

# Vector Tiles mit PostGIS und QGIS

28. Januar 2021 | Astrid Emde | FOSSGIS Update 2021 Online

**Aufgepasst:**

Wieviele Tiere sind in den Folien versteckt? Mitzählen & ein GDAL-T-Shirt gewinnen.  
Die erste richtige Antwort im Chat gewinnt.

# Astrid Emde

WhereGroup Bonn seit 2002, GIS-Consultant  
FOSS-Academy-Schulungen  
PostgreSQL/PostGIS, MapServer, GeoServer, QGIS,  
QGIS Server, Mapbender, PostNAS-Suite  
OSGeo Board, FOSSGIS e.V., Mapbender PSC,  
OSGeoLive PSC, QGIS-DE

[astrid.emde@wherogroup.com](mailto:astrid.emde@wherogroup.com)



40+ Mitarbeiter an 3 Standorten in Bonn, Berlin u.  
Freiburg - Projektleiter:innen, Consultants,  
Trainer:innen, Softwareentwickler:innen

Dienstleister in den Bereichen WebGIS, GDI, Kataster,  
Datenbanken mit freier Software

Schulungen, Infoveranstaltungen, Konferenzen

Unterstützer von OSGeo und FOSSGIS

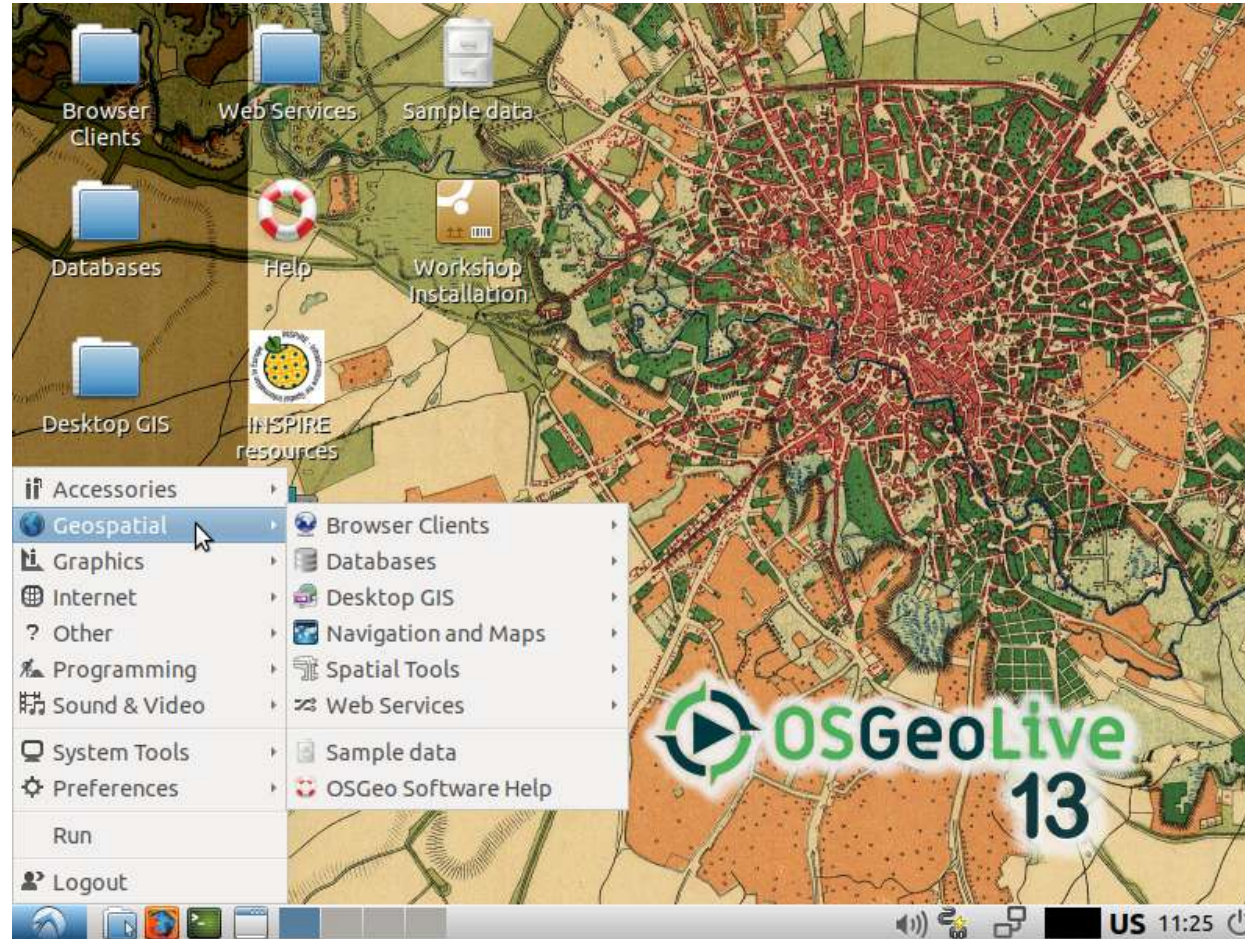
<https://wherogroup.com>

<https://fossacademy.com>



# OSGeoLive

*Einfach mit OSGeoLive ausprobieren*



<https://live.osgeo.org> (Download Version 14)

# Vector Tiles allgemein

# Was sind Vector Tiles

- ▶ Quadratische Kacheln
- ▶ MVT Standard <https://docs.mapbox.com/vector-tiles/specification/>
- ▶ Auslieferung in EPSG:3857 - WGS 84/Pseudo-Mercator
- ▶ Feature werden als Vektoren .pbf an den Client übergeben
- ▶ Client übernimmt das Rendering
- ▶ [http://127.0.0.1:7800/public.buildings\\_a/{z}/{x}/{y}.pbf](http://127.0.0.1:7800/public.buildings_a/{z}/{x}/{y}.pbf)



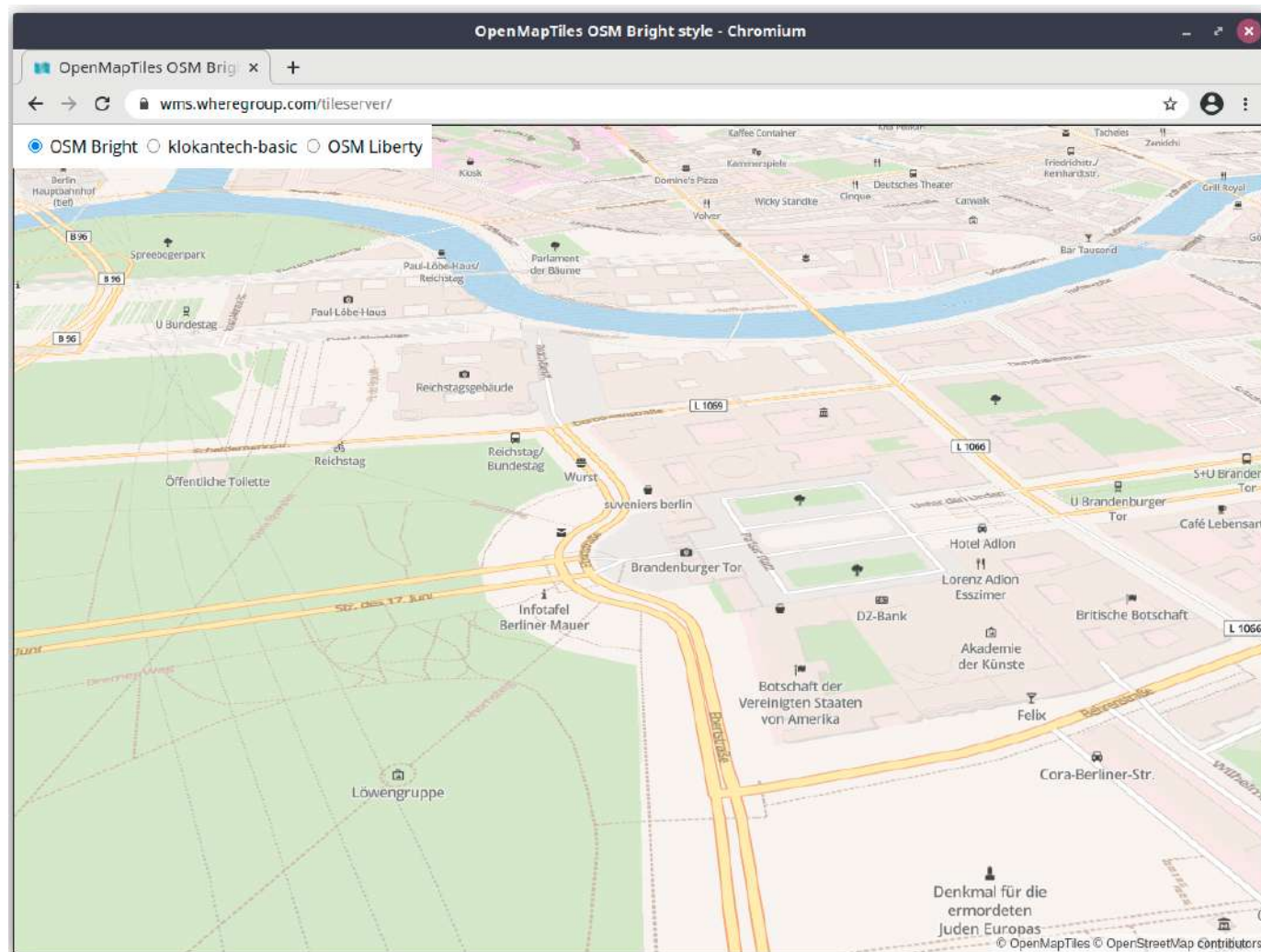
## Warum Vector Tiles?

- ▶ Schnell, wenig Speicher, weniger Bandbreite
- ▶ Attraktiv für den mobilen Einsatz
- ▶ Gestaltung im Client ist möglich
- ▶ Verschiedene Styles können leicht angewendet werden
- ▶ Drehen
- ▶ Wechsel der Sprache
- ▶ Darstellung von Höhen
- ▶ Filtern im Client
- ▶ Attributinformation liegt im Client vor
- ▶ Offline nutzbar

# Anwendungen mit Vector Tiles

## WhereGroup TileServer

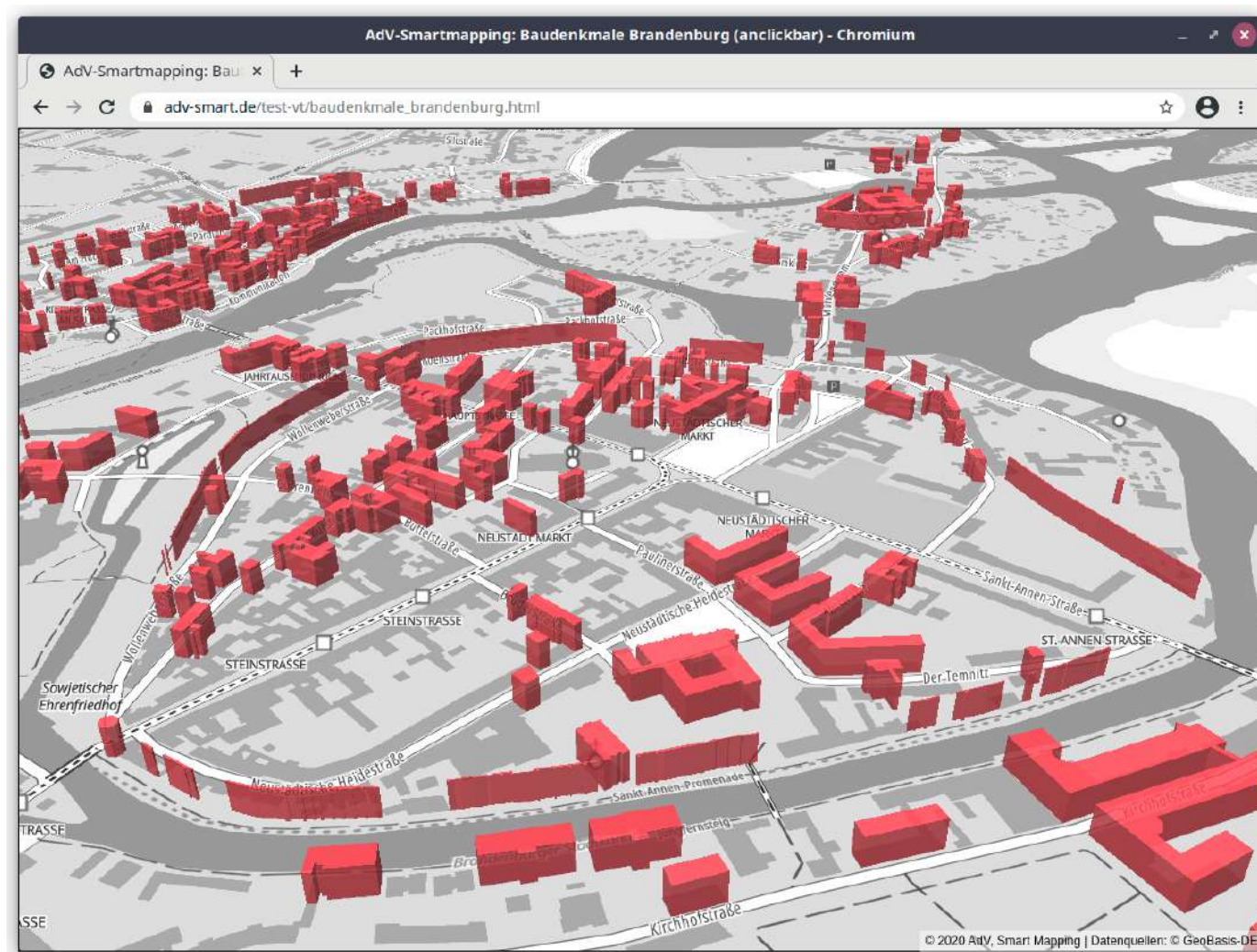
<https://wms.wherogroup.com/tileservet/>





## *AdV Smart Mapping (03/2020 veröffentlicht)*

<https://adv-smart.de/>



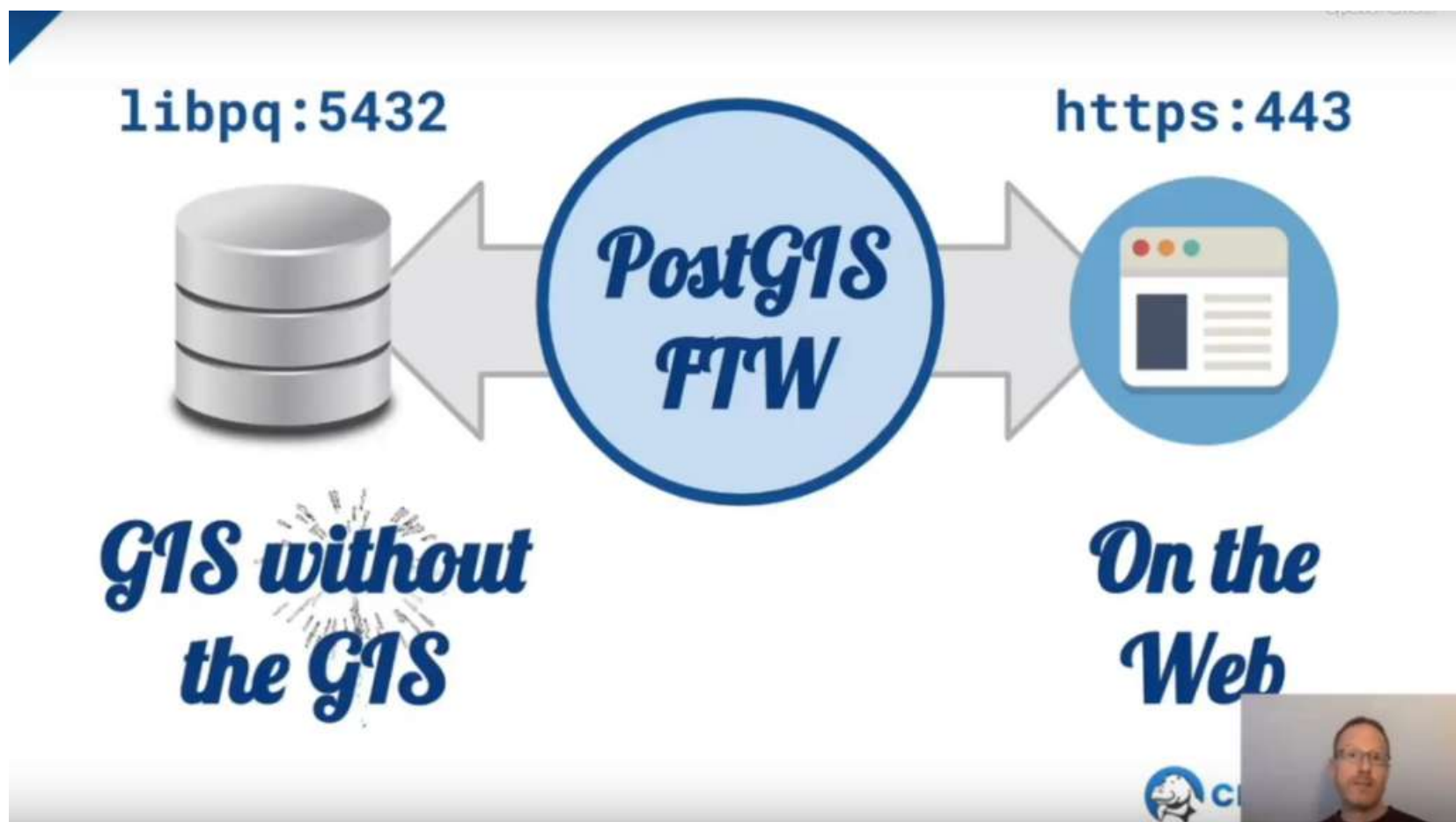
## Clients für Vector Tiles

- ▶ OpenLayers
- ▶ Leaflet über Plugins
- ▶ MapLibre - Fork von MapBox GL 1.13.x
- ▶ QGIS (ab Version 3.14)
- ▶ MapServer (ab 7.6)
- ▶ GDAL ab 2.3
- ▶ & ....

## Server für Vector Tiles

- ▶ GDAL ab 2.3
- ▶ MapServer ab Version 7.2
- ▶ GeoServer über Vector Tiles Extension
- ▶ t-rex
- ▶ pg\_tileserv (V0.1 01/2020)
- ▶ & ....

# PostGIS FTW



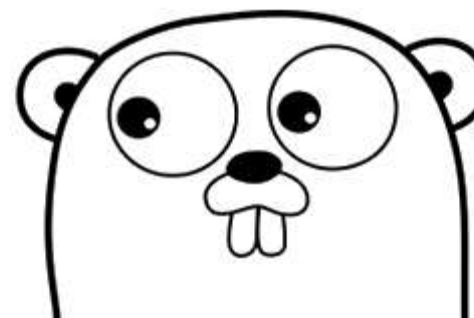
@ Paul Ramsey PostGIS Day 2020

# pg\_tileserv

"PostGIS only Tile Server in Go"



Paul Ramsey  
Firma Crunchy Data



Go Project

# pg\_tileserv

## "PostGIS only Tile Server in Go"

- ▶ Vector Tile Server
- ▶ GIS wird nicht benötigt!
- ▶ Leichtgewichtig & schnell
- ▶ Zustandslos
- ▶ http-Zugriff auf die PostgreSQL-Datenbank
- ▶ Zwischenschicht zwischen Datenbank & Web Mapping Applikation
- ▶ Aktuelle Version v1.0.5 12/2020



# pg\_tileserv

## PostGIS ony Tile Server in Go

- ▶ PostGIS Mapbox Vector Tiles über Funktion `ST_AsMVT` seit Version 2.4 (09/2017)
- ▶ Erste Lösung von Paul Ramsey in Python entwickelt als Proof of Concept
- ▶ Erste Version von pg\_tileserv v0.1 (01/2020)
- ▶ Sehr gute Dokumentation [https://access.crunchydata.com/documentation/pg\\_tileserv/latest/](https://access.crunchydata.com/documentation/pg_tileserv/latest/)
- ▶ Gehört zur Reihe "PostGIS for the Web" - kurz "PostGIS FTW"  
(nicht zu verwechseln mit PostgreSQL FDW)
- ▶ Voraussetzung PostgreSQL  $\geq 9.5$  PostGIS  $\geq 2.4$



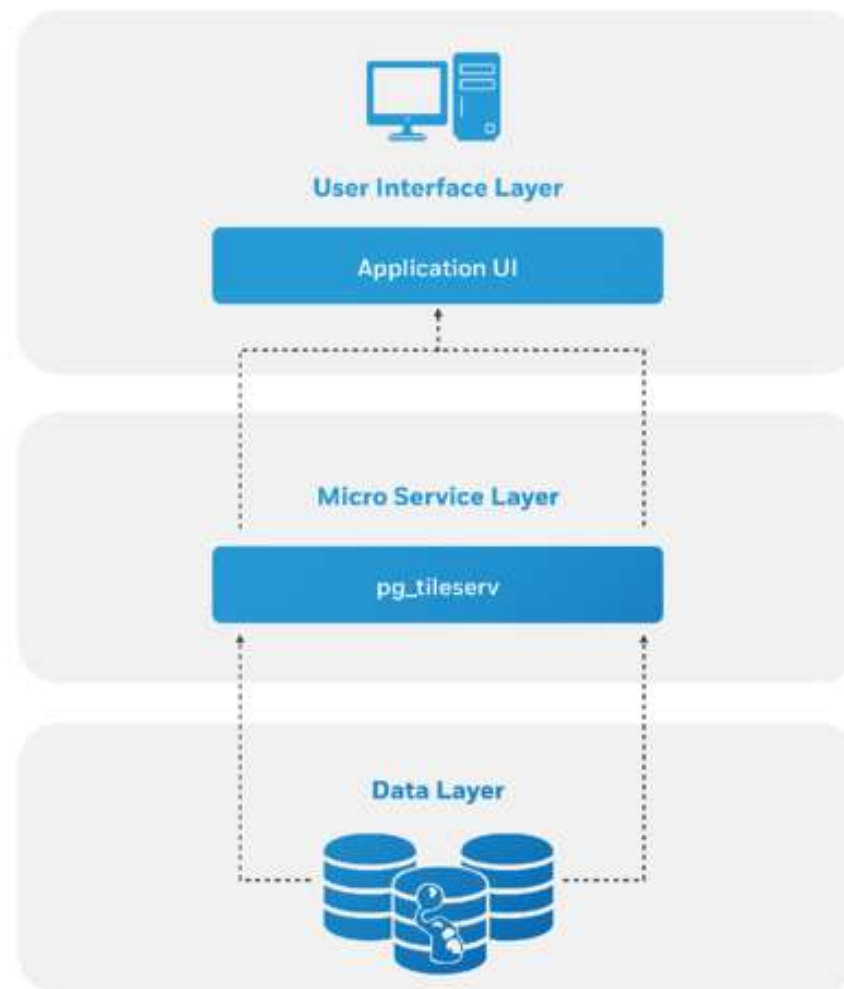
## pg\_tileserv

- ▶ Automatische Konfiguration über Datenbankberechtigungen
- ▶ Einbindung von Funktions-Layern
- ▶ Client zur Datenvisualisierung
- ▶ Metadaten - menschen- und maschinenlesbar

# pg\_tileserv

## Installation

- ▶ Leicht zu installieren
- ▶ Downloadpaket für Windows, Linux, OSX, Docker
- ▶ Ablage im beliebigen Verzeichnis
- ▶ Nutzbar in verschiedenen Architekturen:  
Desktop - Container - Cloud Kubernetes



@ pg\_tileserv Dokumentation

# pg\_tileserv

## Start des Servers

```
export DATABASE_URL=postgresql://username:password@host/dbname  
./pg_tileserv
```

- ▶ Erreichbar über <http://localhost:7800/>

# pg\_tileserv

## Start mehrerer Server parallel

- Konfiguration über toml-Datei - Dokumentation

```
./pg_tileserv --config /opt/pg_tileserv/pg_tileserv_freiburg.toml
```

# pg\_tileserv

## toml-Datei zur Konfiguration

```
# pg_tileserv

# Database connection
# postgresql://username:password@host/dbname
DbConnection = "postgresql://tileserv_reader:abcdefg@localhost/freiburg"

# Close pooled connections after this interval
# 1d, 1h, 1m, 1s, see https://golang.org/pkg/time/#ParseDuration
# DbPoolMaxConnLifeTime = "1h"

# Hold no more than this number of connections in the database pool
# DbPoolMaxConns = 4

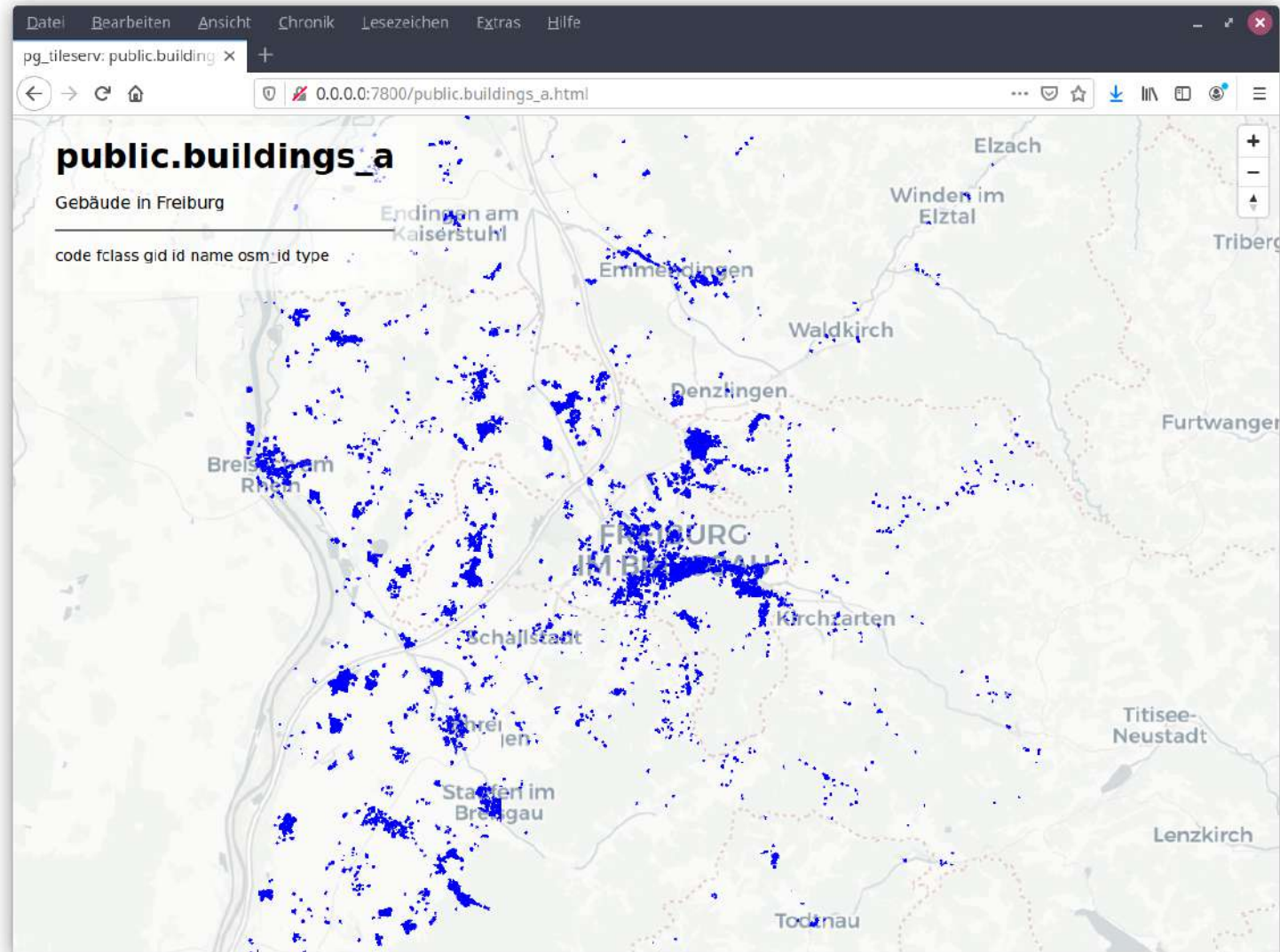
# Cancel a tile request if a tile can't be rendered in this time (seconds)
# DbTimeout = 10

# Look to read html templates from this directory
# AssetsPath = "/usr/share/pg_tileserv/assets"
```



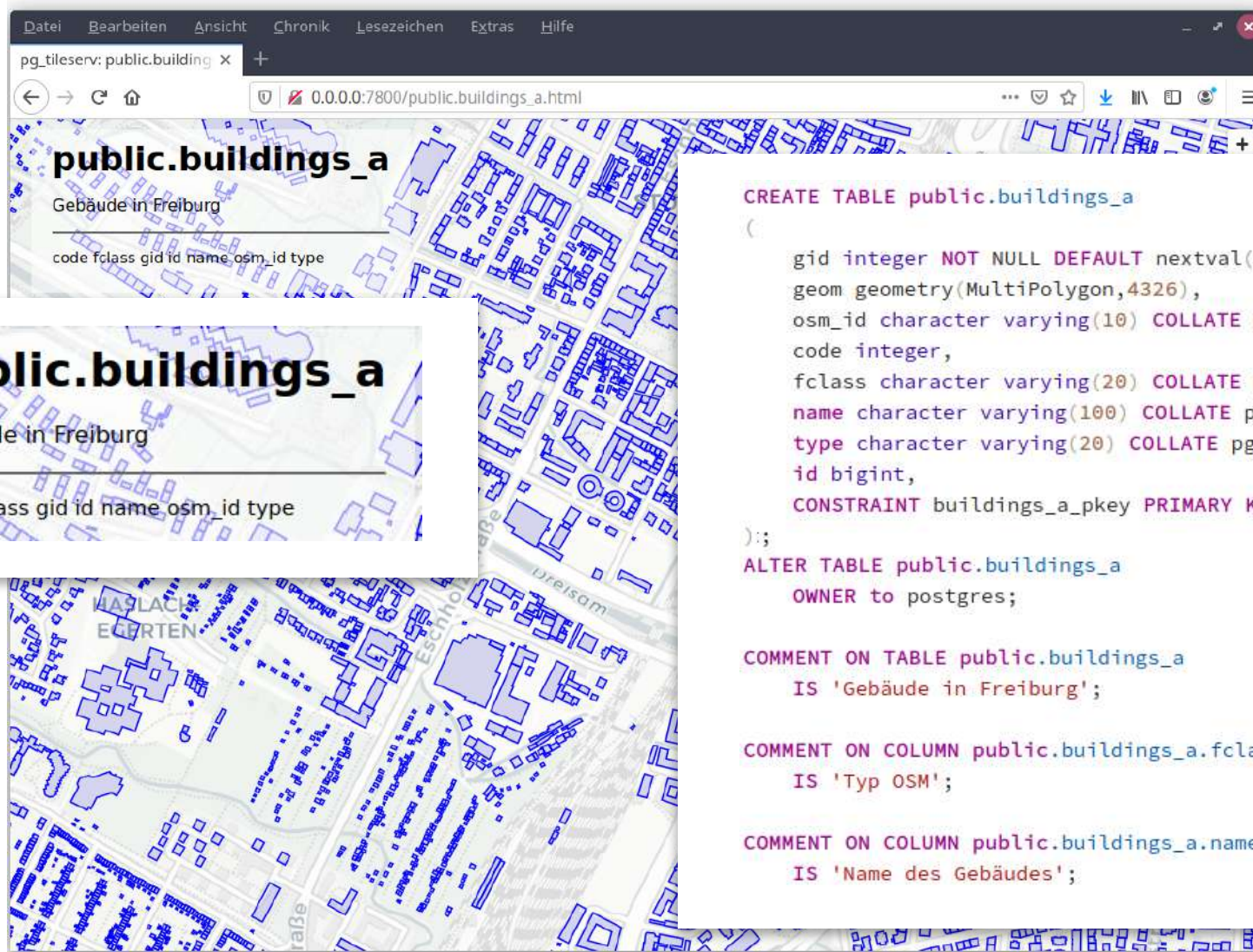
# pg\_tileserv

## Client zur Datenvisualisierung



# pg\_tileserv

## Anzeige Name | Beschreibung | Attribute



**public.buildings\_a**  
Gebäude in Freiburg

code	fclass	gid	id	name	osm_id	type
------	--------	-----	----	------	--------	------

```
CREATE TABLE public.buildings_a
(
    gid integer NOT NULL DEFAULT nextval('buildings_a_gid_seq'::regclass),
    geom geometry(MultiPolygon,4326),
    osm_id character varying(10) COLLATE pg_catalog."default",
    code integer,
    fclass character varying(20) COLLATE pg_catalog."default",
    name character varying(100) COLLATE pg_catalog."default",
    type character varying(20) COLLATE pg_catalog."default",
    id bigint,
    CONSTRAINT buildings_a_pkey PRIMARY KEY (gid)
);
ALTER TABLE public.buildings_a
    OWNER to postgres;

COMMENT ON TABLE public.buildings_a
    IS 'Gebäude in Freiburg';

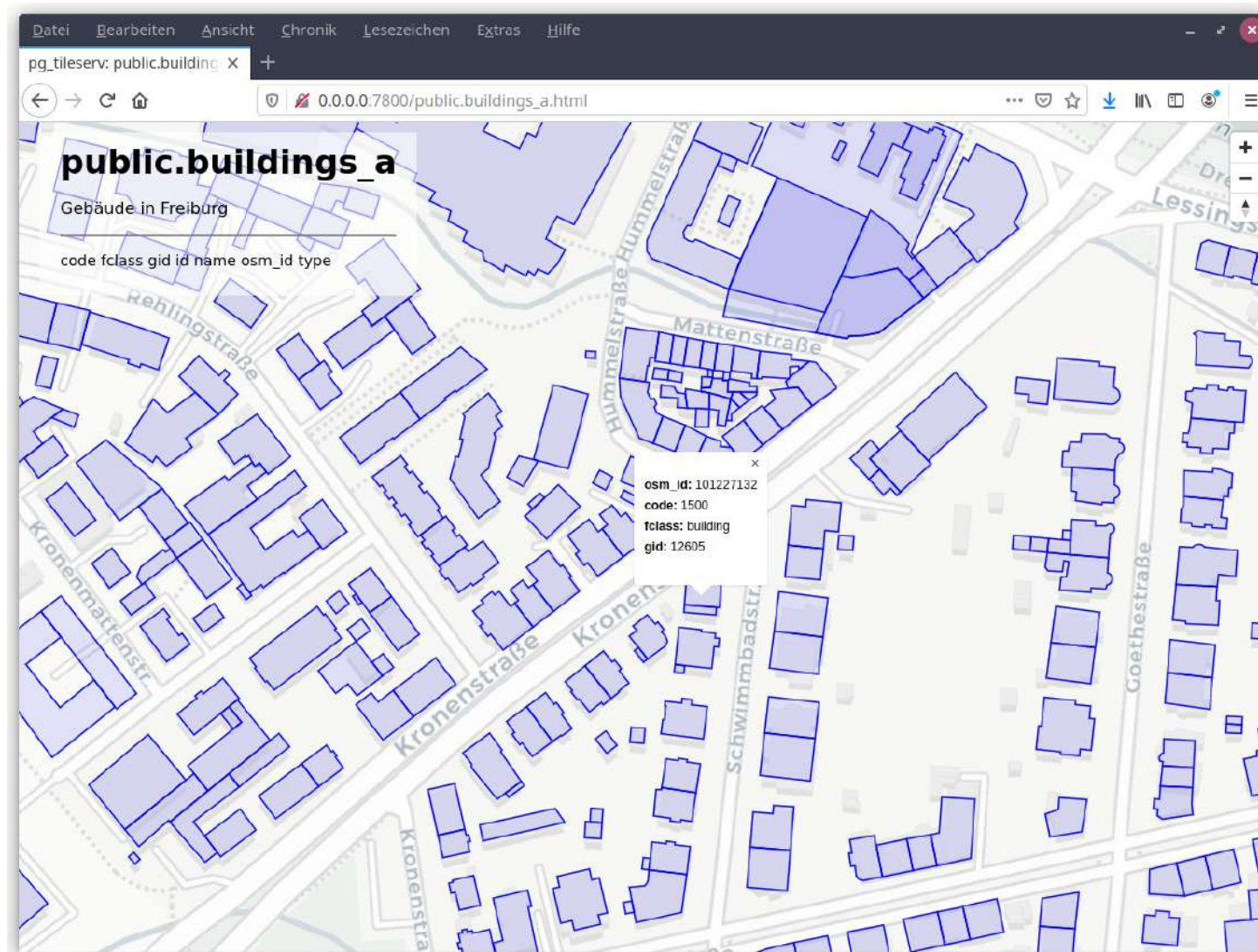
COMMENT ON COLUMN public.buildings_a.fclass
    IS 'Typ OSM';

COMMENT ON COLUMN public.buildings_a.name
    IS 'Name des Gebäudes';
```



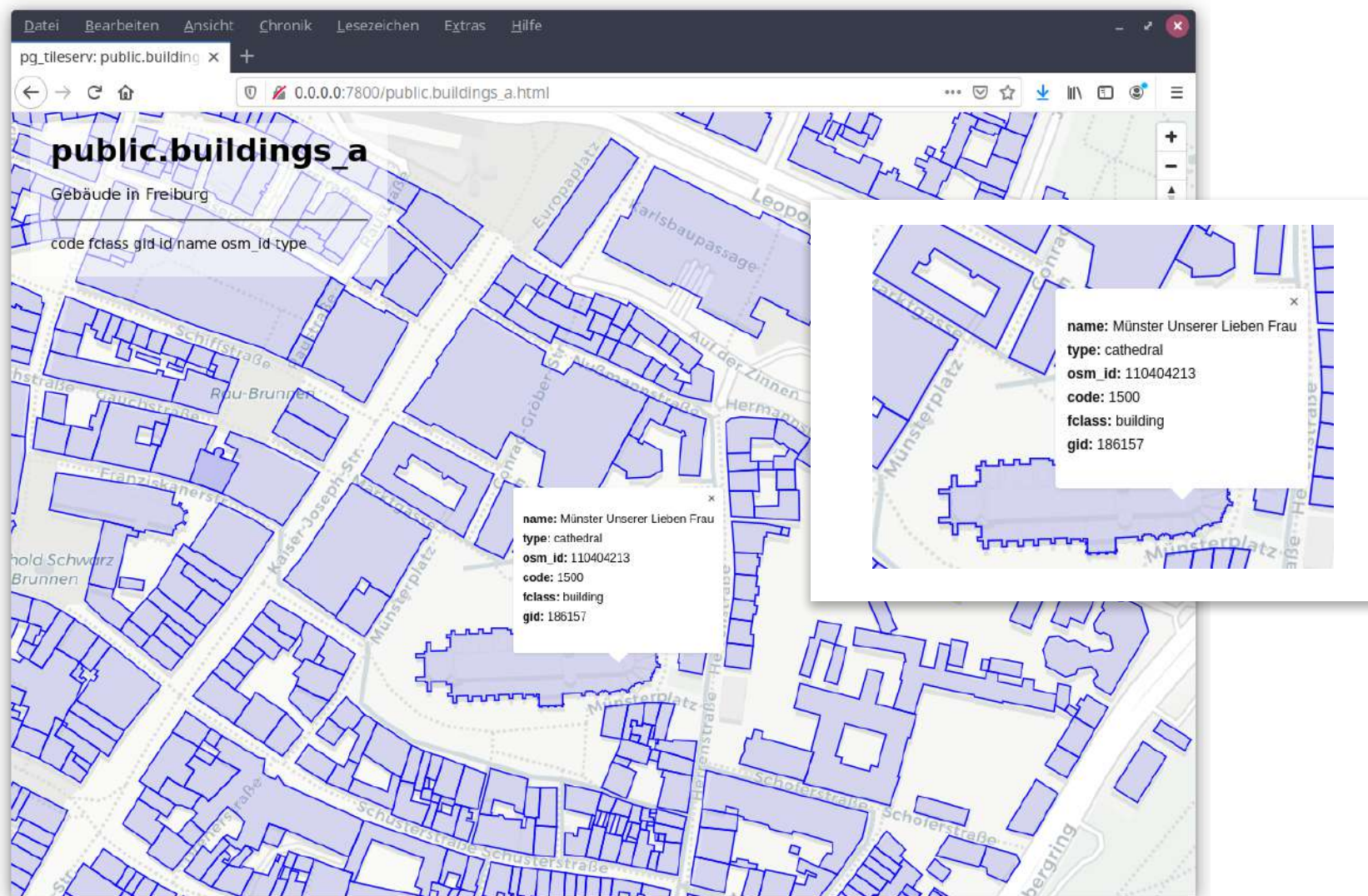
# pg\_tileserv

Anzeige Name | Beschreibung | Attribute



# pg\_tileserv

Anzeige Name | Beschreibung | Attribute

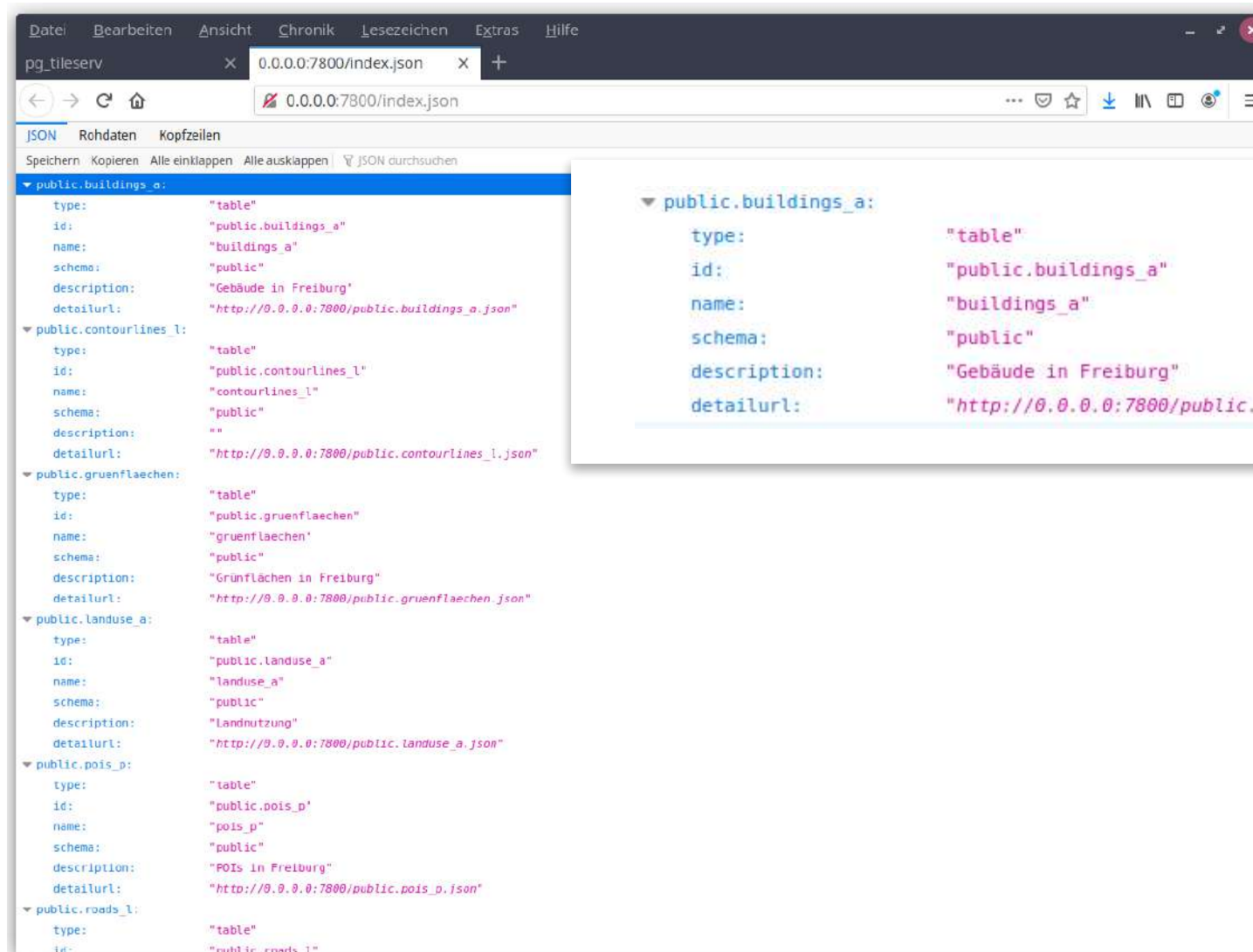


**Das war menschenlesbar.  
Nun kommt maschinenlesbar.**



## Übersicht

<http://localhost:7800/index.json>



The screenshot shows a web browser window with the address bar displaying `0.0.0.0:7800/index.json`. The page content is a JSON file named `index.json` with a tree view. The first entry, `public.buildings_a`, is highlighted. A callout box shows the details of this entry:

```

{
  "type": "table",
  "id": "public.buildings_a",
  "name": "buildings_a",
  "schema": "public",
  "description": "Gebäude in Freiburg",
  "detailurl": "http://0.0.0.0:7800/public.buildings_a.json"
}

```

The full JSON structure visible in the browser is as follows:

```

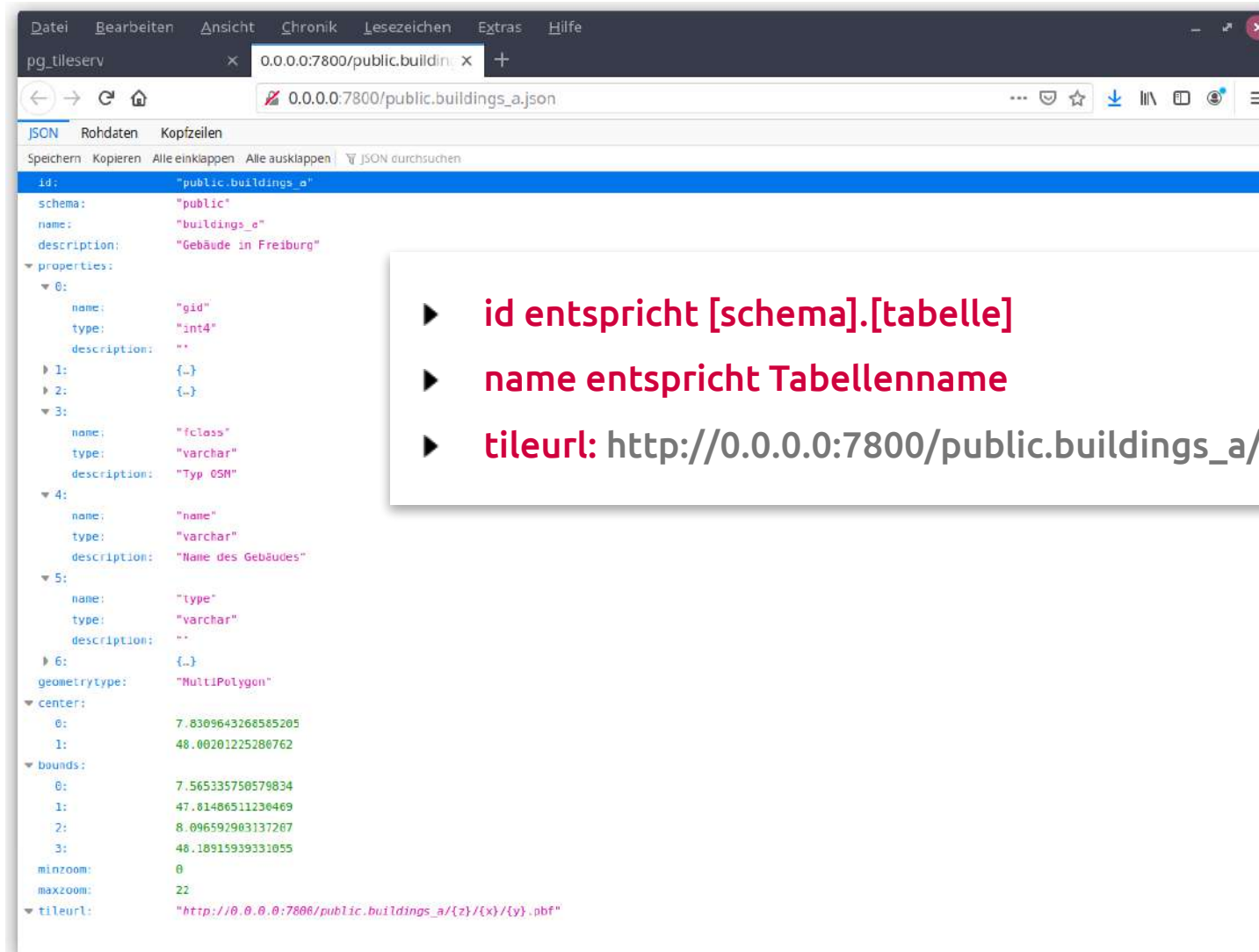
{
  "public.buildings_a": {
    "type": "table",
    "id": "public.buildings_a",
    "name": "buildings_a",
    "schema": "public",
    "description": "Gebäude in Freiburg",
    "detailurl": "http://0.0.0.0:7800/public.buildings_a.json"
  },
  "public.contourlines_l": {
    "type": "table",
    "id": "public.contourlines_l",
    "name": "contourlines_l",
    "schema": "public",
    "description": "",
    "detailurl": "http://0.0.0.0:7800/public.contourlines_l.json"
  },
  "public.gruenflaechen": {
    "type": "table",
    "id": "public.gruenflaechen",
    "name": "gruenflaechen",
    "schema": "public",
    "description": "Grünflächen in Freiburg",
    "detailurl": "http://0.0.0.0:7800/public.gruenflaechen.json"
  },
  "public.landuse_a": {
    "type": "table",
    "id": "public.landuse_a",
    "name": "landuse_a",
    "schema": "public",
    "description": "Landnutzung",
    "detailurl": "http://0.0.0.0:7800/public.landuse_a.json"
  },
  "public.pois_p": {
    "type": "table",
    "id": "public.pois_p",
    "name": "pois_p",
    "schema": "public",
    "description": "POIs in Freiburg",
    "detailurl": "http://0.0.0.0:7800/public.pois_p.json"
  },
  "public.roads_l": {
    "type": "table",
    "id": "public.roads_l",
    "name": "roads_l",
    "schema": "public",
    "description": "",
    "detailurl": "http://0.0.0.0:7800/public.roads_l.json"
  }
}

```



## Details

"detailurl" : [http://0.0.0.0:7800/public.buildings\\_a.json](http://0.0.0.0:7800/public.buildings_a.json)



- ▶ **id** entspricht [schema].[tabelle]
- ▶ **name** entspricht Tabellenname
- ▶ **tileurl**: [http://0.0.0.0:7800/public.buildings\\_a/{z}/{x}/{y}.pbf](http://0.0.0.0:7800/public.buildings_a/{z}/{x}/{y}.pbf)

# pg\_tileserv

## Filter können im Aufruf übergeben werden

- ▶ [https://access.crunchydata.com/documentation/pg\\_tileserv/1.0.3/usage/table-layers/](https://access.crunchydata.com/documentation/pg_tileserv/1.0.3/usage/table-layers/)
- ▶ Defaultwerte überschreiben (Defaults siehe auch poml-Datei)
- ▶ [http://localhost:7800/public.ne\\_50m\\_admin\\_0\\_countries/{z}/{x}/{y}.pbf?  
limit=100000&properties=name,long\\_name](http://localhost:7800/public.ne_50m_admin_0_countries/{z}/{x}/{y}.pbf?limit=100000&properties=name,long_name)

```
?properties=attr1,attr2  
?limit=1000  
?resolution=4096  
?buffer=256
```

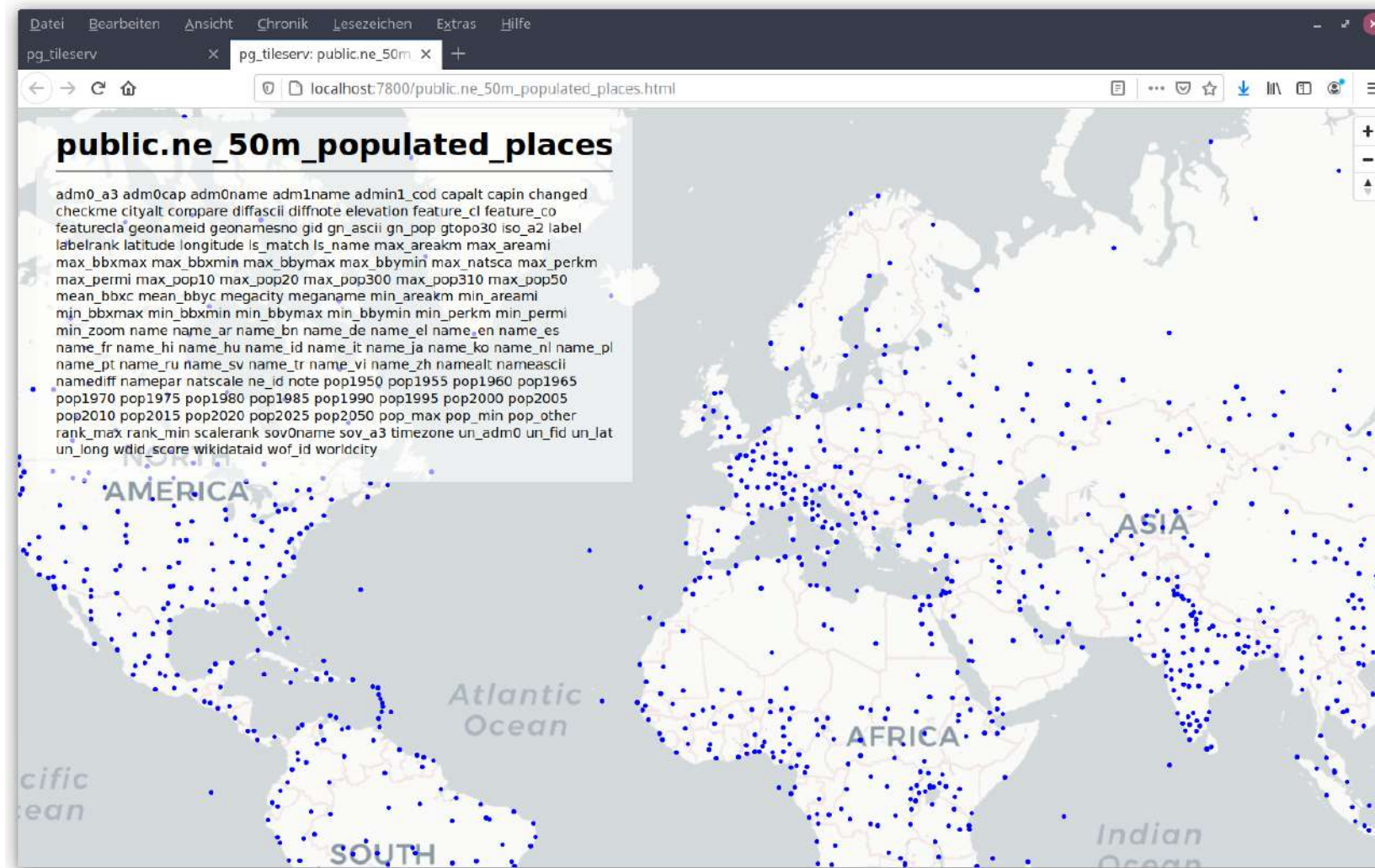
# pg\_tileserv

## Nutzung von Funktionen als Layer

- ▶ Sehr mächtig
- ▶ Versteckt die Datenquelle vor dem Server
- ▶ Optionaler Parameter kann im Aufruf übergeben werden
- ▶ Funktion kann in jedem Schema sein, muss aber MVT bytea zurückgeben
- ▶ Tiles werden in der Funktion selbst erstellt und zurückgegeben
- ▶ Dokumentation `pg_tileserv function layer`
- ▶ FedGeo Day 2020: `pg_tileserv` / `pg_featureserv` mit Adam Timm

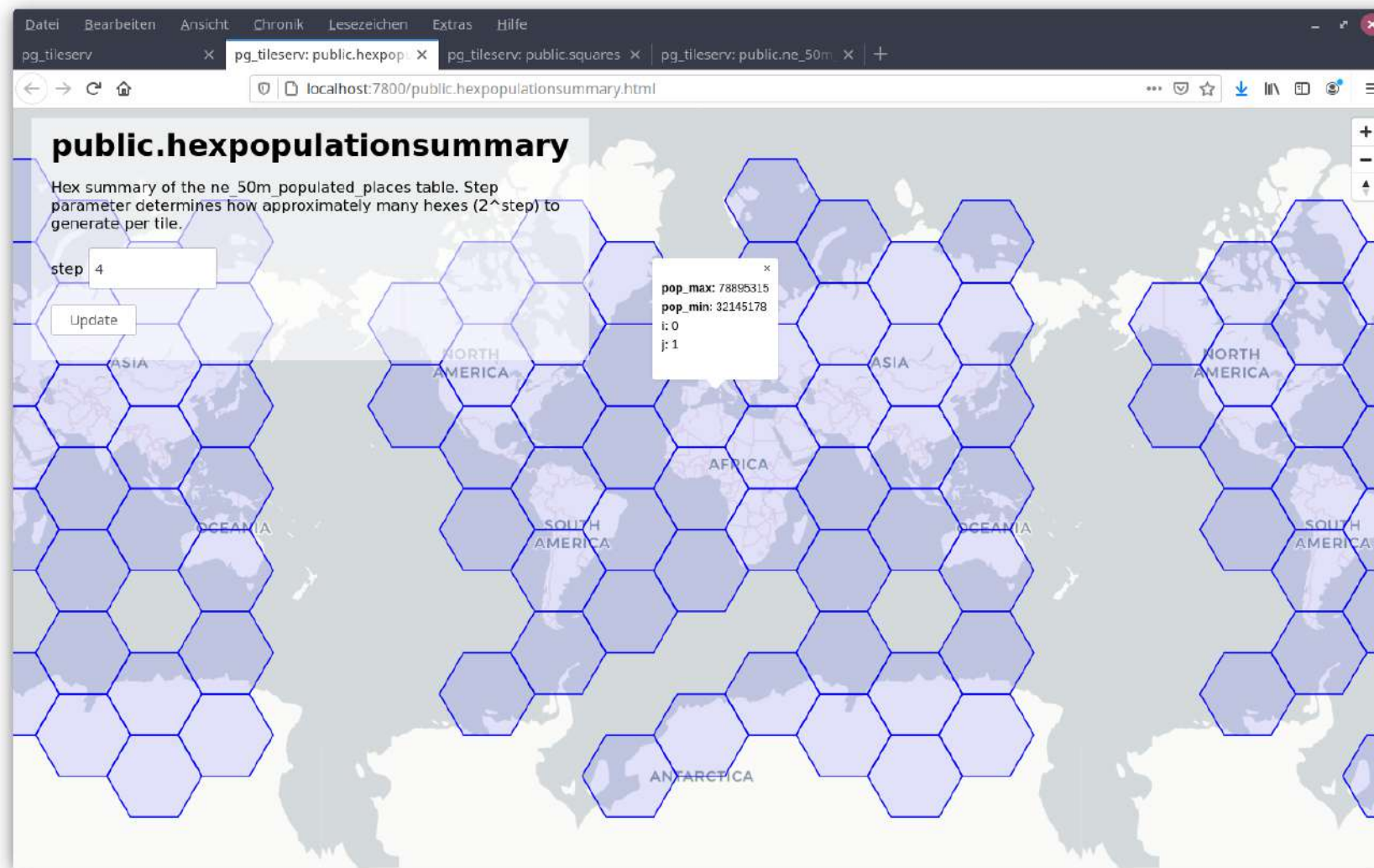
## Nutzung von Funktionen als Layer

pg\_tileserv - Advanced Function Layer



## Nutzung von Funktionen als Layer

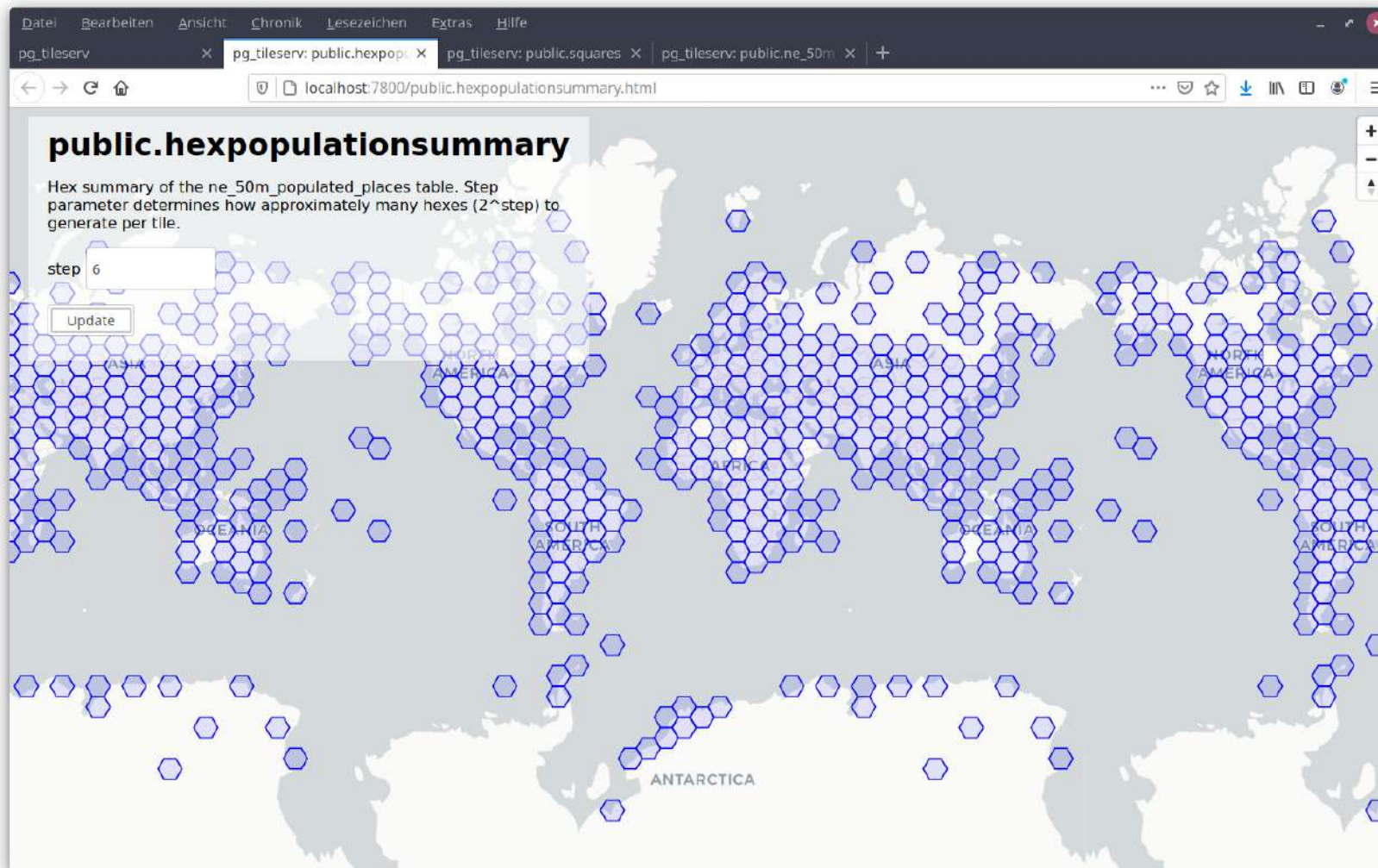
pg\_tileserv - Advanced Function Layer





## Nutzung von Funktionen als Layer

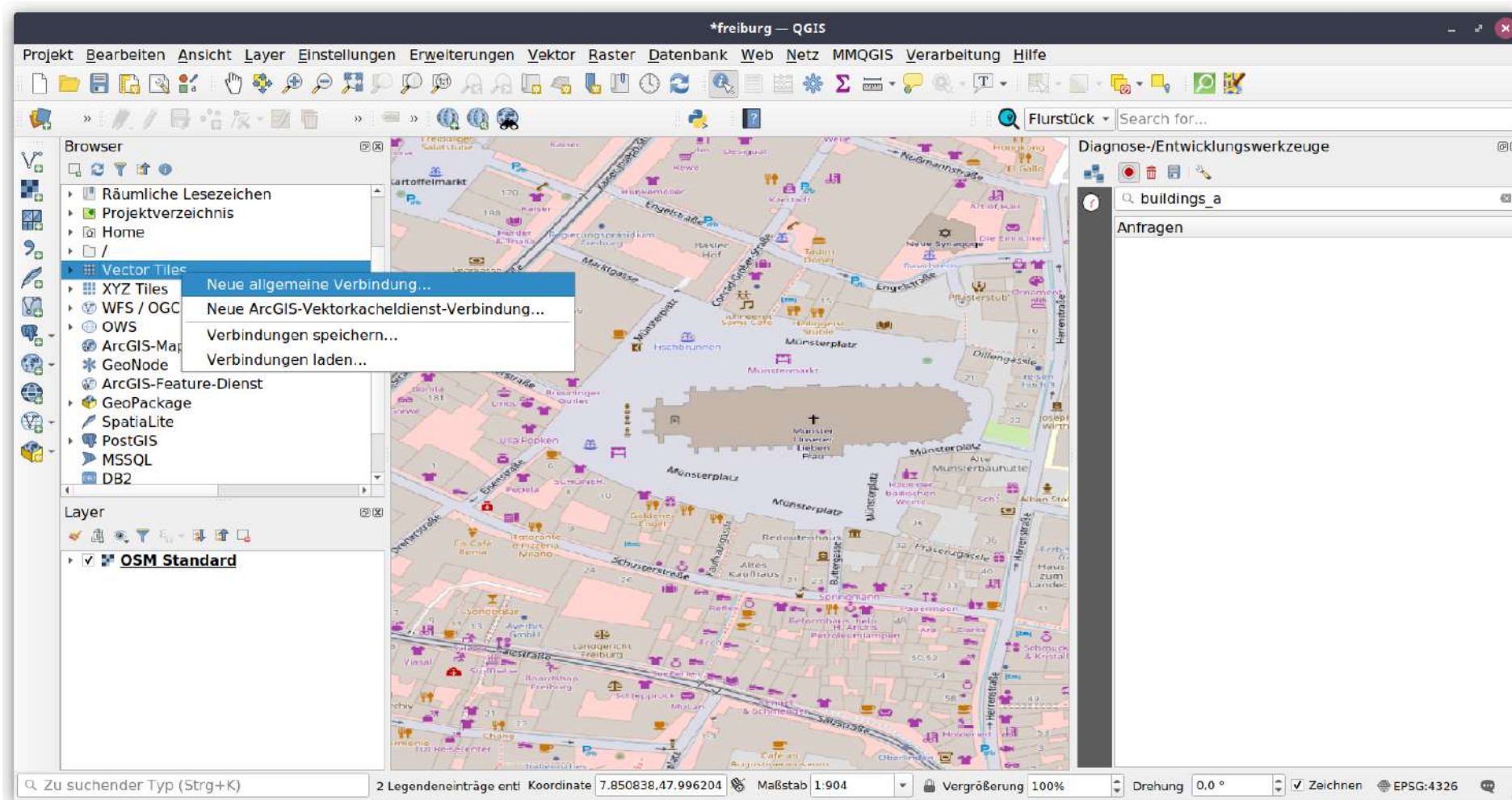
pg\_tileserv - Advanced Function Layer





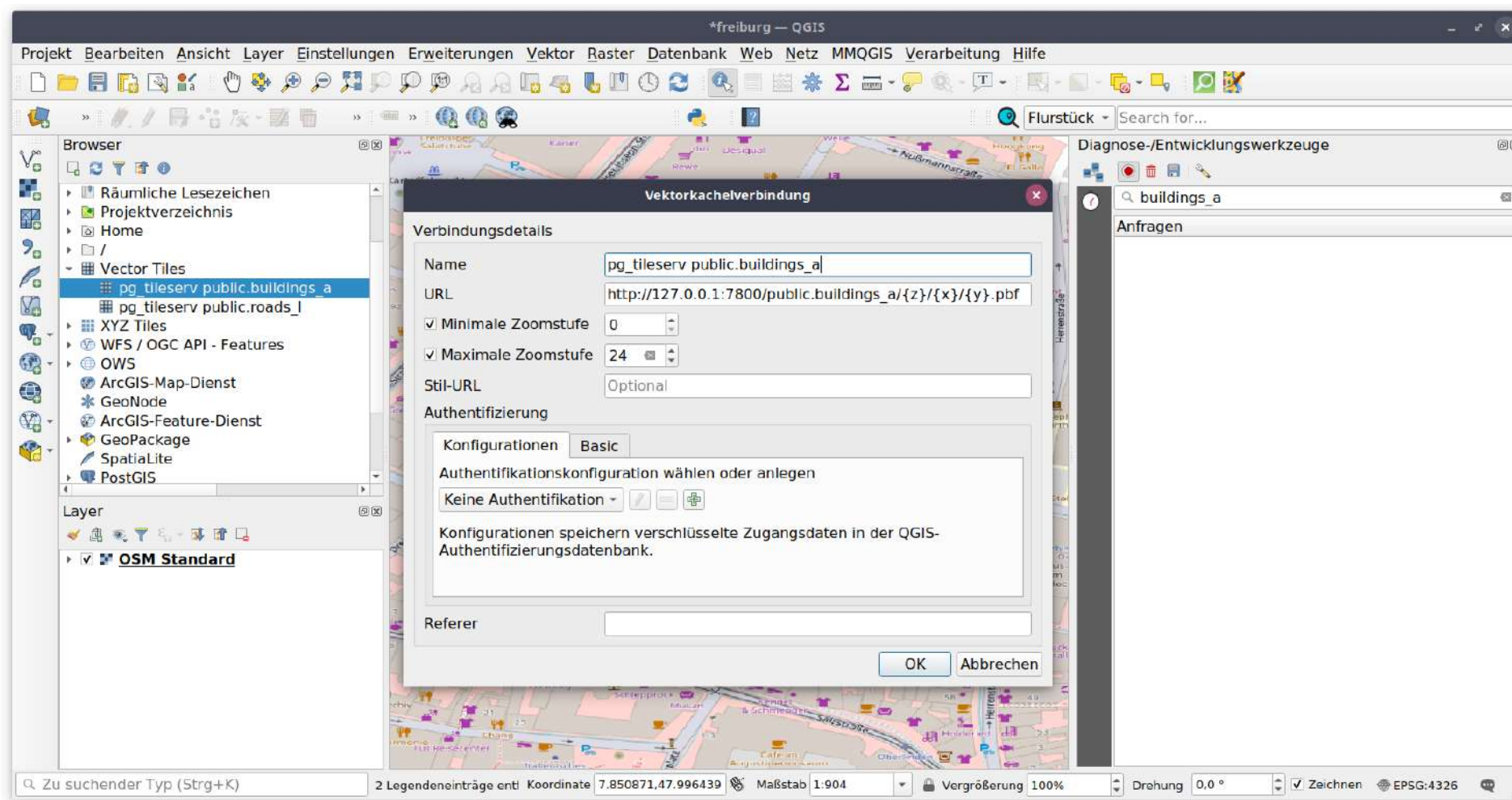
# QGIS & pg\_tileserv

## Vector Tiles über den Browser



# QGIS & pg\_tileserv

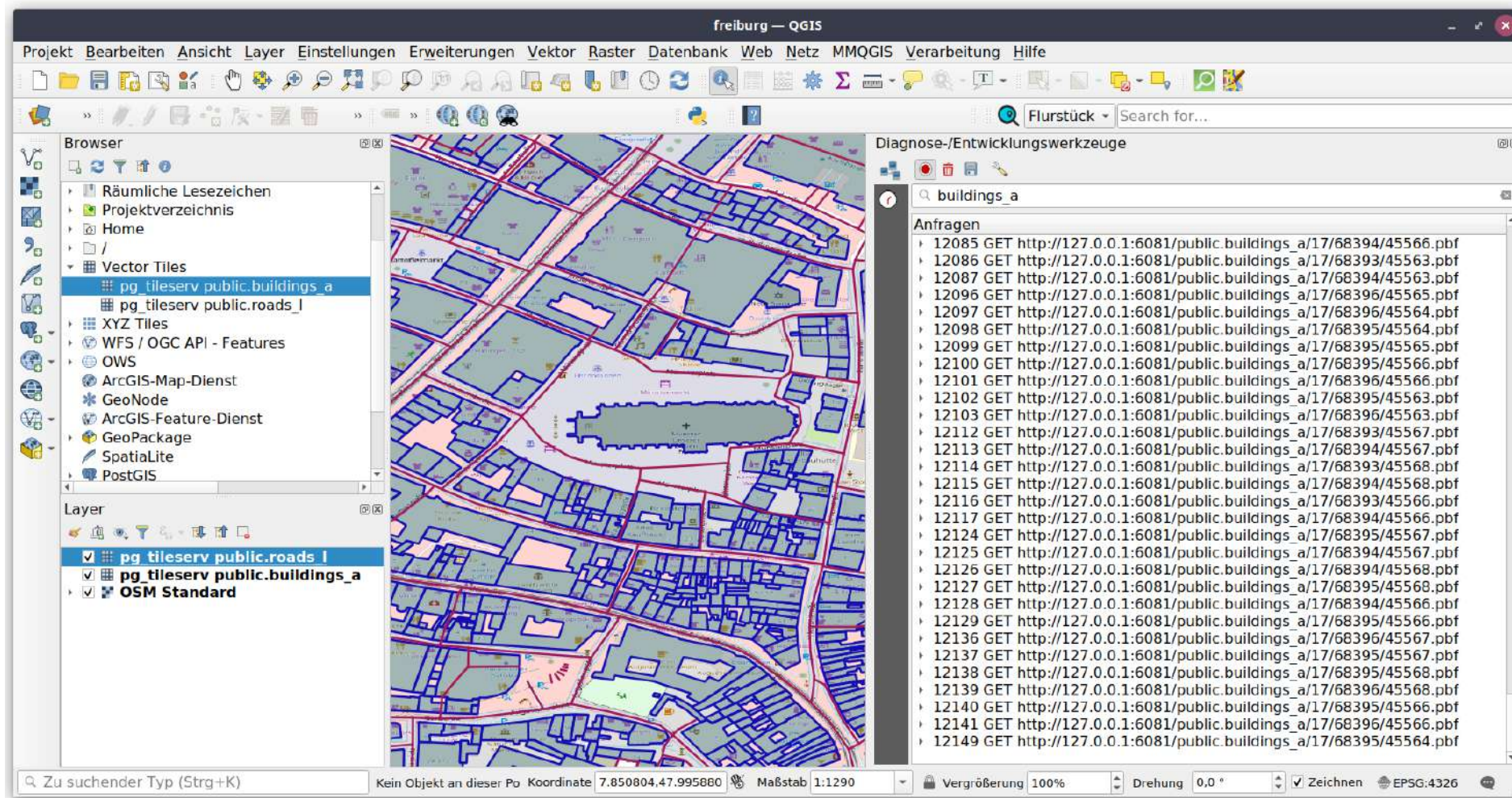
## Vector Tiles über den Browser





# QGIS & pg\_tileserv

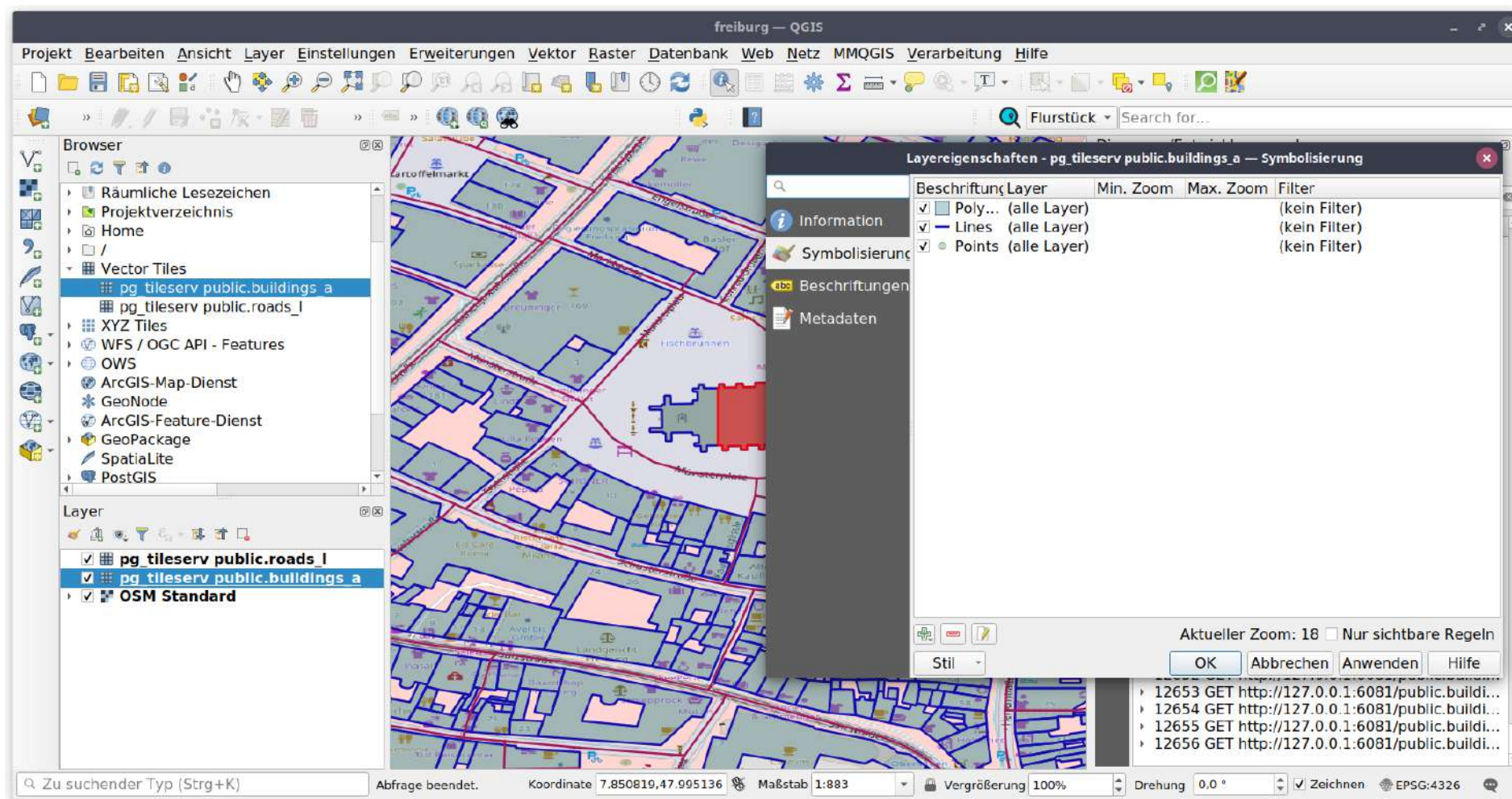
## F12 - Diagnose-/Entwicklerwerkzeuge





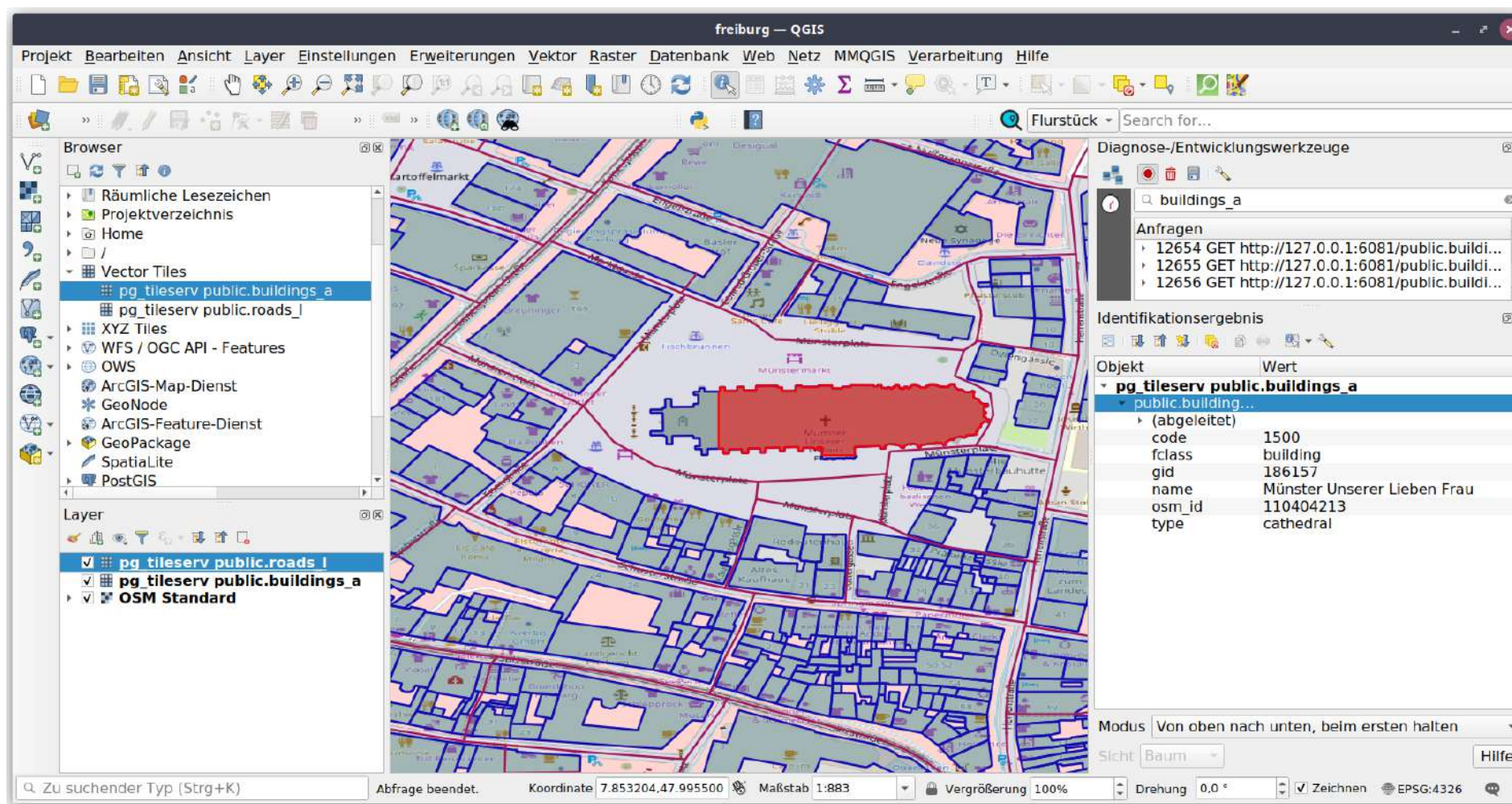
# QGIS & pg\_tileserv

## Stilanpassungen



# QGIS & pg\_tileserv

## Zugriff auf die Attribute



The screenshot shows the QGIS interface with the title 'freiburg — QGIS'. The main map area displays a street map of Freiburg, Germany, with a large red building highlighted. The left sidebar contains a 'Browser' panel with a tree view showing various data sources. Under 'Vector Tiles', 'pg\_tileserv public.buildings\_a' is selected. Below it, 'pg\_tileserv public.roads\_1' and 'OSM Standard' are also listed. The 'Layer' panel at the bottom left shows 'pg\_tileserv public.roads\_1', 'pg\_tileserv public.buildings\_a', and 'OSM Standard' as active layers. The right sidebar features a 'Diagnose-/Entwicklungswerkzeuge' panel. It includes a search bar with 'buildings\_a' entered, a list of requests (Anfragen) showing GET requests to a local server, and an 'Identifikationsergebnis' (Identification result) table. The table lists attributes for the selected building: code (1500), fclass (building), gid (186157), name (Münster Unserer Lieben Frau), osm\_id (110404213), and type (cathedral). At the bottom of the interface, the status bar shows 'Abfrage beendet.', 'Koordinate 7.853204, 47.995500', 'Maßstab 1:883', 'Vergrößerung 100%', 'Drehung 0.0 °', and 'EPSG:4326'.

Objekt	Wert
pg_tileserv public.buildings_a	
public.building...	
(abgeleitet)	
code	1500
fclass	building
gid	186157
name	Münster Unserer Lieben Frau
osm_id	110404213
type	cathedral



# OSM Daten importieren

## Download & Extrakt

```
#osmium 1.11.1
sudo apt install osmium-tool

#Download OSM-Daten
wget https://download.geofabrik.de/europe/germany/baden-wuerttemberg/freiburg-reg

# Extraktion von Freiburg
osmium extract -b 7.565,47.814,8.096,48.189 /data/demo/freiburg-regbez-latest.osm
```

# OSM Daten importieren

## Benutzer osmuser & Datenbank OSM anlegen

```
sudo -u postgres createuser osmuser -U postgres
sudo -u postgres createdb --encoding=UTF8 --owner=osmuser osm
sudo -u postgres psql osm --command='CREATE EXTENSION postgis;'
sudo -u postgres psql osm --command='CREATE EXTENSION hstore;'
```

# OSM Daten importieren

## osm2pgsql (ab Version >= 1.3.0 mit FLEX Output)

<https://osm2pgsql.org/>

Import der Daten

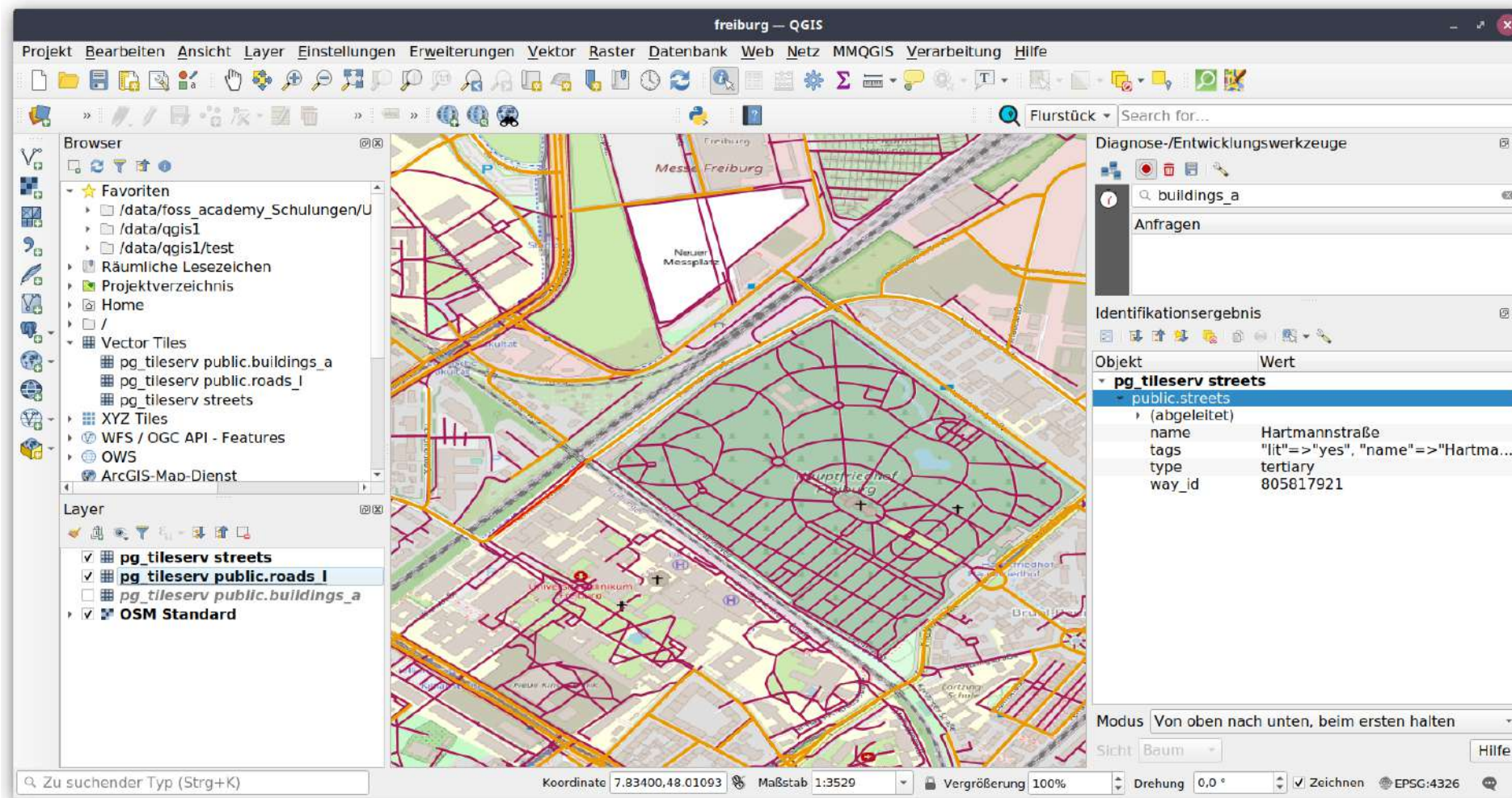
```
cd /data/demo
```

```
osm2pgsql -d osm -U postgres -H localhost -O flex -S streets.lua freiburg.osm.pbf
```

```
2021-01-27 23:53:22 osm2pgsql version 1.4.0 (1.4.0)
2021-01-27 23:53:22 Database version: 12.5 (Ubuntu 12.5-1.pgdg20.04+1)
2021-01-27 23:53:22 Node-cache: cache=800MB, maxblocks=12800*65536, allocation m
Processing: Node(3126k 3126.0k/s) Way(477k 238.50k/s) Relation(10327 10327.0/s)
2021-01-27 23:53:25 Node stats: total(3126931), max(8291671882) in 1s
2021-01-27 23:53:25 Way stats: total(477669), max(892076599) in 2s
2021-01-27 23:53:25 Relation stats: total(10327), max(12133805) in 0s
2021-01-27 23:53:25 No marked ways (Skipping stage 2).
2021-01-27 23:53:25 node cache: stored: 3126931(100.00%), storage efficiency: 49
2021-01-27 23:53:25 Clustering table 'streets' by geometry...
2021-01-27 23:53:26 Creating geometry index on table 'streets'...
2021-01-27 23:53:26 Analyzing table 'streets'...
2021-01-27 23:53:26 All postprocessing on table 'streets' done in 1s.
```

# QGIS & pg\_tileserv

## Zugriff auf die Attribute



## Varnish Cache

- ▶ Caching HTTP Reverse Proxy
- ▶ <https://varnish-cache.org/docs/trunk/index.html>
- ▶ <http://127.0.0.1:6081/>



```
sudo apt-get install varnish
sudo service varnish status

vim /etc/varnish/default.vcl

vcl 4.0;

backend default {
    .host = "127.0.0.1";
    .port = "7800";    # pg_tilserve port
}
```



## Zusammengefasst pg\_tileserv


- ▶ Einfach, flexibel & schnell
- ▶ Leicht installierbar & konfigurierbar
- ▶ Leicht skalierbar

## Links

- ▶ Paul Ramsey PostGIS Day 2020
- ▶ Crunchy Data Blog
- ▶ Blog Paul Ramsey
- ▶ FedGeo Day 2020: pg\_tilserv / pg\_featureserv mit Adam Timm
- ▶ Waiting for PostGIS 3.1 Paul Ramsey

## Fortsetzung folgt voraussichtlich auf der FOSSGIS 2021

### *pg\_featureserv*

- ▶ Serie “**PostGIS for the Web**” (aka “**PostGIS FTW**”)
- ▶ ebenfalls Teil der Crunchy Data Entwicklungen
- ▶ leichtgewichtiger RESTful Geospatial Feature Server
- ▶ Implementiert den OGC API Feature Standard
- ▶ Liefert JSON and GeoJSON 
- ▶ Dokumentation `pg_featureserv`
- ▶ FOSSGIS-Konferenz 2021

## Vielen Dank

Astrid Emde | WhereGroup | [astrid.emde@wherogroup.com](mailto:astrid.emde@wherogroup.com)



FOSDEM 2021 Online 6.-7. Februar 2021

OGC OSGeo ASF Code Sprint 2021 17.-19. Februar 2021

Get your GDAL T-Shirt

Become a Sponsor of OSGeo