WORKSHOP · 3 ORE

Open Data
Dal dato grezzo alla fisicalizzazione

Open Data → API → Visualizzazione → Data Physicalization

@mircopiccin
mirco.piccin@fablabcfv.org

Digital Innovation Hub Vicenza

Il percorso di oggi

01

Open Data

Cosa sono, chi li produce, dove trovarli

02

Interfacciarsi con le API

Postman e Node-RED per leggere i dati

03

Visualizzazione

Dashboard Node-RED e framework per data viz

04

Data Physicalization

Arduino + Firmata: LED e servomotori

⏱️ *~45 min per sezione*

01

Open Data

Cosa sono · Chi li produce · Dove trovarli · Come leggerli

Cosa sono gli Open Data?

"Dati liberamente accessibili, riutilizzabili e ridistribuibili da chiunque, senza restrizioni di copyright, brevetti o altri meccanismi di controllo."

Accessibili

Disponibili online in formati aperti (CSV, JSON, XML)

Riutilizzabili

Licenza aperta (es. CC BY, ODbL) che permette uso commerciale

Redistribuibili

Chiunque può condividerli, anche modificati

Il modello 5 stelle (Tim Berners-Lee)

★	Qualsiasi formato (PDF, immagine)
★★	Formato strutturato (Excel)
★★★	Formato aperto (CSV, JSON)
★★★★	Con URI per identificare le cose
★★★★★	Linked Data (RDF, SPARQL)

Chi li produce e dove trovarli

Pubblica Amministrazione

Comuni, Regioni, ISTAT, Ministeri

Istituzioni scientifiche

NASA, CERN, università, istituti di ricerca

Organizzazioni internazionali

ONU, Banca Mondiale, Eurostat, WHO

Comunità e ONG

OpenStreetMap, Wikipedia, Wikidata

Portali principali:

dati.gov.it · *data.europa.eu* · *opendata.comune.milano.it* · *data.worldbank.org* · *kaggle.com/datasets*

Dove trovare Open Data

Dati.gov.it

Portale nazionale italiano degli open data pubblici. Catalogo di dataset da PA centrale e locale: comuni, regioni, ministeri.

`https://www.dati.gov.it/`

🇮🇹 *Italia · CKAN · JSON / CSV / RDF*

European Data Portal

Catalogo europeo di dataset curati da istituzioni UE e stati membri. Sezione 'curated datasets' con i migliori dataset selezionati per qualità e completezza.

`https://data.europa.eu/en/curated-datasets-overview`

🇪🇺 *Europa · multilingua · JSON-LD / CSV / RDF / SPARQL*

ARPA - Agenzie Regionali

Ogni regione ha la sua ARPA con dati ambientali (aria, acqua, suolo, rumore). Qualità molto variabile per regione - ARPAE (ER) e ARPAV (Veneto) tra le migliori.

`https://it.wikipedia.org/wiki/Agenzia_regionale_per_la_protezione_ambientale#Agenzie_per_regione`

🇮🇹 *Regionale · XML / JSON / CSV · spesso no API key*

NASA - Climate Change

Dati climatici e ambientali globali da satelliti e stazioni a terra. Temperature, CO_2, ghiaccio artico, livello dei mari - serie storiche anche di decenni.

`https://science.nasa.gov/climate-change/`

🌍 *Globale · JSON / CSV / NetCDF · API NASA Open Data*

ISTAT - Banche dati demografiche

Dati demografici italiani: popolazione, famiglie, nascite, morti, stranieri per comune. Demo ISTAT è il portale specifico per i dati territoriali.

`https://demo.istat.it/`

🇮🇹 *Italia · CSV / JSON · storico fino agli anni '80*

JSON vs CSV vs XML

Guida ai formati di scambio dati più diffusi

JSON | CSV | XML

Cos'è ciascun formato?

JSON

Dal 2001

JavaScript Object Notation

Formato leggero basato su coppie chiave-valore, nativo per JavaScript. Ideale per API REST e configurazioni.

CSV

Dal anni '70

Comma-Separated Values

Tabelle di testo in cui i valori sono separati da virgole. Semplicissimo, universale per dati tabulari.

XML

Dal 1998

eXtensible Markup Language

Linguaggio a marcatori gerarchico, estensibile e autodescrittivo. Standard per documenti strutturati complessi.

Lo stesso dato nei 3 formati

Elenco di due utenti rappresentato in JSON, CSV e XML

JSON

```json
[
  { "id": 1, "nome": "Mario" },
  { "id": 2, "nome": "Lucia" }
]
```

CSV

```csv
id,nome
1,Mario
2,Lucia
```

XML

```xml
<utenti>
  <utente>
    <id>1</id>
    <nome>Mario</nome>
  </utente>
  <utente>
    <id>2</id>
    <nome>Lucia</nome>
  </utente>
</utenti>
```

Confronto delle caratteristiche

Caratteristica	JSON	CSV	XML
Leggibilità umana	★★★★	★★★★★	★★★
Struttura dati	Gerarchica	Piana	Gerarchica
Verbosità	Media	Bassa	Alta
Tipi di dato nativi	✔ Sì	✘ No	✘ No
Attributi multipli	✔ Sì	✘ No	✔ Sì
Schema/Validazione	JSON Schema	—	XSD / DTD
Uso principale	API / config	Export tabelle	Documenti / SOAP
Dimensione file	Compatta	Minima	Grande

Quando usare quale formato?

Usa JSON quando…

- Costruisci API REST o web app
- I dati sono annidati o gerarchici
- Lavori con JavaScript / Node.js
- Hai bisogno di tipi (numeri, booleani, null)
- Vuoi configurare applicazioni (package.json, tsconfig…)

Usa CSV quando…

- I dati sono semplici tabelle piatte
- Devi importare/esportare da Excel o fogli di calcolo
- Lavori con grandi volumi di dati tabulari
- Vuoi la massima compatibilità universale
- La struttura non cambierà nel tempo

Usa XML quando…

- Lavori con sistemi enterprise o legacy
- Hai bisogno di validazione con schema (XSD)
- Usi protocolli SOAP o EDI
- I documenti hanno metadati complessi
- Hai bisogno di namespace e spazi dei nomi

Riepilogo

JSON	CSV	XML
Flessibile & moderno	*Semplice & universale*	*Robusto & strutturato*

- JSON è il re delle API moderne: leggero, tipizzato, nidificato. Prima scelta per web e mobile.
- CSV eccelle con dati tabulari puri: dimensioni minime, compatibilità massima con Excel e BI.
- XML domina nei sistemi enterprise e documenti complessi: validazione potente, schema rigoroso.
- Non esiste un formato migliore in assoluto: la scelta dipende dal contesto, dal team e dall'ecosistema.

02

Interfacciarsi con le API

Postman · Node-RED · REST API · JSON

Leggere i dati: strumenti a confronto

POSTMAN

Esplorare e testare le API

- Interfaccia grafica per fare richieste HTTP
- Visualizzare risposta JSON in modo leggibile
- Salvare e condividere le richieste
- Testare parametri e autenticazione (API key)
- Ottimo per capire la struttura di una API

NODE-RED

Automazione a flussi (low-code)

- Nodi HTTP request per chiamare API
- Parsing e trasformazione JSON visuale
- Trigger su timer, eventi, MQTT
- Collegabile a dashboard, database, hardware
- Pensato per prototipazione rapida IoT

03
Rendere visibili i dati

Dashboard Node-RED · Framework di Data Visualization

Strumenti di visualizzazione

NODE-RED DASHBOARD

Rapido, integrato nel flusso

- Nodi pronti: gauge, chart, text, slider
- Aggiornamento real-time via WebSocket
- Nessuna conoscenza web necessaria
- Ottimo per prototipi veloci
- Limitato per layout complessi

FRAMEWORK ESTERNI

Più potenti e personalizzabili

Flourish

Drag & drop, no code, ottimo per storytelling

Observable

Notebook interattivi, D3.js integrato

Grafana

Dashboard professionali, open source

D3.js

Libreria JS potente, full control

HANDS-ON

WEATHER API · NODE-RED

Open-Meteo

Dati meteo in tempo reale · Serie orarie · Mappa geografica

① Dashboard

② Serie oraria

③ Mappa

api.open-meteo.com — gratuito, no API key

① STEP 1

Visualizzare i dati correnti con Dashboard

ENDPOINT

```
GET
https://api.open-meteo.com/v1/forecast?latitude=45.66&longitude=11.94&current=temperature_2m,relative_humidity_2m,wind_speed_10m,weather_code,apparent_temperature
```

VARIABILI CURRENT

Temperatura aria 2m

`temperature_2m`

22.4°C

Vento a 10m

`wind_speed_10m`

18 km/h

Umidità relativa

`relative_humidity_2m`

58%

Parzialmente nuvoloso

`weather_code`

WMO 2

NODI NODE-RED

② STEP 2

Serie temporale oraria (hourly)

STRUTTURA RISPOSTA API

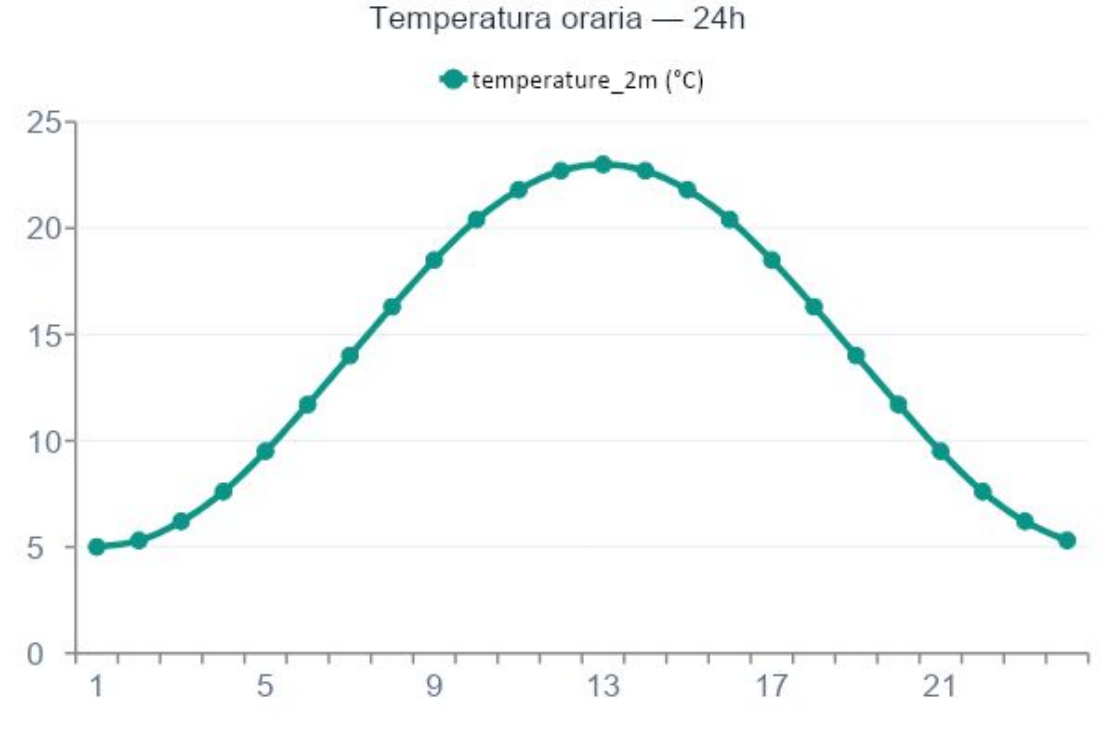
```
{
  "hourly_units": { ... },
  "hourly": {
    "time": [...],
    "temperature_2m": [...]
  }
}
```

hourly_units → asse Y

Unità di misura per ogni variabile
es. temperature_2m: "°C"

hourly.time → asse X

Array ISO 8601 con un valore per ora
es. "2025-05-05T00:00"



In Node-RED: nodo function → msg.payload = { x: msg.payload.hourly.time, y: msg.payload.hourly.temperature_2m }

③ STEP 3

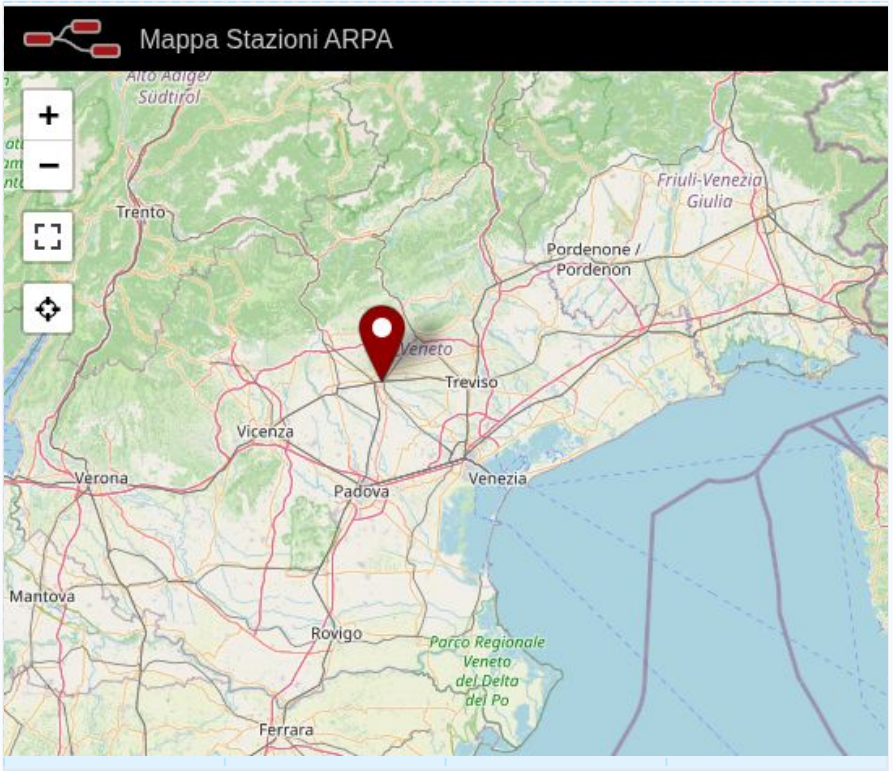
Visualizzare le coordinate su mappa (worldmap)

COORDINATE

`latitude`	**45.66**	Castelfranco Veneto — 2 decimali sufficienti
`longitude`	**11.94**	Precisione ~1 km
`timezone`	**Europe/Rome**	Per orari locali corretti

PAYLOAD PER WORLDMAP NODE

```
msg.payload = {
  name: "Castelfranco Veneto",
  lat: 45.66,
  lon: 11.94,
  label: temp + "°C",
  popup: desc + "<br>" + wind
};
```



→ `http://localhost:1880/worldmap`

Riepilogo

① Dashboard

Inject → HTTP Request → Function → ui_gauge / ui_text
Dati current: temperatura, umidità, vento, codice WMO

② Serie oraria

hourly_units → etichette asse Y · hourly.time → asse X
Chart LINE con 24-168 punti orari

③ Mappa worldmap

lat/lon: 2 decimali bastano · nodo worldmap riceve {lat, lon, label, popup}
Apertura: localhost:1880/worldmap

open-meteo.com — gratuito, open-source, no API key

04
Data Physicalization

Arduino · Firmata · Servomotori · LED

Rendere i dati fisici

Cos'è la data physicalization?

Trasformare dati astratti in oggetti fisici che si possono vedere, toccare e percepire. Un barometro fisico fatto di LED. Un servo che indica la temperatura. Dati che diventano esperienza.

1. Caricare Firmata su Arduino (StandardFirmata.ino)
2. Installare node-red-node-arduino in Node-RED
3. Collegare Arduino via USB → nodo Arduino
4. Mappare un valore dati su un pin PWM (servo/LED)
5. Testare: temperatura → angolo servomotore

La catena:

HANDS-ON #2

API · NODE-RED · ARDUINO

ARPAV
ARPAE
OpenAQ

Livello idrometrico fiumi — ARPAV

FONTE

ARPAV — Agenzia Regionale per la Prevenzione e Protezione Ambientale del Veneto

FORMATO

XML — file scaricabile, nessuna autenticazione richiesta

NOTE

Nessuna API key. Rete di oltre 200 stazioni in Veneto. Aggiornamento ogni 10 minuti, dati non ancora validati. Ultime 48h disponibili. Attenzione: il livello idrometrico è relativo allo zero della stazione, non alla profondità assoluta.

Livello idrometrico fiumi — ARPAV

Sito di riferimento

https://www.arpa.veneto.it/dati-ambientali/open-data
https://www.arpa.veneto.it/dati-ambientali/dati-in-diretta/meteo-idro-nivo/livelloidrometrico

Entry point / API URL

```
https://www.arpa.veneto.it/api/risorse/data-meteo/xml/Ultime48ore.xml    ← livello idrometrico ultime 48h
https://www.arpa.veneto.it/risorse/data-bollettini/meteo/bollettini/it/xml/bollettino_utenti.xml   ← dati
meteo ultime 72h
```

Esempio risposta XML — idro (semplificato)

```
<STAZIONE>
    <IDSTAZ>6</IDSTAZ>
    <NOME>"Cordevole a Saviner"</NOME>
    <X>11.98688899</X>
    <Y>46.44264642</Y>
    <QUOTA>1014</QUOTA>
    <DATI ISTANTE="202605160010">
        <VM>0.10</VM>
    </DATI>
    <DATI ISTANTE="202605160020">
        <VM>0.10</VM>
    </DATI>
```

Livello idrometrico fiumi — ARPAV

Schema di processamento

`inject`	Trigger con repeat ogni 600 secondi — allineato all'aggiornamento ARPAV
`http request`	ret: text (NON obj) — il file è XML, non JSON. Node-RED non lo parsa automaticamente
`xml`	Nodo built-in Node-RED: converte l'XML in oggetto JavaScript navigabile
`function`	Scorre Stazioni → [{ora, livello, serie: nome_stazione}]
`ui-chart`	x: ora (time), y: livello, series: nome stazione — mostra più fiumi sovrapposti

Qualità dell'aria — ARPAE Emilia-Romagna

FONTE

ARPAE — Agenzia Prevenzione Ambiente Energia Emilia-Romagna

FORMATO

JSON / CSV — REST API CKAN + Web Service NRT (Near Real Time)

NOTE

Nessuna API key richiesta. Dati Near Real Time aggiornati frequentemente. Regione limitrofa al Veneto — stazioni a Ferrara, Rovigo, Bologna. Il resource_id si trova sulla scheda del dataset nella colonna 'Esplora'.

Qualità dell'aria — ARPAE Emilia-Romagna

Sito di riferimento

https://dati.arpae.it/dataset/qualita-dell-aria-rete-di-monitoraggio

Entry point / API URL

https://dati.arpae.it/api/3/action/datastore_search?resource_id=<id>&limit=100

es: `https://dati.arpae.it/api/3/action/datastore_search?resource_id=8e207eff-aaec-4589-bd9d-b08e958a98c1&limit=5`

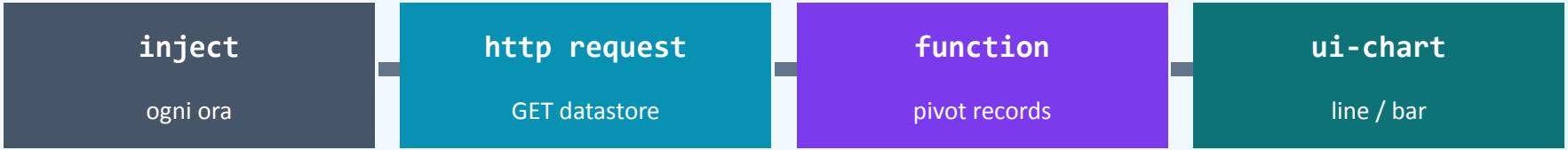
Bollettino NRT diretto (HTML/JSON): https://apps.arpae.it/qualita-aria/bollettino-qa/

Esempio risposta JSON — datastore_search


```
"result": {
   "include_total": true,
   "limit": 5,
   "records_format": "objects",
   "resource_id": "8e207eff-aaec-4589-bd9d-b08e958a98c1",
   "total_estimation_threshold": null,
   "records": [
     {
       "_id": 1,
       "station_id": "7000046",
       "variable_id": "131",
       "reftime": "2026-05-17T16:00:00+01:00",
       "value": "359.4",
       "v_flag": "0"
     },
```

Qualità dell'aria — ARPAE Emilia-Romagna

Schema di processamento



`inject`	Trigger con repeat ogni 3600 secondi
`http request`	ret: obj — passa resource_id e limit come query params nell'URL
`function`	result.records[] → [{data_ora, valore, stazione, serie: parametro}] per Dashboard 2
`ui-chart`	x: data_ora (time), y: valore, series: stazione — una linea per stazione

Qualità dell'aria — OpenAQ v3

FONTE

OpenAQ — aggregatore globale open data qualità aria (da reti ARPA, EPA, ARPAV e altre agenzie mondiali)

FORMATO

JSON — REST API con autenticazione tramite API key gratuita

NOTE

Registrazione gratuita su openaq.org per ottenere la key. Copre stazioni italiane incluso Veneto. Aggiornamento quasi real-time. Parametri disponibili: PM2.5, PM10, NO2, O3, SO2, CO.

Qualità dell'aria — OpenAQ v3

Sito di riferimento

```
https://openaq.org
https://docs.openaq.org
```

Entry point / API URL

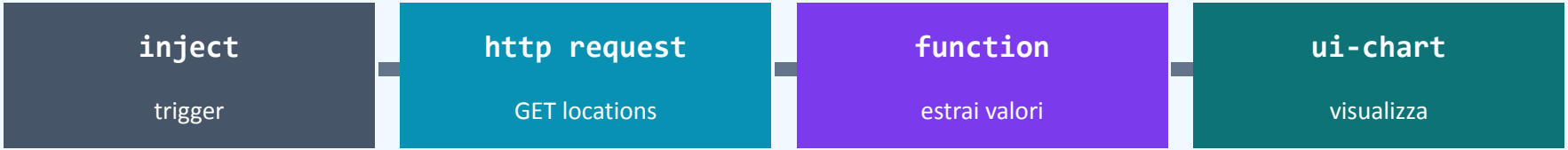
```
https://api.openaq.org/v3/locations?country=IT&bbox=10.6,44.8,12.8,46.7&limit=20
Header richiesto:  X-API-Key: <la_tua_key>
```

Esempio risposta JSON — /v3/locations


```
{
  "results": [
    {
      "id": 225482,
      "name": "IT1539A",
      "locality": "Venezia",
      "country": { "code": "IT" },
      "coordinates": { "latitude": 45.437, "longitude": 12.332 },
      "sensors": [
        { "parameter": { "name": "pm25", "units": "µg/m³" } },
        { "parameter": { "name": "no2",  "units": "µg/m³" } }
      ],
      "datetimeLast": { "local": "2024-05-04T14:00:00+02:00" }
    }
  ]
}
```

Qualità dell'aria — OpenAQ v3

Schema di processamento



`inject`	Avvia il flusso — manuale o ogni ora con repeat
`http request`	Aggiunge header X-API-Key, imposta ret: obj per parsare il JSON automaticamente
`function`	Loop su results[].sensors → array [{stazione, parametro, valore}] per Dashboard 2
`ui-chart`	Bar chart categorico — x: stazione, y: valore, series: parametro

Workshop completato!

01 **Open Data**

Cosa sono, chi li produce, dove trovarli

02 **API con Postman & Node-RED**

Chiamate HTTP, parsing JSON, automazione a flussi

03 **Visualizzazione**

Dashboard Node-RED e framework esterni

04 **Data Physicalization**

Arduino + Firmata per LED e servomotori

Buona sperimentazione! 🚀