

Data Ingestion Tutorial

Corso: Sistemi Distribuiti, UNIFI, DISIT lab



Data Ingestion Step

- **HOW TO Create an IOT Device Model:** <https://www.snap4city.org/591>
- **HOW TO: Create an IOT Device Instance:** <https://www.snap4city.org/590>
[Entity/IoT Directory tool - IoTAPP]
- **HOW TO Develop an IOT Application for Data Ingestion** <https://www.snap4city.org/593>
- **HOW TO Managing Notifications on IOT Application** <https://www.snap4city.org/142>

Data Ingestion Example

Open Weather data, Open Pollution data and Open Sea Condition data for each pilot.

Data Source	Data Source Main Address	Periodicity for Data ingestion	Variables
Open Weather	https://openweathermap.org/api	30 mins	<ul style="list-style-type: none">• airHumidity• airTemperature• cloudCoverPerc• feelsLike• groundLevel• maxTemperature• minTemperature• pressure• seaLevel• sunrise• sunset• visibility• windDirection• windSpeed

Data Ingestion Example

Data Source	Data Source Main Address	Periodicity for Data ingestion	Variables
Sea Conditions	https://open-meteo.com/	60 mins	<ul style="list-style-type: none">• oceanCurrentDirection• oceanCurrentVelocity• swellWaveDirection• swellWaveHeight• swellWavePeakPeriod• swellWavePeriod• waveDirection• waveHeight• wavePeriod• windWaveDirection• windWaveHeight• windWavePeakPeriod• windWavePeriod

Data Ingestion Example

Data Source	Data Source Main Address	Periodicity for Data ingestion	Variables
Open Pollution	https://openweathermap.org/api/air-pollution	30 mins	<ul style="list-style-type: none">• CO• NO• NO2• NH3• O3• SO2• PM2.5• PM10

Dashboard:

<https://www.snap4city.org/dashboardSmartCity/view/newTheme.php?iddashboard=NDMzNQ==>

Data Visualization in a Dashboard

Devices selector

TOURISMO Malta Wed 5 Feb 11:59:21

openweathersMalta
Sea conditions malta
Air pollution Malta

OW_2562735
VALUE NAME: OW_2562735
Last update: 2025-02-05 11:54:42.000+01:00

Description	Value	Button				
airHumidity	72	Last	4h	24h	7d	30d
airTemperature	14.82	Last	4h	24h	7d	30d
cloudCoverPerc	75	Last	4h	24h	7d	30d
dateObserved	2025-02-05T10:54:42.000Z	Last	4h	24h	7d	30d
feelsLike	14.24	Last	4h	24h	7d	30d
groundLevel	1026	Last	4h	24h	7d	30d
maxTemperature	15.08	Last	4h	24h	7d	30d
minTemperature	14.82	Last	4h	24h	7d	30d
pressure	1028	Last	4h	24h	7d	30d

14.8 °C

airTemperature - Day

14:00 16:00 18:00 20:00 22:00 5. Feb 02:00 04:00 06:00 08:00 10:00 12:00

MRDD Foundation

Partner link

Selected device

Related values

Service map

Data Trend

HOW TO Create an IOT Device Model

Edit Model - SirSensors

General Info | IoT Broker | Static Attributes | Values

General Info

SirSensors
Name: Ok
Description: model for Sir Sensors data

weather
Device Type: Ok
Sensor: Kind

SIR
Producer: Ok
Frequency: 900

Refresh Rate: 300
Healthiness Criteria: Healthiness Value

Automatically generated: Ok
Key Generation: Edge-Gateway Type

Save as Cancel Confirm

Edit Model - SirSensors

General Info | IoT Broker | Static Attributes | Values

Static Attributes

Device in Mobility

Subnature: Weather Sensor (Environment)

Add Attribute

Save as

Static Attributes

Edit Model - SirSensors

dateObserved | Timestamp (timestamp) | timestamp in millisecond | string

Value Name: Ok | Value Type: Ok | Value Unit: Ok | Data Type: Ok

Refresh rate: 900 | Healthiness Criteria: Healthiness Value | Remove Value

temperature | Temperature (temperat) | Celsius (°C) | float

Value Name: Ok | Value Type: Ok | Value Unit: Ok | Data Type: Ok

Refresh rate: 900 | Healthiness Criteria: Healthiness Value | Real Time: | Remove Value

rainDelta15 | Rain (rain) | Millimeter (mm) | float

Value Name: Ok | Value Type: Ok | Value Unit: Ok | Data Type: Ok

Values

Edit Model - SirSensors

General Info | IoT Broker | Static Attributes | Values

Broker

orionUNIFI | ContextBroker | ngsi | Protocol

json | Format

Service/Tenant: only ngsi w/MultiService supports Service/Tenant selection | ServicePath: only ngsi w/MultiService supports ServicePath

Save as Cancel Confirm

HOW TO Create an IOT Device instance

iotdirectory-new-device-from-model

Edit device - SIRSensor_TOS30355400

Info	IoT Broker	Position	Static Attributes	Values	Status
SIRSensor_TOS30355400					
Device Identifier			Model		
weather			Mac Address		
Device Type Ok			Edge-Gateway URI		
Edge-Gateway Type			MyOwnPublic		
SIR			Ownership		
Producer			Service URI		
900			http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT		
Frequency			031b1802-0fbc-4341-b52c-a932ba6afef2		
KEY 1					
6ab2b7c1-00a8-4977-938c-11e994518163					
KEY 2					
031b1802-0fbc-4341-b52c-a932ba6afef2					

Buttons: Save as, Cancel, Confirm

Edit device - SIRSensor_TOS30355400

Info	IoT Broker	Position	Static Attributes	Values	Status
44.171			10.2081		
Latitude Ok			Longitude Ok		

Buttons: Save as

Edit device - SIRSensor_TOS30355400

Info	IoT Broker	Position	Static Attributes	Values	Status
<input type="checkbox"/> Device in Mobility					
Subnature					
Weather Sensor (Environment)					
Locality		Minucciano		Value	Remove
Region		LU		Value	Remove
Is in road		http://www.disit.org/km4city/resou		Value	Remove
Altitude		666		Value	Remove
River name		Bagnone		Value	Remove
Add Attribute					

Buttons: Save as, Cancel, Confirm

HOW TO Create an IOT Device instance

The screenshot displays the Node-RED web interface. On the left, a sidebar shows the user profile for 'envdatacollection' and a navigation menu with 'Processing Logics / IOT App' selected. The main workspace contains a flow with the following components:

- Flow 1:** Starts with a 'Tutti' trigger, followed by a 'timestamp' node, a 'function' node, and a series of nodes for sensor data: 'anemo', 'termo', 'igro', and 'pluvio'. Each sensor node is followed by a 'function' node.
- Flow 2:** Starts with a 'function' node, followed by a 'split' node, a 'limit 1 msg/20s' node, and another 'function' node.
- Flow 3:** Starts with an 'http request' node, followed by a 'json' node, a 'function' node, and an 'iotdirectory-new-device-from-model' node.
- Flow 4:** Starts with a 'timestamp' node, followed by a 'function' node, an 'http request' node, and a 'json' node.
- Flow 5:** A 'msg payload' node that receives input from the 'iotdirectory-new-device-from-model' node and the 'function' node in Flow 4.

The right sidebar shows the 'help' panel with a search bar and a list of help topics, including 'Node-RED v2.2.2' and various contrib modules.

HOW TO Create an IOT Application for Data Ingestion

The screenshot displays the Snap4City Node-RED interface. The left sidebar shows the user profile for 'envdatacollection' and a list of dashboards and processing logics. The main workspace shows a flow titled 'SIR' with the following components:

- Input:** A 'timestamp' node followed by a 'function' node.
- Processing:** A series of nodes for different sensor types: 'anemo', 'terno', 'igro', and 'pluvio', each followed by a 'function' node.
- Rate Limiting:** A 'limit 1 msg/s' node is highlighted with a red circle.
- Output:** A 'msg.payload' node and a 'function' node that triggers an 'ibware onion out api v2' node, which sends data to the email 'lucianoalessandro.ipsaropalesi@unifi.it'.

The right sidebar shows the Node-RED v2.2.2 help menu with various contrib nodes listed.

HOW TO Create an IOT Application for Data Ingestion

FORMATO ISO - GMT

```
{"id":"SIRSensor_TOS30355400",  
 "type":"weather",  
 "dateObserved":{"type":"string","value":"2024-05-30T07:45:00.000Z"},  
 "humidity":{"type":"float","value":""},  
 "rainDelta15":{"type":"float","value":0},  
 "temperature":{"type":"float","value":""},  
 "windDirection":{"type":"float","value":""},  
 "windGust":{"type":"float","value":""},  
 "windSpeed":{"type":"float","value":""}}
```

Entity Instances, IoT Devices

Model: SIRSensor
Longitude: 10.2081
Device Uri: http://www.disit.org/km4city/resource/iot/orionUNIFI/DISIT/SIRSensor_TOS30355400
Organization: DISIT
Owner: undefined
Created on: 2022-10-28 11:47:02

Producer: SIR
Latitude: 44.171

PAYLOAD NGSI v1
PAYLOAD NGSI v2
K1: 6ab2b7c1-00a8-4977-938c-11e994518163
K2: 031b1802-0fbc-47...32c-a932ba6afef2

Location	View
	VIEW
	VIEW
	VIEW
	VIEW
	VIEW
	VIEW

Showing 461 to 467 of 467 entries

Previous 1 ... 43 44 45 46 **47** Next

HOW TO Managing Notifications on IOT Application

```

if(msg.payload.status.statusCode!=200){
msg.topic="Check Acquisition SirSensor Sensor»
msg.payload="Problem with"+JSON.stringify(msg.payload)
return msg;
}
return null;

```

IoTApp

The screenshot shows the Snap4City Node-RED interface. The left sidebar contains a navigation menu with categories like 'Dashboards', 'Data Management', and 'Processing Logics / IOT App'. The main workspace displays a Node-RED flow with several function nodes and a 'fiware orion out api v2' node. A 'delegate-my-device' node is highlighted with a blue box. A red circle highlights the 'Deploy' button in the top right corner. A tooltip for the 'delegate-my-device' node is shown on the right, detailing its inputs and outputs.

delegate-my-device
It allows to delegate a device.

Inputs
A JSON with these parameters:

- id** (string): the nome of the kpi device (you MUST have the ownership of the device)
- kind** (string): Kind of delegation. You can choose between READ_ACCESS, READ_WRITE and MODIFY.
- usernameDelegated** (string): Username of the person to be delegated to view the device
- groupDelegated** (string): Group to be delegated to view the device

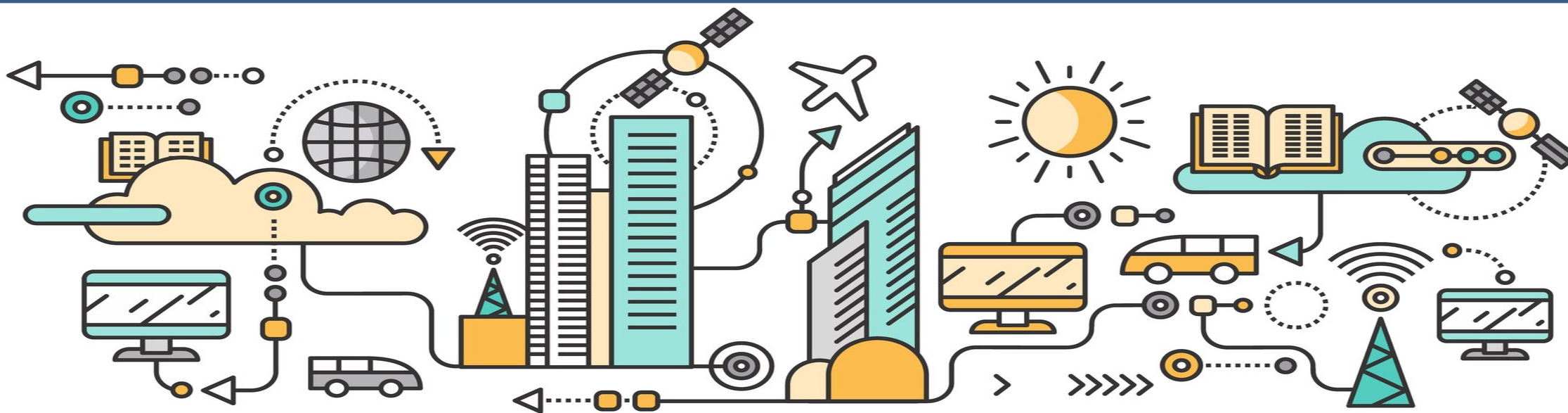
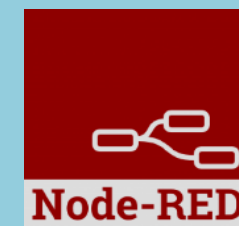
An example of the JSON array filled with correct data:

```
{  
  "id": "nameDevice",  
  "usernameDelegated": "username",  
  "kind": "READ_ACCESS"  
}
```

Outputs
Returns an object containing the delegation

Details
The node can receive a JSON with the parameters described in the Inputs section and with them generate the output JSON. If the values are not present in the input JSON, these are read by those in the configuration. If they are not

Node-RED Libraries



<https://flows.nodered.org/search?term=>



Node-RED [home](#) [about](#) [blog](#) [documentation](#) [forum](#) **[flows](#)** [github](#)

Search library [+](#) [Sign in with GitHub](#)

[nodes](#) **[flows](#)** [collections](#) [recent](#) [downloads](#) [rating](#)

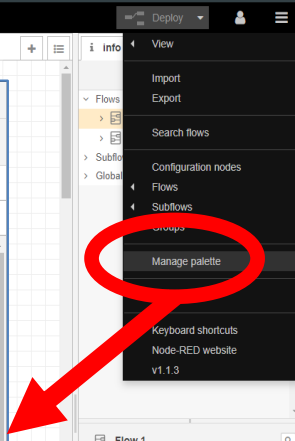
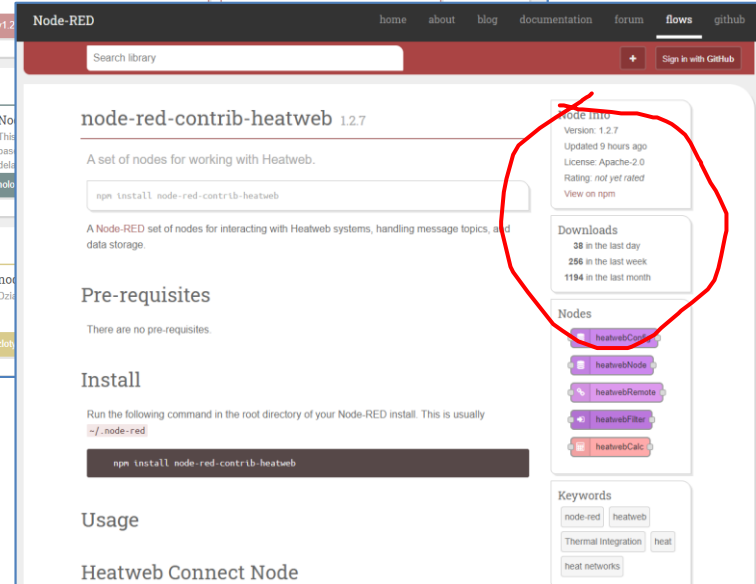
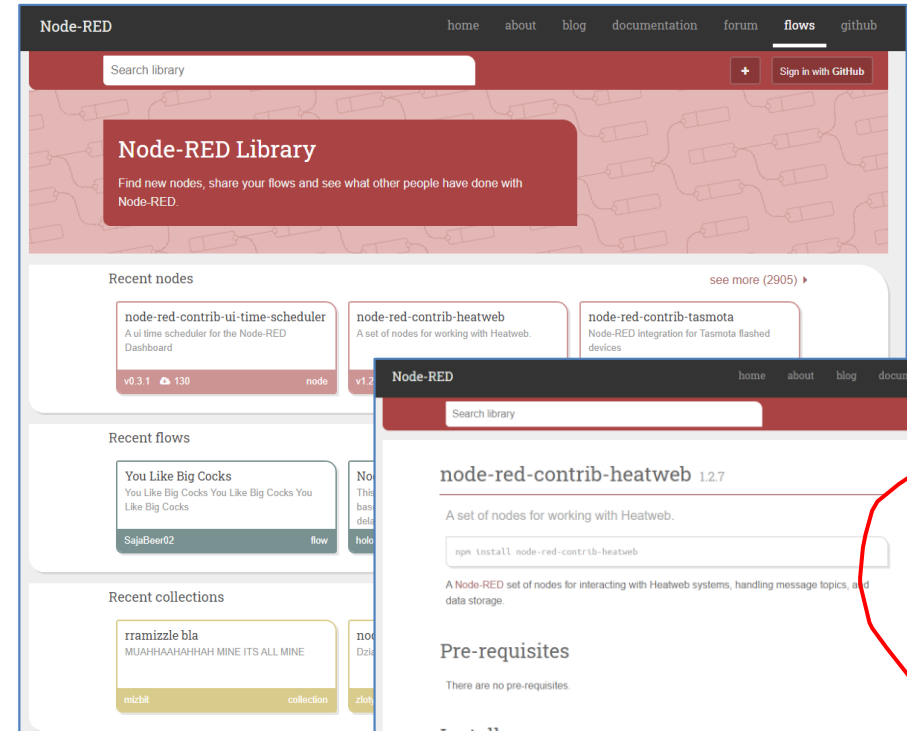
node-red-contrib-websocket-header Custom Websocket with Header v0.5.2 144 node	node-red-contrib-mobilealerts This provides a node for receiving Mobile Alerts status infos. v3.0.5 71 5.0 node	node-red-contrib-cx-alarm-log A Node-RED industrial alarm parser for simple HMI applications. v1.1.0 16 5.0 node
node-red-contrib-websocket-header-acknowledge Custom Websocket with Header v0.0.1 0 node	node-red-contrib-websocket-header-subscriber Custom Websocket with Header v0.0.1 0 node	node-red-contrib-message-queue Message queueing for Node-RED v1.1.4 11 node
node-red-contrib-zigbee2mqtt Zigbee2mqtt connectivity nodes for node-red v2.0.9 1326 4.6 node	@mschaeffler/node-red-asterisk-ami-manager Transfer Asterisk AMI events to json object string representation v1.1.2 6 node	node-red-contrib-sendmail send emails with help of a local sendmail command. v1.0.5 16 node
node-red-contrib-nooperation just do nothing. v1.0.6 6 node	node-red-contrib-sun-position NodeRED nodes to get sun and moon position v2.1.1 1259 4.8 node	node-red-contrib-websocket-header-test Custom Websocket with Header v0.0.1 0 node
@nikolay_kuropatkin/node-red-contrib-dynamic-file-path A simple node that generate a file by dynamic file path v0.0.8 164 5.0 node	node-red-contrib-miio-localdevices Node for Node-Red to control Mi Devices locally via node-mihome (Humidifiers, Purifiers, Heaters, Lights - list of devices to be enlarged). v0.4.1 270 1.9 node	node-red-contrib-daylight-rgbw Daylight RGBW Color control for Node RED v2.1.3 128 node

1 of 429 [Next](#)

Load Library from Palette

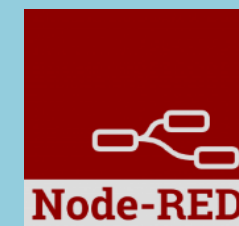


<https://flows.nodered.org/>



Two views of the same libraries

Node-RED in SnaP4City



How to Design

1. **Business Logic** is going to be implemented in Proc.Logic (IoT App), with a set of flows.
2. **Decompose your problem** and sequence diagram in single Data/event Flows, from client side and server side.
3. **Identify the single Data/Event Flow**, as those that start from a certain event (periodic or provoked from other messages), and that finish with: sending of data in the storage, change status, send an event, provide a message into a dashboard, send an email, etc.
4. **Design the single Data/Event Flows** with a mixt of possible **activities**.
 1. The design can be performed using data flow diagrams.
 2. It can have sequences, switch, serialization, packing, joining, distribution, communication, transformation, search, etc.
5. When the design of Data/Event Flow mechanism is clear the designers can pass to directly sketch the flow in Node-RED which is a visual programming.
6. **Incrementally improve the Proc.Logic** (IoT App) Node-RED flows by adding nodes needed
7. **Once obtained the Proc.Logic** (IoT App) Node-RED flows in the correct data model you can send data to the ingestion broker, but also perform many other actions on several services.



Sept 2024 collection

Two Snap4City Libraries



Navigation menu on the left:

- > common
- > function
- > network
- > input
- > output
- > sequence
- > parser
- > storage
- > social
- > advanced
- > Advanced FTP
- > location
- > NGSI
- > Iwm2m
- > S4C SearchDev
- > S4C Utility
- > S4C Mapping
- > S4C Management
- > S4C Data Analytic
- > S4C Big Data
- > S4C IoT App
- > S4C Open Maint
- > S4C IoT
- > S4C Whatif
- > S4C Search
- > S4C Data
- > S4C KPI Data
- > S4C Dashboard
- > S4C Sigfox
- > S4C LogDev
- > S4C View
- > S4C Social
- > dashboard
- > time

Library categories and their contents:

- S4C SearchDev**
 - service search
 - service search near gps position
 - service search near service
 - service search within gps area
 - service search within wkt area
 - service search within stored wkt area
 - service search by municipality
 - service search by queryid
 - full text search dev
 - full text search within wkt area
- S4C Utility**
 - full text search within gps area
 - full text search near gps position
 - full text search exp
 - event search dev
 - event search exp
 - event search within wkt area
 - event search within gps area
 - event search near gps position
 - address search near gps position
 - geometry search near gps position
 - address poi search by text
- S4C Mapping**
 - address poi search by text exp
 - address poi search by text near gps position
 - bus routes search
 - bus routes search near gps position
 - bus routes search within gps area
 - bus routes search within wkt area
 - bus routes
- S4C Data Analytic**
 - point within polygon
 - routing
 - heatmap picker
 - coordinates to address
 - service info
 - edge-tunnel-to-cloud
 - service info mapped
 - mapping
 - set mapping
 - check exist job
 - check exist trigger
 - is in standby mode
 - is shutdown
 - is started
 - get currently executing jobs
- S4C Search**
 - service search near marker
 - service search within circle
 - service search within polygon
 - service search along path
 - full text search within circle
 - full text search within polygon
 - full text search along path
 - full text search usr
 - event search near marker
 - event search within circle
 - event search near marker
 - event search within circle
 - event search near marker
 - event search within circle
 - bus routes search near marker
 - bus routes search within circle
 - bus routes search within polygon
 - bus routes search within polygon
 - tpl agencies
 - tpl lines
- S4C Data**
 - tpl routes by agency
 - tpl routes by line
 - tpl stops by route
 - tpl stop timeline
 - recommendatio within circle
 - value type search near marker
 - value type search within circle
 - value type search within polygon
 - value type search along path
 - get my data
 - get my delegator
 - get my delegated
 - get my activity
- S4C IoT App**
 - portia crawler
 - iotapp restart
 - iotapp upgrade
 - ownership

<https://flows.nodered.org/search?term=snap4city>

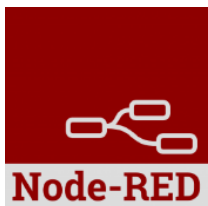
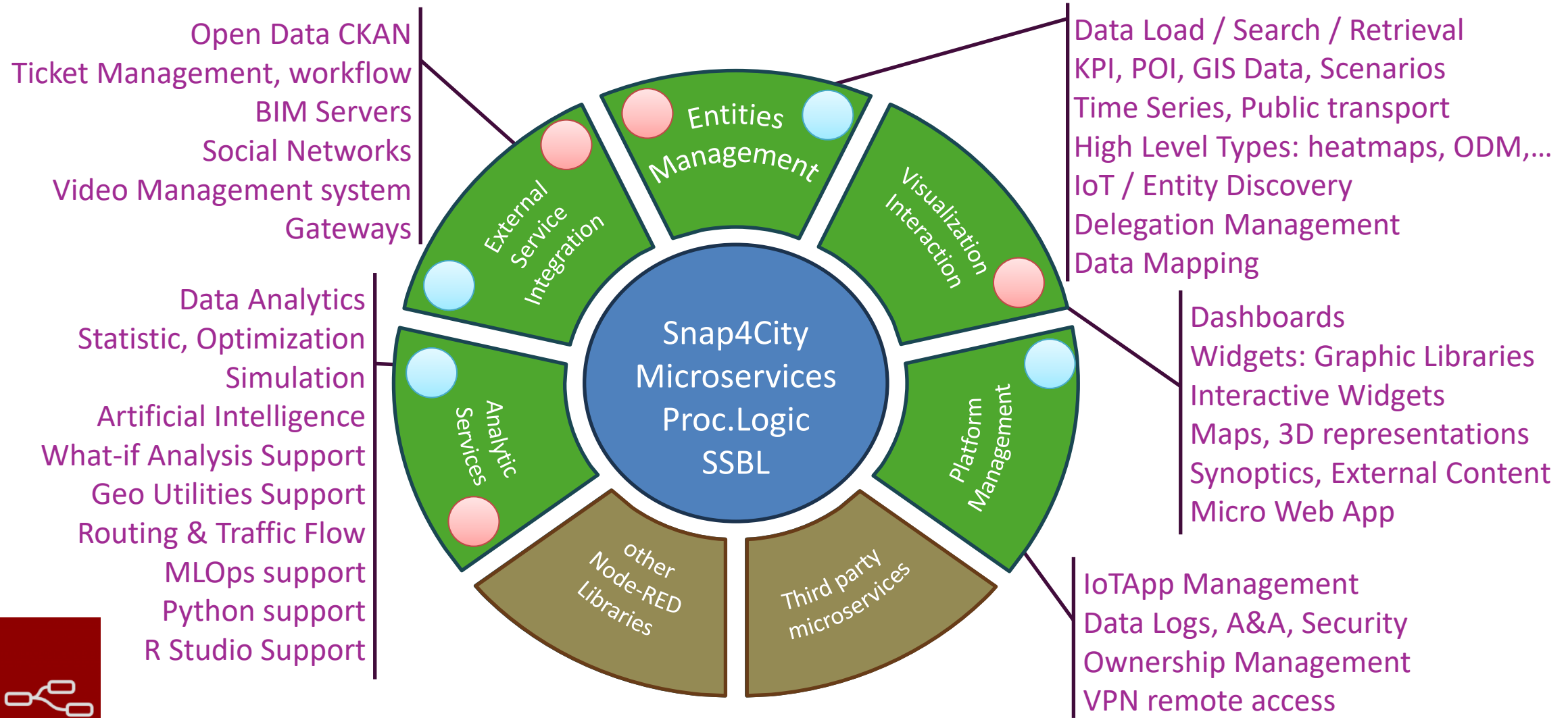
The Processing Logic (IoT App) microservices

Actually, there are more than 180 nodes/blocks in the Snap4City libraries on Processing Logic (IoT App) which can really facilitate your life and save you time in producing Smart Applications for composition of the following microservices and using those that you can install from internet, thousands of functionalities:

- **Data ingestion:** more than 100 protocols IOT and Industry 4.0, web Scraping, external services, any protocol database, etc.
- **Data access:** save/retrieve data, query search on expert system, georeverse solution, search on expert system Km4City ontology, call to Smart City API, etc.
- **Data Transformation/transcoding:** binary, hexadecimal, XML, JSON, String, any format
- **Integration:** CKAN, Web Scraping, FTP, Copernicus satellite, Twitter Vigilance, Workflow OpenMaint, Digital Twin BIM Server, any external service REST Call, etc.
- **Manipulation of complex data:** heatmaps, scenarios, typical time trend, multi series, calendar, maps, etc.
- **Access to Smart City Entities and exploitation of Smart City Services:** transport, parking, POI, KPI, personal data, scenarios, etc.
- **Data Analytic:** managing Python native, calling and scheduling Python/Rstudio containers as snap4city microservices (predictions, anomaly detection, statistics, etc.)
- **User interaction on Dashboard:** get data and message from the user interface, providing messages to the user (form, buttons, switches, animations, selector, maps, etc.), send data to special graphical widgets: D3, Highcharts, etc.
- **Custom Widgets:** SVG, synoptics, animations, dynamic pins on maps, etc
- **Event management:** Telegram, Twitter, Facebook, SMS, WhatsApp, CAP, etc.
- **Special tools as:** routing, georeverse, Twitter Vigilance and sentiment analysis, etc.
- **Hardware Specific Devices:** Raspberry Pi, Android, Philips, video wall management, etc.
- **Etc. etc.**

> 60.000 downloads

Areas



examples

Node shape	Description	Snap4City or standard
	To generate injection messages into a flow, scheduled/periodic or on manual demand by click it on left.	standard
	DATA TRANSFORM A JavaScript function , from a JSON input to one or more JSON outputs, which can be produced by setting it.	standard
	SAVE to STORAGE via internal BROKER To send an Entity Message of an Entity Instance into the storage. The Entity Instance has to be registered on Entity Directory (IoT Directory) and you have to be the owner or to be delegated in READ-WRITE to send messages to it. The node represents the broker, so that the same node can be used to send any Entity Message you need. Please manage the error in output.	Snap4city
	SUBSCRIBE to an Entity change on BROKER To subscribe the Processing Logic (IoT App) to receive event-driven notifications related to Entity Instances changes. The node is substantially a listener connected to an Orion Broker. You can subscribe to many Entities and then to get all of them from the output of the listener. The new version will go to provide an input port to send at this listener multiple subscriptions. PLEASE NOTE THAT ALL THAT YOU CAN DO IN MQTT CAN BE DONE IN ORION BROKER NGSI. Moreover, Orion broker is authenticated, in SSO, provides JSON, etc. This node-red block allows you to subscribe to a topic / device and get event driven actions on IoT App directly. Please manage the error in output.	Snap4city
	READ from STORAGE Query call to Smart City API to get any information about a SURI, ServiceURI. There are many other Nodes which can be used to pose Smart City API queries in very simple manner and recover vectors of ServiceURIs. Please manage the error in output.	Snap4city

Saving Data on Storage

Even Driven

Get Data from Storage

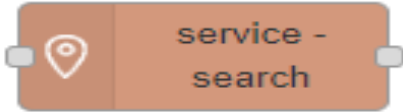

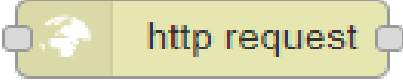
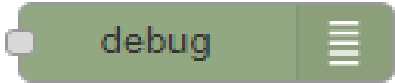
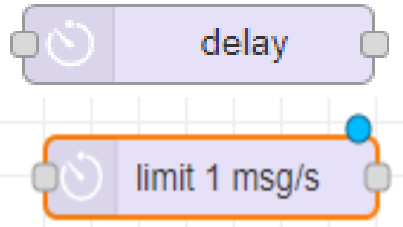
Get Data from Storage

Send email

Gen/access to HTTP, HTML pages

Monitor messages

Stream Delay, limiting rate

	<p>SEARCH on STORAGE</p> <p>To perform queries on the storage to obtain a list of ServiceURI. The nodes of this family can allow you to perform searching queries by filtering for distance, area, subnature/category, values of attributes, time period, etc. Please manage the error in output.</p>	<p>Snap4city</p>
	<p>Send email. With other nodes you can send Telegram, SMS, etc.</p>	<p>standard</p>
	<p>To send a REST CALL (get, post, etc.). Please USE THIS NODE ONLY for the access at external API and not to access at the Snap4City API for which a lot of MicroServices are accessible as NODEs/Blocks in the Processing Logic and they are simpler to be used and ready to use. Please manage the error in output.</p>	<p>standard</p>
	<p>A block which is printing on debug view the data JSON passed in its input. Please note that the node can be tuned to provide only msg.payload or the full JSON message, change configuration of the node.</p>	<p>standard</p>
	<p>A node to insert a delay to each message arriving, or to limit the rate of messages in output. In some cases, the node creates a buffer of messages regularizing the rate in output if the rate in input is greater in some moments.</p>	<p>standard</p>

	A block which is printing on debug view the data JSON passed in its input. Please note that the node can be tuned to provide only <u>msg.payload</u> or the full JSON message, change configuration of the node.	standard
	To create an Entity Instance (device instance) from a model prepared on Entity Directory (IoT Directory).	Snap4city
	To change the ownership of an Entity Instance (IoT Device).	Snap4city
	To delegate a certain Entity Instance (IoT Device) to some other user for which you <u>have to know</u> the Nickname. Delegations can be: <u>Read access</u> , <u>Read write</u> , Modify (to modify the Entity Instance structure)	Snap4city
	To show something on Snap4City dashboard with a single content widget (one of the simplest widgets). A large set of dashboard nodes/widgets to send and retrieve data to/from dashboards are provided. This specific Nodes allows to send on dashboard HTML formatted messages with some limitations. Full HTTP widget is also accessible. See in the following section for the Full list of Nodes for Snap4City Dashboard	Snap4city
	MQTT broker listener , to receive messages from the Broker. Another similar node can be used to send MQTT messages to the MQTT broker. This node allows to perform a subscription to a topic of the MQTT broker.	standard
	DATA ANALYTICS Request performed on a Container including a Python data analytics, which is loaded into the node and the container is created at the first Deploy of the Processing Logic. Similar Approach is performed for RStudio Data Analytics.	Snap4city
	SPLIT: This block takes in input a buffer, or an array, or an object and split it on a set of messages in output, for each line in the buffer, each element of the array, each element in the object, respectively.	standard
	JOIN: This block takes in input a set of messages and join/merge them into a single message (string, buffer, <u>array</u> or object, etc.), on the basis of specific criteria.	standard

*USage To
be trained*



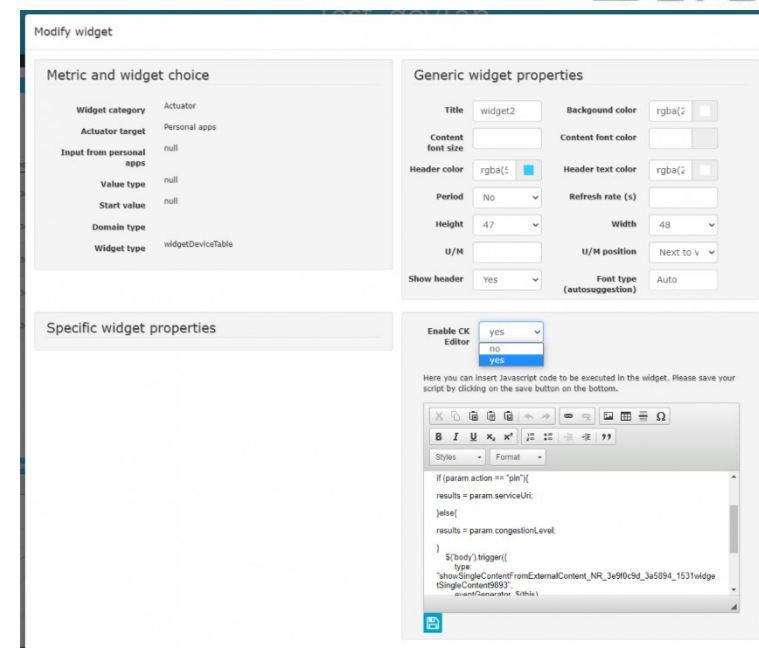
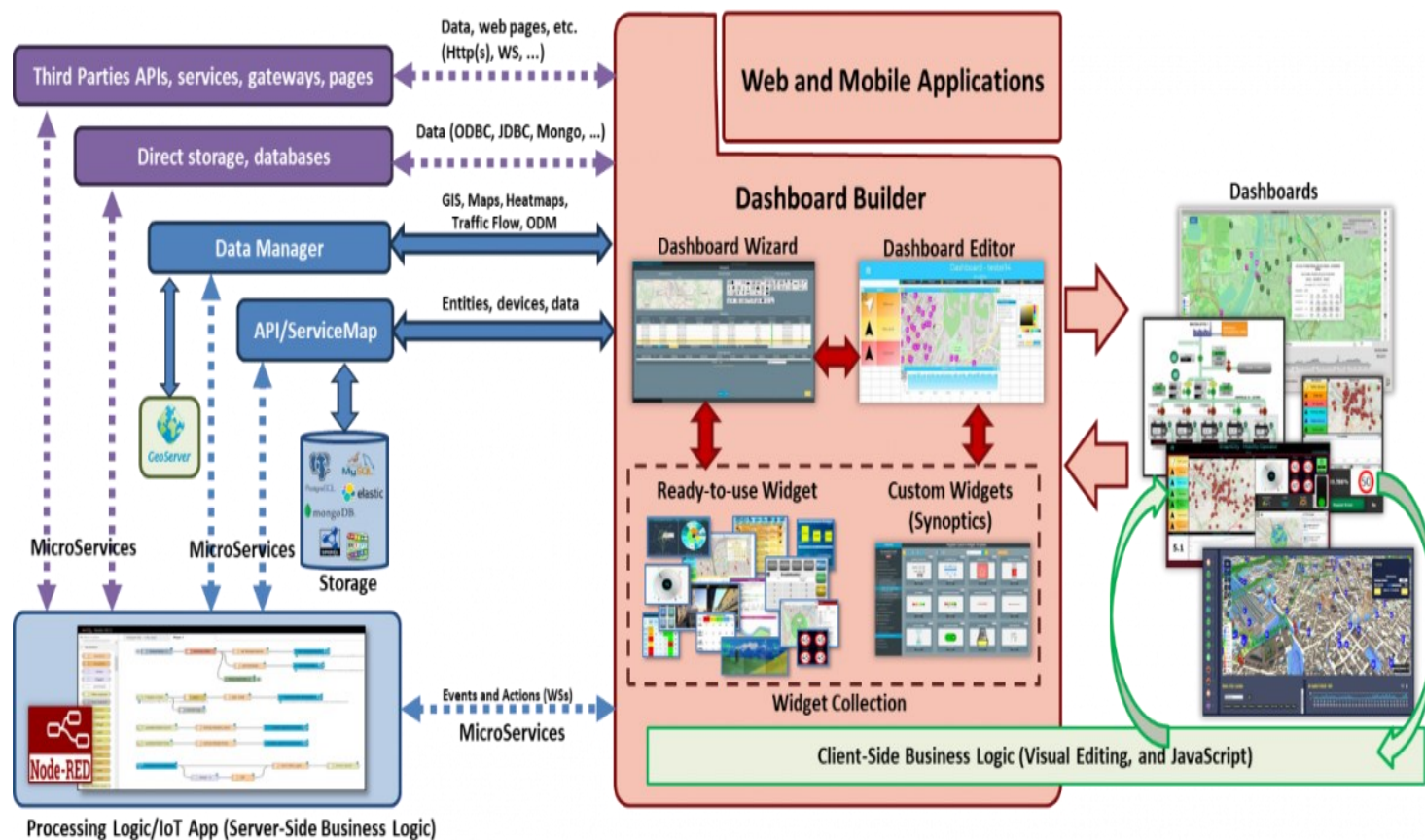
Esempi Data Ingestion

CSBL Tutorial



Client-Side Business Logic on Snap4city.org platform

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



Computing Predictions and Heatmaps

The screenshot displays the SNAP4CITY web application interface. At the top, the title "Computing Predictions And Heatmaps - Cloned Last" is shown along with the date "Wed 28 May 16:06:18".

Left Sidebar: A navigation menu with icons and labels for "Scenario Editor", "Air quality Sensors", "Weather Sensors", "Traffic Sensors", "OpenWeather", and "Traffic Flow".

Map Area: A map of Florence, Italy, with numerous red circular markers representing sensors. A "Selector - Map" panel is overlaid on the map, containing a "MAPS" dropdown, zoom controls, and checkboxes for "Show Road graph", "Show Traffic Sensors", and "Draw sub-area". A "Filter by road types" option is also present.

Scenario Panel: A dark overlay panel titled "Scenario" with the following details:

- Load Scenario: Init Acc
- Scenarios waiting to be processed: AirQuality
- Scenario version: 2024-07-30 14:46:12
- Buttons: Load Scenario, Clean

Form Invio: A panel on the right with the following controls:

- Buttons: Compute Predictions, Compute Heatmaps, Show Heatmaps
- Heat Map Model: heatMapModel_traffic
- Heat Map: AlessandroTryHeatmap4_vehicleFlow
- Buttons: Heatmap Update, Show Heatmap, Remove Heat Map

Data Visualization: Two charts are displayed at the bottom:

- DISIT:OrionUNIFI:METRO1059 - VehicleFlow:** A line chart showing vehicle flow (car/h) from July to November. The y-axis ranges from 0 to 1k. The x-axis shows dates from 29 Jul to 11 Nov. The chart shows a clear daily cycle with peaks around 1000 car/h.
- Selected Trend And Predictions:** A line chart showing anomaly levels from December 2024 to May 2025. The y-axis ranges from 40 to 200. The x-axis shows dates from Dec '24 to May '25. The chart compares "METRO468 - anomalyLevel" (blue line) and "METRO468 - Predicted - anomalyLevel" (yellow line).



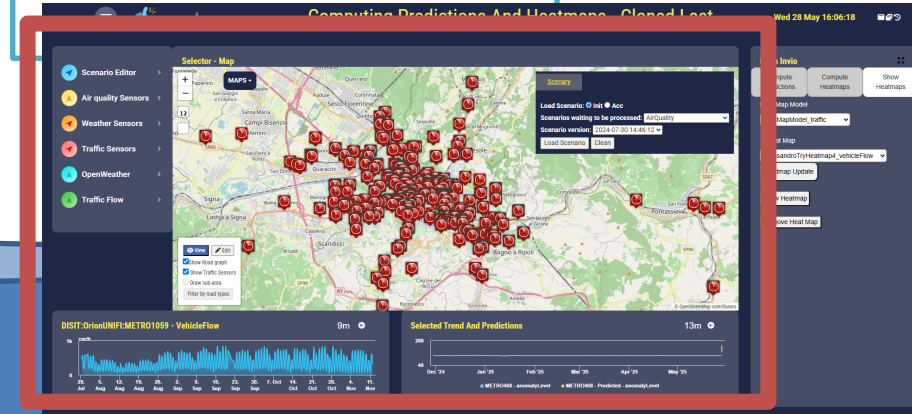
CSBL

API PYTHON

CSBL



Storage



Javascript Request

javascript

```
$.ajax({  
  url: 'https://api.example.com/data',  
  type: 'POST',  
  contentType: 'application/json',  
  data: JSON.stringify({  
    name: 'John',  
    age: 30  
  }),  
  headers: {  
    'Authorization': 'Bearer YOUR_ACCESS_TOKEN'  
  },  
  success: function(response) {  
    console.log('Success:', response);  
  },  
  error: function(xhr, status, error) {  
    console.error('Error:', error);  
  }  
});
```

url: Target API endpoint.

type: 'POST': Method of the request.

contentType: Tells the server the format of the data (application/json).

data: Payload sent to the server (must be stringified).

headers: Include Authorization if needed.

success: Callback for a successful response.

error: Callback for failures.

API Flask

Introduzione a Flask

Cos'è Flask?

- Flask è un micro web framework per Python.
- Leggero e flessibile, ideale per creare API e applicazioni web.
- Basato su Werkzeug (server) e Jinja2 (templating engine).

Caratteristiche principali:

- Minimalista: solo il necessario per iniziare.
- Estendibile con plugin e librerie.
- Ottimo per progetti piccoli o medi.

Componenti principali:

- Routing: associa URL a funzioni
- Request/Response: gestisce dati da form, JSON, URL
- Struttura modulare per grandi progetti
- Tante estensioni utili.

```
python

from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
    return "Ciao, Flask multi-thread!"

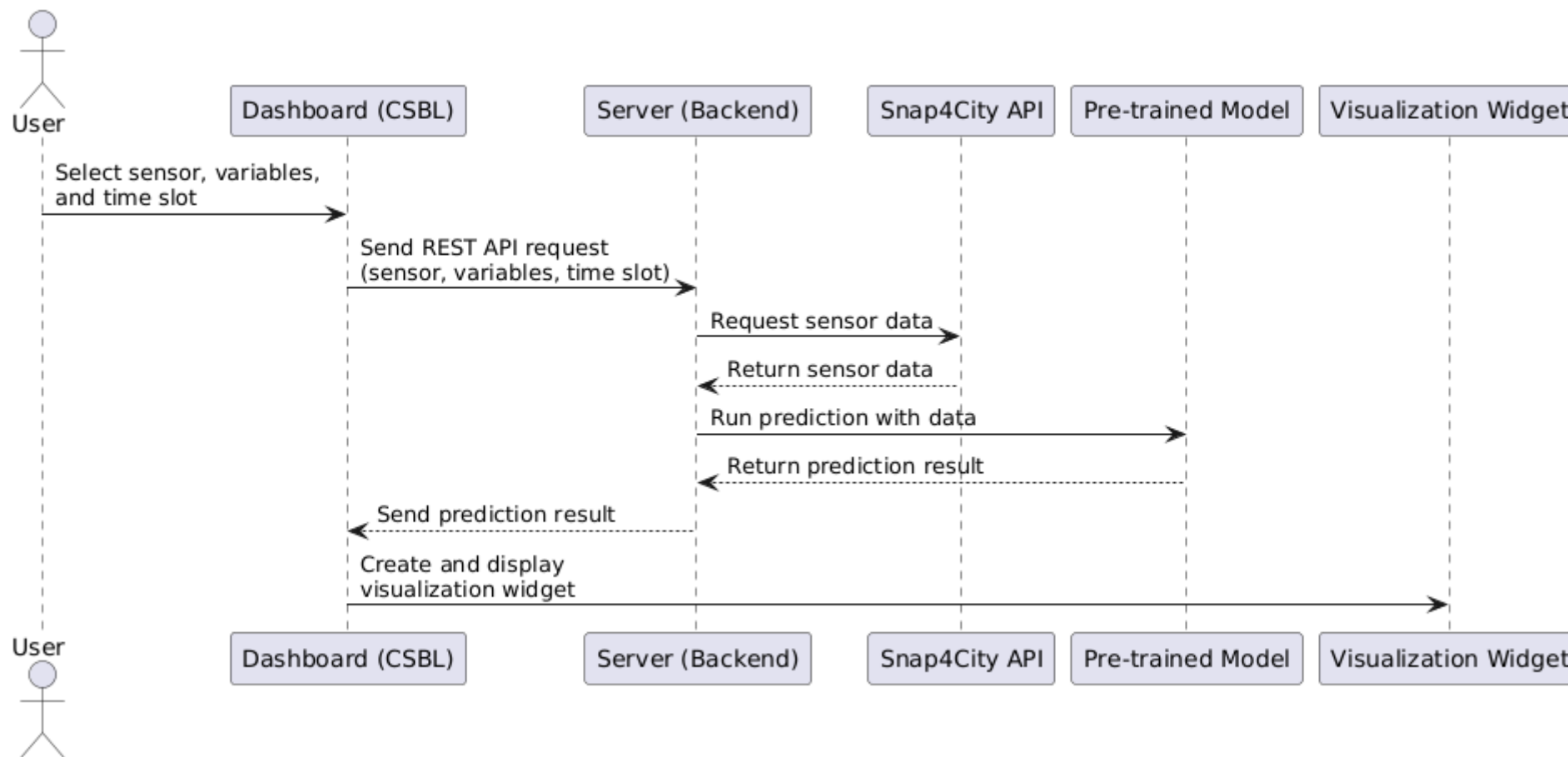
if __name__ == "__main__":
    app.run(debug=True, threaded=True)
```

threaded=True

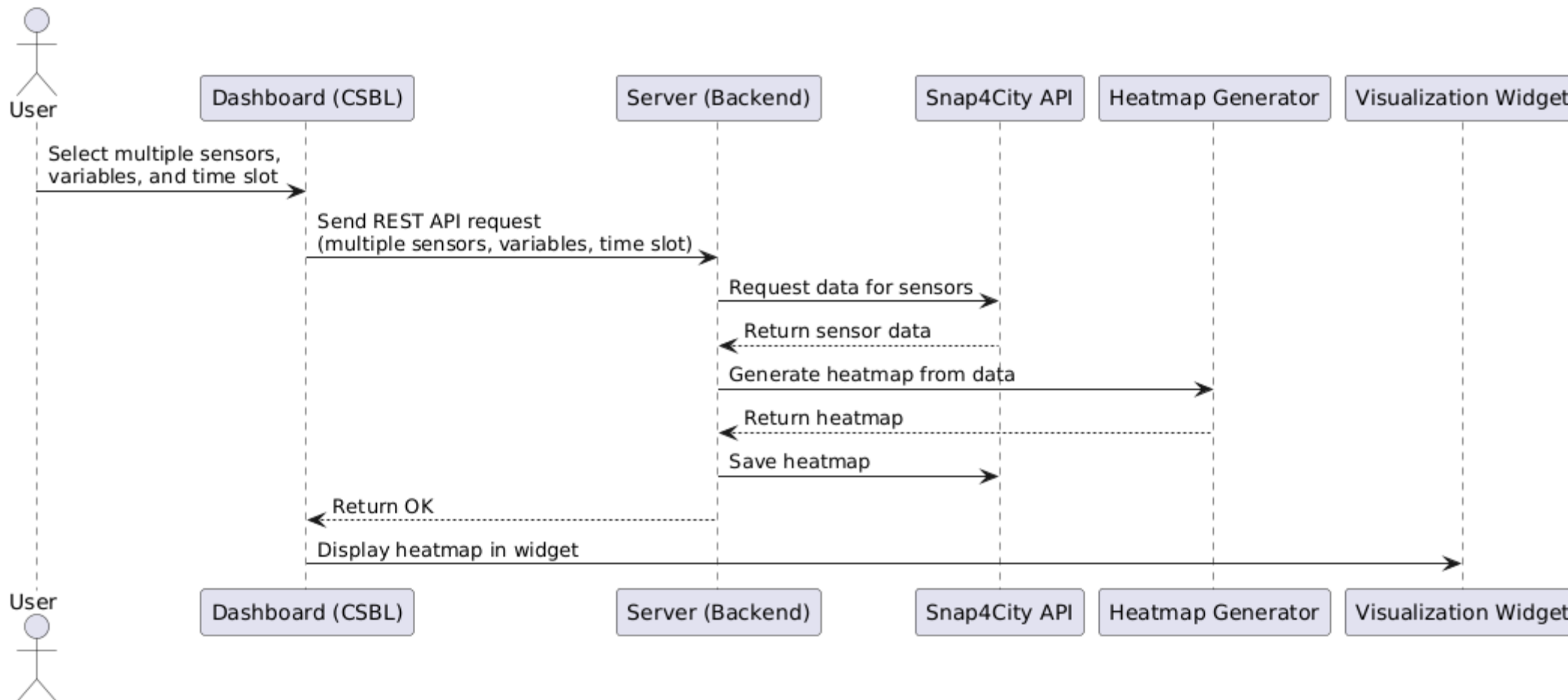
Abilita la gestione multi-thread:
ogni richiesta viene gestita in un thread
separato.

Alternativa: `processes=n` per gestione con
multiprocessi, ma meno comune per Flask in
sviluppo.

Prediction



Heatmap



Computing Predictions and Heatmaps

The screenshot displays the SNAP4CITY dashboard interface. At the top, the title "Computing Predictions And Heatmaps - Cloned Last" is visible along with the date "Wed 28 May 16:06:18". The main area features a map of Florence with numerous red circular markers representing traffic sensors. A "Selector - Map" panel is active, showing a "MAPS" dropdown and a "Scenario Editor" sidebar. The sidebar includes options for "Air quality Sensors", "OpenWeather", and "Traffic Flow". A "Form Invio" panel on the right contains buttons for "Compute Predictions", "Compute Heatmaps", and "Show Heatmaps", along with a "Heat Map Model" dropdown set to "heatMapModel_traffic" and a "Heatmap Update" button. At the bottom, two charts are displayed: "DISIT:OrionUNIFI:METRO1059 - VehicleFlow" (9m) and "Selected Trend And Predictions" (13m). The latter chart shows a line graph with two data series: "METRO468 - anomalyLevel" (blue circles) and "METRO468 - Predicted - anomalyLevel" (yellow diamonds).

<https://www.snap4city.org/dashboardSmartCity/view/Geo-Night.php?iddasboard=NDYwNA==>