

# FROST

Martina Forconi – Beatrice Olivari



This project has received funding from the European Union's HE research and innovation programme under the grant agreement No. 101057497

# Free and Open Source SensorThings API Implementations

## Whiskers [\[ edit \]](#)

In March 2016 [SensorUp](#) and the GeoSensorWeb Lab at the University of Calgary submitted an open source software project proposal to the Eclipse Foundation and has been approved. The project is called [Whiskers](#).<sup>[15]</sup> Whiskers is an OGC SensorThings API framework. It will have a [JavaScript](#) client and a light-weight server for IoT gateway devices (e.g., Raspberry Pi or BeagleBone). Whiskers aim to foster a healthy and open IoT ecosystem, as opposed to one dominated by proprietary information silos. Whiskers aims to make SensorThings development easy for the large and growing world of IoT developers.

## GOST [\[ edit \]](#)

GOST<sup>[16]</sup> is an open source implementation of the SensorThings API in the [Go programming language](#) initiated by Geodan. It contains easily deployable server software and a JavaScript client. Currently (June 2016) it is in development but a first version can already be downloaded and deployed. The software can be installed on any device supporting Docker or Go (e.g. Windows, Linux, Mac OS and Raspberry Pi). By default sensor data is stored in a [PostgreSQL](#) database.

## FROST [\[ edit \]](#)

FROST-Server<sup>[17]</sup> is an Open Source server implementation of the OGC SensorThings API. FROST-Server implements the entire specification, including all extensions. It is written in Java and can run in Tomcat or Wildfly and is available as a Docker image. Among its many features is the ability to use String or UUID based entity IDs.

FROST-Client<sup>[18]</sup> is a Java client library for communicating with a SensorThings API compatible server.

## SensorThings HcDT Charting SDK [\[ edit \]](#)

SensorThings HcDT<sup>[19]</sup> is a JavaScript charting library for the OGC SensorThings API. It is based on the open source<sup>[clarification needed]</sup> [Highcharts](#) library and [DataTables](#). It is a front-end charting library enable developers to connect to datastreams from any OGC SensorThings API service, and display the sensor observations in charts, tables, or dashboard widgets for web applications.

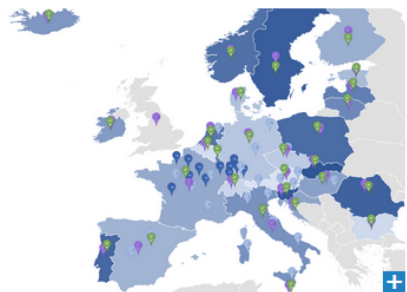
## Mozilla STA [\[ edit \]](#)

[Mozilla](#) developed a node implementation of the OGC SensorThings API.<sup>[20]</sup>

## 52°North STA [\[ edit \]](#)

52N SensorThingsAPI<sup>[21]</sup> is an open source implementation of the OGC SensorThings API. Its core features are the interoperability with the [52N SOS](#) implementing the [OGC Sensor Observation Service](#), customizable database mappings and several convenience extensions. It can be deployed as a Docker container, inside an [Apache Tomcat](#) or as a standalone application.

# FROST<sup>®</sup>-Server - An open source implementation of OGC SensorThings API




Locations of the data sources connected in API4INSPIRE

## Description of the project

Against the background of the requirements from numerous applications in the mega trend topic "Internet of Things", the Fraunhofer IOSB decided to develop a server for the Standard SensorThingsAPI. The objective was to achieve high performance with low resource consumption and an openness that facilitates usability both in the research environment and in commercial applications.

Performance with low resource consumption and an openness that facilitates usability in both the research environment and commercial applications. The desire for openness led to the decision to design the implementation as open source software right from the start, in order to make the way as free as possible for innovation. The result is the FROST<sup>®</sup>-Server. The name is an acronym and stands for "Fraunhofer Open Source SensorThings API Server". But the name is also intended to suggest that your data is kept "fresh and available"

Meanwhile, the "SensorThings API" is recommended by the European Commission as "Good Practice" for the deployment of measurement data according to the INSPIRE guidelines: "[OGC SensorThings API as an INSPIRE download service](#)" 

<https://www.iosb.fraunhofer.de/en/projects-and-products/frost-server.html>

# FROST Server Deployment

The FROST-Server comes in three packages, and thus also in three docker images:

- [fraunhoferiosb/frost-server](https://github.com/fraunhoferiosb/frost-server) The all-in-one package
- [fraunhoferiosb/frost-server-http](https://github.com/fraunhoferiosb/frost-server-http) The HTTP-only package
- [fraunhoferiosb/frost-server-mqtt](https://github.com/fraunhoferiosb/frost-server-mqtt) The MQTT-only package

The FROST-Server can be run in **Tomcat** or as a [docker image](#)  
You will also need a **PostgreSQL** server with **PostGIS** extensions

<https://fraunhoferiosb.github.io/FROST-Server/deployment/architecture-packages.html>



# PostgreSQL

PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to [install](#) and [use](#) PostgreSQL through the [official documentation](#). The [open source community](#) provides many helpful places to become familiar with PostgreSQL, discover how it works, and find career opportunities. Learn more on how to [engage with the community](#).

<https://www.postgresql.org/>



## PostgreSQL: The World's Most Advanced Open Source Relational Database

# PostGIS

PostGIS extends the capabilities of the [PostgreSQL](#) relational database by adding support storing, indexing and querying geographic data.

PostGIS features include:

- **Spatial Data Storage:** Store different types of spatial data such as points, lines, polygons, and multi-geometries, in both 2D and [3D](#) data.
- **Spatial Indexing:** Quickly [search](#) and retrieve spatial data based on its location.
- **Spatial Functions:** A wide range of spatial functions that allow you to filter and analyze spatial data, measuring [distances](#) and [areas](#), [intersecting](#) geometries, [buffering](#), and more.
- **Geometry Processing:** Tools for [processing](#) and [manipulating](#) geometry data, such as [simplification](#), [conversion](#), and generalization.
- **Raster Data Support:** Storage and processing of [raster data](#), such as elevation data and weather data.
- **Geocoding and Reverse Geocoding:** Functions for [geocoding](#) and reverse geocoding.
- **Integration:** Access and work with PostGIS using third party tools such as QGIS, GeoServer, ArcGIS, Tableau, and MapServer.

<http://postgis.net/>



## **TimescaleDB is an extension of Postgres for time-series.**

- You could continue to use ubiquitous SQL to perform your queries.
- The particular benefits of TimescaleDB include high compression ratios achieved through type-specific compression (> 10x compression) along with much more performant time-series queries than standard Postgres.

**Time-series data** represents how a system, process, or behavior changes over time. For example, if you are taking measurements from a temperature gauge every five minutes, you are collecting time-series data.

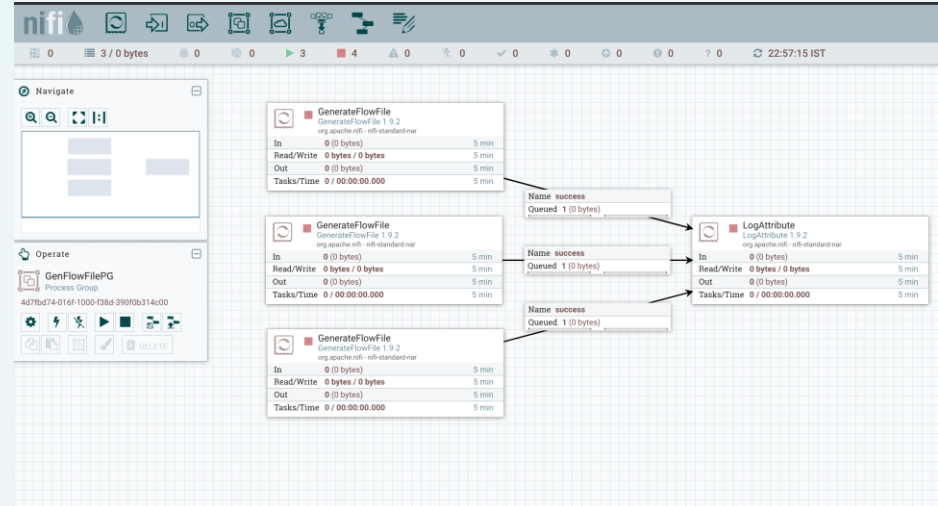
As these measurements change over time, each data point is recorded alongside its timestamp, allowing it to be measured, analyzed, and visualized.

Specialized time-series databases, like Timescale, are designed to handle large amounts of database writes, so they work much faster. They are also optimized to handle schema changes, and use more flexible indexing, so you don't need to spend time migrating your data whenever you make a change.

**Hypertables** are PostgreSQL tables that automatically partition your data by time. You interact with hypertables in the same way as regular PostgreSQL tables, but with extra features that makes managing your time-series data much easier.

# Other optional components: Apache nifi

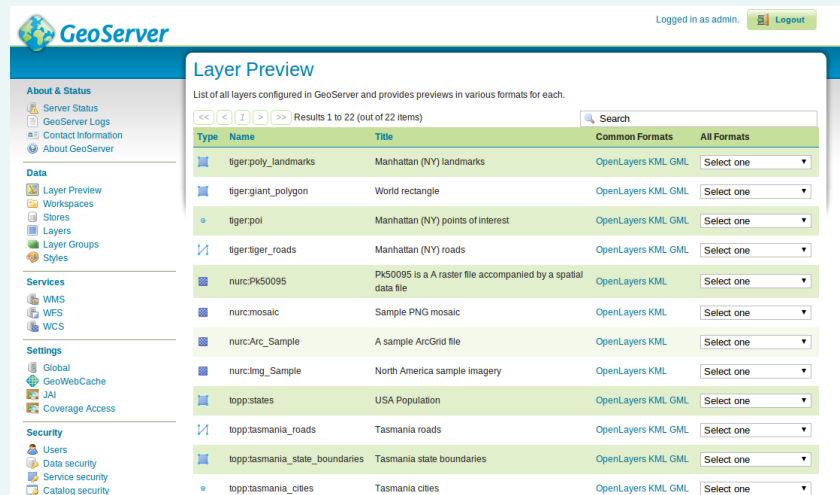
- Apache nifi is a software project from the **Apache Software Foundation**
- It was designed to **automate the flow of data between software systems.**
- It leverages the concept of extract, transform, load (**ETL**).
- The software design is based on the **flow-based programming model.**





# Other optional components: GeoServer

- GeoServer is an **open-source** server written in Java that allows users to **share, process and edit geospatial data**.
- Using **open standards** set forth by the Open Geospatial Consortium (**OGC**), GeoServer allows for flexibility in map creation and data sharing.
- Designed for **interoperability**, it publishes data from any major spatial data source using open standards.
- GeoServer **can also display data on any of the popular mapping applications** such as **Google Maps, Google Earth, Microsoft Bing Maps, and MapBox**. In addition, GeoServer can connect with traditional GIS architectures such as ESRI ArcGIS



GeoServer

Logged in as admin. Logout

### Layer Preview

List of all layers configured in GeoServer and provides previews in various formats for each.

<< < > >> Results 1 to 22 (out of 22 items) Search

Type	Name	Title	Common Formats	All Formats
	tiger.poly_landmarks	Manhattan (NY) landmarks	OpenLayers KML GML	Select one
	tigergiant_polygon	World rectangle	OpenLayers KML GML	Select one
	tigerpoi	Manhattan (NY) points of interest	OpenLayers KML GML	Select one
	tigertiger_roads	Manhattan (NY) roads	OpenLayers KML GML	Select one
	nurc:PK50095	PK50095 is a A raster file accompanied by a spatial data file	OpenLayers KML	Select one
	nurc:mosaic	Sample PNG mosaic	OpenLayers KML	Select one
	nurc:Arc_Sample	A sample ArcGrid file	OpenLayers KML	Select one
	nurc:img_Sample	North America sample imagery	OpenLayers KML	Select one
	topp:states	USA Population	OpenLayers KML GML	Select one
	topp:tassmania_roads	Tasmania roads	OpenLayers KML GML	Select one
	topp:tassmania_state_boundaries	Tasmania state boundaries	OpenLayers KML GML	Select one
	topp:tassmania_cities	Tasmania cities	OpenLayers KML GML	Select one



GeoServer

# USE CASE: Air Quality Index for AirBreak

**Goal:** Creating an Index of Air Quality for the city of Ferrara that shows hourly average values for 5 polluting substances.

<https://airbreakferrara.net/che-aria-tira/>

LabService Analytica (LAS) sensors measure **AQ parameters** and makes them available through **proprietary API**

nifi

Data from LAS are ingested into the FROST server of **Municipality Ferrara** to be **STA compliant**

nifi

We then use nifi again to build our **hourly Air Break Index**

GeoServer

The Index is then published as **WFS** and **WMS** thanks to GeoServer

Using ANY **WMS/WFS-COMPLIANT** client, we display that on a map

We publish the aggregated data as a layer to the Open Data portal of the municipality of Ferrara ([link](#))