

Indice

[Analisi Dataset - Open Data Hub](#)

[Accessibilità](#)

[Domini di Appartenenza](#)

[tourism](#)

[mobility](#)

[Weather Department by South Tyrol and Trentino - Open Data Hub API](#)

[Vantaggi](#)

[Copertura Territoriale](#)

[Frequenza di Aggiornamento dei dati](#)

[Dati disponibili](#)

[Possibili endpoint dell'API](#)

[Esempio di utilizzo](#)

[Weather Forecast by District - Open Data Hub API](#)

[Vantaggi](#)

[Copertura Territoriale](#)

[Frequenza di Aggiornamento dei Dati](#)

[Dati Disponibili](#)

[Possibili endpoint dell'API](#)

[Esempio di utilizzo](#)

[Weather Forecast by Municipality - Open Data Hub API](#)

[Vantaggi](#)

[Copertura Territoriale](#)

[Frequenza di Aggiornamento dei Dati](#)

[Dati Disponibili](#)

[Possibili endpoint dell'API](#)

[Esempio di utilizzo](#)

[Real-Time Weather - Open Data Hub API](#)

[Vantaggi](#)

[Copertura Territoriale](#)

[Frequenza di Aggiornamento dei Dati](#)

[Dati Disponibili](#)

[Possibili endpoint dell'API](#)

[Esempio di utilizzo](#)

Analisi Dataset - Open Data Hub

Di seguito analizzeremo diversi dataset forniti da **Open Data Hub**, rilevanti per l'integrazione nel sistema MinervaS dei dati metereologici, che coprono il territorio del Trentino-Alto Adige:

- **Weather Department by South Tyrol and Trentino**
- **Weather Forecast by District**
- **Weather Forecast by Municipality**
- **Real-Time Weather Data**

Accessibilità

Ogni Dataset è accessibile tramite **interfaccia grafica** o tramite la **documentazione interattiva dell'API REST su Swagger** che permette di testare endpoint e vedere la risposta in tempo reale (JSON). I link per ogni Dataset sono i seguenti:

Weather Department by South Tyrol and Trentino

- [Data Browser - Weather Department](#)
- [Swagger - Weather Department](#)

Weather Forecast by District

- [Data Browser - Weather Forecast by District](#)
- [Swagger - Weather Forecast by District](#)

Weather Forecast by Municipality

- [Data Browser - Weather Forecast by Municipality](#)
- [Swagger - Weather Forecast by Municipality](#)

Real-Time Weather Data

- [Data Browser - Real-Time Weather](#)
- [Swagger - Real-Time Weather](#)

Domini di Appartenenza

L'Open Data Hub suddivide i suoi dataset in base al **dominio di appartenenza**. I due di cui fanno parte le API citate sopra sono i seguenti:

tourism

Fornisce **previsioni meteorologiche** aggregate e localizzate, aggiornate quotidianamente e utili per il turismo. Sono API con struttura fissa: i dati vengono **tutti restituiti in blocco**, senza la possibilità di filtrare su tipo di dato, intervallo temporale, o struttura di risposta. Endpoint fissi e non modulari:

Unset

/v1/Weather/District

/v1/Weather/Forecast

/v1/Weather/Forecast/{id}

Parametri non supportati:

- intervalli temporali personalizzati
- filtri avanzati su dati meteorologici specifici
- rappresentazioni modulari tipo **flat, node, series...**

Parametri supportati:

- **language** → lingua della risposta (**en, de, it**)

Dataset associati:

- Weather Forecast by District
- Weather Forecast by Municipality

mobility

Raccoglie **dati osservati in tempo reale** provenienti da sensori e stazioni meteorologiche. Sono API **modulari e flessibili**: permettono l'utilizzo di parametri opzionali per filtrare i risultati. Struttura dell'endpoint:

Unset

/v2/{representation}/{stationTypes}/{dataTypes}/{from}/{to}

- **{representation}**

Definisce la **struttura e il formato del dato restituito** in modo da personalizzare la struttura del risultato a seconda del sistema che lo utilizzerà. Le principali rappresentazioni disponibili sono:

- **flat** → Restituisce i dati in formato **tabellare**: una riga per ciascuna misura.
- **node** → Include **metadati strutturati** in formato JSON annidato.
- **series** → Fornisce **serie temporali** aggregate per sensore.
- **tree** → Rappresenta i dati in **struttura gerarchica navigabile** (stazione → sensori → misure).

N.B. Le rappresentazioni indicate sono combinabili tra loro per ottenere rappresentazioni ibride (es. **flat,node** → Combinano dati e metadati)

- **{stationTypes}** & **{dataTypes}**

Serve per ottenere informazioni **filtrate** in base al **tipo di stazione** e al **tipo di misurazione** (es. temperatura, umidità...).

N.B. E' possibile inserire più tipi di stazioni o tipi di dati come elenchi separati da virgole, per filtrare in base a più parametri.

- **/from/{from}/to/{to}**

Serve per **recuperare dati storici** registrati da una o più stazioni, in un intervallo temporale specifico.

- **/latest**

Serve per **ottenere solo l'ultima misurazione disponibile**.

- **/metadata**

Serve per recuperare i **metadati**: le **informazioni sulle stazioni e sui dati registrati**, ma non le misure stesse.

- **/edgeTypes**

Restituisce **le relazioni tra gli oggetti** (stazioni, sensori, attributi...) nel sistema.

Dataset associati:

- Weather Department by South Tyrol and Trentino
- Real-Time Weather

Weather Department by South Tyrol and Trentino

- Open Data Hub API

Il dataset **Weather Department by South Tyrol and Trentino** dell'Open Data Hub fornisce **dati meteorologici osservati (non previsionali)**, raccolti da stazioni meteorologiche ufficiali gestite dal **Dipartimento Idrografico della Provincia Autonoma di Bolzano (Alto Adige)**, dalla **Provincia Autonoma di Trento** e da **EURAC Research**.

I dati restituiti comprendono informazioni dettagliate sulle stazioni metereologiche e sulle condizioni meteo come precipitazioni, temperatura dell'aria, umidità, vento e altri parametri climatici, fornendo coordinate geografiche e riferimenti temporali precisi.

Vantaggi

1. Controllo totale sui dati meteorologici osservati (non previsionali)

- Fornisce misure reali rilevate da stazioni meteo ufficiali.
- Include variabili come: temperatura, umidità, vento, pressione, radiazione, neve...

2. Architettura flessibile e modulare

Gli endpoint sono componibili e filtrabili, il che consente di:

- Recuperare solo le ultime misure (**/latest**)
- Recuperare misurazioni in un intervallo temporale (**/ {from} / {to}**)
- Recuperare metadati senza le misurazioni associate (**/metadata**)
- Visualizzare relazioni tra oggetti (**/edgeTypes**)

3. Esplorabilità e filtraggio avanzato

E' possibile filtrare per:

- tipo di stazione (**stationTypes**)
- tipo di misura (**dataTypes**)
- intervallo di tempo (**/ {from} / {to}**, **/latest**)

Copertura Territoriale

Il dataset copre **interamente il territorio delle due Province Autonome del Trentino-Alto Adige: Trento e Bolzano**.

Frequenza di Aggiornamento dei dati

I dati sono aggiornati con frequenza variabile a seconda della stazione, ma generalmente **almeno una volta al giorno** (tipicamente intorno alle 09:00).

Dati disponibili

Dati meteo osservati:

- Temperatura media, minima e massima (°C)
- Umidità relativa (%)
- Precipitazioni (mm)
- Velocità (m/s) e direzione del vento (°)
- Pressione atmosferica (hPa)
- Radiazione solare (W/m²)
- Altezza della neve (cm)
- Timestamp della rilevazione

Metadati delle stazioni:

- ID e nome della stazione
- Coordinate geografiche (latitudine, longitudine)
- Altitudine
- Tipologia della stazione (automatica o manuale)
- Disponibilità di dati storici e metadati associati

Possibili endpoint dell'API

Gli endpoint dell'API seguono una **struttura modulare basata su combinazioni di parametri dinamici**. Il formato generale è:

Unset

`/v2/{representation}/{stationTypes}/{dataTypes}/{from}/{to}`

Esempio di utilizzo

Python

```
import requests

import pandas as pd

# Parametri API

base_url = "https://mobility.api.opendatahub.com/v2"

representation = "flat,node"

station_type = "MeteoStation"

data_type = "air-temperature-min"

start_date = "2024-01-01"

end_date = "2024-01-05"

# Componi URL

url =
f"{base_url}/{representation}/{station_type}/{data_type}/{start_d
ate}/{end_date}"

# Esegui richiesta

response = requests.get(url, headers={"accept":
"application/json"})

# Parsing JSON

if response.status_code == 200:

    data = response.json()
```

```
    print("✅ Dati ricevuti correttamente!")
else:
    print(f"❌ Errore: {response.status_code}")
    exit()

# Estrai solo la lista di osservazioni meteo
records = data["data"]

# Normalizza in un DataFrame
df = pd.json_normalize(records)

# Estrai e rinomina solo le colonne che ti servono
df_clean = df[[
    "_timestamp", "mvalue", "sname",
    "scoordinate.x", "scoordinate.y"
]].rename(columns={
    "_timestamp": "timestamp",
    "mvalue": "min_temperature",
    "sname": "station_name",
    "scoordinate.x": "longitude",
    "scoordinate.y": "latitude"
})
```



```
# Mostra i primi 5 risultati
```

```
print(df_clean.head())
```

```
### OUTPUT: -----
```

```
✅ Dati ricevuti correttamente!
```

	timestamp	min_temperature	station_name
longitude latitude			
0	2024-01-01 00:00:00.000+0000 10.4979 46.8064	-9.3	Curon Belpiano
1	2024-01-01 00:00:00.000+0000 10.4878 46.7767	-12.5	Curon Cima Undici
2	2024-01-01 00:00:00.000+0000 10.5289 46.7759	-8.7	S. Valentino alla Muta
3	2024-01-01 00:00:00.000+0000 10.5213 46.7057	-4.4	Monte Maria
4	2024-01-01 00:00:00.000+0000 10.4615 46.6373	-6.8	Tubre

Weather Forecast by District - Open Data Hub API

Il dataset **Weather Forecast by District** dell'Open Data Hub fornisce **previsioni meteorologiche giornaliere aggregate** per ciascuno dei **7 distretti dell'Alto Adige** aggiornate regolarmente. Le previsioni sono elaborate e fornite dal **Servizio Meteo della Provincia Autonoma di Bolzano**.

Si tratta di dati previsionali (non osservati), appartenenti al **dominio “tourism”**, pensati per essere facilmente leggibili e utili in applicazioni territoriali come sistemi di supporto alla guida, turismo, mobilità e logistica.

Vantaggi

- **Previsioni meteo ufficiali, localizzate e leggibili**

Le previsioni sono fornite dal **Servizio Meteo della Provincia di Bolzano**, quindi **affidabili e ufficiali**.

- **Copertura completa e ben segmentata**

Vengono coperti **tutti i 7 distretti dell'Alto Adige**, ognuno con un **ID univoco** e un **nome localizzato**.

Copertura Territoriale

Sono coperti i **7 principali distretti dell'Alto Adige**:

1. Alta Val Venosta
2. Bassa Atesina
3. Oltradige-Bassa Atesina
4. Val Pusteria
5. Valle Isarco
6. Burgraviato
7. Salto-Sciliar

Frequenza di Aggiornamento dei Dati

Le previsioni sono aggiornate **quotidianamente** e riguardano l'arco delle 24 ore successive.

Dati Disponibili

Il dataset fornisce previsioni strutturate secondo i seguenti parametri:

- Temperatura media, minima, massima
- Copertura nuvolosa
- Precipitazioni (quantità attesa)
- Vento (velocità e direzione)
- Data di previsione e orari
- ID del distretto e nome localizzato

Possibili endpoint dell'API

La struttura dell'endpoint è la seguente:

Unset

/v1/Weather/District

Supporta il parametro: `locfilter` → per filtrare un singolo distretto (parametro esclusivo per **/v1/Weather/District**).

Risposta: Restituisce un array contenente le **previsioni per tutti e 7 i distretti** dell'Alto Adige. Per ciascun distretto restituisce:

- ID e nome localizzato
- Data di aggiornamento
- Previsioni giornaliere suddivise in **4 fasce orarie**:
 - **Part1** → Mattino presto
 - **Part2** → Tarda mattinata
 - **Part3** → Pomeriggio
 - **Part4** → Sera

Ogni parte è un numero intero da 0 a 3 e rappresenta **un indice sintetico delle condizioni meteo** in quella fascia oraria.

- **0** → Tempo buono / stabile

- 1 → Variabile
- 2 → Tempo instabile / peggioramento
- 3 → Condizioni avverse

Esempio di utilizzo

Python

```
import requests

# Endpoint API

url = "https://tourism.api.opendatahub.com/v1/Weather/District"

params = {
    "language": "en",
}

response = requests.get(url, params=params)

if response.status_code == 200:
    data = response.json()

    for district in data:
        print("=" * 50)
        print(f"DISTRETTO: {district.get('DistrictName')}")
        print(f>Data aggiornamento: {district.get('date')}")
```

```

print("ID Distretto:", district.get('Id'))

print("Lingua:", district.get('Language'))


print("\n🇮🇹 PREVISIONI:")

for forecast in district.get('BezirksForecast', []):

    print("-" * 30)

    print(f"    Giorno: {forecast.get('date')}")

    print(f"    Descrizione meteo: {forecast.get('WeatherDesc')}")

    print(f"    Temperatura massima: {forecast.get('MaxTemp')} °C")

    print(f"    Temperatura minima: {forecast.get('MinTemp')} °C")

    print(f"    Pioggia prevista: da {forecast.get('RainFrom')} a {forecast.get('RainTo')} mm")

    print(f"    Icona meteo: {forecast.get('WeatherImgUrl')}")

    print(f"    Parti del giorno (1-4): {[forecast.get('Part1'), forecast.get('Part2'), forecast.get('Part3'), forecast.get('Part4')]}")

    print(f"    Fulmini previsti: {forecast.get('Thunderstorm')}")

    print(f"    Affidabilità: {forecast.get('Reliability')}")

print("\n🔗 Link API distretto:", district.get('Self'))

```

```
        print("📝 Licenza:", district.get('LicenseInfo',  
{})).get('License'))  
  
        print("=" * 50 + "\n")  
  
else:  
    print(f"Errore {response.status_code}: {response.text}")
```

OUTPUT: ---

=====

DISTRETTO: Bolzano, Überetsch and Unterland

Data aggiornamento: 2025-04-22T09:00:00

ID Distretto: 1

Lingua: en

 PREVISIONI:

Giorno: 2025-04-22T00:00:00

Descrizione meteo: Partly cloudy

Temperatura massima: 24 °C

Temperatura minima: 9 °C

Pioggia prevista: da 0 a 5 mm

Icona meteo:

https://api-weather.services.sia.g.it/api/v2/graphics/icons/imgsource/wetter/icon_2.png

Parti del giorno (1-4): [0, 0, 1, 1]

Fulmini previsti: 0

Affidabilità: None

Giorno: 2025-04-23T00:00:00

...

Giorno: 2025-04-27T00:00:00

Descrizione meteo: Cloudy

Temperatura massima: 22 °C

Temperatura minima: 11 °C

Pioggia prevista: da 0 a 5 mm

Icona meteo:

https://api-weather.services.sia.g.it/api/v2/graphics/icons/imgsource/wetter/icon_3.png


Parti del giorno (1-4): [0, 0, 1, 1]

Fulmini previsti: 0

Affidabilità: None

 Link API distretto:

<https://tourism.api.opendatahub.com/v1/Weather/1>

 Licenza: CC0

=====

Weather Forecast by Municipality - Open Data Hub API

Il dataset **Weather Forecast by Municipality** dell'Open Data Hub fornisce **previsioni meteorologiche dettagliate** per ciascuno dei **comuni (municipalità)** della provincia Autonoma di **Bolzano - Alto Adige**, con un elevato grado di granularità spaziale e temporale. Fa parte del dominio tourism dell'Open Data Hub ed è utile per scenari in cui è importante adattare la guida in modo preciso alle condizioni previste in specifici tratti urbani o extraurbani.

Vantaggi

- **Alta granularità**

Fornisce **previsioni meteorologiche specifiche per 110 municipalità** dell'Alto Adige, includendo previsioni **ogni 3 ore**.

- **Supporto GIS**

Le previsioni possono essere **associate a coordinate precise** e **visualizzate su mappa** per simulare percorsi.

Copertura Territoriale

- Tutti i **comuni** della provincia autonoma di Bolzano – Alto Adige
- Oltre **110 municipalità** coperte

Frequenza di Aggiornamento dei Dati

- Le **previsioni sono aggiornate quotidianamente**
- Coprono diverse **fasce orarie** per la giornata successiva

Dati Disponibili

- Temperatura media, minima e massima (°C)
- Copertura nuvolosa (%)
- Precipitazioni (quantità stimata in mm)
- Velocità (m/s) e Direzione (°) del vento
- Lingua localizzata del contenuto
- Nome della municipalità

- Id della municipalità
- Timestamp dell'ultimo aggiornamento delle previsioni
- Data e orario della previsione

Possibili endpoint dell'API

La seguente struttura recupera le previsioni meteorologiche per tutte le municipalità disponibili.

Unset

/v1/Weather/Forecast

La seguente struttura, invece recupera le previsioni meteorologiche per uno specifico comune, identificato dall'**id**.

Unset

/v1/Weather/Forecast/{id}

Supporta i parametri:

- **id** → parte dell'URL path per ottenere le previsioni di un singolo comune, identificato dall'**id**.
- **limit** → numero massimo di record restituiti.

Risposta: Restituisce un array di oggetti contenenti:

- **MunicipalityName** → Nome del comune.
- **ForeCastDaily** → Previsioni giornaliere.
- **Forecast3HoursInterval** → Previsioni ogni 3 ore.
- **_Meta** → Metadati.

Esempio di utilizzo

Python

```
import requests
```

```

from datetime import datetime

url = "https://tourism.api.opendatahub.com/v1/Weather/Forecast"

params = {
    "language": "en",
    "limit": 3
}

response = requests.get(url, params=params)

if response.status_code == 200:
    data = response.json()

    for forecast in data:
        name = forecast.get("MunicipalityName", {}).get("en",
"Unknown")

        print("=" * 70)

        print(f"🌐 Comune: {name}")

        print(f"📄 ID previsione: {forecast.get('Id')}")

        print(f"📅 Ultimo aggiornamento: {forecast.get('_Meta',
{}).get('LastUpdate')}")

        print()

        # Previsioni giornaliere

        print(f"📅 Previsioni giornaliere:")

        for day in forecast.get("ForeCastDaily", []):


```

```
...  
  
# Previsioni orarie (ogni 3 ore)  
  
...  
  
print("=" * 70 + "\n")  
  
else:  
  
    print(f"Errore {response.status_code}: {response.text}")
```


Output atteso (esempio)

Unset

 Comune: Welsberg-Taisten

 ID previsione: forecast_021052

 Ultimo aggiornamento: 2025-04-16T12:00:00+00:00

 Previsioni giornaliere:

- Giorno: 2025-04-17

 Min: 4°C | Max: 12°C

 Sole: 5 h

 Pioggia: 3 mm (Prob: 55%)

 Meteo: cloudy with moderate rain

 Icona: <https://...>

 Previsioni ogni 3 ore:

- 2025-04-17 01:00:

 Temp: 8.6°C

 Pioggia: 0.8 mm (Prob: 50%)

 Meteo: overcast with light rain

 Vento: 2 km/h da 85°

 Icona: <https://...>

...

Real-Time Weather - Open Data Hub API

Il dataset **Real-Time Weather** dell'Open Data Hub fornisce **dati meteorologici aggiornati in tempo reale** provenienti da stazioni distribuite in tutta la **provincia autonoma di Bolzano - Alto Adige**. I dati sono ideali per applicazioni che necessitano informazioni ambientali aggiornate costantemente, come i sistemi di guida assistita o i modelli predittivi a breve termine.

Vantaggi

- **Reattività in tempo reale** → I dati vengono aggiornati ogni 5–10 minuti a seconda della stazione.
- **Supporto a decisioni immediate** → Utile per rilevare **eventi meteo critici** (es. vento laterale, pressione in calo, visibilità ridotta).
- **Compatibile con sistemi predittivi** → Ottimo per alimentare modelli AI o sistemi IoT context-aware.

Copertura Territoriale

Stazioni meteorologiche attive in tempo reale nella **provincia autonoma di Bolzano - Alto Adige**.

Frequenza di Aggiornamento dei Dati

I dati vengono aggiornati con frequenza variabile, tipicamente ogni **5-10 minuti**, a seconda della stazione.

Dati Disponibili

- Velocità (m/s) e Direzione (°) del vento
- Altezza della neve (cm)
- Precipitazioni (quantità stimata in mm)
- Pressione dell'aria (hPa)
- Temperatura dell'acqua e dell'aria (°C)
- Durata del sole (hh:mm)
- Visibilità (m)
- Radiazione globale (W/m²)
- Portata del flusso (corsi d'acqua) (m³/s)
- Umidità relativa (%)
- Nome della stazione
- Id della categoria
- Coordinate geografiche (latitudine e longitudine)
- Altitudine

Possibili endpoint dell'API

Gli endpoint dell'API seguono una **struttura modulare basata su combinazioni di parametri dinamici**.

L'endpoint base per accedere ai dati realtime di tutte le stazioni è il seguente:

Unset

/v2/weather

Il formato generale è:

Unset

/v2/{representation}/{stationTypes}/{dataTypes}/{from}/{to}

Parametri supportati:

- **stationId** → ID della stazione specifica.
- **attributes** → lista di variabili da includere (es. umidità, pressione...)
- **limit** → numero massimo di risultati da restituire.

Esempio di utilizzo

Python

```
import requests

from datetime import datetime, timedelta, timezone
from collections import defaultdict

# ♦ Intervallo temporale fisso
end = datetime.now(timezone.utc).replace(microsecond=0)
start = end - timedelta(days=7)

start_iso = start.isoformat().replace("+00:00", "Z")
end_iso = end.isoformat().replace("+00:00", "Z")

# Richiesta dataset disponibili
base_url = "https://mobility.api.opendatahub.com/v2/flat,event"
response = requests.get(base_url)

categorie_eventi = defaultdict(list)
```

```

if response.status_code == 200:
    datasets = response.json()

    print(f"Trovati {len(datasets)} dataset disponibili:\n")

    for d in datasets:
        dataset_id = d.get("id", "Sconosciuto")
        events_url = d.get("self.events")

        print(f"🔍 Analizzo dataset: {dataset_id}")
        print(f"➡ URL: {events_url}")

    .....

# 3. Stampa eventi raggruppati per categoria
print("\nEventi raggruppati per categoria:\n")

for categoria, eventi in categorie_eventi.items():
    print(f"📁 Categoria: {categoria} ({len(eventi)} eventi)")
    print("-" * 60)


    ....


```

Output atteso (esempio)


Python


Trovati **2** dataset disponibili:

 Analisi dataset: A22


 URL: <https://mobility.api.opendatahub.com/v2/flat,event/A22>


 Misure trovate: **0**


 Analisi dataset: PROVINCE_BZ

 URL: https://mobility.api.opendatahub.com/v2/flat,event/PROVINCE_BZ

 Misure trovate: **5**

 Eventi raggruppati per categoria:

 Categoria: Controlli radar_Name of the secondary category on italian was not found | Radarkontrolle_Name of the secondary category on german was not found (**5** eventi)

 Evento **#1**

Descrizione IT: Prestate sempre la massima attenzione al rispetto dei limiti di velocità, **in** questo momento **in** particolare a Settequerce.

Strada: Bolzano - Merano

Inizio: **2024-03-08 00:00:00.000+0000**

Fine: **2024-03-08 00:00:00.000+0000**

Coordinate: lat **46.510092546** | lon **11.275777106**
