

# TRABALHO PRÁTICO

## Sistemas de Informação Geográfica

Patrícia Pereira -22304 e Rui Morão- 20854

*Engenharia da Computação Gráfica e Multimédia*



# Índice

Índice.....	2
Introdução.....	2
Local.....	3
Objetivos .....	3
Modelo de Dados.....	4
Realização .....	9
No QGIS .....	11
Conclusão .....	14
Bibliografia.....	14

## Introdução

Este relatório documenta o desenvolvimento de um projeto na disciplina de Sistemas de Informação Geográfica (SIG), focado na criação de uma aplicação para consultar informações sobre uma área geográfica específica. Na fase inicial, definimos o tema e o modelo de dados, optando pelo PostgreSQL e utilizando o PGAdmin4 como ferramenta de administração. O projeto utiliza o QGIS para criar um mapa interativo da área de residência, abrangendo informações como rede viária, pontos de interesse, habitações, sinais verticais e zonas verdes. Destacaremos as fases de desenvolvimento, desde a definição do modelo de dados até a implementação prática, abordando decisões e desafios enfrentados.

### \*PostgreSQL:

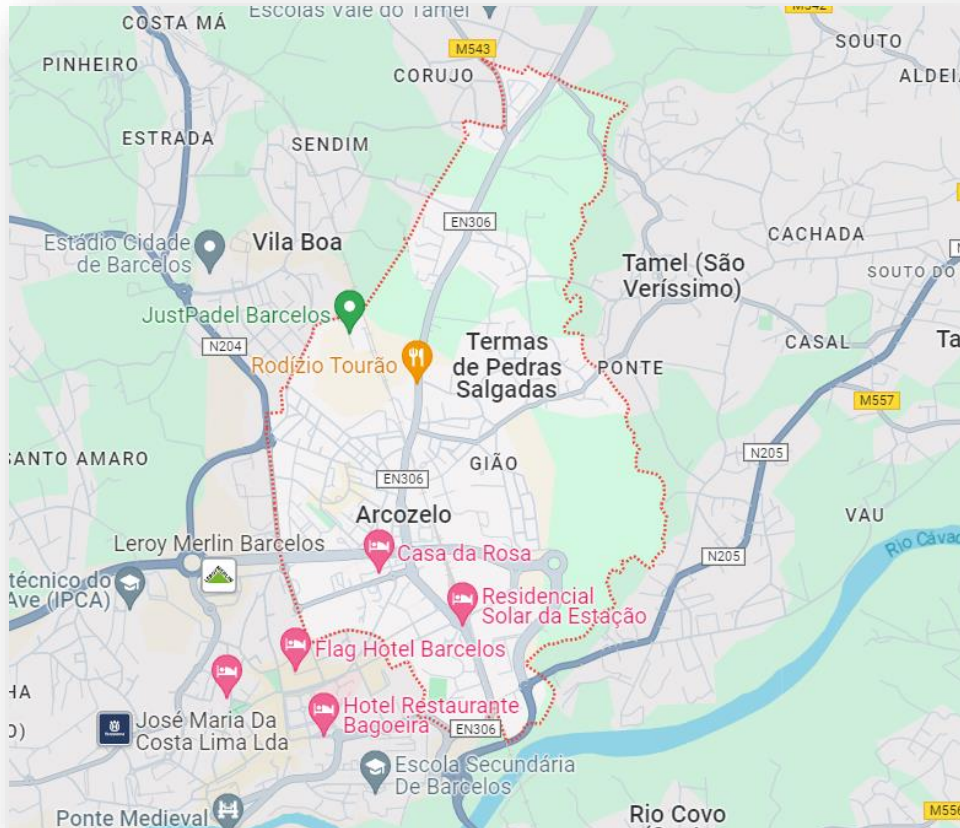
O PostgreSQL é um banco de dados relacional de código aberto, seguro e eficiente, originado no projeto POSTGRES da UC Berkeley em 1986.

### \*QGIS (Sistema de Informação Geográfica Quantum):

O QGIS é um SIG de código aberto para visualização, edição e análise de dados geoespaciais, amplamente usado por profissionais e entusiastas. Versátil e robusto, é valioso para manipular dados geográficos.

## Local

Para este trabalho escolhemos a freguesia de Arcozelo do município de Barcelos, zona de residência do membro do grupo Rui Morão. Nesta região existem alguns pontos de interesse como é o caso da Igreja Velha de São Mamede de Arcozelo e da Igreja das Irmãs Franciscanas Missionárias de Maria, o Estádio Adelino Ribeiro Novo e o Campo de Futebol NC Os Andorinhas de Arcozelo e JustPadel.



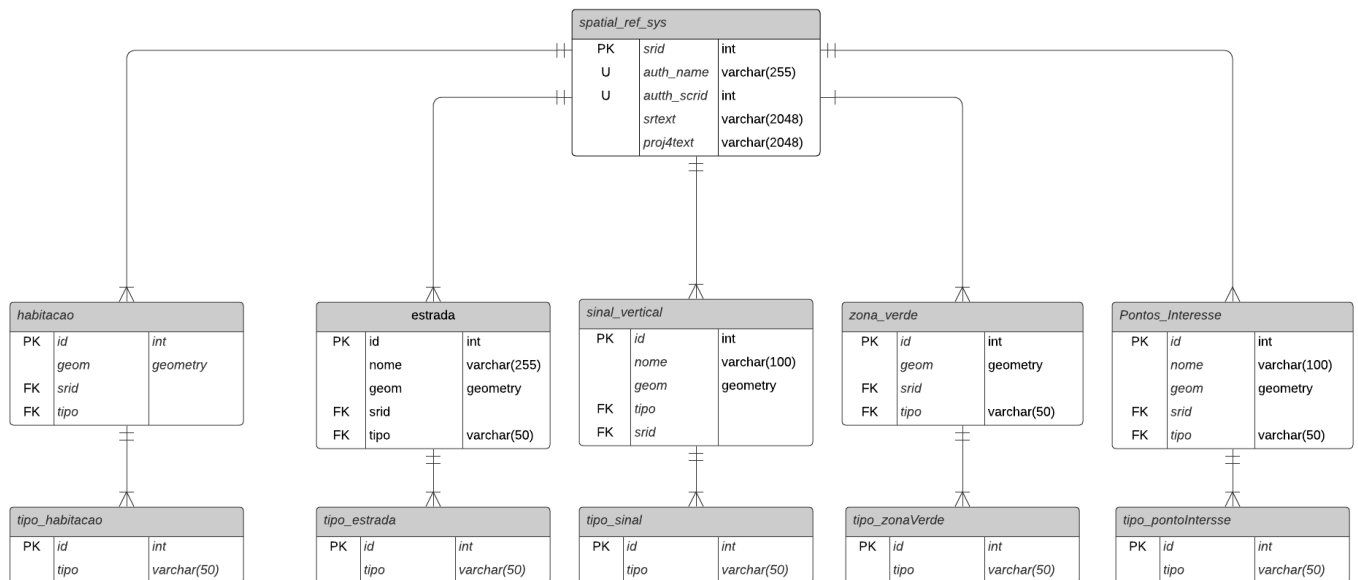
i

## Objetivos

- Criação de uma base de dados geográficos no PostGIS através da execução de scripts;
- Digitalização de dados para uma determinada área geográfica;

# Modelo de Dados

O modelo de Dados foi criado no Lucid App.



```

CREATE TABLE IF NOT EXISTS public.estrada
(
    estradaid integer NOT NULL DEFAULT nextval('estrada_estradaid_seq'::regclass),
    nome character varying(100) COLLATE pg_catalog."default",
    tipoestradaid integer,
    geom geometry(LineString) NOT NULL,
    srid integer,
    CONSTRAINT estrada_pkey PRIMARY KEY (estradaid),
    CONSTRAINT estrada_srid_fkey FOREIGN KEY (srid)
        REFERENCES public.spatial_ref_sys (srid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT estrada_tipoestradaid_fkey FOREIGN KEY (tipoestradaid)
        REFERENCES public.tipoestrada (tipoestradaid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.estrada
    OWNER to postgres;
    
```

```

CREATE TABLE IF NOT EXISTS public.habitacao
(
    habitacaooid integer NOT NULL DEFAULT nextval('habitacao_habitacaooid_seq'::regclass),
    tipohabitacaooid integer,
    geom geometry(Polygon) NOT NULL,
    srid integer,
    CONSTRAINT habitacao_pkey PRIMARY KEY (habitacaooid),
    CONSTRAINT habitacao_srid_fkey FOREIGN KEY (srid)
        REFERENCES public.spatial_ref_sys (srid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT habitacao_tipohabitacaooid_fkey FOREIGN KEY (tipohabitacaooid)
        REFERENCES public.tipohabitacao (tipohabitacaooid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.habitacao
    OWNER to postgres;

```

```

CREATE TABLE IF NOT EXISTS public.pontointeresse
(
    pontointeresseid integer NOT NULL DEFAULT nextval('pontointeresse_pontointeresseid_seq'::regclass),
    nome character varying(100) COLLATE pg_catalog."default",
    tipo pontointeresseid integer,
    geom geometry(Polygon) NOT NULL,
    srid integer,
    CONSTRAINT pontointeresse_pkey PRIMARY KEY (pontointeresseid),
    CONSTRAINT pontointeresse_srid_fkey FOREIGN KEY (srid)
        REFERENCES public.spatial_ref_sys (srid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT pontointeresse_tipo pontointeresseid_fkey FOREIGN KEY (tipo pontointeresseid)
        REFERENCES public.tipo pontointeresse (tipo pontointeresseid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.pontointeresse
    OWNER to postgres;

```

```

CREATE TABLE IF NOT EXISTS public.sinalvertical
(
    sinalverticalid integer NOT NULL DEFAULT nextval('sinalvertical_sinalverticalid_seq'::regclass),
    nome character varying(100) COLLATE pg_catalog."default",
    tiposinalid integer,
    geom geometry(Point) NOT NULL,
    srid integer,
    CONSTRAINT sinalvertical_pkey PRIMARY KEY (sinalverticalid),
    CONSTRAINT sinalvertical_srid_fkey FOREIGN KEY (srid)
        REFERENCES public.spatial_ref_sys (srid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT sinalvertical_tiposinalid_fkey FOREIGN KEY (tiposinalid)
        REFERENCES public.tiposinal (tiposinalid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.sinalvertical
    OWNER to postgres;

```



```

CREATE TABLE IF NOT EXISTS public.spatial_ref_sys
(
    srid integer NOT NULL,
    auth_name character varying(256) COLLATE pg_catalog."default",
    auth_srid integer,
    srtext character varying(2048) COLLATE pg_catalog."default",
    proj4text character varying(2048) COLLATE pg_catalog."default",
    CONSTRAINT spatial_ref_sys_pkey PRIMARY KEY (srid),
    CONSTRAINT spatial_ref_sys_srid_check CHECK (srid > 0 AND srid <= 998999)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.spatial_ref_sys
    OWNER to postgres;

REVOKE ALL ON TABLE public.spatial_ref_sys FROM PUBLIC;

GRANT SELECT ON TABLE public.spatial_ref_sys TO PUBLIC;

GRANT ALL ON TABLE public.spatial_ref_sys TO postgres;

CREATE TABLE IF NOT EXISTS public.tipoestrada
(
    tipoestradaid integer NOT NULL DEFAULT nextval('tipoestrada_tipoestradaid_seq'::regclass),
    nome character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT tipoestrada_pkey PRIMARY KEY (tipoestradaid),
    CONSTRAINT tipoestrada_nome_key UNIQUE (nome)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tipoestrada
    OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.tipohabitacao
(
    tipohabitacaoid integer NOT NULL DEFAULT nextval('tipohabitacao_tipohabitacaoid_seq'::regclass),
    nome character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT tipohabitacao_pkey PRIMARY KEY (tipohabitacaoid),
    CONSTRAINT tipohabitacao_nome_key UNIQUE (nome)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tipohabitacao
    OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.tipopontointeresse
(
    tipopontointeresseid integer NOT NULL DEFAULT nextval('tipopontointeresse_tipopontointeresseid_seq'::regclass),
    nome character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT tipopontointeresse_pkey PRIMARY KEY (tipopontointeresseid),
    CONSTRAINT tipopontointeresse_nome_key UNIQUE (nome)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tipopontointeresse
    OWNER to postgres;

```

```
CREATE TABLE IF NOT EXISTS public.tiposinal
(
    tiposinalid integer NOT NULL DEFAULT nextval('tiposinal_tiposinalid_seq'::regclass),
    nome character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT tiposinal_pkey PRIMARY KEY (tiposinalid),
    CONSTRAINT tiposinal_nome_key UNIQUE (nome)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tiposinal
    OWNER to postgres;
```

```
CREATE TABLE IF NOT EXISTS public.tipozonaverde
(
    tipozonaverdeid integer NOT NULL DEFAULT nextval('tipozonaverde_tipozonaverdeid_seq'::regclass),
    nome character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT tipozonaverde_pkey PRIMARY KEY (tipozonaverdeid),
    CONSTRAINT tipozonaverde_nome_key UNIQUE (nome)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tipozonaverde
    OWNER to postgres;
```

```

CREATE TABLE IF NOT EXISTS public.zonaverde
(
    zonaverdeid integer NOT NULL DEFAULT nextval('zonaverde_zonaverdeid_seq'::regclass),
    tipozonaverdeid integer,
    geom geometry(Polygon) NOT NULL,
    srid integer,
    nome character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT zonaverde_pkey PRIMARY KEY (zonaverdeid),
    CONSTRAINT zonaverde_srid_fkey FOREIGN KEY (srid)
        REFERENCES public.spatial_ref_sys (srid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT zonaverde_tipozonaverdeid_fkey FOREIGN KEY (tipozonaverdeid)
        REFERENCES public.tipozonaverde (tipozonaverdeid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.zonaverde
    OWNER to postgres;

```

```

INSERT INTO tipohabitacao (Nome) VALUES
    ('Residencial'),
    ('Comercial'),
    ('Industrial');

```

```

INSERT INTO tipoestrada (Nome) VALUES
    ('Rodovia'),
    ('Rua'),
    ('Avenida');

```

```

INSERT INTO tipozonaverde (Nome) VALUES
    ('Parque'),
    ('Jardim'),
    ('Praça');

```

```

INSERT INTO tipopontointeresse (Nome) VALUES
    ('Igreja'),
    ('Restaurante'),
    ('Estação de Comboio');

```



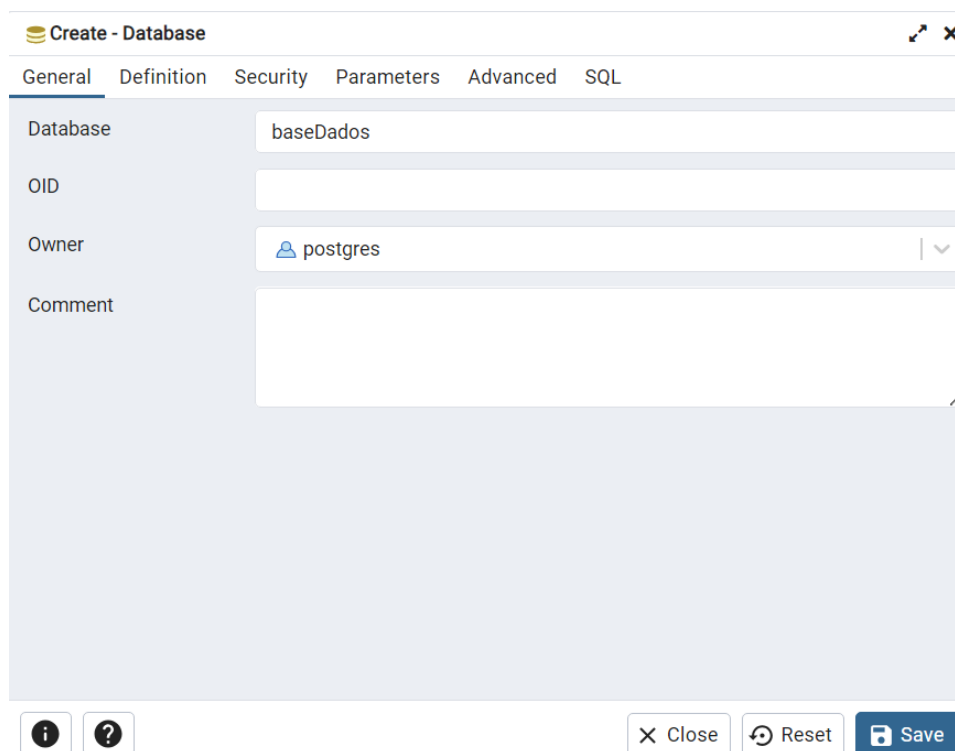
# Realização

## No PostGIS

Primeiramente foi criado um modelo de dados para posteriormente criar uma base de dados no PostgreSQL.

Para criar uma base de dados PostgreSQL fomos ao nosso servidor clicar na seta para mostrar mais e depois Databases. Depois clicar no botão direito do mouse , Create

-> DataBase, depois inserimos um nome á DataBase que foi baseDados.



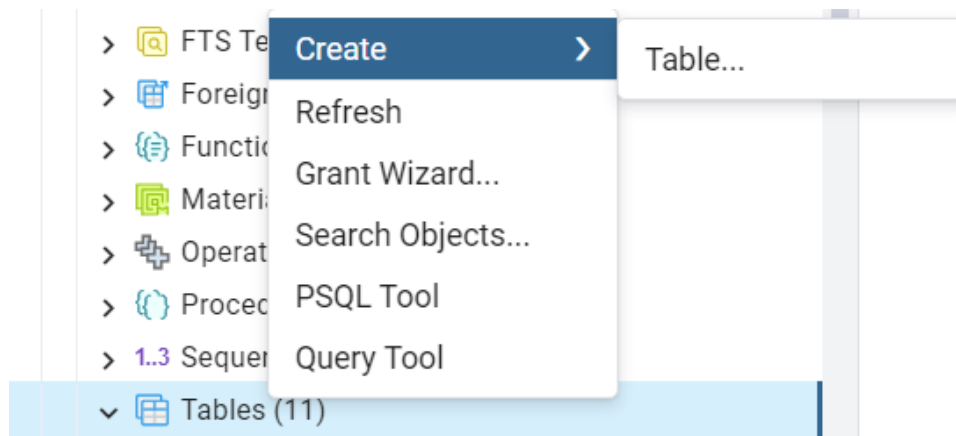
The screenshot shows a 'Create - Database' window with the following fields and options:

- Database:** baseDados
- OID:** (empty)
- Owner:** postgres (with a dropdown arrow)
- Comment:** (empty text area)
- Buttons:** Close, Reset, Save

Posteriormente é necessário importar uma extensão chamada de postgis, esta extensão é responsável pela criação da Tabela spatial\_ref\_sys , esta tabela permite a criação de dados espaciais. Algo essencial para realização do nosso trabalho.

```
CREATE EXTENSION IF NOT EXISTS postgis;
```

Agora é necessário criar as restantes tabelas do nosso modelo de dados, para isso basta Clicar em Tables, Create , Table



Em todas as tabelas há uma chave estrangeira com o srid correspondente da tabela spatial\_ref\_sys. Cada tabela tem também uma parte geométrica, podemos fazer isso adicionando uma coluna do tipo geometry que irá conter o tipo de geom que usei para cada tabela. Estes atributos criados com ajuda do PostGIS, permitem o armazenamento de formas multipartes, formas tridimensionais e formas que possuem uma medida (ou valor M) associada a seus vértices. As geometrias mais simples e usadas neste trabalho são as seguintes:

- POINT , neste trabalho usei pontos para marcar os diferentes tipos de sinais existentes numa área
- LINESTRING , para digitalizar estradas, usei linhas
- POLYGON , para construções, desde habitações a parques, usei polígonos

```
INSERT INTO Habitacao (TipoHabitacaoID, Geom, SRID) VALUES
```

```
(1, 'POLYGON((XX YYY, ZZ WWW, AA BBB, XX YYY))', 4326)
```

```
INSERT INTO PontoInteresse (TipoPontoInteresseID, Nome, Geom, Descricao, SRID)
VALUES
```

```
('Igreja', 1, 'POLYGON((XX YYY, ZZ WWW, AA BBB, XX YYY))', 4326),
```

```
INSERT INTO SinalVertical (TipoSinalID, Nome, Geom, SRID) VALUES
```

```
('Stop', 1, 'POINT(XX YYY)', 4326)
```

```
INSERT INTO Estrada (Nome, TipoEstradaID, Geom, SRID) VALUES
```

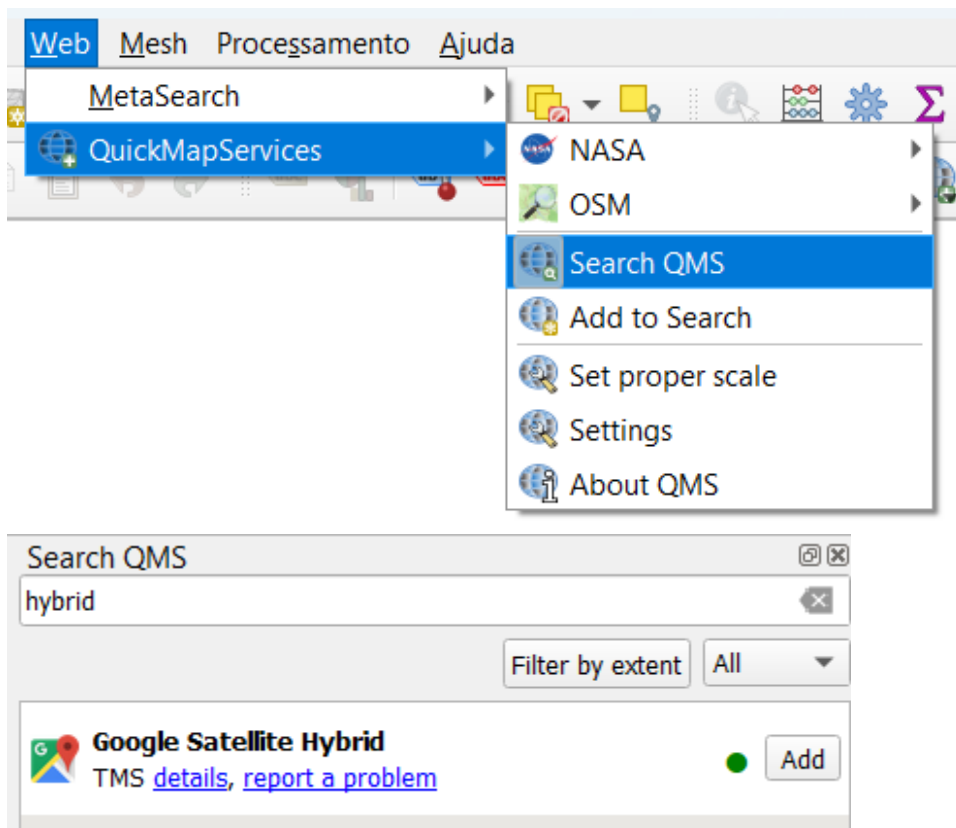
```
('Autoestrada A1', 1, 'LINESTRING(XX YYY, ZZ WWW)', 4326)
```

```
INSERT INTO ZonaVerde (TipoZonaVerdeID, Geom, SRID) VALUES
```

```
(1, 'POLYGON((XX YYY, ZZ WWW, AA BBB, XX YYY))', 4326),
```

## No QGIS

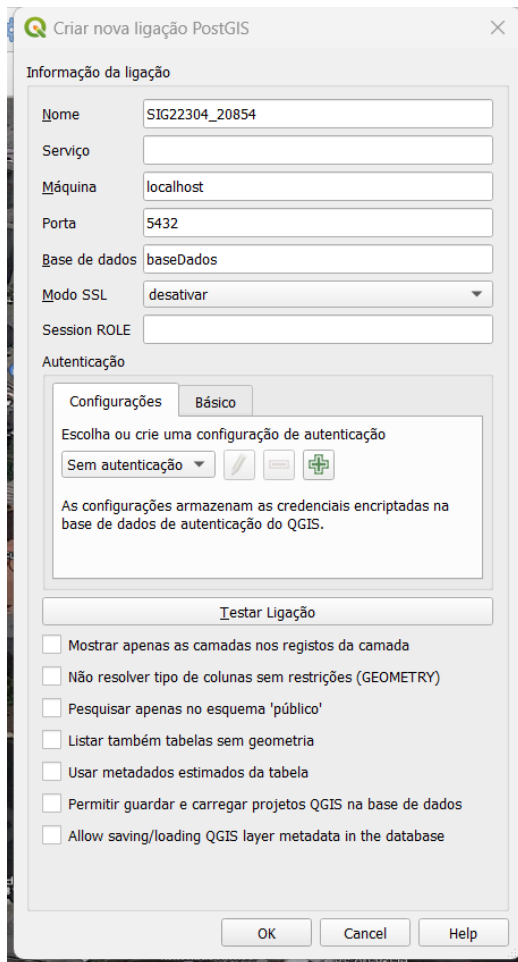
Após a criação do projeto instalei uma extensão do Google designada por: Google Satellite Hybrid para maior facilidade na digitalização.



Logo após a criação do projeto liguei este há minha de base de dados criada anteriormente no PostGis

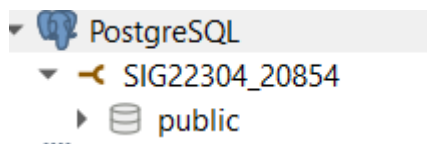
, para isso basta clicar com o botão direito numa espécie que elefante (logo do PosGis) e selecionar “nova ligação”.

Após o click em Nova Ligação vai aparecer uma janela igual ou parecida á seguinte:

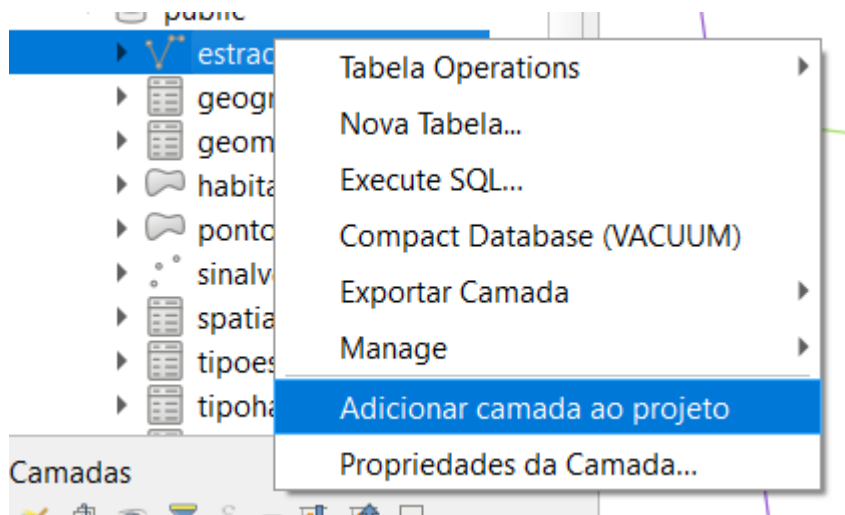


Na aba do Nome, é simplesmente o nome da ligação, demos o nome “SIG22304\_20854”. No serviço no nosso caso não vale a pena inserir nada pois não vamos usar nenhum serviço deste trabalho. No nosso caso usamos o XAMP como servidor local, ou seja, a nossa “Maquina”/Servidor será o localhost. A porta é inserida no form automaticamente é posto o nome da base de dados, nome que demos á nossa base de dados no PostGis que foi “baseDados”.

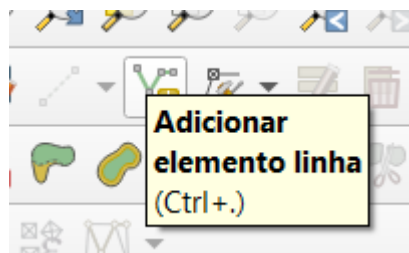
Quando a ligação for efetuada irá aparecer a nossa base de dados por baixo no logo do PosGis irá aparecer a nossa base de dados com o nome da nossa ligação.



Adicionamos a tabela como camada para o nosso projeto, para assim começar a digitalização de dados que serão inseridos na camada selecionada.



De seguida para digitalizar dados selecionamos a camada, ativar a edição nesta, e no painel superior escolhemos a opção de adicionar o tipo geométrico correspondente ao tipo desta camada.



Após a conclusão de toda a digitalização na área respetiva, o resultado foi o seguinte:





