thead>
<tr>
<th>IOT Broker</th>
<th>Access Link</th>
<th>Access Port</th>
<th>Protocol</th>
<th>Ownership</th>
<th>Organization</th>
<th>Owner</th>
<th>Created</th>
<th>Edit</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>iotobsf-smartbed</td>
<td>192.168.1.47</td>
<td>8443</td>
<td></td>
<td>DELEGATED</td>
<td>SmartBed</td>
<td>angelo.difino</td>
<td>2019-11-29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mqttUNIFI</td>
<td>192.168.110</td>
<td>1883</td>
<td>mqtt</td>
<td>DELEGATED</td>
<td>DISIT</td>
<td>iotdirectory.dist</td>
<td>2018-02-07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mqttUNIMI</td>
<td>195.149.129.184</td>
<td>1884</td>
<td>mqtt</td>
<td>DELEGATED</td>
<td>DISIT</td>
<td>iotdirectory.dist</td>
<td>2018-04-30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orionAntwerp-UNIFI</td>
<td>broker3.snap4city.org</td>
<td>8080</td>
<td></td>
<td>PUBLIC</td>
<td>Antwerp</td>
<td>iotdirectory.antwerp</td>
<td>2019-06-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orionFinland</td>
<td><a href="https://ngsi.fff.fi">https://ngsi.fff.fi</a></td>
<td></td>
<td></td>
<td>PUBLIC</td>
<td>Helsinki</td>
<td>iotdirectory.helsinki</td>
<td>2018-11-21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orionFirenze-UNIFI</td>
<td>192.168.117</td>
<td>8443</td>
<td></td>
<td>PRIVATE</td>
<td>Firenze</td>
<td>iotdirectory.firenze</td>
<td>2019-10-28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orionHelsinki-UNIFI</td>
<td>broker2.snap4city.org</td>
<td>8080</td>
<td></td>
<td>PUBLIC</td>
<td>Helsinki</td>
<td>iotdirectory.helsinki</td>
<td>2019-06-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orionLonatoDelGarda-UNIFI</td>
<td>192.168.3.46</td>
<td>8443</td>
<td></td>
<td>DELEGATED</td>
<td>LonatoDelGarda</td>
<td>iotdirectory.lig</td>
<td>2019-10-28</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Details:**
- **IP:** 192.168.112
- **Latitude:** 43.76666
- **Longitude:** 11.26242
- **Login:** shadow
- **Password:** shadow

Showing 1 to 10 of 17 entries
Register a New IOT Broker
Please note on IOT Brokers

• An IOT Broker
  – is associated with an Organization
    • Each Organization has a Knowledge Base of reference (KB, ServiceMap)
    • Each KB may host multiple Organizations and addresses multiple Geographic areas
  – can be an IOT Orion Broker which
    • is compliant with a specific NGSI version: V1, V1-2, V2, etc...
  – can be an IOT Orion Broker configure to
    • expose different authentication methods: K1/K2, Certificate, etc.
    • be accessible from IOT Devices and IOT App in Cloud only
    • be accessible from Internet
IOT Orion Broker Network

- **IOT Broker can be Internal (on Snap4City Cloud)**
  - Registered and automatically connected on NIFI and Data Shadow
  - Registration of IOT Devices is passing from IOT Directory

- **IOT Broker can be External (managed by third party)**
  - Registration of IOT Devices is managed by third parties
  - The registered IOT Devices can be collected and queried from the IOT Directory as well

- **IOT Brokers can be networked**
  - Services, Service paths: for managing the IOT Broker network (in progress)
  - Multi-tenancy: more than one user/org on the same IOT Broker (to be released)
IOT Device Model
Snap4city Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server

periodic

IOT Device

pull

IOT App

sporadic

(1c) static

IOT Directory

Dynamic

IOT Orion Broker

IOT Orion Broker

push

IOT App

Adapter

push

IOT Device

Adapter

push

IOT Device

Adapter

push

IOT Brokers

IOT Brokers

push

IOT App

Knowledge Base

Semantic Reasoners

Indexing and Aggregating
NIFI, Elastic Search

SURI Link

subscription note

Real Time

Information, File

IOT Apps

Snap4City Tools

IOT Device/Gateway

Snap4City (C), October 2020
IOT Device Data Model (1)

- **IOT Broker**
  - Name of the Brokers: among those registered
  - Protocol: NGSI, AMQP, MQTT, etc..
  - Format: CSV, JSON, XML.
  - Service/Tenant:
  - ServicePath:

- **Info**
  - Name (Identifier)
  - Model: Custom or Model ID
  - DeviceType: ..a string..
  - MAC address: ...optional...
  - Edge-GW: Raspberry, Android, ...
  - Edge-GW: URI
  - Producer
  - Owner
  - Freq: ..... Sec
  - Keys: K1, K2
IOT Device Data Model (2)
## IOT Device Data Model (3): Attributes

<table>
<thead>
<tr>
<th>Where</th>
<th>IOT Device Model</th>
<th>IOT Device</th>
<th>A Temporal Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOT Broker</td>
<td>Broker: <strong>OrionUNIFI</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOT Broker</td>
<td>Protocol: <strong>NGSI</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td>ID: string</td>
<td>ID: <strong>park45</strong></td>
<td><strong>park45</strong></td>
</tr>
<tr>
<td>Position</td>
<td>GPS: lat, long</td>
<td>GSP Position: <strong>43.12, 11.34</strong></td>
<td>GSP Position: <strong>44.12, 11.12</strong></td>
</tr>
<tr>
<td>Static attribute</td>
<td>Description: string</td>
<td>Description: <strong>“parking massaia”</strong></td>
<td></td>
</tr>
<tr>
<td>Static attribute</td>
<td>Location: string</td>
<td>Location: <strong>“Via Massaia”</strong></td>
<td></td>
</tr>
<tr>
<td>Static attribute</td>
<td>Civic Number: string</td>
<td>Civic Number: <strong>3</strong></td>
<td></td>
</tr>
<tr>
<td>Static attribute</td>
<td>MaxCapacity: number, cars</td>
<td>MaxCapacity: <strong>456</strong></td>
<td></td>
</tr>
<tr>
<td>Values</td>
<td>dateObserved: Timestamp</td>
<td></td>
<td><strong>23-12-2019T20:13:12...</strong></td>
</tr>
<tr>
<td>Values</td>
<td>FreeSlots: Integer, #</td>
<td></td>
<td><strong>345</strong></td>
</tr>
<tr>
<td>Values</td>
<td>Humidity: float, %</td>
<td></td>
<td><strong>25,5</strong></td>
</tr>
<tr>
<td>Values</td>
<td>Temperature: float, celsius</td>
<td></td>
<td><strong>34</strong></td>
</tr>
</tbody>
</table>
Model meaning

• **ID:** is the unique identifier for reconnecting Temporal Instances with registered IOT Devices

• **Static Attributes:**
  – Are typically associated with instances of the IOT Device. E.g.: You have a set of parking areas, each of them is located in a specific street, and has its one name, etc.
  – Different kinds of attributes can be set for each SubNature. Their definition has to be prepared into the Knowledge Base 😊 for automated indexing.

• **Values:** they are time varying variables (temporal values/instances)
  – They change over time, the time stamp of the time serie is conventionally «dateObserved» in Snap4City
  – In new *SensorMobile* HLT, also GPS can be changing over time as in the MyKPI
IOT Device Registration
many possibilities
Snap4city Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server

dynamic

(1c) static

IOT Directory

Knowledge Base
Semantic Reasoners

Indexing and Aggregating
NIFI, Elastic Search

SURI Link

IOT Orion Broker

Real Time

IOT Orion Broker

registration
triples

subscription note

IOT Apps

Snap4City Tools

IOT Device/Gateway

IOT Brokers

IOT Devices

IOT App

Adapter

push

Automatic

IOT App

Adapter

push

IOT App

push

IOT App

push

IOT App

push

IOT App

pull

sporadic

IOT App

pull

periodic

IOT App

push

Information, File

IOT Apps

IOT Device/Gateway

periodic

IOT Device

pull

push

IOT Device

pull

push

IOT Device

pull

push

IOT Device

pull

push

Snap4City (C), October 2020
Activities for IOT Device Registration

• Registration of
  – an IOT Device **Manually** from Zero or from an IOT Device Model
  – a set of IOT Devices with the same IOT Device Model from IOT App
  – a set of IOT Devices in **BULK** loading a CSV (with or without a reference IOT Model)

• The IOT Device registration implies the **automated production of the Digital Twin Device into the Knowledge Base**, which implies:
  – Activation of the Storage “DataShadow” for historical data access
  – Activation of all the relationships
  – Activation of Discovery mechanisms via IOT Directory, KB
  – Activation of Dashboard Wizard (after a few minutes), and Data Inspector
  – Etc.
IOT Devices/Sensors
IOT Discovery on IOT Application Development
IOT Device Registration from IOT Model
Snap4City Data Ingestion Flow Diagram

Snap4City (C), October 2020
Many IoT Devices?  

IOT Device Model!!!  

Example: ChargingStationModel

- **Prerequisites**: only for AreaManager users  
- If you have a set of sensors with the same features,  
  - you can create a model and then a set of instances (IoT Devices) in compliance with the model (not time consuming and avoiding errors)  
- IoT Directory and Devices > IoT Device Models > ‘New Model’ button
Add IOT/IOE Devices, exploiting an IOT Device Model
Add IOT/IOE Devices, exploiting an IOT Device Model

Just Buy an IOT Device and register: SigFOX, MQTT, NGSI (FiWare), ...

- Attach them by
  - Models
- A range of protocols, formats, approaches

Create your own devices:
- Arduino,
- Raspberry,
- Android,
- LoraWAN + Arduino,
- etc.

Secure Communication: HTTPS, TLS (K1, K2), Certificates

Snap4City (C), October 2020
Add IOT/IOE Devices, exploiting an IOT Device Model

Addition of Static Attributes of the IOT Device
Only if you enabled from model
IOT Device Registration in Bulk from CSV File
IOT Orion Broker

Knowledge Base
Semantic Reasoners

Indexing and Aggregating
NIFI, Elastic Search

Snap4City Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server

periodic

IOT Device

push

IOT App

pull

sporadic

(1c) static

IOT Directory

registration triples

SURI Link

subscription note

Real Time

Information, File
IOT Apps
Snap4City Tools
IOT Device/Gateway
From CSV → register IoT Devices in BULK

- Create a CSV from the CSV Model provided
- The columns must respect the CSV Model (every field present in the Model)
- One row of the CSV is one new IoT Device
- You have to create to create two keys (called k1, k2) that are necessary to read and write access to the device. They must be different each other.
- Each group of devices, that has the same IoT Model (data set), could/should have the same K1, K2. In this way, it is easier to read or write all the IOT Devices of the same set at the same time.
- These keys are in the UUID v4 format and can be generated online on this website: https://www.uuidgenerator.net

Available example: https://www.snap4city.org/592

File model in https://www.snap4city.org/289
Get a CSV template of your device for filling it

- download a csv template conformant to the IOT Device Model,
  - Go on the IOT Device Model, click on (+) and press the EXPORT button
Register IoT Devices in BULK

- IoT Directory and Devices > IoT Devices Bulk Registration
- Select: Model, Broker
- Upload the CSV file
- Wait
- Verify the presence of your Devices in:
  - IoT Directory and Devices > IoT Devices

https://www.snap4city.org/289
IOT Device Registration from IOT App (automation)
Snap4City Data Ingestion Flow Diagram

Static or quasi Static Data: POI, etc.

IOT Broker Registration

IOT Device Model Reg.

IOT Device Registration

Real Time data in Pull
Any gateway, server
periodic

IOT Orion Broker

IOT Orion Broker

Knowledge Base
Semantic Reasoners

Registration

Triples

Indexing and Aggregating
NIFI, Elastic Search

SURI Link

subscription note

Information, File
IOT Apps
Snap4City Tools
IOT Device/Gateway

IOT Device

push

IOT App

Adapter

IOT App

Adapter

Automatic

periodic

sporadic

(1c) static

pull

pull

push

push

push

IOT App

IOT Directory

Dynamic

Snap4City (C), October 2020
1) Model creation

**Model name:** Florence wifi average person
2) IoT Devices Creation from IOT APP

**BLOCK:** ‘IoTDirectory-new-device-from-model’

**Model name:** Florence wifi average person

Snap4City (C), October 2020
3) Group Creation (more than 200 devices) -> put all the devices in the group and put them as ‘public’ (or they remain private)
4) Send RT data to the IoTDevices
5) Verify RT Data via Snap4City API or via ServiceMap
1) IoTModel

2) Static Flow to create IoTDevices

3) Add the license and Make Public the IoTDevices (according to the license)

3) Search for the Cameras on Map

5) Working on Dynamic Flow to save Average #people every 15 minutes for each IoTDevice
Real Time Data Ingestion via IOT Applications
Snap4City Architecture vs Data Ingestion

**Data Sources, External Services**
- **PULL Data**

**Data Ingestion, aggregation, regularization, reconcile:**
- NIFI, IOT App

**Knowledge base**
- Semantic reasoners

**Indexing and aggregating**
- Elastic search

**Search and Query, Smart City API**
- Facet, semantic search

**Data Analytics, Simulations**
- R, Tensor Flow, Python, ...

**IOT Applications, Business Logic**
- Node-RED + Snap4City MicroServices

**Front-End**
- Rendering
- Acting
- Acting
- User interface, Interactive
- Dashboard, Drill down, maps, heatmaps

**Inform, announce, Act!, warning, alarms, What-IF, ..**

Snap4City (C), August 2020
A range of Possibilities

- Any Data can be ingested from any kind of service via an IOT App by using any kind of protocol
- IOT Devices NGSI, MQTT, … can push data into Internal Brokers Snap4City
- IOT Devices with other protocols can connect with IOT App for adapting their data to the data of the Internal Brokers Snap4City
- IOT Devices can be connected to any IOT Broker of any protocol and the IOT Broker can talk with Internal Brokers Snap4City, with or without adapter

**THUS any data passing on our Internal IOT Brokers is automatically indexed into Elastic Search and Knowledge Base creating the full support for query and data shadow, usable immediately without any config for dashboards, synoptics, data inspection, etc.**
**Dynamic Data**

- **Hyp**: IOT Device created

- Create an IOT App to send the dynamic data to your Device/s (in push or Pull)

- The IOT App is based on the data you have:
  - Data ingestion
  - Data transform to a json referring to the IOT Device/s
  - Choose the correct broker

---

Snap4City (C), October 2020
Dynamic Data

- Charging Station in Florence (Open Data)
  1. Use an inject block to choose the frequency of update
  2. Use http block to download data
  3. Use a set of blocks to convert data from xml to json
  4. Use a function block to create a json referring to your Model and Devices (ChargingStationModel)
  5. Use a fiware Orion block to send the dynamic data:
     - selecting the broker
     - Put your K1, K2

ChargingStationModel: values
Further readings

• HOW TO: create a Dashboard in Snap4City
• HOW TO: add a device to the Snap4City Platform
• HOW TO: add data sources to the Snap4City Platform
• HOW TO: define privacy rules for personal data, produced by the end-users own device
• HOW TO: Develop Smart Applications, Snap4City development Life Cycle
• HOW TO: HLT vs Ingestion, and HLT vs Widgets
• HOW TO: Develop an IOT Application for Data Ingestion
• HOW TO: Upload data into Knowledge Base, ServiceMap (triple upload)
• HOW TO: Create as set of Devices with BulkProcessing
• HOW TO: Create an IOT Device Model
• HOW TO: Create an IOT Device Instance from IOT Directory tool
Further readings

- HOW TO: add a device to the Snap4City Platform
- HOW TO: add data sources to the Snap4City Platform
- HOW TO: add IOT Device data source from external broker to the platform.
- TC9.13: How to upload a local file into your IOT Application
- TC9.2. Managing heterogeneous File Ingestion, protocols, formats via IOT applications, and open standards
- TC2.25. Registering external MicroService calling RestCall services, using it on IOT applications
A Complete Example for Time Series: IOT Device Model + IOT Data Ingestion
I have created an IOT Device Model as:

<table>
<thead>
<tr>
<th>General Info</th>
<th>IoT Broker</th>
<th>Static Attributes</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>statuscorregione</td>
<td>statuscorregione</td>
<td>Description</td>
<td>Ok</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
<td>Description</td>
<td>Ok</td>
</tr>
<tr>
<td>misura</td>
<td>misura</td>
<td>Sensor</td>
<td></td>
</tr>
<tr>
<td>Device Type</td>
<td>Device Type</td>
<td>Kind</td>
<td></td>
</tr>
<tr>
<td>protezione civile</td>
<td>protezione civile</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>Producer</td>
<td>Producer</td>
<td>Frequency</td>
<td></td>
</tr>
<tr>
<td>Healthiness Criteria</td>
<td>Healthiness Criteria</td>
<td>Healthiness Value</td>
<td></td>
</tr>
<tr>
<td>Automatically generated</td>
<td>Automatically generated</td>
<td>Edge-Gateway Type</td>
<td></td>
</tr>
</tbody>
</table>

Edit Model - statuscorregione

Edit Model - statuscorregione

Edit Model - statuscorregione

Snap4City (C), October 2020
For Time Series
- **ValueName:** dateObserved
- **DataType:** time
- **ValueType:** timestamp
- **ValueUnit:** timestamp in millisecond
- **E.g.:** ISO string of the date-time

<table>
<thead>
<tr>
<th>General Info</th>
<th>IoT Broker</th>
<th>Static Attributes</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateObserved</td>
<td>time</td>
<td>Timestamp</td>
<td>timestamp in millisecond</td>
</tr>
<tr>
<td>ValueName</td>
<td>Date Type</td>
<td>Value Type</td>
<td>Value Unit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time</td>
<td>timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number</td>
<td>Number in</td>
</tr>
</tbody>
</table>
From IOT Model I have created some instances: the IOT Devices
They have been created by «Add new Device»
IOT Device from IOT Model by Providing:

- **NAME** (it has to be unique)
- **Select the IOT Model:** «statuscorregione»
  - Thus the K1, K2 appears since the model is associated to an Orion Broker that needs to have them, the tool generate them for you but you can impose if you like
  - See in previous slide the ID name of the IOT Broker used
- **Lat** and **Lon**, GPS coordinates you can:
  - pick on the map
  - Write the coordinates manually and see the pin on map
Once created the IOT Device you may send data on it

- You may create an IOT App, where:
  - Function: is preparing the JSON package
  - Block «Fi-Ware Orion OUT V1» is sending the data to the Orion Broker. Namely: «OrionUNIFI»
  - Please note that several version of IOT ORION Brokers and protocols exists:
    - So that you have to know which protocols you need to use for your broker
• Certificates are automatically loaded at the first authentication
• Done!!

• IP if the Broker is in cloud
• Symbolic address of IOT Broker can be taken from IOT Directory
NGSI versions

- Orion Broker of V2 with NGSI syntax of V1
- Orion Broker of V2 with NGSI syntax of V2
- Orion Broker of V1 with NGSI syntax of V1 + Secure Filter of Snap4city
A Json from the IOT App

**ID**: The Name of the IOT Device: «corveneto»

**Type** as that define in the IOT Device when you created

The **Time stamp**: “dateObserved” to have a time series data

- “str” is a string with the date and time in standard ISO, such as
  - “2020-08-04T04:00:00+02:00”,
  - “2020-08-03T00:00:00.000Z”

And the **vector of “attributes”**

```python
msg = { payload: {
    "id": "corveneto",
    "type": "misura",
    "attributes": [
        { "name": "dateObserved", "value": str, "type": "time" },
        { "name": "stato", "value": "active", "type": "string" },
        { "name": "ricoverati_con_sintomi", "value": 12, "type": "integer" },
        { "name": "terapia_intensiva", "value": 34, "type": "integer" },
        { "name": "totale_ospedalizzati", "value": 34, "type": "integer" },
        { "name": "isolamento_domiciliare", "value": 334, "type": "integer" },
        { "name": "totale_attualmente_positivi", "value": 12, "type": "integer" },
        { "name": "nuovi_attualmente_positivi", "value": 33, "type": "integer" },
        { "name": "dimessi_guariti", "value": 22222, "type": "integer" },
        { "name": "deceduti", "value": 2, "type": "integer" },
        { "name": "totale_casi", "value": 2222, "type": "integer" },
        { "name": "tamponi", "value": 222222344, "type": "integer" }
    ]
}

return msg;
```
Multi Series Widget coming from the same IOT Device

- Over on the serie label to highlight
- Click on the serie label to on/ok
- Over on the graph to see the values

Andamenti Nazionali e Regionali infezione COVID-19
Sulla base dei dati della protezione civile, elaborazioni DISITLab

per evidenziare gli andamenti di vostro interesse: eliminare le curve che non interessano selezionandole in legenda.
Alcuni dati in passato non sono pervenuti alla protezione civile

JSON for Authentication as well

- `msg.auth = {
  "k1": "1ef0e5e8-yyyy-xxxx-9462-0aa4cfcf5e19",
  "k2": "b2b34425-yyyy-xxxx-818d-2d6cac2314a6",
  "apikey": "apikey",
  "basicAuth": "basicAuthKey"
} ;`
You may use other functions from IOT Directory

- IOT Discovery in an area
- Query on IOT Directory
- Get IOT Device Info
- Registering an IOT Device from model
- Delegate an IOT Device
- Change Ownership of an IOT Device
Get IOT Device Info

- You can create smart IOT Applications that on the basis of the list of IOT Devices would request all what you need to load data into **YOUR OWN IOT Devices** including:
  - Service URI
  - K1, K2
  - Authentication
Data Ingestion via API: External services, using HTTP MicroService on IOT App
General solution, bring data from API to Dashboards

- You can use the MicroServices HTTP in get and post to act on REST Call

- OR

- You can **save/consolidate your rest API** transforming it in a MicroService usable for many colleagues into IOT Applications:
  - TC2.25- Registering external MicroService calling RestCall services, using it on IOT applications
    - https://www.snap4city.org/129

IF your REST API is going to use credentials as username and password, we suggest you to save them into MyPersonalData of Snap4City

- so that the code will not provide clear credentials and you can update from user interface on your personal data profile.
- The IOT App can retrieve the Username and Password at the moment in which they are used with the security shield of Snap4City
External REST Call API vs MicroServices

- Each Rest Call API can be automatically transformed into a MicroService for the IOT Applications
Edit MicroService Call: call and help editing

Edit MicroService: Antwerp cameras location.zip

- Nature: Transfer service and renting
- Sub Nature: Monitoring camera
- Licence: Public
- Description: Antwerp cameras location from Open Data

Select Image: Nessun file selezionato

Method: GET

Do you want create a Microservice with Authentication?


Remove Parameter  Add Parameter

Help:

Description of microservice
The service gives the camera location (lat, lon)

Inputs
Microservice input description:
No Parameter

Outputs
json

Details
More details here: https://opendata.antwerpen.be/datasets/kaart

Cancel  Confirm
Usage of the MicroService from IOT App
Data Ingestion via IOT App towards MyKPI
Snap4City Architecture vs Data Ingestion

Data Sources, External Services
PULL Data

Data Ingestion, aggregation, regularization, reconcile:
NIFI, IOT App

Front-End
Rendering
Acting,
Widgets,
MicroApps
User
interface,
Interactive
Dashboard,
Drill down,
maps,
heatmaps

Back-End
Knowledge base
Semantic reasoners

Search and Query,
Smart City API
Facet, semantic
search

Data Analytics, Simulations
R, Tensor Flow, Python, ….

IOT Applications, Business Logic
Node-RED + Snap4City MicroServices

Inform, announce, Act!, warning, alarms, What-IF, ..
• **MyKPI can**
  – be time series + metadata
  – be POI with full metadata decryption, MyPOI
  – be passed in ownership to other users,
  – be delegated in access to other users
  – model daily trajectories from: Mobile Phone Apps, CANBUS data and GPS location from mobiles, PAX Counter Mobile, mobile IOT Devices, etc.
  – be saved and retrieved from IOT Apps
  – create events at their changes towards IOT Apps
  – be saved into: MySQL and/or Elastic Search

Snap4City (C), October 2020
Create your MyKPIs

- My Data, KPI, POI > ‘Add My KPI’ button
- Verify the KPI existence in My Data, KPI, POI
- Create your IoT App sending data to your KPIs
- Example: Lonato Car Park:
  - NumFreeSlots
  - MaxDuration
  - MaxDurationSlotId

Snap4City (C), October 2020
IoT App sending data to your KPIs

- Create your IoT App (ex: ‘SmartParking LonatoDelGarda’)
  1. Use an inject block to choose the frequency of update
  2. Use http block to download data (e.g.)
  3. Use a function of blocks to convert data in a specific json sending data to each KPI
  4. Use a save-my-kpidata-values block
Data Ingestion via Web Scraping
Snap4City Architecture vs Data Ingestion

Data Sources, External Services

Data Ingestion, aggregation, regularization, reconcile:
NIFI, IOT App

Front-End
Rendering Acting, Widgets, MicroApps
User interface, Interactive Dashboard, Drill down, maps, heatmaps

Back-End
Knowledge base
Semantic reasoners

Search and Query, Smart City API
Facet, semantic search

Indexing and aggregating
Elastic search

Data Analytics, Simulations
R, Tensor Flow, Python, ...

IOT Applications, Business Logic
Node-RED + Snap4City MicroServices

Inform, announce, Act!, warning, alarms, What-IF, ..
Web Scraping from Portia

My Scraping process

Knowledge Base, Km4City

IOT App. Editor

Web Scraper PORTIA

Generating WEB Scraping

Sharing/saving reusing Scraping

Resource Manager
See them in the list of IOT Apps
Web scraping

- TC9.16 – Web Scraping to get data from web pages
Data Streams from Smart City API, participatory
• We intend in this cases the data that can be posted on the infrastructure **by using the Smart City API**, such as:
  – MyPersonalData, MyKPI, data from IOT App on mobile, etc.
  – please see slides of Day 7
• Next section discusses those that are **automatically collected from Mobile Phone** and sent to the infrastructure on cloud via the Smart City API. For example:
  – Clicks on App
  – Post of images on POI
  – Post of comments on POI
  – Post of raking on POI
  – Questionnaires, and reactions
  – Eventual scores collected on the basis of the actions performed by the users.
  – Trajectories taken from the mobile phone positions and clicks
  – OBD2 data and positions
  – Etc.
Data Streams from Mobile Devices
Snap4City Architecture vs Data Ingestion

Data Sources, External Services
- PULL Data

Data Sources, Brokers, External Services
- Data Driven, Real Time

Data Ingestion, aggregation, regularization, reconcile:
- NIFI, IOT App

Knowledge base
- Semantic reasoners

Indexing and aggregating
- Elastic search

Search and Query, Smart City API

Data Analytics, Simulations
- R, Tensor Flow, Python, ..

IOT Applications, Business Logic
- Node-RED + Snap4City MicroServices

Inform, announce, Act!, warning, alarms, What-IF, ..
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 ?
- ...

**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
- .....
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, ....
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 buy a bus ticket.

Rules

User context

City context
MyKPI: Tracking of Devices and Mobiles

- Real Time Trajectories for
  - Mobile Phone
  - Moving IOT Devices
  - OBU, Vehicular Kits
  - Multiple tracks
  - Day by day

- Micro Application

![Diagram of tracking system](image)
Load and change data via Smart City API

• Data regarding Mobile Phones:
  – Clicks on App
  – Post of images on POI
  – Post of comments on POI
  – Post of raking on POI
  – Questionnaires, and reactions
  – Trajectories taken from the mobile phone positions and clicks
  – OBD2 data and positions
  – etc.

• They are automatically collected and can be inspected by the user via special tools as presented in the following.
My KPI data view and manipulation

https://www.snap4city.org/mypersonaldata/
Further reading on MyKPI

- **TC1.17.** Object tracking, widget tracker, personal tracking/trajectories, moving sensors
- **TC 2.35** - How manage My KPI with Dashboard
- **Solution:** using PAX Counters, monitoring museum and events
- **HOW TO:** define privacy rules for personal data, produced by the end-users own device
- **TC9.13:** How to upload a local file into the platform (IOT Application and MyKPI)
Data Streams from Dashboards
Snap4City Architecture vs Data Ingestion

Data Ingestion in Push

Data Sources, External Services
PULL Data

Data Sources, Brokers, External Services
Data Driven, Real Time

Data Ingestion, aggregation, regularization, reconcile:
NIFI, IOT App

Back-End
Knowledge base
Semantic reasoners

Indexing and aggregating
Elastic search

Search and Query, Smart City API
Facet, semantic search

Data Analytics, Simulations
R, Tensor Flow, Python, ….

IOT Applications, Business Logic
Node-RED + Snap4City MicroServices

Front-End
Rendering Acting, Widgets, MicroApps
User interface, Interactive Dashboard, Drill down, maps

Inform, announce, Act!, warning, alarms, What-IF, ...

Snap4City (C), October 2020
Dashboards with data driven IOT Applications enforcing intelligence

Snap4City: Builder of Sentient Cities Solutions

IOT and data World

IOT Applications

Dashboards and Apps

My IOT Devices

Big Data Analytics, Artificial Intelligence
IOT Data Driven

Snap4City Platform storage for «Data Shadow» and much more

Towards any IOT Device and/or Dashboard

Managing Public and Private IOT/IOE Devices

IOT Applications

Real Time

Real Time + Historical

Dashboards also provide rendering for sensor values

Sensor Actuator

From Dashboard to IOT Device

From IOT Device to KB

Actuator

Sensors

Snap4City IOT Brokers

Sensors

Lavagnini

Statuto

Green

OFF

45

45
What the Dashboards can directly exploit

From Dashboard to IOT App

From IOT App to Dashboard

Big Data Analytics, Artificial Intelligence Access to all data

Snap4City (C), October 2020
HLT: Sensors-Actuators

- Complex Event
- Dashboard-IOT App
- External Service
- Heatmap
- KPI (Key Performance Indicator)
- MicroApplication
- My Personal Data
- MyKPI
- MyPOI
- POI (Point of Interest)
- Sensor
- Sensor Actuator
- Special Widget
- Wfs (GIS)
From Dashboard to IOT App

MyKPI variable

Synoptics

Impulse button

Numeric keyboard

Switch button

Dimmer

Geolocator

Event driven my kpi

Synoptic read

Synoptic subscribe

Nature
Dashboard - IOT App

From IOT App to Dashboard

IOT Application
<table>
<thead>
<tr>
<th>Widgets ICONS</th>
<th>Widget Name, Description</th>
<th>IOT App</th>
<th>Dashboard-IOT App</th>
<th>KPI (metric)</th>
<th>MyPersonal Data</th>
<th>MyData</th>
<th>MyKPI</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX</td>
<td>Single Content</td>
<td>X (cs)</td>
<td>X (DD)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>50</td>
<td>Speed Limit</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Speedometer</td>
<td>X (cs)</td>
<td>X (DD)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Gauge</td>
<td>X (cs)</td>
<td>X (DD)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Single Bar, V/H</td>
<td>X</td>
<td>X (DD)</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Single and Multiple Bars, stacked or not</td>
<td>X (cs)</td>
<td>X (DD)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>MultiSeries, shaded, staked and non staked</td>
<td>X (cs)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Time Trend (single)</td>
<td>X</td>
<td>X (DD)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Time trend Compare</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SpiderNet, radar, Kiviat</td>
<td>X (cs)</td>
<td>X (DD)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Pie, Donut, 2 layers Donut</td>
<td>X (cs)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Table</td>
<td>X (cs)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
• **IOT APP column in previous table:**
  - **X:** means that from the IOT App you can send a new value or array to the widget directly, without the need to have it stored into Sensor or MYKPI variable, etc.
  - **CS, widget supports Change Source,** in the sense that: from the IOT App it is possible to send a command to the Widget to change the data source. E.g., selecting sources among: Sensors (service URI), MyKPI (ID), any value produced on the IOT App directly. *(cs) recent additions*

• **Dashboard IOT App column in previous table:**
  - **X:** there is a MicroService / node on IOT App to act on those widgets on dashboard. The data are visualized.
  - **DD, widget is Data Driven,** in the sense that new data in push can be sent and the widget is updated in real time on web page without web page reloading.
From-To Custom Widgets / Synoptics to Storage in WS

MyKPI

Sensors

MyKPI

Sensor

[ ]

New Shared Variables

Constant Values
GIS Data Import and Export: WFS and WMS
GIS vs Sna4City

GIS Server can be: ESRI ArcGIS Enterprise, QGIS, GeoServer, ..
GIS Player can be: ESRI ArcGIS Pro, ArcGIS Portal, Snap4City WFS player, ...

- GIS:
  - Geographic Information System
- WMS:
  - Web Map Service
- WFS:
  - Web Feature Services

Any External Service

Data Ingestion
WMS/WFS APIs
Smart City APIs
Big Data and Semantic Storage

IOT & Real Time Streams
Big Data Analytics, Artificial Intelligence

Dashboards and Apps

Dashboard System

GIS Player

GIS vs Sna4City (C), October 2020
Snap4City is interoperable with
- ESRI ArcGIS Enterprise, Portal, Pro/MAP, ...
- other GIS tools supporting WFS, WMS, GeoJSON, GML

Snap4City is interoperable since:
- Provides info/data in WFS, WMS
- Exploits data/info from WFS, WMS
- Imports data/info from WFS/WMS

The Snap4City platform can be installed on premise using Snap4City Appliance
https://www.snap4city.org/drupal/node/471
- StartSNAP4CITYVM includes the Dashboard Builder that is capable to work with WFS WMS protocols for the integration with GIS platforms as ESRI ArcGIS, QGIS, directly or using Snap4City GIS player.
- KBSSMVM includes the Smart City API and WFS API which can be used to data harvest from any GIS servers and GIS desktop tool.
• DISIT Lab has ESRI ArcGIS Enterprise 10.6 installed
• ArcGIS Portal accesses to ArcGIS Enterprise server
• Snap4City Dashboard uses as External Service: ArcGIS ESRI Portal

Snap4City (C), October 2020
GIS data on Dashboard via Snap4City GIS Player

- DISIT Lab has ESRI ArcGIS Enterprise 10.6 installed
- Snap4City has its WFS Player [https://main.snap4city.org/widgets/venezia/index.php](https://main.snap4city.org/widgets/venezia/index.php)
- Snap4City Dashboard uses as External Service: Snap4City GIS viewer via WFS/WMS: [https://main.snap4city.org/view/index.php?iddasboard=MTIxNg==](https://main.snap4city.org/view/index.php?iddasboard=MTIxNg==)
Dash with Snap4City GIS widget and Selector

- DISIT Lab has ESRI ArcGIS Enterprise 10.6 installed
- Snap4City has its WFS / WMS widget / Player
- Snap4City Dashboard shows WFS/WMS data via Special GIS Widget Map:
  - https://www.snap4city.org/dashboardSmartCity/view/index.php?iddashboard=MTQwMw==
  - Snap4City can use Selector to select WFS / WMS sources to be shown from ESRI ArcGIS (as well as from any other WFS service) on Widget map

The Snap4City Widget Map allows to mix WFS GIS sources with Smart City API
https://www.snap4city.org/dashboardSmartCity/view/index.php?iddashboard=MTM5NA==
DISIT Lab, Distributed Data Intelligence and Technologies
Distributed Systems and Internet Technologies
Department of Information Engineering (DINFO)
http://www.disit.dinfo.unifi.it
http://www.disit.org

(D) Dashboard with Orthomaps and shapes layers based on WMS, GeoJSON

- DISIT Lab has ESRI ArcGIS Enterprise 10.6 installed, and GeoServer
- Snap4City main MultidataMap Widget can load WMS background images and shapes, the Orthomaps
- Also Heatmaps calibrated are provided from GeoServer using WMS protocol
- Maps in this case are directly taken from ESRI Server, free of charge
- https://www.snap4city.org/dashboardSmartCity/view/index.php?iddashboard=MTQwNg==#
Snap4City via WFS on top of Smart City API provide data to ESRI ArcGIS Enterprise or ArcGIS Pro, and thus the data become accessible on ArcGIS ESRI Portal

- Snap4City provides a WFS service on top of SmartCity API. Thus providing data to any GIS/WFS client, including ESRI ArcGIS Enterprise, ArcGIS Pro, QGIS, etc.
- In the example, our ingested Helsinki Data have been harvested from ESRI ArcGIS via WFS.
- Once ingested on ESRI ArcGIS can be visualized, by using ArcGIS Portal
- Snap4City Dashboard can show ESRI ArcGIS Portal (A) as External Services in a dashboard.
  - [https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MjIwNg==](https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MjIwNg==)
To test you need to have installed ArcGIS pro on your pc and connect with Snap4City WFS/WMF server or with ArcGIS server which has done the same connection with our server WFS/WMS

On PC:

(i) Get data via WFS connected to ESRI ArcGIS Enterprise or other sources
(ii) Download data via WFS ... from...
(EA) ArcGIS ESRI Pro as GIS / WMS play

To test it you need to have installed ArcGIS pro on your pc and connect with our WFS/WMF server or with ArcGIS server which has done the same connection with our server WFS/WMS

Heatmap taken from Snap4City GeoServer via WMS protocol
Snap4City vs GIS, WFS/WMS

• GIS data:
  – Ingested via WFS/WMS protocols, and then managed as the other data. Data ingestion from GIS server can be performed via ETL processes, or directly from Dashboards
  – Shown on Dashboards via third party GIS tools as External Services
  – Shown on Dashboards using Special GIS Widget Map which directly access to GIS data via WFS/WMS
  – Heatmaps and Maps are distributed via a GeoServer

• Snap4City can interact with ArcGIS Real Time Events via MQTT protocol as well

• Snap4City vs GIS solutions and connections
Integration with CKAN
Open Data Manager and Portal
Data Ingestion via Data Gate

Knowledge Base, Km4City

Knowledge and Storage Data from the Field and City

Federated Network

Saving / Sharing reusing

Data Gate

Data Set: sharing, Harvesting, Loading/Downloading

Data Set:
- Search
- Loading
- Download
- Share
- Publish
- Also automated

https://datagate.snap4city.org/ssologin_handler

Snap4City (C), October 2020
Integrated DataGate/CKAN
Static open data ingestion

Federated Crawling
Federated Distribution

Data Set:
- Search
- Loading
- Download
- Share
- Publish
- Also automated

Automated data regularization
Once the file is create can be ingested into the IOT App above presented
Further readings on Datagate

- Data Set Manager: Data Gate / CKAN federated
- HOW TO: add a device to the Snap4City Platform
- HOW TO: add data sources to the Snap4City Platform
- TC6.1- Managing DataSets via DataGate: ingest, search, download, upload, annotate, share
- TC9.3- Managing data sets with IOT Applications, and exploiting DataGate
- TC6.2- Search on DataGate for Data Sets
- US6. Developing and using processes for data transformation

https://datagate.snap4city.org/ssologin_handler
Integration with CKAN: automated data set (i) ingestion and (ii) production via IOT App
Advanced Snap4City APIs and MicroServices

IoT App – Automatize:
- Import data from CKAN to Snap4City
- Upload Public Data from Snap4City to CKAN
- Data Harvesting
- Dashboards and Mobile/Web Apps creation

Datagate

Open or Private External CKAN Data Portals

Knowledge Base, Km4City

Snap4City Portal and Integrated tools

Snap4City Portal

Knowledge and Storage Data from the Field and City

Real Time Data

Heatmaps
What-IF Analysis
Remote Control
Historical Data

Mob & Web Apps
Snap4City Dashboards
Almost all the calls to CKAN are quite similar
Read more on

- TC9.17 – CKAN vs Snap4City Integration and Interaction
  - automating the Read of a Dataset Info from CKAN
  - automating the Read of a Resource info from CKAN
  - automating the Creation of a Dataset on CKAN
  - automating the Creation of a static Resource in CKAN
  - automating the Creation of a dynamic Resource in CKAN
  - automating the Sending of a json to CKAN from a query to Snap4City to perform any other action on the Smart City

- Data Set Manager: Data Gate / CKAN federated
Social Media data collection and exploitation
Social Media Data

• Data from Social Media can be ingested in different manners:
  • Twitter Data can be:
    – Collected via Twitter Vigilance programming keywords.
    – Collected in real time via IOT Applications
    – Once collected the Metrics (number of Tweets, reTweets, NLP, Sentiment Analysis, etc. can be:
      • Shown, exploited in IOT Applications
      • Exploited for early warning
  • FaceBook data:
    – Can be generated by using IOT App
  • Telegram data:
    – Can be generated by using IOT App
Prediction/Assessment

- Football game results as related to the volume of Tweets
- Number of votes on political elections, via sentiment analysis, SA
- Size and inception of contagious diseases
- Marketability of consumer goods
- Public health seasonal flu
- Box-office revenues for movies
- Places to be visited, most visited
- Number of people in locations like airports
- Audience of TV programmes, political TV shows
- Weather forecast information
- Appreciation of services
• http://www.disit.org/tv
• http://www.disit.org/rttv
• Citizens as sensors to
  – Assess sentiment on services, events, ...
  – Response of consumers wrt, ...
  – Early detection of critical conditions
  – Information channel
  – Opinion leaders
  – Communities
  – Formation
  – Predicting volume of visitors for tuning the services
Twitter Vigilance RT: sentiment analysis

Real time
Early Warning
Twitter Vigilance

Early Warning

Florene downburst

Predictive models

Hot flows

Attendance at long lasting events: EXPO2015

Attendance at recurrent events: TV, football

Snap4City (C), October 2020
IOT Applications Development

- MicroServices collections
- My IOT Applications
- IOT App. Editor
- Generating IOT App With Dashboard
- Sharing/saving reusing IOT App
- Resource Manager
- Knowledge Base, Km4City
- Dashboard Collection, Editor and Wizard
- ServiceMap Discovery
- IOT Discovering

Snap4City (C), October 2020
Overview (paolo)

https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=Mjc3NQ==

Paolo
HeritData TW Demo

Volume Trends

SA trend

Volume Trends hourly

SA trend hourly

RTW TW Volume Trends hourly

SA trend for key

Volume Trends daily
Further reading on Social Media

- TC2.22- Exploiting Twitter Vigilance as External Service, in Dashboard, and as RestCall as MicroService in IOT applications
- TC2.21- IOT Applications with Social Media Actions, and cultural scenarious
- External Services
- TC2.21- IOT Applications with Social Media Actions, and cultural scenarious
Data Ingestion and Transformation via ETL Processes

(only for former versions of Snap4City)
Snap4City Architecture vs Data Ingestion

Data Ingestion in PULL

Data Sources, External Services
PULL Data

Data Sources, Brokers, External Services
Data Driven, Real Time

Big Data Cluster
HDFS, Hbase, Phoenix

Knowledge base
Semantic reasoners

Search and Query,
Smart City API
Facet, semantic search

Indexing and aggregating
Elastic search

Data Analytics
R, Tensor Flow, Python, MapReduce, ...

Visual Analytics

IOT Applications
Node-RED + Snap4City MicroServices

Inform, announce, Act!, warning, alarms, What-IF, ...

Snap4City (C), July 2020
Data Ingestion Flow Guideline, thumb rules

Road Graph Setup

X Data

Static file

yes

no

Regular file

yes

no

Regularize

ETL: Only PULL

Development on Penthao/Spoon

Automated Execution ETL

registration

Knowledge base

Semantic reasoners

- (1) static
- (2) Real time

ETL Execution on DISCES

Big Data Cluster

HDFS, Hbase, Phoenix

Indexing and aggregating

Elastic search
Developers of ETL, Data Manager

Distributed Back Office

DISCES scheduler production

ETL Development Environment

Data sources

Knowledge Base, Km4City

Knowledge and Storage Data from the Field and City

Data Gate

Data Set Saving / Sharing

Resource Manager

Saving / Sharing reusing

Data Set Loading/Downloading

Schedule Execute

Load data or prepare for data ingestion

Snap4City (C), October 2020
Batch Processing for dynamic data ingestion

Integrated ETL development
Classic Data warehouse Architecture
The traditional Km4City Approach

Open Data/other sources

- OD1
- OD2
- OD...N
- RT data...

Phase I
- BigData Store Dati Statici

Phase II
- ETL proc.
- Quality Improvement

Phase III
- km4City Ontology
- R2RML model
- Mapping Process

Knowledge and Storage Data from the Field and City

Knowledge Base, Km4City
ETL Processes

The three phases are:

• **Extracting** data from outside sources (**Ingestion** phase).

• **Transforming** data to fit operational needs which may include improvements of quality levels (**Data Quality Improvement** phase).

• **Loading** data into the end target (database, operational data store, data warehouse, data mart, etc.). So the data can be translated in **RDF triples using a specific ontology** (Static/periodic datasets) or on **NoSQL Databases** (Dynamic datasets)
Phase I: Data Ingestion

• Purpose is to store data in HBase (Big Data NoSQL database).

• Acquisition of wide range of OD/PD: open and private data, static, quasi static and/or dynamic real time data.

• Static and semi-static data include: points of interests, geo-referenced services, maps, accidents statistics, etc.
  – files in several formats (SHP, KML, CVS, ZIP, XML, JSON, etc.)

• Dynamic data mainly data coming from sensors
  – parking, weather conditions, pollution measures, bus position, ...
  – using Web Services

• Using Pentaho - Kettle for data integration (Open Source tool)
  – using specific ETL Kettle transformation processes (one or more for each data source)
Phase II: Data Quality Improvement

• Purpose: add more information as possible and normalize data from ingestion

• Problems kinds:
  – Inconsistencies, incompleteness, typos, lack of standards, multiple standards, ...

• Problems on:
  – Place-name code
  – Street names (e.g., dividing names from numbers, normalize when possible)
  – Dates and Time: normalizing
  – Telephone numbers: normalizing
  – Web links and emails: normalizing
Phase III: Data mapping

• Purpose is to translate data from QI in RDF triples
• We use triples to do inference on data.

• Using **Karma Data Integration tool**, a mapping model from SQL to RDF on the basis of the ontology was created.
  – Data to be mapped first temporary passed from HBase to MySQL and then mapped using Karma (in batch mode)

• The mapped data in triples have to be uploaded (and indexed) to the **RDF Store** (Virtuoso).
• Triples are composed by a subject, a predicate and an object.
Main strengths:

- Collect data from a **variety of sources** (extraction);
- Move and modify data (transport and transform) while cleansing, denormalizing, aggregating and enriching it in the process;
- Frequently (daily) store data (loading) in the final target destination, usually a **large dimensionally modeled database** (or **data warehouse**).

- **Spoon**: graphically oriented end-user tool to model the **flow of data** from input through transformation to output (**transformation**)
- **Pan** is a **command line tool** that executes transformations modeled with Spoon
- **Chef**: a graphically oriented **end-user tool** used to model **jobs** (transformations, FTP downloads etc. placed in a flow of control)
- **Kitchen** is a **command line tool** to execute jobs created with Chef.
Type of Steps in Spoon

Three different kinds of steps:

• **Input**: process some kind of 'raw' resource (file, database query or system variables) and create an output stream of records from it.

• **Output**: (the reverse of input steps): accept records, and store them in some external resource (file, database table, etc.).

• **Transforming**: process input streams and perform particular actions on them (adding new fields/new records); these actions produce one or more output streams.
# Type of Transformations in Spoon

<table>
<thead>
<tr>
<th>Transform</th>
<th>Lookup</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Add a checksum</td>
<td>- Call DB Procedure</td>
<td>- Access Output</td>
</tr>
<tr>
<td>- Add constants</td>
<td>- Check if a column exists</td>
<td>- Delete</td>
</tr>
<tr>
<td>- Add sequence</td>
<td>- Check if file is locked</td>
<td>- Excel Output</td>
</tr>
<tr>
<td>- Add value fields changing</td>
<td>- Check if webservice is available</td>
<td>- Insert / Update</td>
</tr>
<tr>
<td>- Add XML</td>
<td>- Database join</td>
<td>- Json output</td>
</tr>
<tr>
<td>- Calculator</td>
<td>- Database lookup</td>
<td>- LDAP Output</td>
</tr>
<tr>
<td>- Closure Generator</td>
<td>- Dynamic SQL row</td>
<td>- Palo Cells Output</td>
</tr>
<tr>
<td>- Example plugin</td>
<td>- File exists</td>
<td>- Palo Dimension Output</td>
</tr>
<tr>
<td>- Number range</td>
<td>- Fuzzy match</td>
<td>- Properties Output</td>
</tr>
<tr>
<td>- Replace in string</td>
<td>- HTTP client</td>
<td>- RSS Output</td>
</tr>
<tr>
<td>- Row denormaliser</td>
<td>- HTTP Post</td>
<td>- Salesforce Delete</td>
</tr>
<tr>
<td>- Row flattener</td>
<td>- Stream lookup</td>
<td>- Salesforce Insert</td>
</tr>
<tr>
<td>- Row Normaliser</td>
<td>- Table exists</td>
<td>- Salesforce Update</td>
</tr>
<tr>
<td>- Select values</td>
<td>- Web services lookup</td>
<td>- Salesforce Upsert</td>
</tr>
<tr>
<td>- Sort rows</td>
<td>- Utility</td>
<td>- Serialize to file</td>
</tr>
<tr>
<td>- Split field to rows</td>
<td>- Change file encoding</td>
<td>- SQL File Output</td>
</tr>
<tr>
<td>- Split Fields</td>
<td>- Clone row</td>
<td>- Synchronize after merge</td>
</tr>
<tr>
<td>- String operations</td>
<td>- Delay row</td>
<td>- Table output</td>
</tr>
<tr>
<td>- Strings cut</td>
<td>- Execute a process</td>
<td>- Text file output</td>
</tr>
<tr>
<td>- Unique rows</td>
<td>- If field value is null</td>
<td>- Update</td>
</tr>
<tr>
<td>- Unique rows (HashSet)</td>
<td>- Mail</td>
<td>- XML Output</td>
</tr>
<tr>
<td>- Value Mapper</td>
<td>- Metadata structure of string</td>
<td>- Write to log</td>
</tr>
</tbody>
</table>

---

Snap4City (C), October 2020
Sequential vs parallel

SNAP4CITY

START  Get Data from CSV File  Phones  Addresses  Websites  Put Data to Database  Successo

HBase Input_strnum  Get Variables 5  Add constants 5  Table output_strnum
HBase Input_manov  Get Variables 6  Add constants 6  Table output_manov
HBase Input_giunzioni  Get Variables 9  Add constants 9  Table output_giunzioni

Snap4City (C), October 2020
• ‘HTTP’ step downloads the dataset defined in $PARAM$ (URL) field.
• If a file with the same name already exists, we compare them.
  - If they are the same, we delete the folders created before.
  - Otherwise we unzip the file and ‘HBase_firstInsert’ is called.
HBase_firstInsert

- ‘Get KML’ loads the file just downloaded.
- ‘Get data from XML’ and ‘Row denormaliser’ extract fields from source file.
- In the JS step, we create an identifier (it will be use as key in HBase).
- ‘HBase Output’ saves the information in a HBase table.
- ‘Insert/Update’ updates the last ingestion date in MySQL table process_manager2.
Quality Improvement

- ‘HBase Input’ gets back data saved at the ingestion end.
- ‘Fix address’ is used to correct typing error (e.g. Giovambattista instead of Giovanbattista) or to simplify search of right toponym code.
- ‘Modify_*’ transformations normalize address, CAP, website, e-mail, phone number, ...
- ‘Add constants’ adds two fields (address_syn, codice_toponimo) which we will use in the next job.
- ‘HBase Output’ saves in a new HBase table the quality improvement result.
- For the rows which have an un-empty streetAddress, the steps below extract a word from streetAddress which we will use to find the right toponym code.
To get toponym codes we use `tbl_toponimo_bis` MySQL table. It contains road names, toponym codes, town codes, .... Each road has his center coordinates (lat, long).

- **Table input** does the query on that table using the word created before.
- **Fuzzy match** calculates similarity (from 0.2 up to 1) between the query result and the address acquired during ingestion phase.
- Following steps calculate distance from ingestion coordinates and query ones and select the closest.
- **HBase Output** fills `address_syn` and `codice_toponimo` in QI HBase table.
The method used for toponym code extraction sometimes produces wrong results.

Given a word, it might be impossible to determine the right toponym code (e.g., query using ‘Brunelleschi’ returns ‘Via dei Brunelleschi’, ‘Piazza Brunelleschi’).

Summarize roads with their centers and calculate distances could not return right results (if the point of interest is far from his road center, it could be closer to another road center).
Since we use Karma to generate models, we have to move data from HBase to MySQL.

- ‘DROP’ and ‘CREATE’ respectively deletes MySQL table if already exists and creates a new one.
- ‘getTime’ returns last triple generation timestamp for a specific process (using MySQL `process_manager2` table).
- ‘HBaseToMySQL’ moves data from HBase to MySQL only if timestamp just calculated is older than the date of last ingestion (it prevents to generate triples based on the same data).
- ‘create RegioneCSVTriples’ calls Karma script to generate triples based on the model.
- ‘update last_triples’ updates last triple generation timestamp in `process_manager2`. 
Scheduling Real Time Ingestions

• For **Real Time data** (car parks, road sensors, etc.), the ingestion and triple generation processes should be performed periodically (no for **static data**).

• A scheduler is used to manage the periodic execution of ingestion and triple generation processes;
  – this tool throws the processes with a predefined interval determined in phase of configuration.
## Scheduler: DISCES

<table>
<thead>
<tr>
<th>SCHEDULER NAME ID</th>
<th>FIRE INSTANCE ID</th>
<th>DATE</th>
<th>JOB NAME</th>
<th>JOB GROUP</th>
<th>JOB DATA</th>
<th>STATUS</th>
<th>PROGRESS</th>
<th>TRIGGER NAME</th>
<th>TRIGGER GROUP</th>
<th>PREV FIRE TIME</th>
<th>NEXT FIRE TIME</th>
</tr>
</thead>
</table>
Triplification Overview
Triplification Overview

java -cp target/karma-offline-0.0.1-SNAPSHOT-shaded.jar edu.isi.karma.rdf.OfflineRdfGenerator
--sourcetype DB
--modelfilepath "${MODELPATH}/${processName}_agency.ttl"
--outputfile ${TRIPLESDESTDIRECT}/agency.n3
--dbtype MySQL
--hostname ${IPADDRESSMASTER}
--username ${USERNAMEMYSQL}
--password ${PSWMYSQL}
--portnumber ${PORTMYSQL}
--dbname ${DATABASEMYSQL}
--tablename ${processName}_agency
Karma
An introductory guide for generating RDF triples from relational data in minutes

Karma ([http://usc-is-i2.github.io/karma/](http://usc-is-i2.github.io/karma/)) is a data integration tool developed at the Center of Knowledge Graphs of the Information Science Institute at the University of Southern California. Here at DISIT, we exploit Karma for triplifying relational data.
Karma is a mapping model based on ontology (km4city) from MySQL tables to RDF.

- Triples are uploaded to Virtuoso, an RDF Store.
- It can import MySQL tables but no HBase ones.

Here are the steps to have your triples in your pocket in minutes:
1. Get one of the ready-to-use DISIT VMs
2. Launch the Karma server
3. Build your own model:
   1. Load vocabularies
   2. Load relational tables
   3. (Optional) Load R2RML models
4. Define mappings
5. Export your model
6. Launch the command-line tool
7. Enjoy!
Get one of the ready-to-use DISIT VMs

The DISIT Lab makes available through its Drupal portal a set of ready-to-use virtual machines specifically oriented to data integration. Here is how you can get one of them:

1. Connect to [http://www.disit.org/drupal/?q=node/6690](http://www.disit.org/drupal/?q=node/6690) and scroll down to the section “MACCHINA VIRTUALE, VMSDETL, GIA' PRONTA”

2. Get the “Versione del 2017/2018 0.8 con Phoenix” at [http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8.rar](http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8.rar), or the “Versione del 2017/2018 0.8 con Phoenix per Virtualbox” at [http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8-ovf.rar](http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8-ovf.rar), unless you have a good reason for picking a different one

3. Wait for the download to complete, and extract the archive

4. Launch the VM player of your choice

5. Open the VM

6. Run it


Launch the Karma server

Do the following to run the Karma server:

1. Open a shell
2. Move to ~/programs/Web-Karma-master/karma-web
3. Run mvn -Djetty.port=9999 jetty:run
4. Wait while the Jetty server comes up
5. Connect to localhost:9999 where you will find the Web application for building your model
For that Karma could produce the RDF triples for you, it is required that you instruct it about how relational data should be mapped to semantic data. Documents that describe such a mapping are called R2RML models. Models are built operating a dedicated Karma Web application, and they are exported as ttl files.

Steps:
1. Load Vocabularies
2. Load Relational Tables
3. Load R2RML Models
4. Define Mappings
Have you connected to http://localhost:9999? Are you displaying something similar to this? Right, you are ready to load your vocabularies.

Identify classes and properties that you wish to appear in RDF triples that will be the result of the whole process. Identify vocabularies where such classes and properties are defined. Load them.

Below here is how you load a vocabulary:

1. Hit Import, at the top left corner of the Web page
2. Hit From File
3. Select the vocabulary file (it can be an OWL, RDF/XML, or TTL file)
4. Leave OWL Ontology selected, and hit Next
5. Indicate the correct file encoding if the proposed one is not, and hit Import

You should now see your newly imported vocabulary displayed in the Command History (left column).
Identify tables in your RDB where source data can be found. Load them in your model.

Below here is how you load a table:

1. Hit **Import**, at the top left corner of the Web page
2. Hit **Database Table**. The **Import Database Table** dialog should open.
3. Fill in the form with authentication data and RDB name, and hit **OK**
4. A table listing should appear below the form
5. Put the mouse pointer over the table of your interest
6. **Buttons Import and Preview should appear at the right of the table name**. Hit **Import**.
7. Confirmation message “Table imported in the workspace!” should appear. Hit **OK**.
8. Repeat steps 5 – 8 for each table where source data are found
9. Hit **Close at the bottom right corner of the Import Database Table dialog to dismiss it**
If you already have built and exported a model in the past, and you now just need to make a modification over it, you can start loading and applying your existing model, instead of rebuilding it from scratch.

Below here is how you apply an existing model:

1. Identify the table to which the model has to be applied, and hit the triangle that is displayed next to the table name.
2. Select Apply R2RML Model, and then From File.
3. Select the ttl file that contains your model, and hit Open.
4. Done. Classes and links should appear in the workspace.
Below here is how you specify that a **column** of an RDB table maps to a **data property** of a semantic class, and how to specify that a **column** contains an identifier that can be used for building the URIs of instances of the semantic class:

1. Below RDB table name, identify the blue box that contains the **column** name written in white.
2. Hit the white triangle that you can see next to the **column** name.
3. Hit Set Semantic Type.
4. Pick the checkbox at left of **property of Class**.
5. Hit the Edit button at right of **property of Class**.
6. Select the **semantic class** from the All Classes list. Use the Class textbox for filtering.
7. Select the **property** from the All Properties list. Use the Property textbox for filtering.
8. If the **column** is a key, pick the Mark as key for the class checkbox.
9. You can map the **column** to a **typed** literal, filling the textbox below the label **Literal type**.
10. When you are done, hit Save. Repeat the procedure for each of the **columns** to be mapped.
**Karma**

**Define Mappings**

Below here is how you specify instead that a foreign key of a table in a relational database corresponds to an object property of a semantic class. Scenario: a relational table `stops`, that corresponds to a semantic class `Stop`, has a column `agency_id` where the unique identifier of the agency that manages the stop can be found. Each value in `agency_id` corresponds to one and only one value in a column, let’s say `id`, that can be found in the relational table `agencies`. Table agencies corresponds to the semantic class `Agency`. We wish resources of class `Stop` to be linked each to the appropriate resource of class `Agency`, through the property `gtfs:agency`. For such a purpose, we will do the following:

1. Load relational table `stops` to workspace
2. Map data properties, linking columns in table `stops`, to the class `Stop`, through appropriate properties, as outlined above
3. Identify the grey box with rounded angles that has the name of the class `Stop` written within. It should locate in the workspace. Identify the black triangle that should locate near the right margin of the box. Click it.
4. Select Add Outgoing Link. A popup window should open.
5. Type `gtfs:agency` in the box labelled Property, and `gtfs:Agency` in the box labelled To Class
6. Click Save in the bottom right corner of the popup window to dismiss it.
7. A new grey box, related to class `Agency`, will appear in the workspace, linked to the grey box related to class `Stop` through a link labelled `agency` for brevity
8. Map column `agency_id` as a data property of class `Agency`, also specifying that it is a unique identifier, as described in the above paragraph
9. Repeat for all foreign keys to be mapped, then go to next step *(Export your model)*.
Once you have defined all needed mappings, you have to export your model to a ttl file, so that you can provide it as a parameter to the command-line Karma tool that performs the triplification. Here is how you can export your model:

1. Identify the RDB table whose model you wish to export
2. Hit the black triangle at the right of the table name
3. Select Publish, and then Model
4. A popup should appear at the top right corner of the window, saying “R2RML Model published”
5. Hit Manage Models, in the menu bar at the top of the page
6. A listing should appear of all models that you have exported in the current session
7. Identify the row corresponding to the last model exported, based on the File Name (the name of the RDB table) and the Publish Time.
8. Cut the URL that you can find in the rightmost column of the prospect, and open it in a new tab
9. Save As... the page that you have opened at step 8.
10. Done. The file that you have saved at step 9 is your ready-to-use R2RML model.
Launch the command-line tool

Once you have exported your R2RML model as a **ttl file**, you are ready to perform the triplification:

1. Open a shell
2. Move to `/home/ubuntu/programs/Web-Karma-master/karma-offline`
3. Launch the following as a single line command, customizing parameter values in bold:

```
mvn exec:java
-Dexec.mainClass="edu.isi.karma.rdf.OfflineRdfGenerator"
-Dexec.args="--sourcetype DB --modelfilepath /path/to/model.ttl
--outputfile /path/to/output_triples_file.n3 --dbtype MySQL
--hostname mysql_srv_hostname_or_ip_address --username mysql_user
--password mysql_pwd --portnumber 3306 --dbname mysql_dbname
--tablename mysql_table_name" -Dexec.classpathScope=compile
```
ETL SDK Virtual machine

- videos: [https://www.snap4city.org/drupal/node/139](https://www.snap4city.org/drupal/node/139)
- example on Github: [https://github.com/disit/smart-city-rtl](https://github.com/disit/smart-city-rtl)
- [https://www.snap4city.org/download/video/Snap4City-ETL-VM.rar](https://www.snap4city.org/download/video/Snap4City-ETL-VM.rar)
- “Versione del 2017/2018 0.8 con Phoenix” at [http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8.rar](http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8.rar), or the “Versione del 2017/2018 0.8 con Phoenix per Virtualbox” at [http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8-ovf.rar](http://www.disit.org/vmsdetl/VMSDETL-2017-v0-8-ovf.rar), unless you have a good reason for picking a different one
**Further readings on ETL**

- ETL Development: [https://www.snap4city.org/drupal/node/24](https://www.snap4city.org/drupal/node/24)
- US6. Developing and using processes for data transformation
- HOW TO: add a device to the Snap4City Platform
- HOW TO: add data sources to the Snap4City Platform
- TC6.3. Creating ETL processes for automated data ingestion and data transformation
- TC6.4. Managing ETL processes via Resource Manager, upload, execute, monitor
- TC6.5. Managing Heterogeneous File Ingestion via ETL processes
- TC6.6. Producing data-sets in Bundle via ETL
- TC6.8. ETL processes for data transformation, and exploiting MicroServices/API/RestCall
- TC6.9. ETL processes for multiprotocol and format data ingestion, see on GITHUB for library
- TC6.10. ETL Applications using multiple protocols, and formats for files and to calling services using REST and WS
- TC6.11. Add a new ETL coping with a new Protocol
- ETL processes for massive Data Ingestion and Transformation
Acknowledgements
Main running projects

- Sii-Mobility $\rightarrow$ mobility and transport, sustainability
- REPLICATE $\rightarrow$ ICT, smart City Control room, Energy, IOT
- RESOLUTE $\rightarrow$ Resilience, ICT, Big Data
- GHOST $\rightarrow$ Strategies, smart city
- TRAFAIR $\rightarrow$ Environment & transport
- MOSAIC $\rightarrow$ mobility and transport
- WEEE Life $\rightarrow$ Smart waste, environment
- Smart Garda Lake $\rightarrow$ Castelnuovo del Garda
- 5G $\rightarrow$ Industry 4.0 vs SmartCity
- Green Impact $\rightarrow$ Industry 4.0, Chemical Plant
- SmartBed (laid $\rightarrow$ smart health
- Green Field Peas (soda) $\rightarrow$ Industry 4.0, Chemical plant
- MobiMart and PISA Agreement $\rightarrow$ data aggregation, Living Lab
- Lonato del Garda $\rightarrow$ smart parking, environment
- Herit Data $\rightarrow$ tourism, culture and management
- ISPRA JRC $\rightarrow$ site management and services

Snap4City (C), October 2020
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.
DISIT thanks to

Herit Data: Tourism and Mng.  [https://herit-data.interreg-med.eu/](https://herit-data.interreg-med.eu/)
Snap4City: IOT/IOE smart city  [www.snap4city.org](http://www.snap4city.org)
Trafair: CEF project with several Cities  [http://trafair.eu/](http://trafair.eu/)
Mosaic: Mobility and transport model  [http://www.km4city.org](http://www.km4city.org)
Km4City:  [http://www.km4city.org](http://www.km4city.org)
REPLICATE H2020, SCC1, EC flagship  [http://replicate-project.eu/](http://replicate-project.eu/)
Sii-Mobility SCN MIUR:  [http://www.sii-mobility.org](http://www.sii-mobility.org)
Feedback: retail and GDO Big Data analytics
5G with 3G-Wind, Open Fiber, Estra
Coll@bora Social Innovation, MIUR:  [http://www.disit.org/5479](http://www.disit.org/5479)
TRACE-IT, RAISSS, TESYSRAIL, ...
Mobile Emergency:  [http://www.disit.org/5404](http://www.disit.org/5404)
Further readings

- HOW TO: create a Dashboard in Snap4City
- HOW TO: add a device to the Snap4City Platform
- HOW TO: add data sources to the Snap4City Platform
- HOW TO: define privacy rules for personal data, produced by the end-users own device
- HOW TO: Develop Smart Applications, Snap4City development Life Cycle
- HOW TO: HLT vs Ingestion, and HLT vs Widgets
- HOW TO: Develop an IOT Application for Data Ingestion
- HOW TO: Upload data into Knowledge Base, ServiceMap (triple upload)
- HOW TO: Create as set of Devices with BulkProcessing
- HOW TO: Create an IOT Device Model
- HOW TO: Create an IOT Device Instance from IOT Directory tool
Be smart in a SNAP!

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