TiPg

OGC Features and Tiles API for PostGIS

Vincent Sarago - DevelopmentSeed
David Bitner - DevelopmentSeed

FOSS4G - Prizren - June 2023

Geospatial Engineer @ Developmentseed

COG Tzar

Self-Taught Python dev

Creator of @RemotePixel

MSc in Earth Sciences

Bike & Coffee

Another OGC Tiles/Features server

https://github.com/geopython/pygeoapi

https://github.com/CrunchyData/pg_featureserv

https://github.com/CrunchyData/pg_tileserv

https://github.com/azavea/franklin

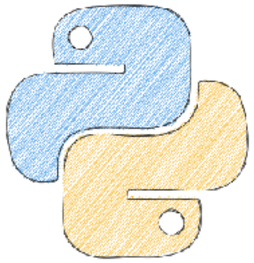
https://github.com/maplibre/martin

https://github.com/t-rex-tileserver/t-rex/

https://github.com/microsoft/ogc-api-fast-features

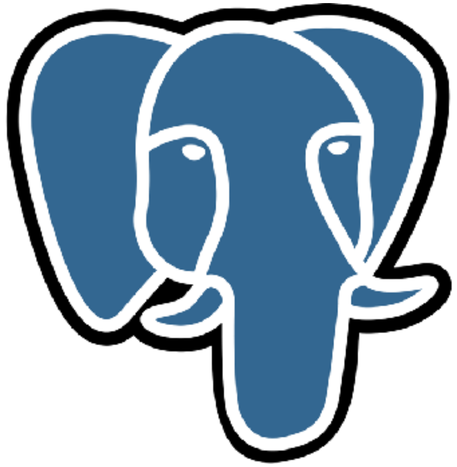
https://github.com/opengeospatial/ogcapi-tiles/blob/master/implementations.adoc

Why another?



Python >=3.8

FastAPI



Why another?

Customization

Integration

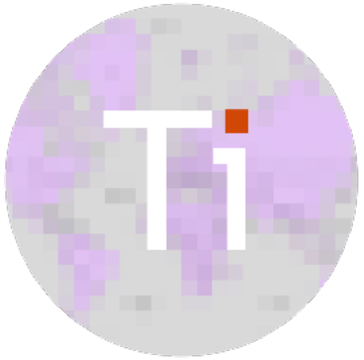
Performance

Why another?

TiPg
Vector / Features Services

https://github.com/developmentseed/tipg

History

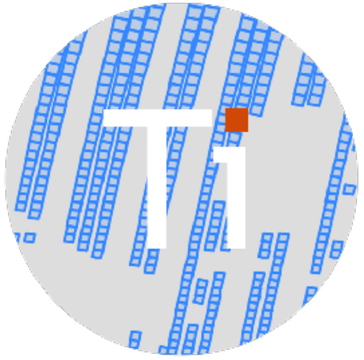


TiMVT
Vector Tiles

https://developmentseed.org/timvt/

TiPg
Vector / Features Services

https://github.com/developmentseed/tipg



TiFeatures
OGC Features

https://developmentseed.org/tifeatures/

Ti-* Family

TiTiler
Raster Services

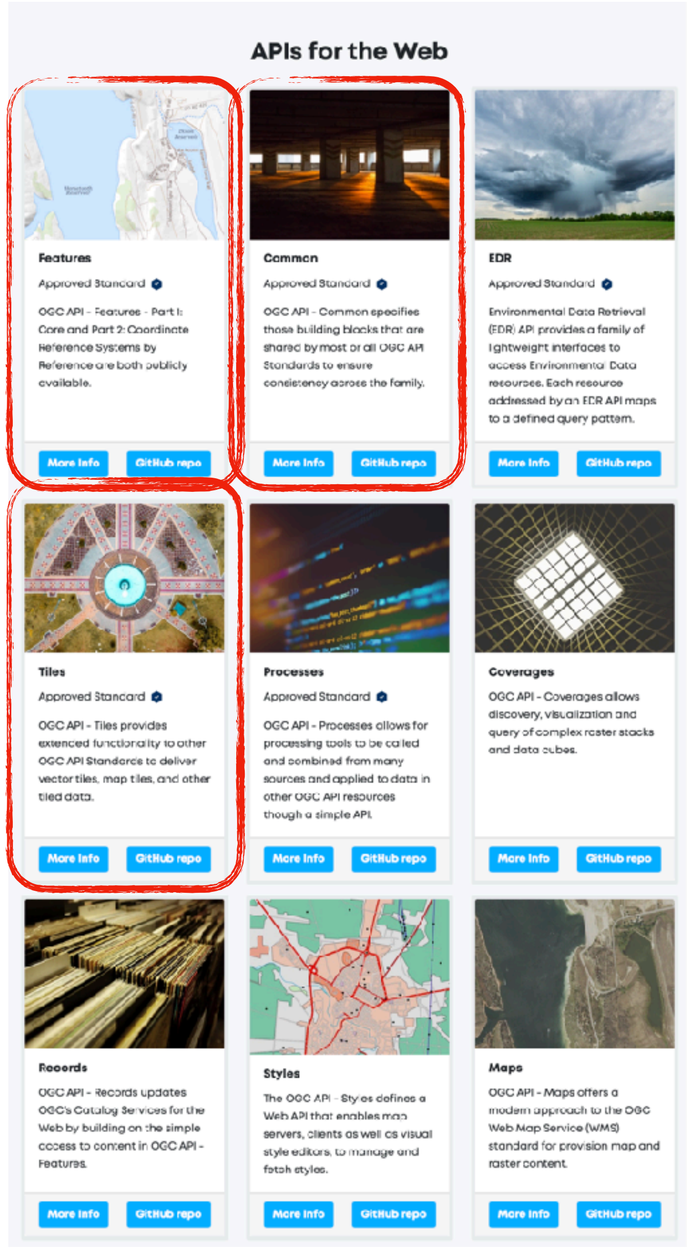
https://developmentseed.org/titiler/

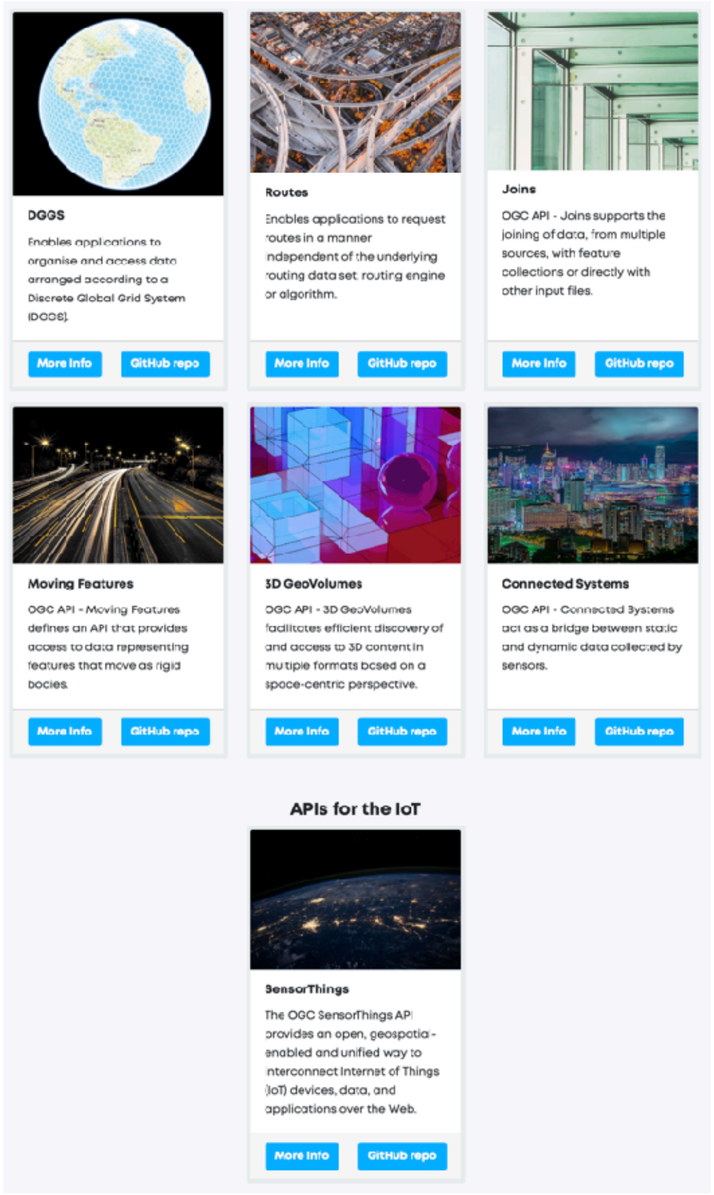
TiPg
Vector / Features Services

https://github.com/developmentseed/tipg

Specifications

Open Geospatial Consortium





Specifications

OGC Common Part 1: Core	✅
OGC Common Part 2: Geospatial Data	✅
OGC Features Part 1: Core	✅
OGC Features Part 2: CRS by Reference	❌
OGC Features Part 3: Filtering / CQL2	✅
OGC Tiles Part 1: Core	✅

Opinions

The project authors choose not to implement the Part 2 of the specification to avoid the introduction of CRS based GeoJSON. This might change in the future.

While the authors tried to follow OGC specifications to the letter,
some API endpoints might have more capabilities (e.g geometry column selection).

Endpoints

OGC API - Common - Part 1: Core

GET / - Retrieves the landing page.
GET /api - Retrieve the API definition
GET /conformance - Provides a list declaring the modules that are implemented by the API.

OGC API - Common - Part 2: Geospatial Data

GET /collections - Retrieves information which describes the set of supported Collections.
GET /collections/{collectionId} - Retrieves descriptive information about a specific Collection.

OGC API - Features - Part 1: Core

GET /collections - List the collections of data.
GET /collections/{collectionId}/items - Retrieves features for a specific Collection.
GET /collections/{collectionId}/items/{idemId} - Retrieves a single feature for a specific Collection.

OGC API - Features - Part 3: Filtering

Adds advanced CQL2 `filtering`

GET /collections/{collectionId}/queryables - Retrieves queryable properties for a specific Collection.

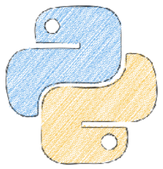
Endpoints

OGC API - Tiles - Part 1: Core

GET /tileMatrixSets - List available TileMatrixSets supported.
GET /tileMatrixSets/{tileMatrixSetId} - Retrieve metadata for a specific TileMatrixSet.
GET /collections/{collectionId}/tiles - List available TileSet for a specific TileMatrixSet.
GET /collections/{collectionId}/tiles/{tileMatrixSetId} - Retrieve metadata for a specific TileSet.
GET /collections/{collectionId}/tiles/{tileMatrixSetId}/{z}/{x}/{y} - Retrieve a Vector Tile for a specific TileSet.

Features

- OGC Features + Tiles API
- Modular (Features or Tiles or both)
- Support custom SQL layers
- Support non-geo tables
- Allow for multiple geometry columns
- No admin role needed on the database (even for custom SQL)
- Multiple TileMatrixSets supported (via morecantile)
- Web viewer
- Customizable

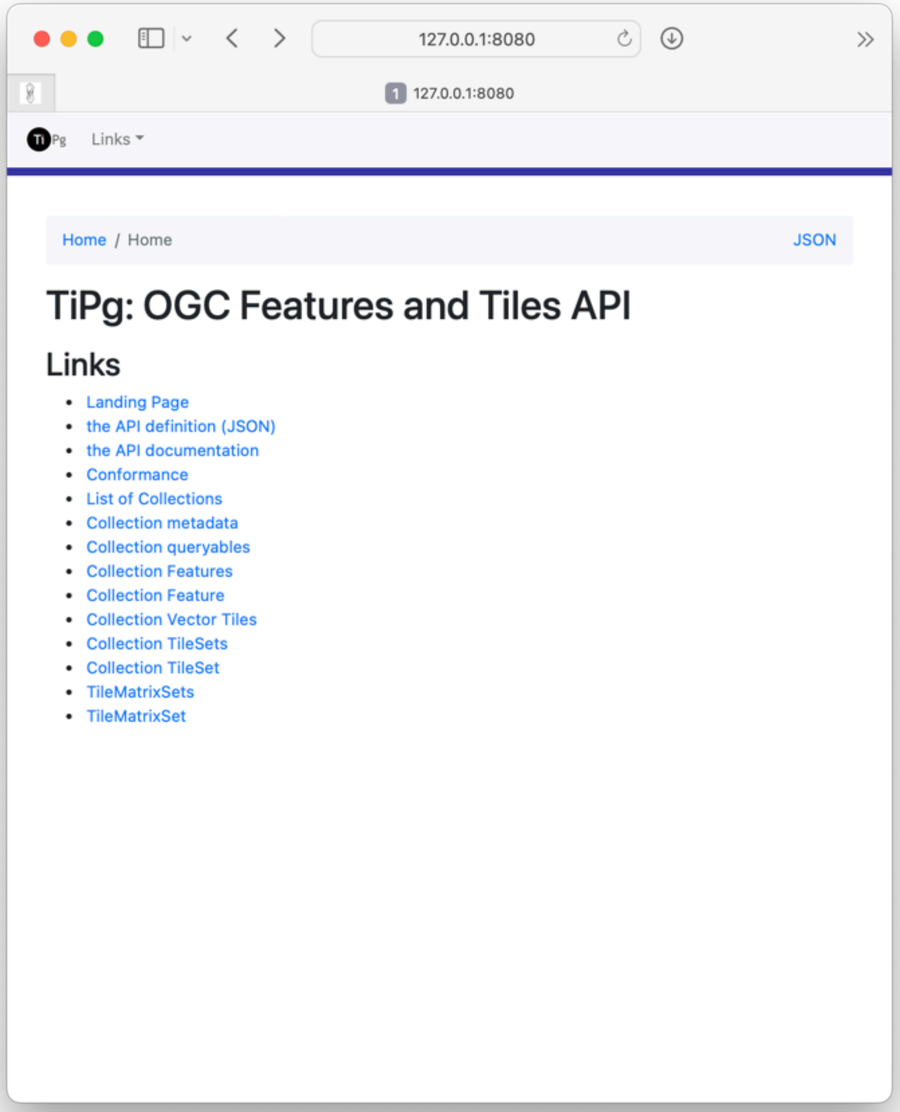


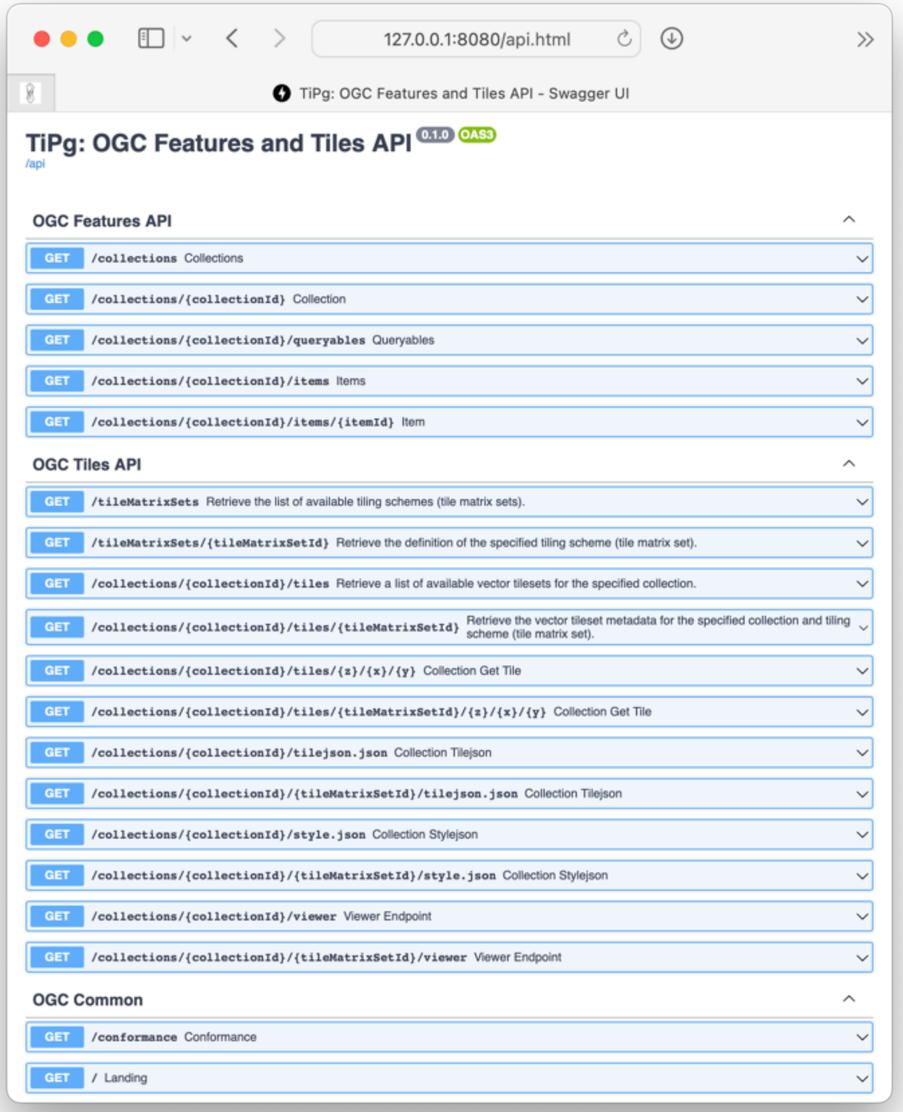
Python >=3.8

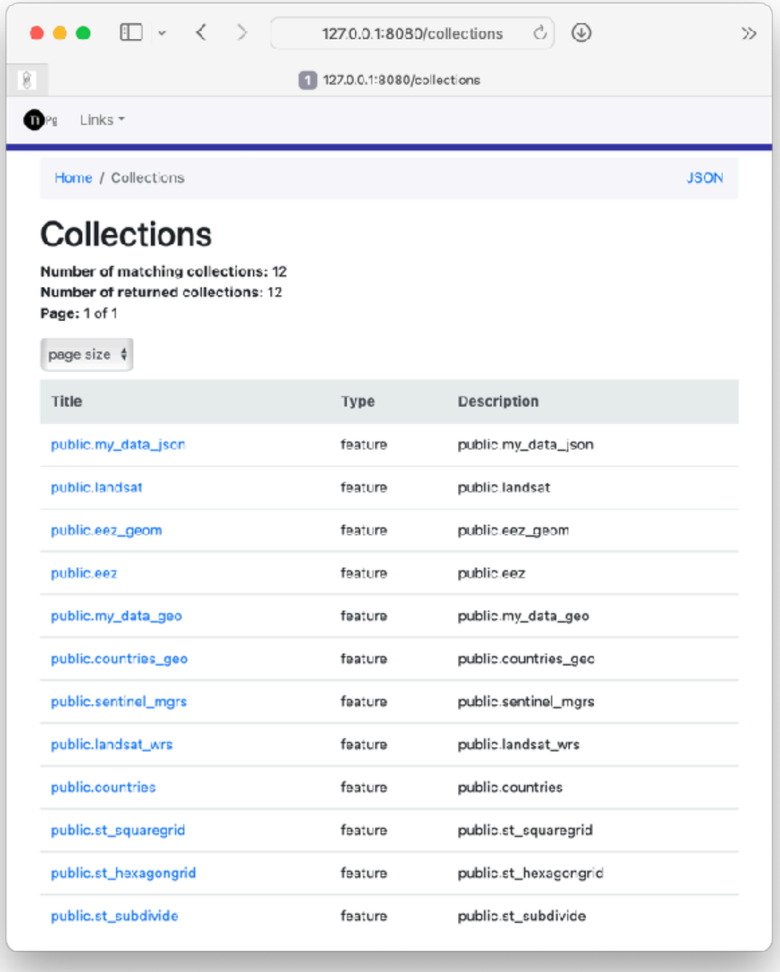
python -m pip install tipg

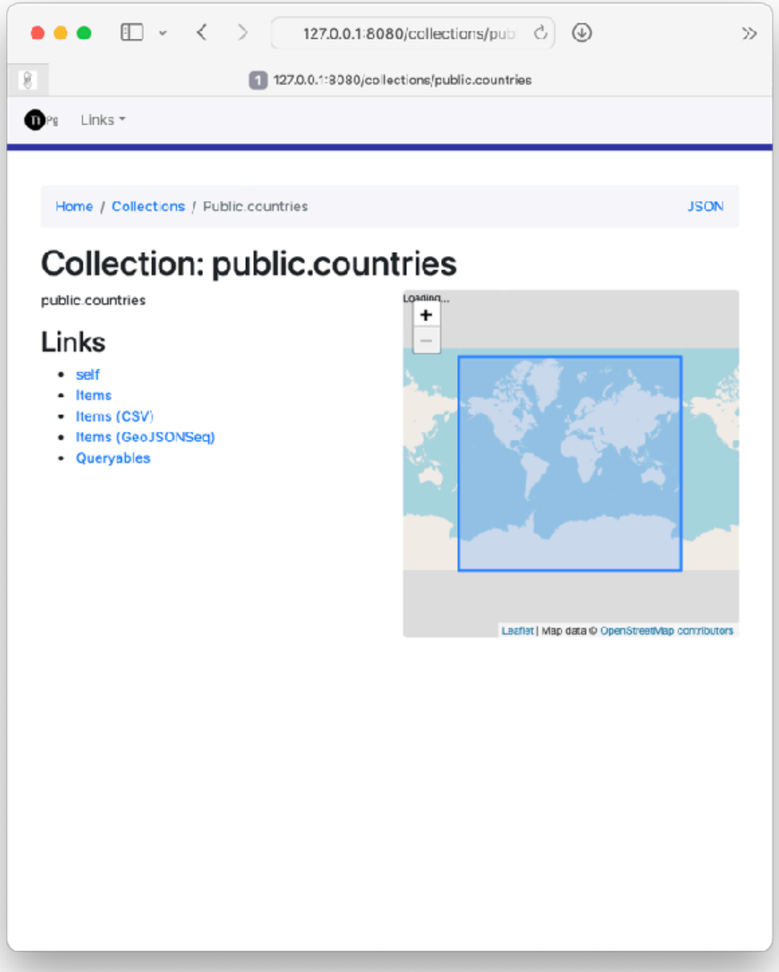
Launch

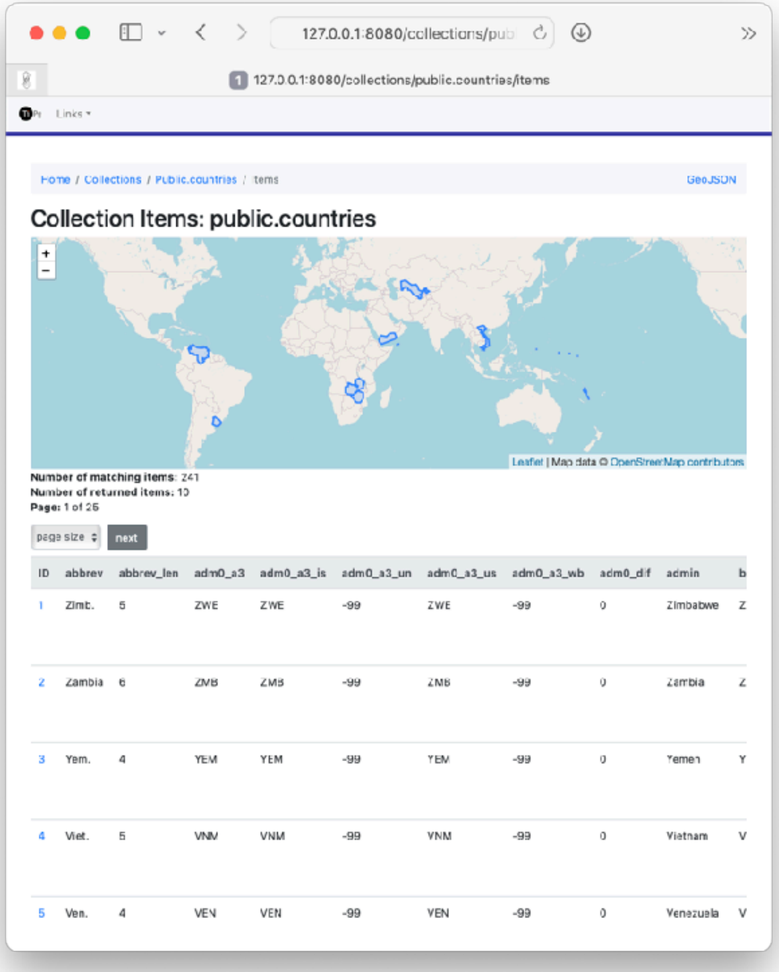
```
pip install tipg uvicorn

export DATABASE_URL=postgresql://username:password@0.0.0.0:5432/postgis
uvicorn tipg.main:app --port 8080 --reload

or

docker run \
    -p 8081:8081 \
    -e PORT=8081 \
    -e HOST=127.0.0.1 \
    -e DATABASE_URL=postgresql://username:password@0.0.0.0:5432/postgis \
    -e DEBUG=TRUE \
    --rm ghcr.io/developmentseed/tipg:latest
```


127.0.0.1:8080
127.0.0.1:8080
TiPg
Links
Home / Home
JSON
TiPg: OGC Features and Tiles API
Links
Landing Page
the API definition (JSON)
the API documentation
Conformance
List of Collections
Collection metadata
Collection queryables
Collection Features
Collection Feature
Collection Vector Tiles
Collection TileSets
Collection TileSet
TileMatrixSets
TileMatrixSet


127.0.0.1:8080/api.html
TiPg: OGC Features and Tiles API - Swagger UI
TiPg: OGC Features and Tiles API 0.1.0 OAS3
/api
OGC Features API
GET /collections Collections
GET /collections/{collectionId} Collection
GET /collections/{collectionId}/queryables Queryables
GET /collections/{collectionId}/items Items
GET /collections/{collectionId}/items/{itemId} Item
OGC Tiles API
GET /tileMatrixSets Retrieve the list of available tiling schemes (tile matrix sets).
GET /tileMatrixSets/{tileMatrixSetId} Retrieve the definition of the specified tiling scheme (tile matrix set).
GET /collections/{collectionId}/tiles Retrieve a list of available vector tilesets for the specified collection.
GET /collections/{collectionId}/tiles/{tileMatrixSetId} Retrieve the vector tileset metadata for the specified collection and tiling scheme (tile matrix set).
GET /collections/{collectionId}/tiles/{z}/{x}/{y} Collection Get Tile
GET /collections/{collectionId}/tiles/{tileMatrixSetId}/{z}/{x}/{y} Collection Get Tile
GET /collections/{collectionId}/tilejson.json Collection Tilejson
GET /collections/{collectionId}/{tileMatrixSetId}/tilejson.json Collection Tilejson
GET /collections/{collectionId}/style.json Collection Stylejson
GET /collections/{collectionId}/{tileMatrixSetId}/style.json Collection Stylejson
GET /collections/{collectionId}/viewer Viewer Endpoint
GET /collections/{collectionId}/{tileMatrixSetId}/viewer Viewer Endpoint
OGC Common
GET /conformance Conformance
GET / Landing


127.0.0.1:8080/collections
Links
Home / Collections
JSON
Collections
Number of matching collections: 12
Number of returned collections: 12
Page: 1 of 1
page size
Title
Type
Description
public.my_data_json
feature
public.my_data_json
public.landsat
feature
public.landsat
public.eez_geom
feature
public.eez_geom
public.eez
feature
public.eez
public.my_data_geo
feature
public.my_data_geo
public.countries_geo
feature
public.countries_geo
public.sentinel_mgrs
feature
public.sentinel_mgrs
public.landsat_wrs
feature
public.landsat_wrs
public.countries
feature
public.countries
public.st_squaregrid
feature
public.st_squaregrid
public.st_hexagongrid
feature
public.st_hexagongrid
public.st_subdivide
feature
public.st_subdivide


127.0.0.1:8080/collections/public.countries
Links
Home / Collections / Public.countries
JSON
Collection: public.countries
public.countries
Links
self
Items
Items (CSV)
Items (GeoJSONSeq)
Queryables
Leaflet | Map data © OpenStreetMap contributors


127.0.0.1:8080/collections/public.countries/items
Links
Home / Collections / Public.countries / Items
GeoJSON
Collection Items: public.countries
Leaflet | Map data © OpenStreetMap contributors
Number of matching items: 241
Number of returned items: 10
Page: 1 of 25
page size
next
ID
abbrev
abbrev_len
adm0_a3
adm0_a3_is
adm0_a3_un
adm0_a3_us
adm0_a3_wb
adm0_dif
admin
1
Zimb.
5
ZWE
ZWE
-99
ZWE
-99
0
Zimbabwe
2
Zambia
6
ZMB
ZMB
-99
ZMB
-99
0
Zambia
3
Yem.
4
YEM
YEM
-99
YEM
-99
0
Yemen
4
Viet.
5
VNM
VNM
-99
VNM
-99
0
Vietnam
5
Ven.
4
VEN
VEN
-99
VEN
-99
0
Venezuela

127.0.0.1:8080/collections/public.countries/items/2

Home / Collections / Public.countries / Items / 2 GeoJSON

Collection Item: 2

Properties

- **ID:** 2
- **abbrev:** Zambia
- **abbrev_len:** 6
- **adm0_a3:** ZMB
- **adm0_a3_is:** ZMB
- **adm0_a3_un:** -99
- **adm0_a3_us:** ZMB
- **adm0_a3_wb:** -99
- **adm0_dif:** 0
- **admin:** Zambia
- **brk_a3:** ZMB
- **brk_diff:** 0
- **brk_group:** None
- **brk_name:** Zambia
- **continent:** Africa
- **economy:** 7. Least developed region
- **featurecla:** Admin-0 country
- **fips_10_:** ZA
- **formal_en:** Republic of Zambia
- **formal_fr:** None
- **gdp_md_est:** 65170.0
- **gdp_year:** 2016
- **geou_dif:** 0
- **geounit:** Zambia
- **gid:** 2
- **gu_a3:** ZMB
- **homepart:** 1
- **income_grp:** 4. Lower middle income
- **iso_a2:** ZM
- **iso_a3:** ZMB
- **iso_a3_eh:** ZMB
- **iso_n3:** 894
- **labelrank:** 3
- **lastcensus:** 2010
- **level:** 2
- **long_len:** 6
- **mapcolor13:** 13
- **mapcolor7:** 5
- **mapcolor8:** 8
- **mapcolor9:** 6
- **max_label:** 8.0
- **min_label:** 3.0
- **min_zoom:** 0.0
- **name:** Zambia
- **name_alt:** None
- **name_ar:** زامبيا

127.0.0.1:8080/collections/public.countries/tiles

Home / Collections / Public.countries / Tiles JSON

TileSet list

Links

- 'public.countries' tileset tiled using CanadianNAD83_LCC TileMatrixSet
- 'public.countries' tileset tiled using EuropeanETRS89_LAEAQuad TileMatrixSet
- 'public.countries' tileset tiled using LINZAntarticaMapTilegrid TileMatrixSet
- 'public.countries' tileset tiled using NZTM2000Quad TileMatrixSet
- 'public.countries' tileset tiled using UPSAntarcticWGS84Quad TileMatrixSet
- 'public.countries' tileset tiled using UPSArcticWGS84Quad TileMatrixSet
- 'public.countries' tileset tiled using UTM31WGS84Quad TileMatrixSet
- 'public.countries' tileset tiled using WGS1984Quad TileMatrixSet
- 'public.countries' tileset tiled using WebMercatorQuad TileMatrixSet
- 'public.countries' tileset tiled using WorldCRS84Quad TileMatrixSet
- 'public.countries' tileset tiled using WorldMercatorWGS84Quad TileMatrixSet

127.0.0.1:8080/collections/public.countries/tiles/WebMercatorQuad

Home / Collections / Public.countries / Tiles / WebMercatorQuad JSON

'public.countries' tileset tiled using WebMercatorQuad TileMatrixSet

Properties

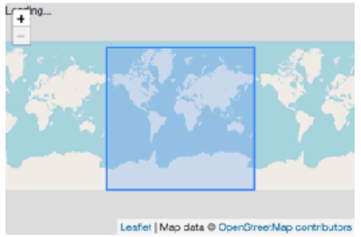
- **DataType:** vector
- **CRS:** http://www.opengis.net/def/crs/EPSG/0/3857

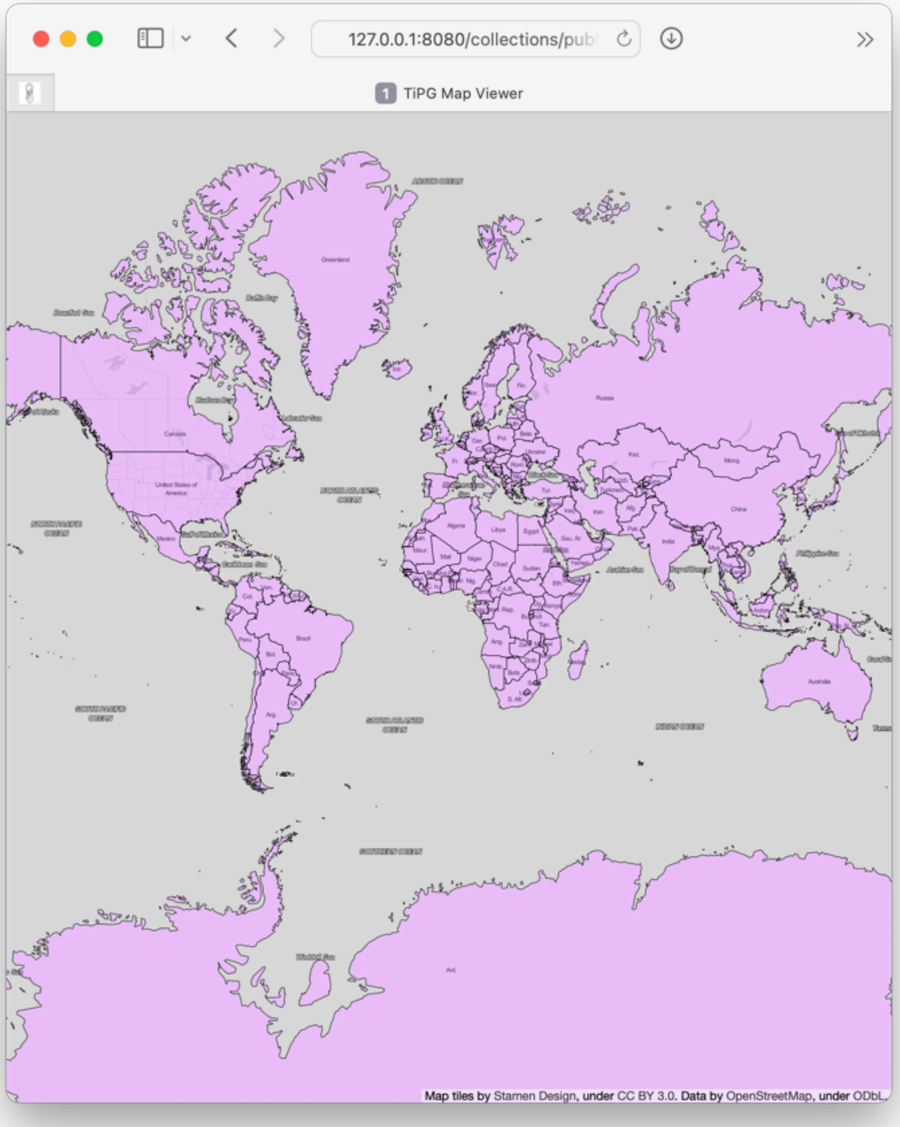
TileMatrixSet Limits

Level	minTileRow	maxTileRow	minTileCol	maxTileCol
0	0	1	0	1
1	0	2	0	2
2	0	4	0	4
3	0	8	0	8
4	0	16	0	16
5	1	32	0	32
6	2	64	0	64
7	5	128	0	128
8	10	256	0	256
9	20	512	0	512
10	41	1024	0	1024
11	83	2048	0	2048
12	167	4096	0	4096
13	335	8192	0	8192
14	671	16384	0	16384
15	1343	32768	0	32768
16	2687	65536	0	65536
17	5374	131072	0	131072
18	10748	262144	0	262144
19	21496	524288	0	524288
20	42992	1048576	0	1048576
21	85985	2097152	0	2097152
22	171971	4194304	0	4194304
23	343943	8388608	0	8388608
24	687886	16777216	0	16777216

Links

- Definition of 'WebMercatorQuad' tileMatrixSet
- Templated link for retrieving Vector tiles




127.0.0.1:8080/collections/pub
TiPG Map Viewer
Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

Build your own application

```python
from contextlib import asynccontextmanager
from tipg.database import close_db_connection, connect_to_db
from tipg.collections import register_collection_catalog
from tipg.errors import DEFAULT_STATUS_CODES, add_exception_handlers
from tipg.factory import OGCFeaturesFactory
from tipg.settings import DatabaseSettings

from fastapi import FastAPI


@asynccontextmanager
async def lifespan(app: FastAPI):
    """FastAPI Lifespan

    - Create DB connection POOL and `register` the custom tipg SQL function within `pg_temp`
    - Create the collection_catalog
    - Close the connection pool when closing the application

    """
    await connect_to_db(
        app,
        settings=PostgresSettings(database_url="postgres://...."),
        schemas=["public"],
    )
    await register_collection_catalog(app, schemas=["public"])

    yield

    await close_db_connection(app)


app = FastAPI(openapi_url="/api", docs_url="/api.html", lifespan=lifespan)

endpoints = OGCFeaturesFactory(with_common=True)
app.include_router(endpoints.router, tags=["OGC Features API"])

add_exception_handlers(app, DEFAULT_STATUS_CODES)
```

Build your own application

```python
app = FastAPI(openapi_url="/api", docs_url="/api.html", lifespan=lifespan)

endpoints = OGCFeaturesFactory(with_common=True)
app.include_router(endpoints.router, tags=["OGC Features API"])
```

Create a FastAPI application

Create a set of endpoints

Register the endpoints to the application

Build your own application

tipg.factory.OGCFeaturesFactory ← OGC Features API

tipg.factory.OGCTilesFactory ← OGC Tiles API

tipg.factory.Endpoints ← OGC Features & Tiles API

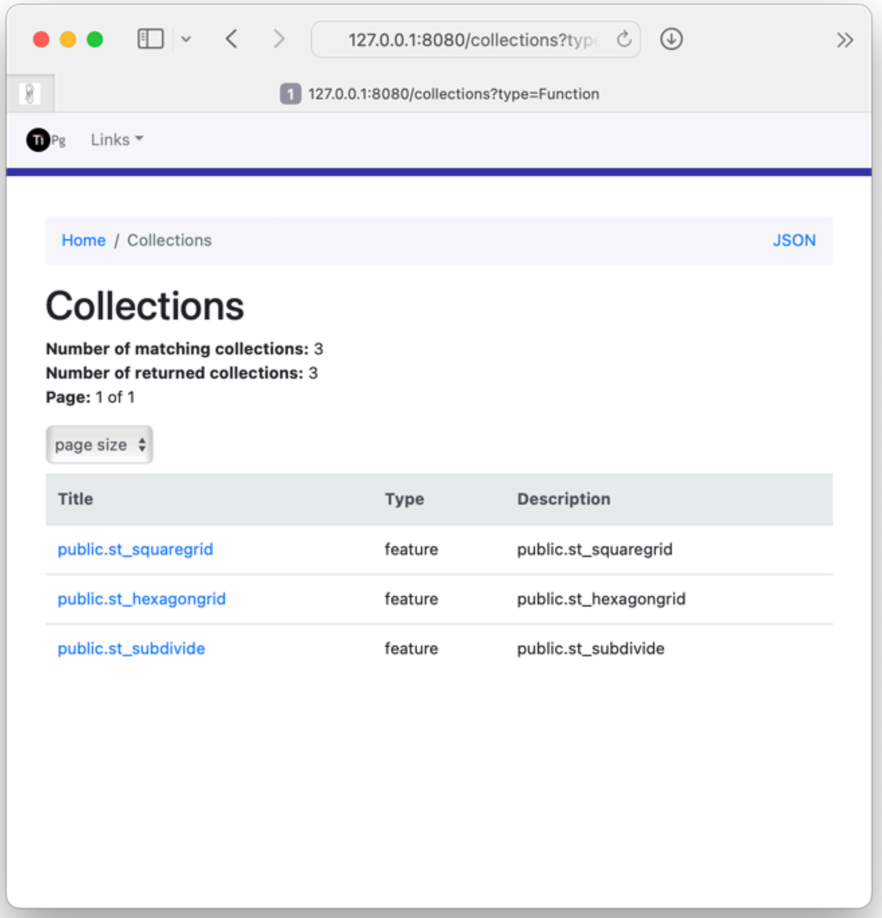
Custom SQL Layer

`SQL Functions` are any procedural functions defined in the database that match the following criteria:

- Must be defined to return `SETOF`
- Functions defined to return `RECORD` must include typed OUT definitions in the function signature
- All arguments (`IN` and `OUT`) must be named
- Functions that return a `geometry` will be usable as Features and Vector Tiles, those that do not, will be available to return json/csv
- Functions that match these qualifications will be found based on the DB visibility settings (schemas)

```
CREATE FUNCTION hexagons(
    IN size int DEFAULT 10,
    IN bounds geometry DEFAULT 'srid=4326;POLYGON((-180 -90,-180 90,180 90,180 -90,-180 -90))'::geometry,
    OUT geom geometry,
    OUT i integer,
    OUT j integer
) RETURNS SETOF RECORD AS $$
    SELECT * FROM st_hexagongrid(size, bounds);
$$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
```

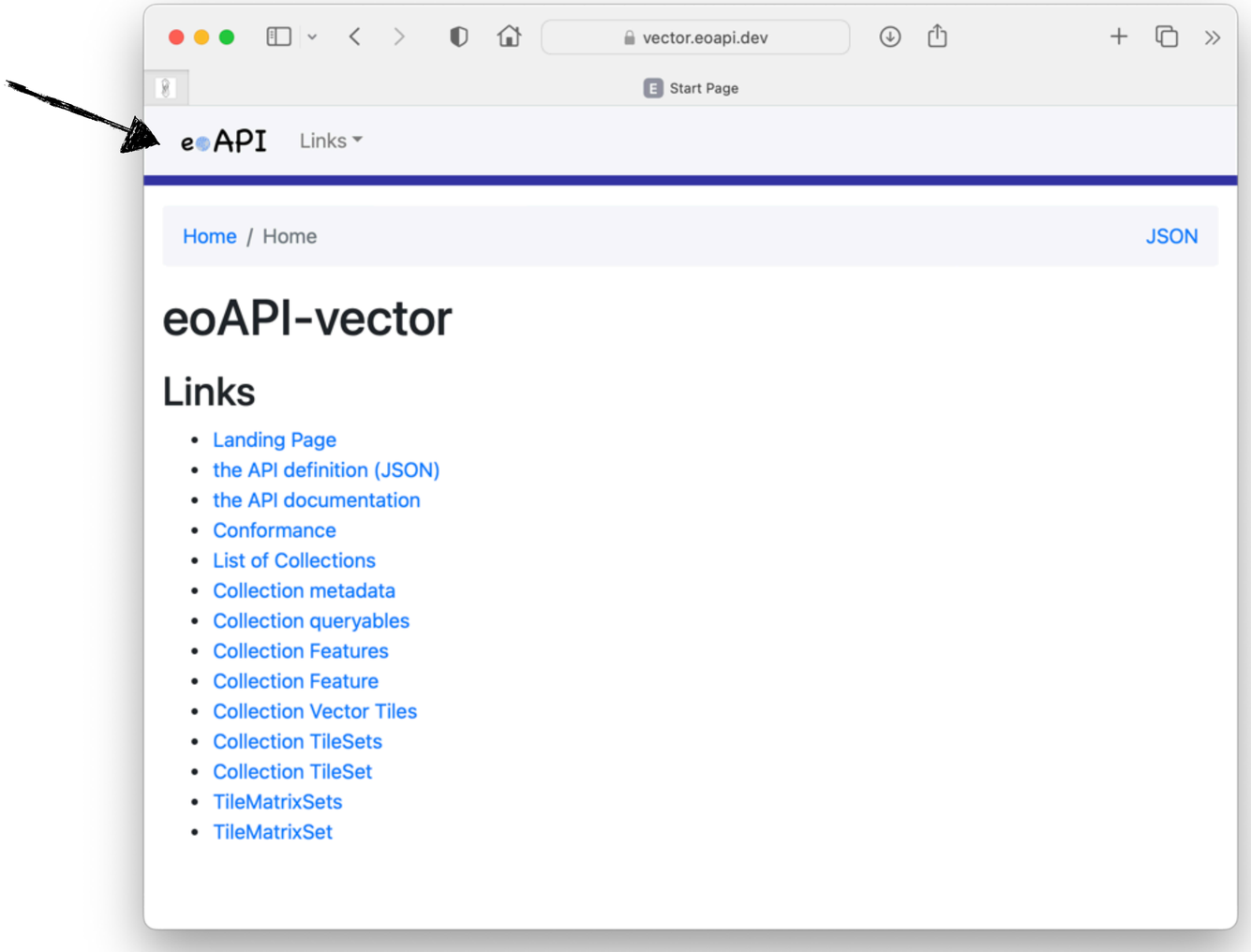
Custom SQL Layer



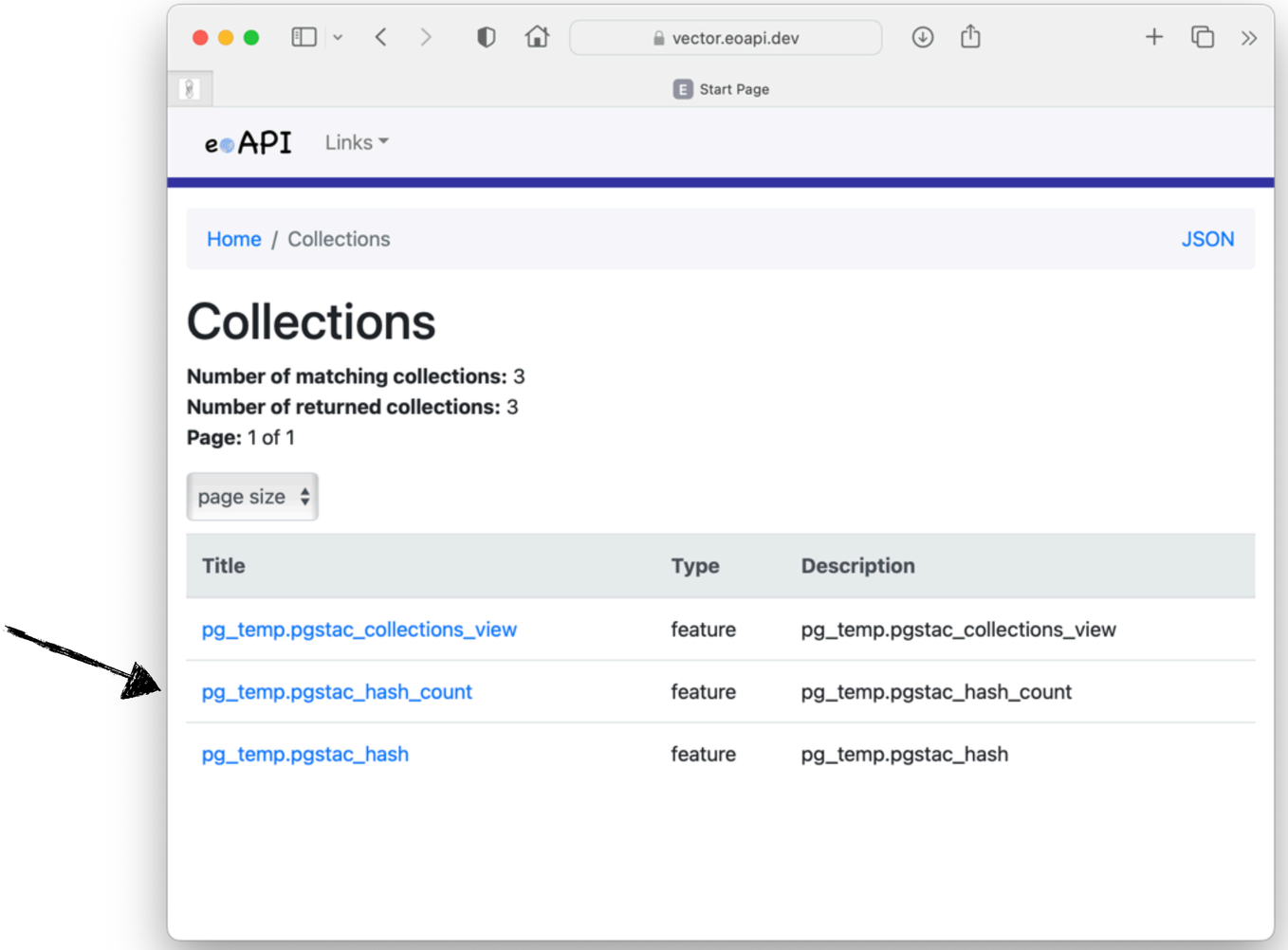
Custom SQL Layer

`SQL Functions` can be pre-existent in the database,
or you can tell `tipg` to register SQL code dynamically to
the `pg_temp` schema at startup

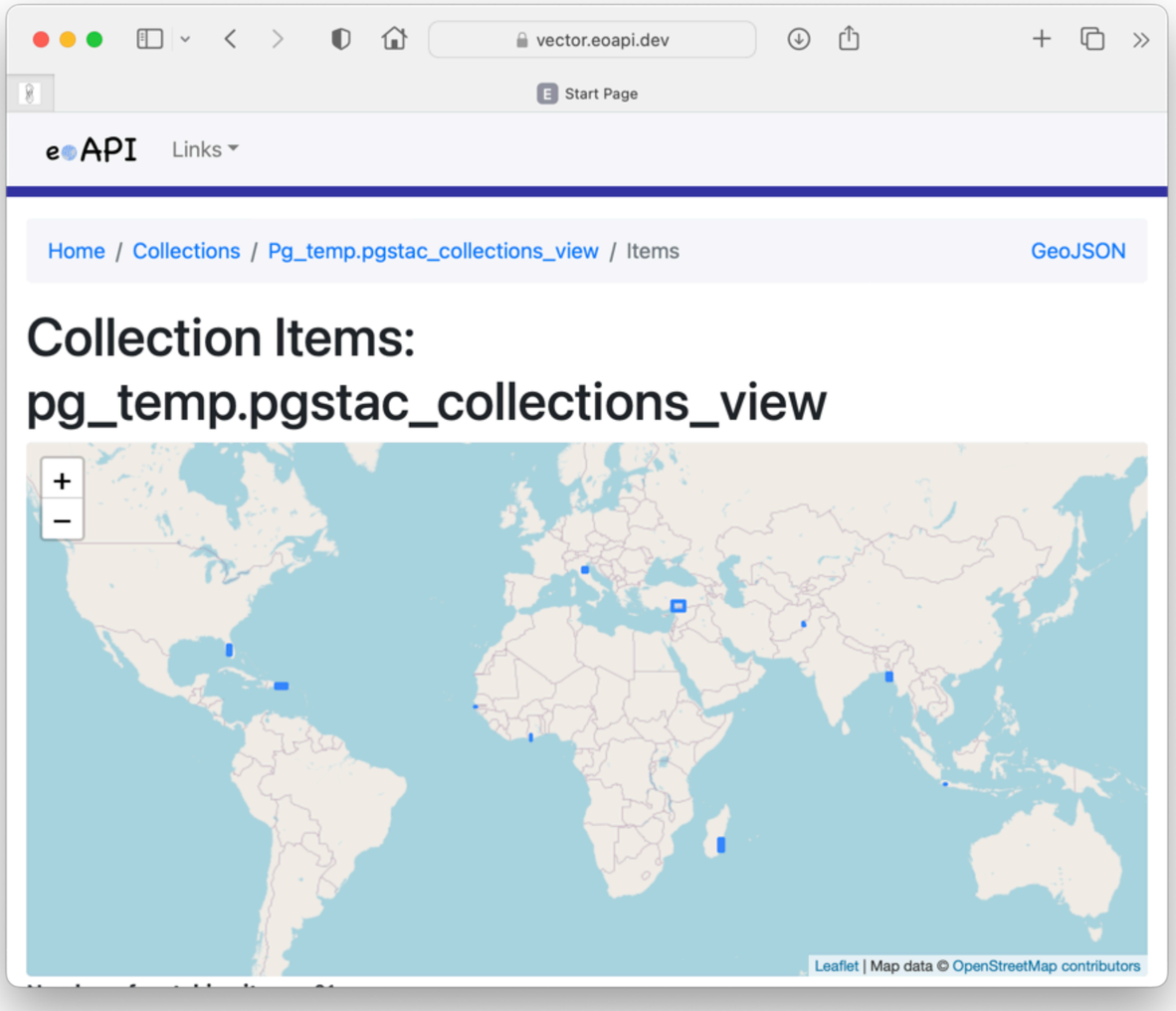
The eoAPI example



The eoAPI example



The eoAPI example



https://github.com/developmentseed/tipg

Ping Me!

@_VincentS_
@cogeotiff
@developmentseed

Join the team & make a better planet.
https://developmentseed.org/careers