

# GeoAP

---

FastAPI + PostGIS service to store Wi-Fi access points and query nearest neighbors.

## Features

- Ingests hierarchical `data.json` into PostGIS as `POINTZ` geography with upsert protection on `ap_code`.
- Nearest AP endpoint ordered by distance (meters) with optional limit and max distance filters.
- Dockerized stack with PostGIS, automatic schema bootstrap, and one-off data loader.

## Stack

- Python 3.11, FastAPI, SQLAlchemy/GeoAlchemy2, asyncpg.
- PostgreSQL + PostGIS (postgis/postgis:16-3.4).
- Docker Compose for orchestration.

## Project layout

- `src/app/` — API code, models, repositories, services.
- `scripts/init_db.sql` — schema + PostGIS extension + indexes.
- `scripts/load_data.py` — one-off loader for `data.json` (upserts).
- `data.json` — source dataset.
- `docker-compose.yml` — services: `db`, `data-loader`, `api`.

## Prerequisites

- Docker and Docker Compose.
- Copy environment template and adjust as needed:

Dr. Fa

```
cp .env.example .env
# edit credentials/limits if desired
```

## Run with Docker

Build and start the stack; this will:

1. Start PostGIS and apply `init_db.sql` automatically.
2. Run `data-loader` once to ingest `data.json`.
3. Start the API on port 8000.

```
docker compose up --build
```

Check health:

```
curl http://localhost:8000/health
```

Query nearest APs (example):

```
curl "http://localhost:8000/aps/BALZAY-AU1-PB-155/nearest?limit=5"
```

Optional radius filter:

```
curl "http://localhost:8000/aps/BALZAY-AU1-PB-155/nearest?  
max_distance_m=50"
```

Logs:

```
docker compose logs -f api  
docker compose logs -f data-loader
```

Stop:

```
docker compose down
```

## Local development (without Docker)

Dr. Fa

- Create a PostGIS-enabled database and run `scripts/init_db.sql`.
- Export `DATABASE_URL` pointing to your DB (asyncpg format).
- Install deps: `pip install -r requirements.txt`.
- Ingest data: `python scripts/load_data.py`.
- Run API: `uvicorn app.main:app --reload --host 0.0.0.0 --port 8000` (from repo root, ensure `PYTHONPATH=./src`).

## Testing

```
pytest
```

## Notes

- Access point uniqueness is enforced by `ap_code`.

- Geography is stored as **POINTZ** (lon/lat/alt, SRID 4326) and distance uses meters via PostGIS.