

Open Shelter API

Open Data Management & The Cloud

De Nardin Carlo

Index

- Introduction
- Data Life Cycle
 - Planning
 - Collecting & Processing
 - Sharing
 - Preserving
 - Reusing
- FAIR principles
- Service implementation
- Cloud implementation
- Demo

Introduction

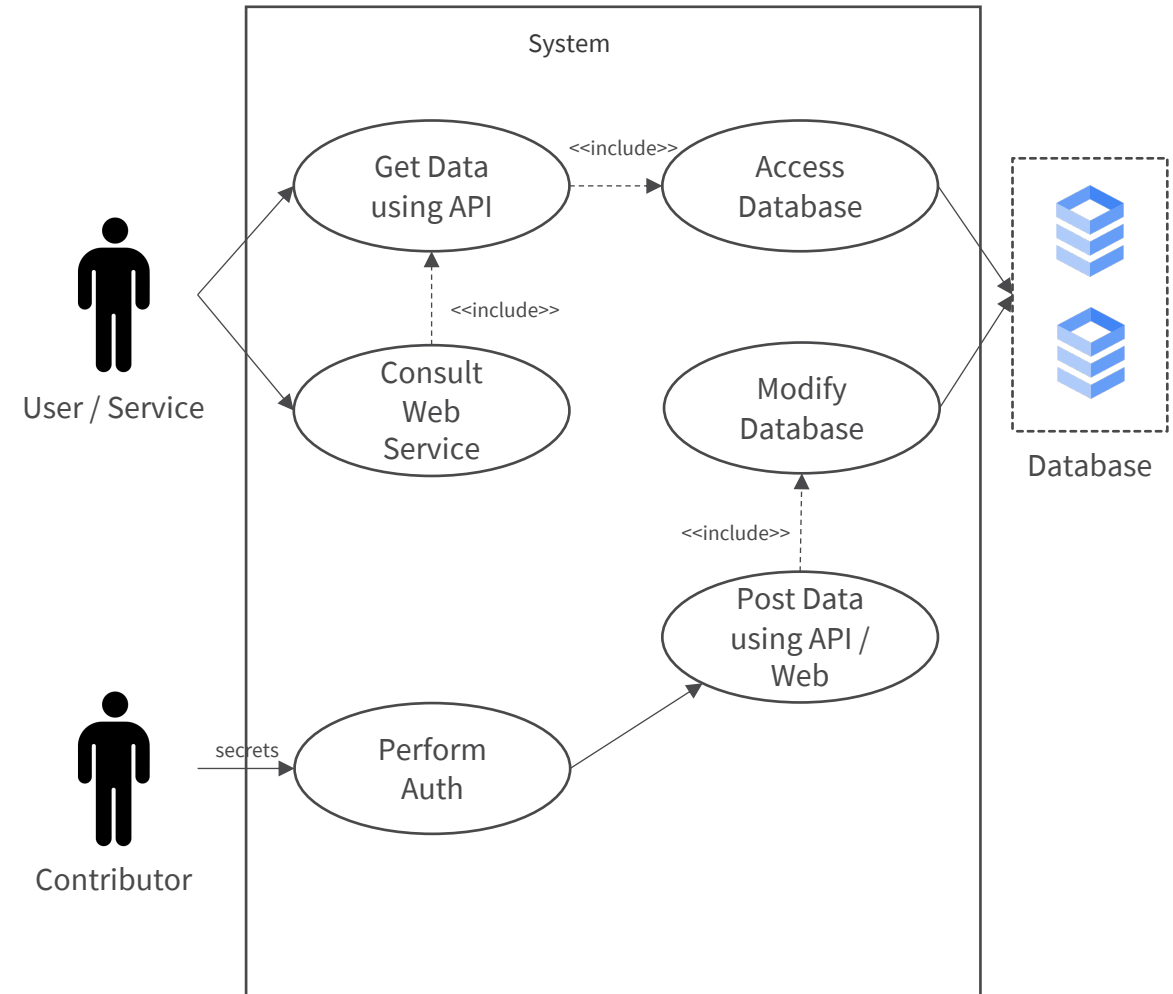
Open Shelter API is a service that enables users to access and retrieve information on **mountain shelters**. This is achieved through a **collaborative platform** that allows users to generate and share data using a web service or API calls.

Open Shelter API relies on **user participation** to create a comprehensive and constantly updated database of information on mountain shelters that is **open and accessible to everyone**. Open Shelter API is built on the principles of OpenStreetMap, an open data platform, to ensure that the information shared is reliable and up-to-date.

Introduction

The **UML Use Case** model on the right represents the functionalities, requirements, and interactions of the service from a user perspective.

The service provides two types of APIs: a **public API** that allows anyone to **access data**, and an **authenticated API** that is restricted to contributors for **inserting and modifying data**. This approach enables the service to provide public access to data while maintaining the security and integrity of the information through a separate API exclusively for contributors.



Data Life Cycle



Planning

1. **Refining objectives:** Enhance the accessibility of mountain shelter information to benefit both common users and external services like tourist offices
2. **Defining (meta)data to be published:** The service will publish information about mountain shelters including their name, geographical location (province, region, coordinates), website link (★★★★★), and available services (restaurant, Wi-Fi, electricity, ...)
3. **Data distribution license:** The Open Database License (ODbL) V1.0 will be used to ensure that the data is open for public use, while also protecting and accrediting the data source.
 - **Copyleft:** all modifications made to the original data must be released under the same license
 - **Attribution:** attribution of the original work to the authors of the geospatial data
 - **Data Sharing:** derived geospatial data must be released in an open and accessible manner
 - **Commercial use:** allows for commercial use of geospatial data
 - **Disclaimer of warranty:** does not provide any warranties regarding the information contained in the data

<https://opendatacommons.org/licenses/odbl/1-0/>

01

02

03

04

05

Collecting & Processing

1. **Data collection:** Authenticated users through the web service or with external services linked with the rest API can upload shelter data
2. **Data processing:** A user data may contain errors or low-quality information; it is important to verify and validate the data. To achieve this there are several solutions:
 - Making certain fields mandatory such (e.g., name, geographical information, ..) when a shelter is created
 - Checking the coherence of the data with other sources (e.g., verifying if the coordinates match the provided province/region, checking the shelter website)
 - Creating a versioning system that allows users to validate / modify information provided by other users

01

02

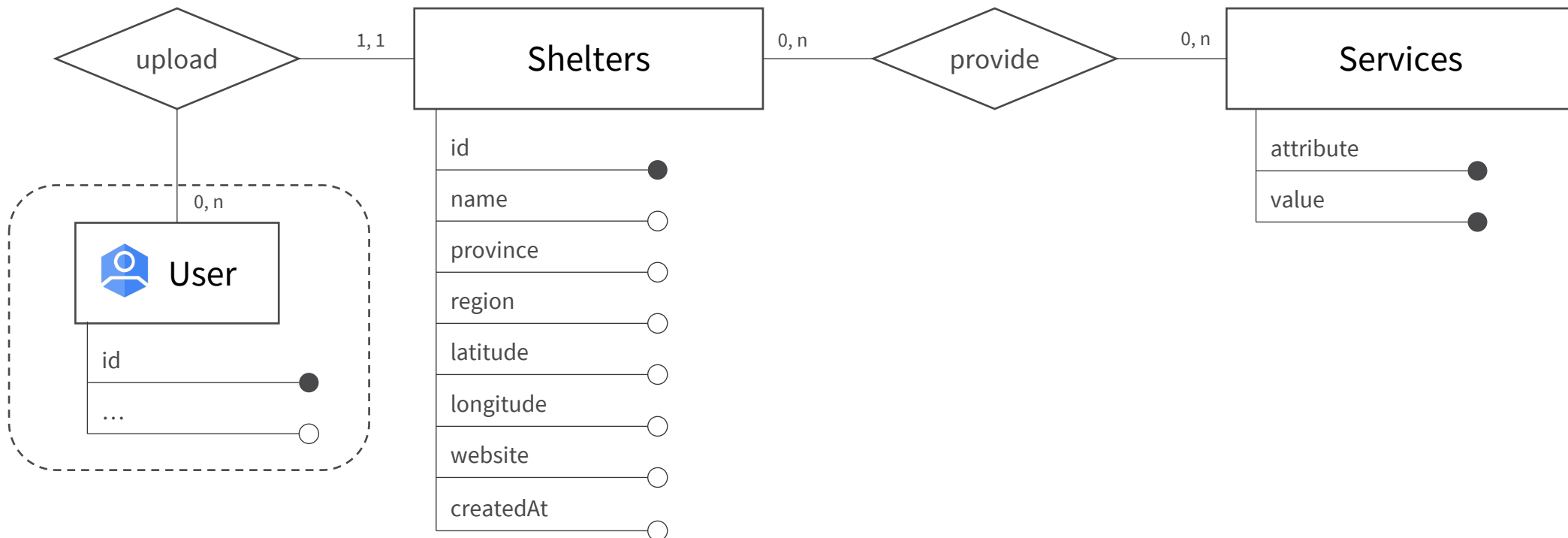
03

04

05

Collecting & Processing

3. **ER model:** An ER model has been chosen to represent the database design. The user entity is not present in the above schema as a cloud solution has been utilized for user management



01

02

03

04

05

Collecting & Processing

5. **Relational model:** After designing the ER model, it was mapped into a relational model
 - Shelters(id, name, province, region, latitude, longitude, website, createdAt, user)
 - Services(attribute, value)
 - Provide(id, attribute, value)
6. **EAV services:** Shelter services management is facilitated through an **EAV (Entity-Attribute-Value)** schema, which offers the necessary flexibility and extendibility for the services management

Shelters

<u>id</u>	name	...	createdAt
1	Carestiato	...	09/03/23
2	Vazzoler	...	03/03/23
...

Provide

<u>id</u>	<u>attribute</u>	<u>value</u>
1	Restaurant	Available
1	Wi-Fi	Not Available
2	Beds	10

Services

<u>attribute</u>	<u>value</u>
Restaurant	Available
Restaurant	Not Available
Wi-Fi	Available
Wi-Fi	Not Available
Beds	10
...	...

Collecting & Processing

7. Versioning schema: As mentioned earlier, a versioning database is crucial for saving modifications made to the data. In this project, **history tables** were used to save the modifications made to both shelters and services. These tables are a copy of the tables in the main database but use **creation date as a key** to ensure data univocity.

- SheltersHistory(id, name, province, region, latitude, longitude, website, createdAt, user)
- Services(attribute, value)
- ProvideHistory(id, createdAt, attribute, value)

SheltersHistory

<u>id</u>	name	...	<u>createdAt</u>
1	Carestiato	...	09/03/23
1	Carestiato	...	04/03/23
...

ProvideHistory

<u>id</u>	<u>createdAt</u>	<u>attribute</u>	<u>value</u>
1	09/03/23	Restaurant	Available
1	09/03/23	Wi-Fi	Available
1	04/03/23	Wi-Fi	Not Available
...

Services

<u>attribute</u>	<u>value</u>
Restaurant	Available
Restaurant	Not Available
Wi-Fi	Available
Wi-Fi	Not Available
...	...

Sharing

- 1. Provide data:** Shelters data is provided through a customized web service. This service allows users to **view shelter data on a map** (geographical data) or retrieve data by making a call to the available **rest API** to obtain information about a single shelter or a group of shelters.
- 2. Data sharing format:** Initially, the chosen format to represent the data was **GeoJson**, which ensures **syntactic interoperability** between different systems.
- 3. Metadata enrichment: Dublin Core** metadata is adopted to provide additional information that enriches the data

```
dc:title : 1 italian shelters
dc:description : List of italian shelters that are located in region [Friuli Venezia Giulia]
dc:creator : Open Shelter API
dc:identifier : https://rest-service-hnlijallya-oa.a.run.app/v1/shelters/clesjse7d000ms676liphxe9
dc:date : 2023-03-08T09:32:42.618Z
dc:format : .jsonld
dc:language : en
dc:source : https://www.openshelterapi.com
dc:rights : Open Data Commons Open Database License (ODbL) – https://opendatacommons.org/licenses/odbl/1.0/
```

Sharing

4. **Semantic interoperability:** GeoJson does not guarantee semantic interoperability because it lacks a metadata specification. To add meaning to the data, the **JSON-LD** format has been chosen while maintaining the natural structure of the GeoJson format.



Sharing

5. **Ontology:** To achieve semantic interoperability the widely adopted **schema.org** ontology has been used in combination of **custom vocabularies** to describe shelter services. This allows for a better understanding of the data's meaning and facilitates its integration with other data sources

01

02

03

04

05

▼ 2 {4}

@type : LocationFeatureSpecification

schema:name : Beds

schema:value : 10

schema:url : <https://storage.cloud.google.com/vocabularies/openshelterapi/vocabularies/Beds.jsonld>

► @context {1}

@type : schema:Service

schema:name : Beds

schema:description : Service that provides information about the number of total beds

▼ schema:numberOfBeds {3}

@type : schema:QuantitativeValue

schema:description : Number of total beds

schema:value : integer

Preserving

- 1. Long term archive:** In order to ensure the longevity and accessibility of the collected data, it is important to guarantee its preservation and future availability. For this purpose, in this project, it has been decided to store shelter data in a cloud storage solution (Google) called Archive. This storage solution provides long-term conservation of data at a low cost and with low performance requirements, making it suitable for archiving of large amounts of data.
- 2. Data backup:** Data security is a top priority during the preservation phase, and regular backups are essential to protect against data loss or corruption. To archive this, a regular backup strategy has been implemented using the data backup services provided by Google Cloud.

01

02

03

04

05

Reusing

- 1. Reuse data:** Collected data can be utilized for various purposes, such as creating applications, performing data analysis, generating reports, and more. The possibilities for reusing the collected data are vast and can be leveraged to generate valuable insights and drive innovation in the tourism industry.

01

02

03

04

05

FAIR principles

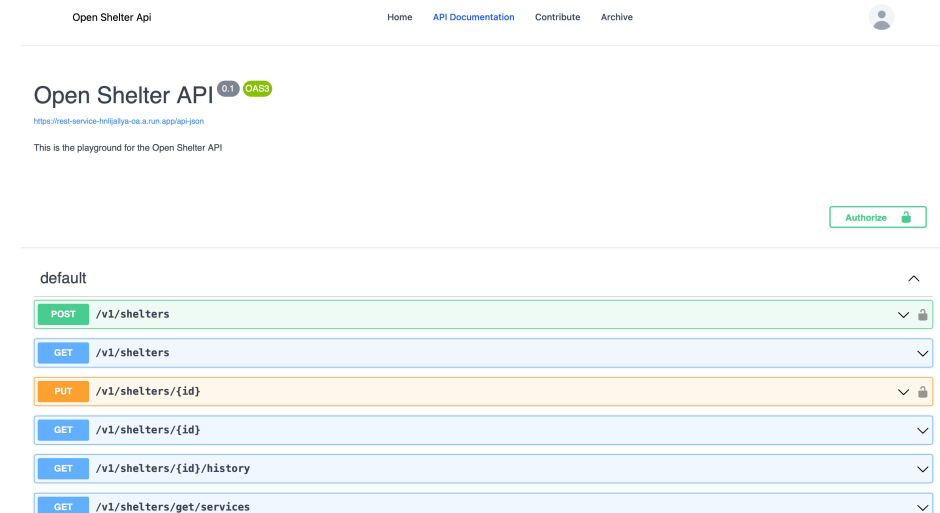
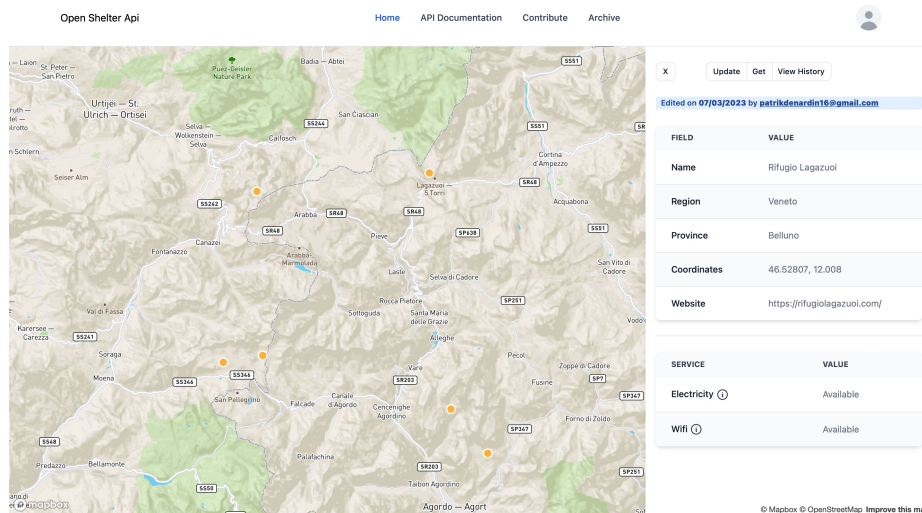
- 🔍 **Findable:** (meta)data are assigned a unique identifier (current data) and are publicly available and accessible, ensuring their findability
 - 👤 **Accessible:** (meta)data are retrievable using their identifier through an open communication protocol (HTTP), which also allows for authentication procedures
 - 🔗 **Interoperable:** (meta)data use a formal, accessible, and shared language (json-ld).
Furthermore, vocabularies that comply with the FAIR principles are employed
 - ♻️ **Reusable:** The data are made made available under an open license without usage restrictions, allowing for integration and sharing with other services
- ⚠️ **Respecting all the FAIR principles may require additional resources and expertise, but the long-term benefits of using FAIR data often outweigh the initial costs and challenges.**

Service implementation

Open Shelter API is based on a microservices architecture, which is composed of 2 microservices:

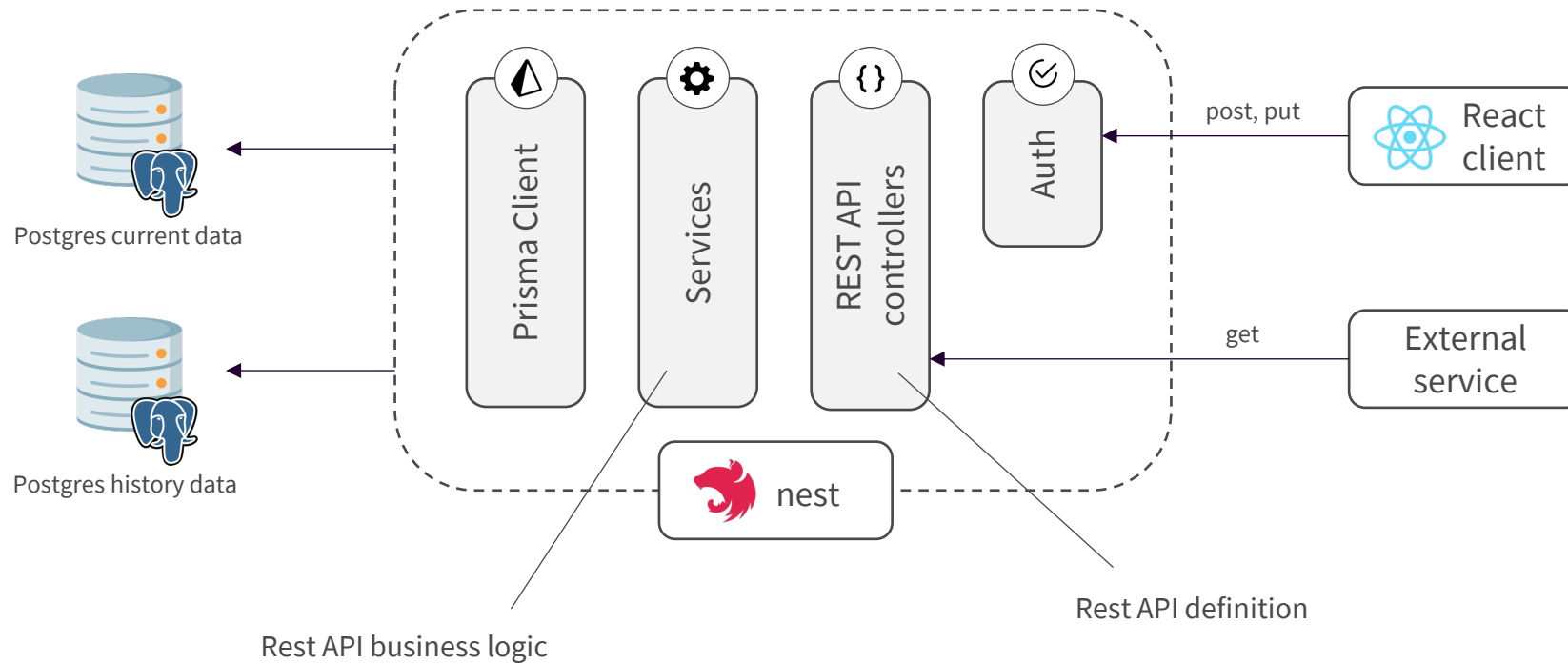
1. **Web service:** Main interface for accessing the data and is built using the **React framework**.

Given the geographic nature of the data, the service includes a map for displaying the location of the shelters. Furthermore, a **Swagger (documentation service) playground** is available for interacting with the API.



Service implementation

2. **Rest API service:** Provides an API for accessing and interact with the service data. It is built using the **Nest framework** and the **Prisma ORM**.



Service implementation

3. **Prisma ORM:** The decision to use an ORM (Object-Relational Mapping) was made to achieve greater efficiency in the development of the application, including table creation, queries, and migrations. In particular, the choice of **Prisma** was motivated by its **flexibility** in supporting different types of databases, including PostgreSQL, MySQL, and SQLite, as well as its support for **TypeScript**. This makes Prisma a very versatile option for developing backend applications, allowing for the use of different technologies while also providing increased security and flexibility.

```
model shelter {
  id      String      @id @default(cuid())
  name    String
  province String
  region  String
  url     String
  latitude Float
  longitude Float
  author  String
  createdAt DateTime @default(now())
  amenities shelter_service[]
}
```









```
Carlo De Nardin, 7 days ago | 1 author (Carlo De Nardin)
-- CreateTable
CREATE TABLE "shelter" (
  "id" TEXT NOT NULL,
  "name" TEXT NOT NULL,
  "province" TEXT NOT NULL,
  "region" TEXT NOT NULL,
  "url" TEXT NOT NULL,
  "latitude" DOUBLE PRECISION NOT NULL,
  "longitude" DOUBLE PRECISION NOT NULL,
  "author" TEXT NOT NULL,
  "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,

  CONSTRAINT "shelter_pkey" PRIMARY KEY ("id")
);
```

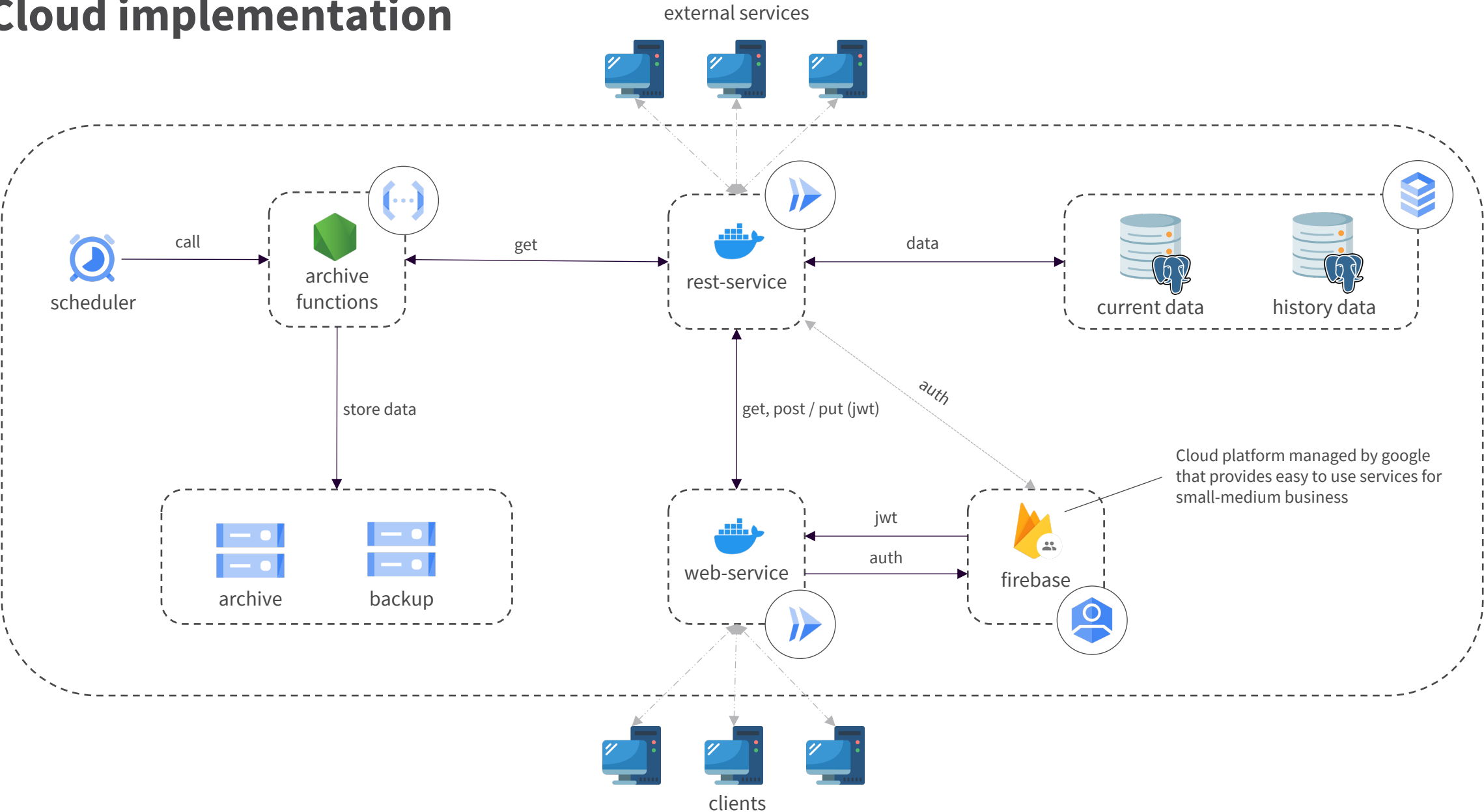
```
const shelter = await this.prismaServiceDb.shelter.findUnique({
  where: {
    id,
  },
  include: {
    amenities: true,
  },
});
```

Cloud implementation

The service has been provided by using a cloud provider, in this case **google cloud platform** has been used. Different google cloud **paas** services have been used:

-  **Google Identity Platform:** provides identity and access management (Firebase)
-  **Cloud SQL:** relational database service (PostgreSQL)
-  **Cloud Run:** serverless compute platform that automatically scales stateless containers
-  **Container Registry:** container registry service that allows to store and manage container images
-  **Secret Manager:** allows to securely store and manage secrets such as API keys and passwords
-  **Cloud Storage:** storage service that allows to store and access data from the web
-  **Cloud Functions:** allows to run code in response to events
-  **Cloud Scheduler:** allows to schedule jobs that automate recurring tasks

Cloud implementation



Cloud Cost

Cloud SQL for PostgreSQL

DB-CUSTOM-1-4096 HA

of instances: 1

Instance type: db-custom-1-4096

Location: Zurich

730.0 total hours per month

SSD Storage: 10.0 GiB

Backup: 10.0 GiB

EUR 129.85

DB-G1-SMALL

of instances: 1

Instance type: db-g1-small

Location: Zurich

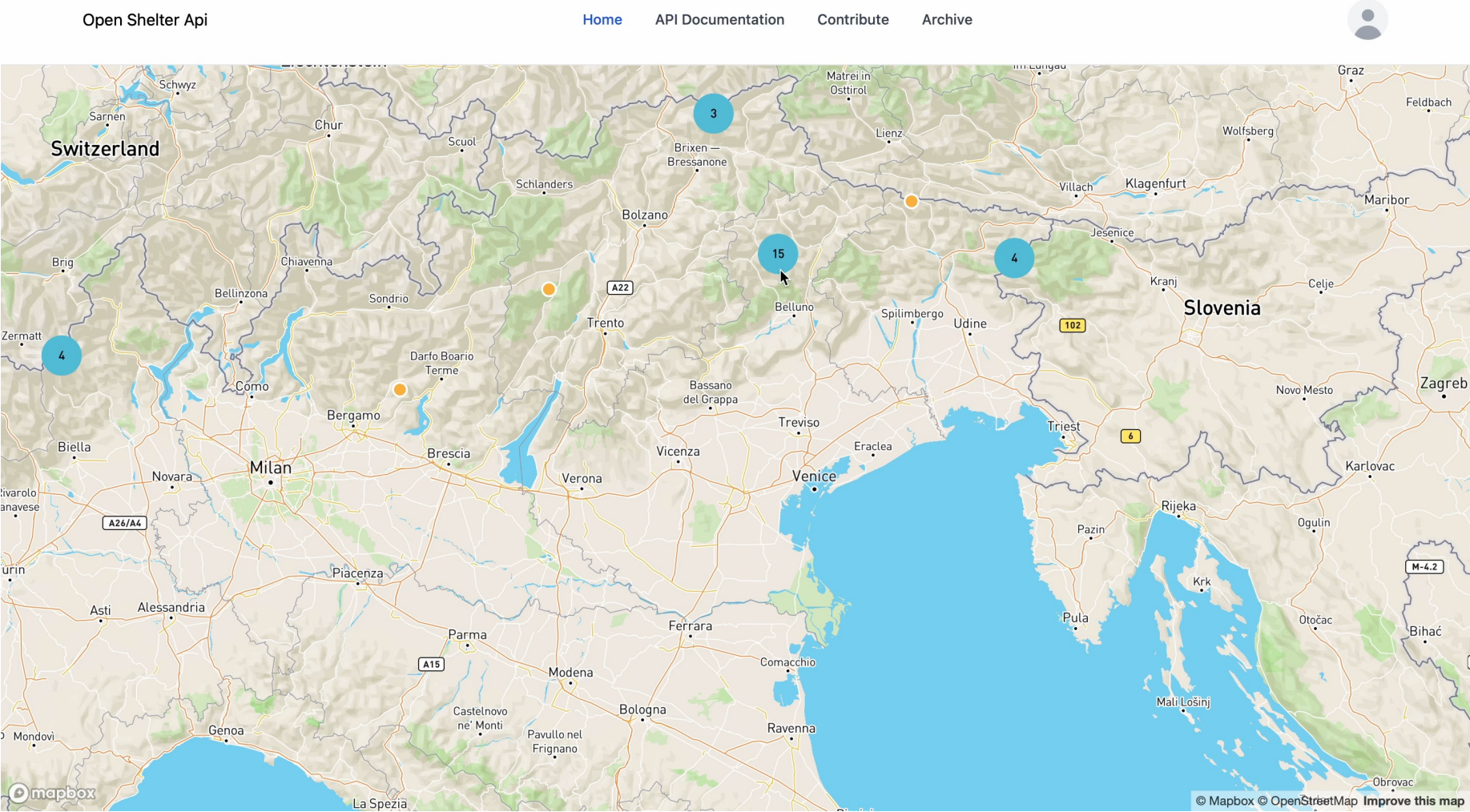
730.0 total hours per month

HDD Storage: 100.0 GiB

Backup: 100.0 GiB

EUR 52.43

Total Estimated Cost: EUR 182.28 per 1 month



Conclusions

In conclusion, the Open Shelter API has made it possible to make information about mountain shelters available and accessible to interested users and services. Thanks to the **ODbL license**, the data has been made open and free, promoting the diffusion of information.

Furthermore, the **syntactic** and **semantic interoperability** of the API allows for external services to efficiently manage and utilize the data. This feature enhances the overall functionality and usefulness of the Open Shelter API, making it a valuable resource for those interested in mountain shelters.

References

- <https://ukdataservice.ac.uk/learning-hub/research-data-management>
- https://en.wikipedia.org/wiki/Entity-attribute-value_model
- <https://www.dublincore.org/>
- <https://json-ld.org>
- <https://schema.org>
- <https://nestjs.com>
- <https://www.prisma.io>
- <https://cloud.google.com>
- <https://web-service-hnlijallya-oa.a.run.app>
- <https://github.com/orgs/ODC-units/repositories>