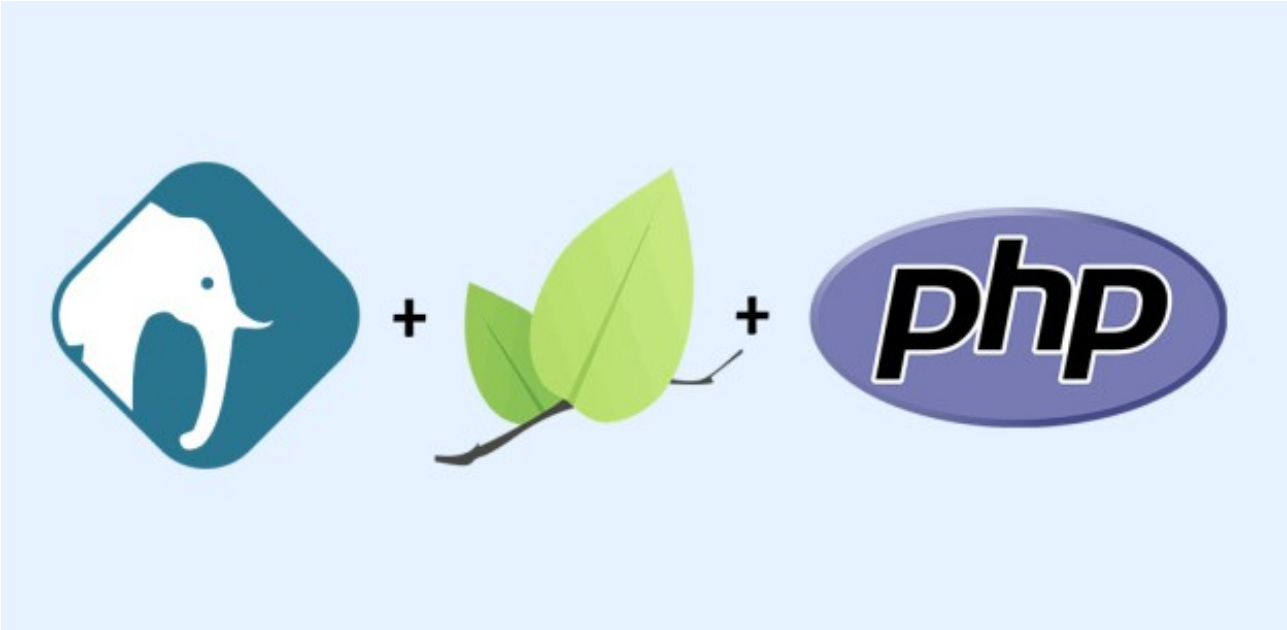
Como montar um Servidor de dados espaciais e como trabalhar com ele.



André Luiz Lima Costa

Preparando um servidor para trabalhar com geoprocessamento

Veremos os passos necessários para preparar um servidor (linux ou UNIX) para Geoprocessamento e processamento de dados em geral. Estou descrevendo aqui o processo no Centos 7 (o processo no RedHat e Fedora é exatamente o mesmo) Podemos usar outras distros (debian-Ubuntu, openSuse, etc) com as devidas substituições no programa instalador de cada distro (apt-get, zypper).

Os passos abaixo comentados descrevem o processo de instalação dos programas necessários no servidor. O programa R e módulos associados a ele são opcionais mas caso se interesse ele dá uma versatilidade enorme ao sistema de geoprocessamento. Executar como root, ou usando **sudo** antes de cada comando.

PASSO 1

ATUALIZANDO O INSTALADOR **yum** E INSTALANDO PACOTES DE PROGRAMAS BASE

```
yum install -y epel-release
yum update
yum install -y perl
yum install -y wget
yum install -y unzip
```

PASSO 2

OBTENDO E INSTALANDO O POSTGRESQL

```
rpm -Uvh https://yum.postgresql.org/11/redhat/rhel-7-x86_64/pgdg-centos11-
11-2.noarch.rpm
yum install -y postgresql11-server
yum groupinstall -y “Development Tools”
yum install -y postgresql11-devel

#INICIANDO O POSTGRESQL
/usr/pgsql-11/bin/postgresql-11-setup initdb
systemctl enable postgresql-11.service
systemctl start postgresql-11.service

#ADICIONANDO ENTRADAS NA CONFIGURAÇÃO PERMITINDO ACESSO REMOTO
#COM SENHA AO BANCO SEM A UTILIZAÇÃO DE MÁSCARA DE REDE
#****NOTA: O SERVIDOR ESTARÁ ABERTO A ACESSOS EXTERNOS MEDIANTE
# AUTENTICAÇÃO – ACESSOS MAIS RESTRITOS PODEM SER ESTABELECIDOS***
sed -i "\$a host all all 0.0.0.0/0 md5" /var/lib/pgsql/11/data/pg_hba.conf
sed -i "\$a listen_addresses = '*'" /var/lib/pgsql/11/data/postgresql.conf

# INICIANDO O POSTGRESQL E ADICIONANDO AO initd
systemctl restart postgresql-11.service
systemctl enable postgresql-11.service

# MUDE PARA O USUÁRIO postgres e execute os comandos abaixo para
# criar um superusuário do banco de de dados, modifique o conteúdo
# em laranja de acordo com sua necessidade
sudo -i -u postgres
psql template1 -c "CREATE USER usuário WITH PASSWORD 'segredo';"
psql template1 -c "ALTER USER usuário WITH SUPERUSER;"
exit

#como o novo usuário criado acima, crie o seu banco de dados
createdb meubanco -O usuário –encoding=utf-8
```

```
# Crie um um grupo que somente permite leitura dos dados (e
# visualização) sem permissões para alterar o conteúdo e crie um
# usuário de leitura.
psql -d meubanco -c "DO 'BEGIN IF NOT EXISTS (SELECT 1 FROM pg_group WHERE
groname = ''readaccess'') THEN CREATE ROLE readaccess;GRANT usage on SCHEMA
public to readaccess;  END IF; END';DO 'BEGIN IF NOT EXISTS (SELECT 1 FROM
pg_roles WHERE rolname = ''usuárioquesólê'') THEN CREATE USER usuárioquesólê
WITH PASSWORD ''segredo2''; GRANT readaccess TO usuárioquesólê;END IF;END';"
```

PASSO 3

```
#ANTES DE INSTALAR O POSTGIS VAMOS INSTALAR ALGUNS PROGRAMAS DE SUPORTE
yum install -y libxml2-devel
yum install -y geos37.x86_64
yum install geos37-devel.x86_64
yum install -y proj-devel.x86_64
yum install -y libstdc++.so.6
yum install python36 python36-devel
yum install python-devel

#O GDAL DEVE SER COMPILADO PARA TERMOS A ÚLTIMA VERSÃO
wget https://download.osgeo.org/gdal/2.4.0/gdal-2.4.0.tar.gz
tar xzf gdal-2.4.0.tar.gz
cd gdal-2.4.0
./configure --with-python --with-jpeg --with-png --with-geotiff --with-pg --
with-geos=/usr/geos36/bin/geos-config --without-sfcgal
make
make install
cd ..

#AGORA VAMOS COMPILAR O POSTGIS
yum install -y centos-release-scl-rh
yum install -y llvm-toolset-7
wget https://download.osgeo.org/postgis/source/postgis-2.5.2.tar.gz
tar xzf postgis-2.5.2.tar.gz
cd postgis-2.5.2
./configure --with-pgconfig=/usr/pgsql-11/bin/pg_config --with-projdir=/usr/
proj49 --with-geosconfig=/usr/geos37/bin/geos-config
--with-gdalconfig=/usr/local/bin/gdal-config
make
# -------------IMPORTANTE CRIAR ESTE LINK----------------------
sudo ln -s /opt/rh/llvm-toolset-7/root/usr/bin/llvm-lto
/usr/lib64/llvm5.0/bin/llvm-lto
make install
cd ..

# ADICIONE A EXTENSÃO POSTGIS AO SEU BANCO DE DADOS CRIADO ACIMA
psql -d meubanco -c "CREATE EXTENSION postgis;"
psql -d meubanco -c "CREATE EXTENSION postgis_topology;"
# SE O SEGUINTE ERRO APARECER :
# ERROR: could not load library “/usr/pgsql-11/lib/rtpostgis-
# 2.5.so”: libgdal.so.20:
# CRIE ESTE LINK
ln -s /usr/local/lib/libgdal.so.20.5.0 /usr/pgsql-11/lib/libgdal.so.20
```

PASSO 4

```
#--------------- Ambiente R (OPCIONAL)--------------------
yum install -y R
R -e 'install.packages("raster")'
R -e 'install.packages("rgdal", configure.args =
"--with-gdal-config=/usr/local/bin/gdal-config")'
R -e 'install.packages("rgeos", configure.args =
"--with-geos-config=/usr/geos37/bin/geos-config")'
#--------copiar usr/pgsql-11/include em usr/includes e fazer o
# mesmo com lib
R -e 'install.packages("RPostgreSQL")'
R -e 'install.packages("rpostgis")'
```

PASSO 5

```
#-------------Finalizamos instalando Apache e PHP
yum -y install httpd
systemctl start httpd.service
systemctl enable httpd.service
yum -y install php
yum -y install php-pgsql php-gd
systemctl restart httpd.service
systemctl enable httpd.service
```

Tudo pronto! Já temos tudo que precisamos para carregar os dados e criar ferramentas para interagir com esses dados. Nos próximos passos colocaremos dados no banco e começaremos a implementar as ferramentas de visualização dos dados do sistema.

Carregando os dados espaciais no banco Postgis (Parte 2)

Na parte anterior vimos como criar um banco de dados para geoprocessamento. Agora vamos mostrar como carregar informações de arquivos para dentro do banco de dados. Podemos alcançar esse objetivo de diversos modos.

Primeiro vamos obter o dado usando o comando wget. Esse dado é sobre processos minerários no estado do Amazonas por exemplo, mas pode ser qualquer arquivo de dado.

wget "http://sigmine.dnpm.gov.br/sirgas2000/AM.zip"
unzip -o AM.zip

Os dados estão em SIRGAS2000 (EPSG:4674).

Modo 1

De forma rápida e eficiente usando ***shp2pgsql*** na linha de comando:

/usr/pgsql-11/bin/shp2pgsql -s 4674 -d AM public.am | psql -d meubanco

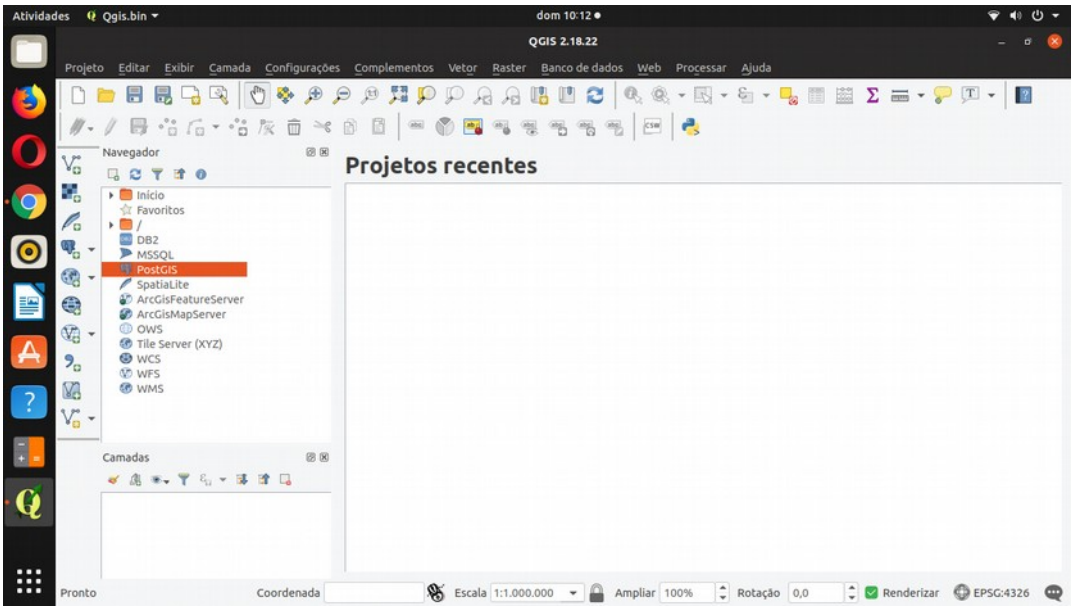
Ou usando o programa ogr2ogr para os outros formatos. O ogr2ogr identifica qual é o formato a ser convertido e, existindo um drive para esse formato, o arquivo será carregado no banco de dados (inclusive shapefiles podem ser carregados dessa forma):

ogr2ogr -f "PostgreSQL" PG:"host=seuhost user=usuário dbname=meubanco password=segredo" -a_srs"EPSG:4674" arquivo.tab -nlm nomeDaTabela

Modo 2

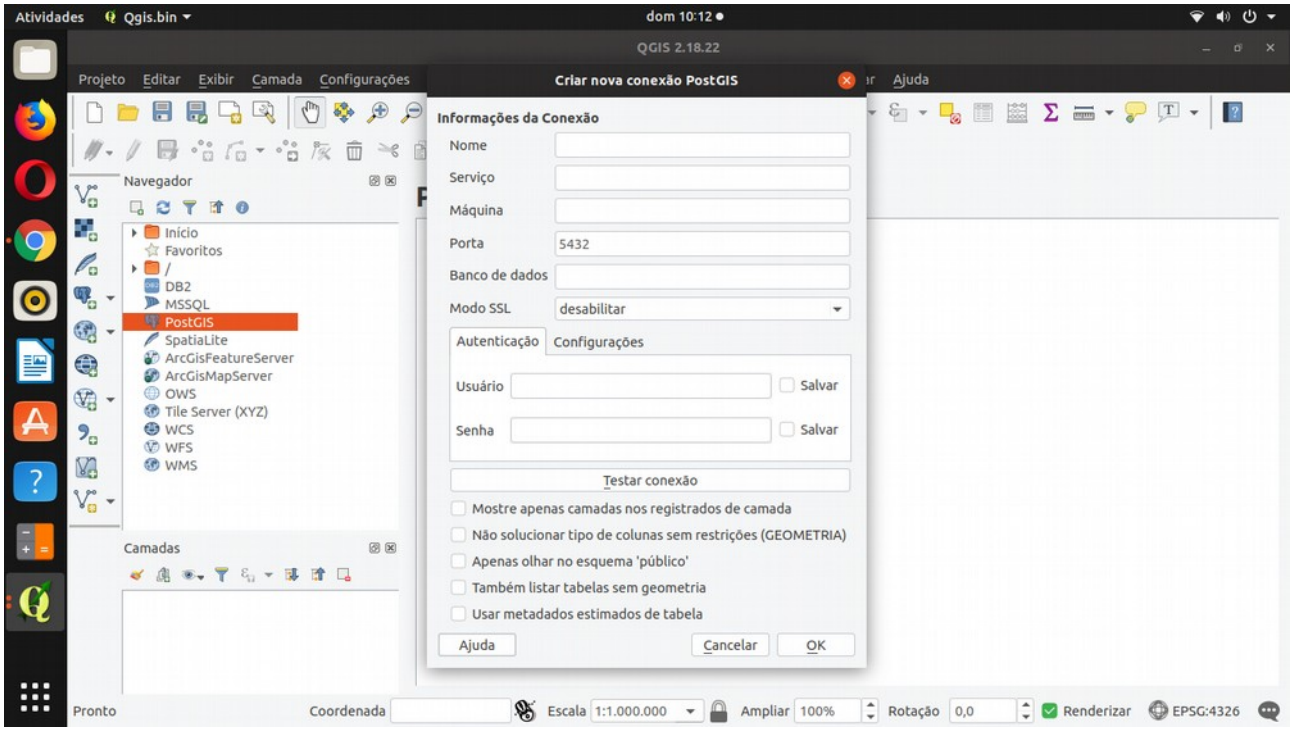
Usando QGIS conectado ao banco de dados. Alguma versões de QGIS não suportam o formato de dados 3D do shapefile. Por essa razão vamos converter o arquivo para 2D antes de carregar ele no banco de dados com QGIS usando ogr2ogr na linha de comando.

ogr2ogr -f "ESRI Shapefile" AM2.shp AM.shp -dim 2

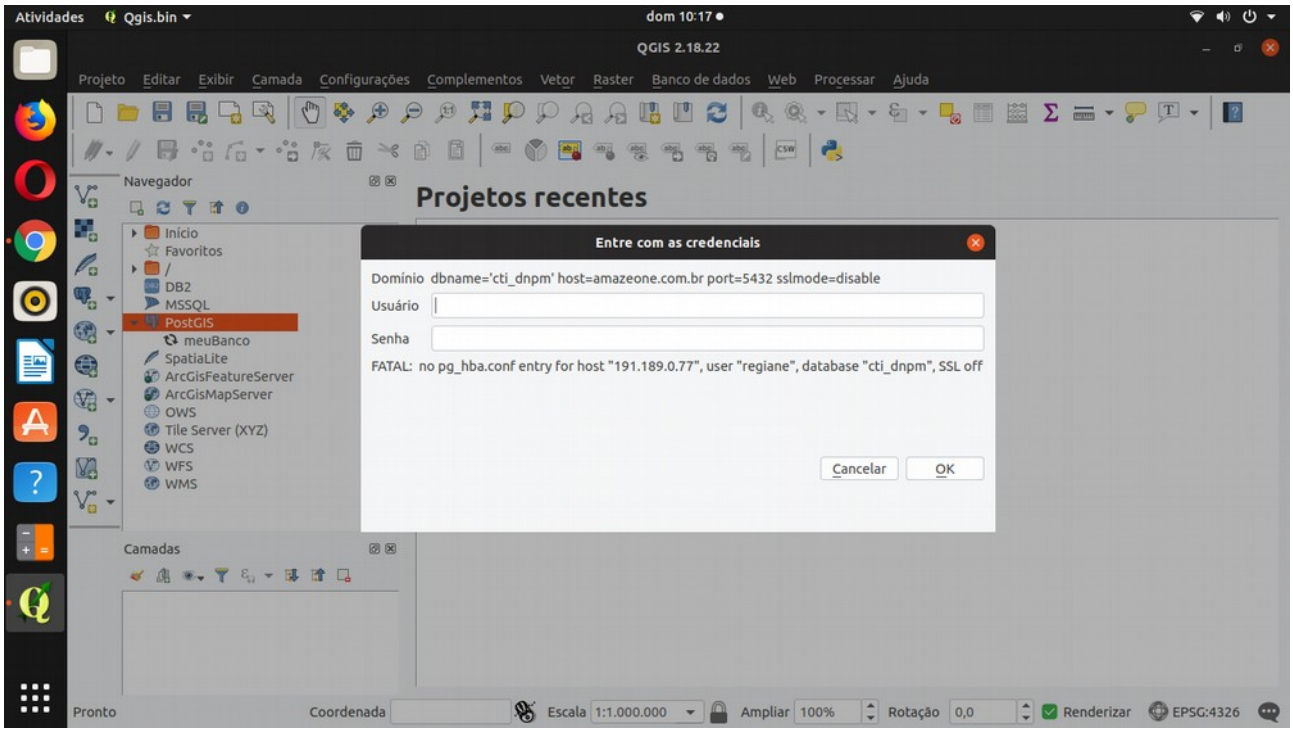


Abra o QGIS e crie uma conexão nova de postgis clicando com o botão direito no Elefante e clique em nova conexão:

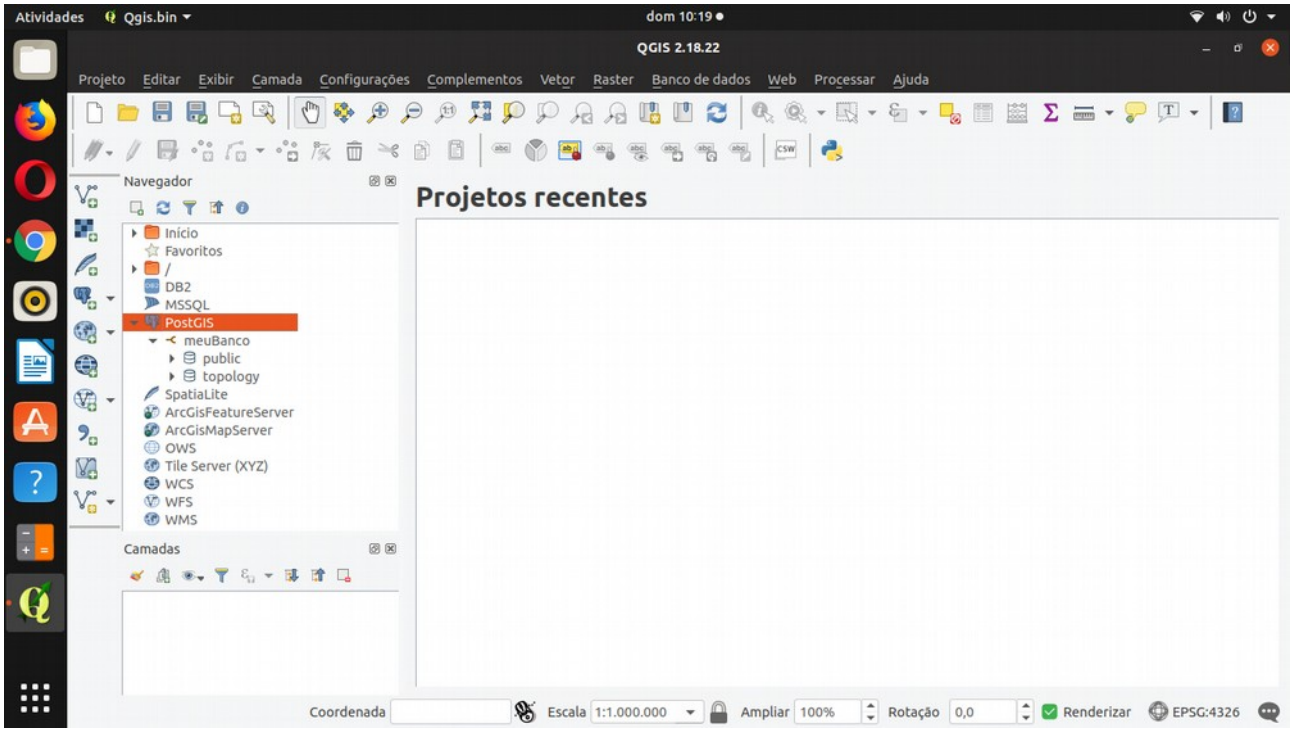
Entre com os parâmetros de conexão onde **Máquina** é o IP ou nome do domínio do banco de dados e **Banco de dados** é o nome do seu banco de dados. Dê um nome a essa conexão:



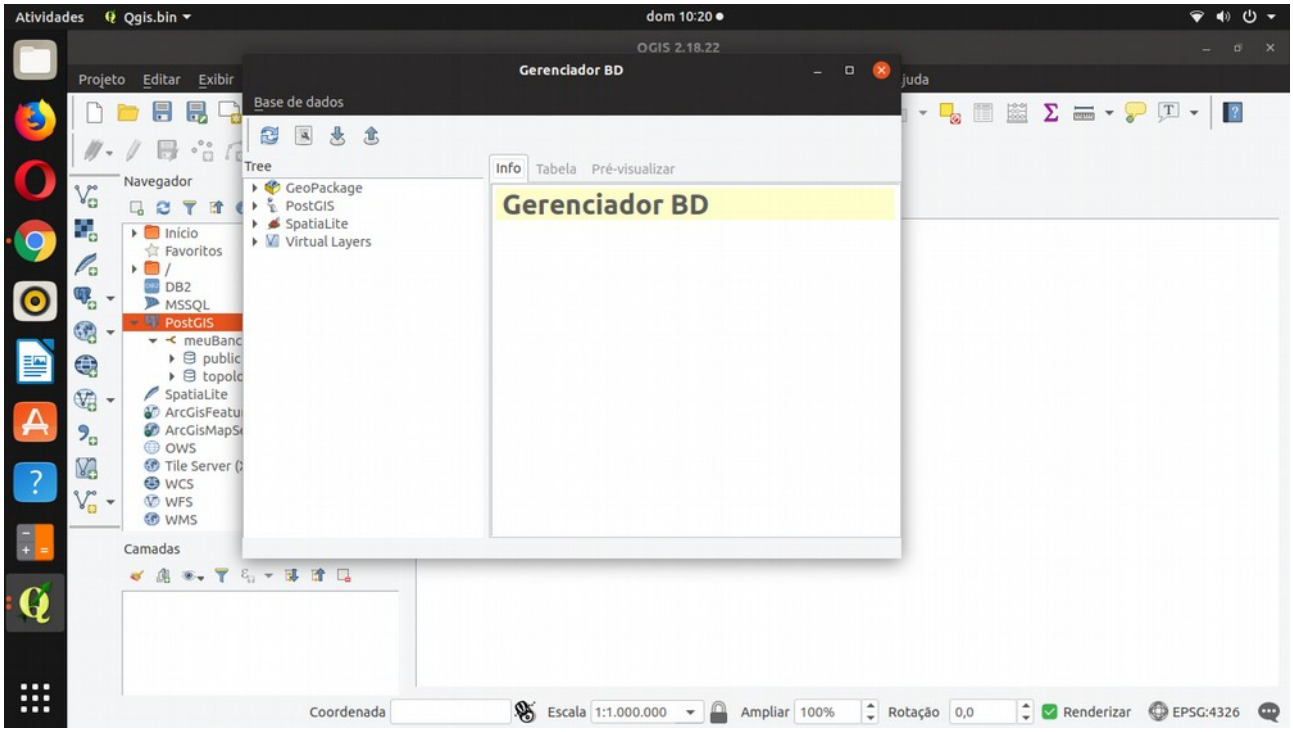
Clique na seta da nova entrada de banco e entre com ao usuário e a senha do seu banco de dados postgis:



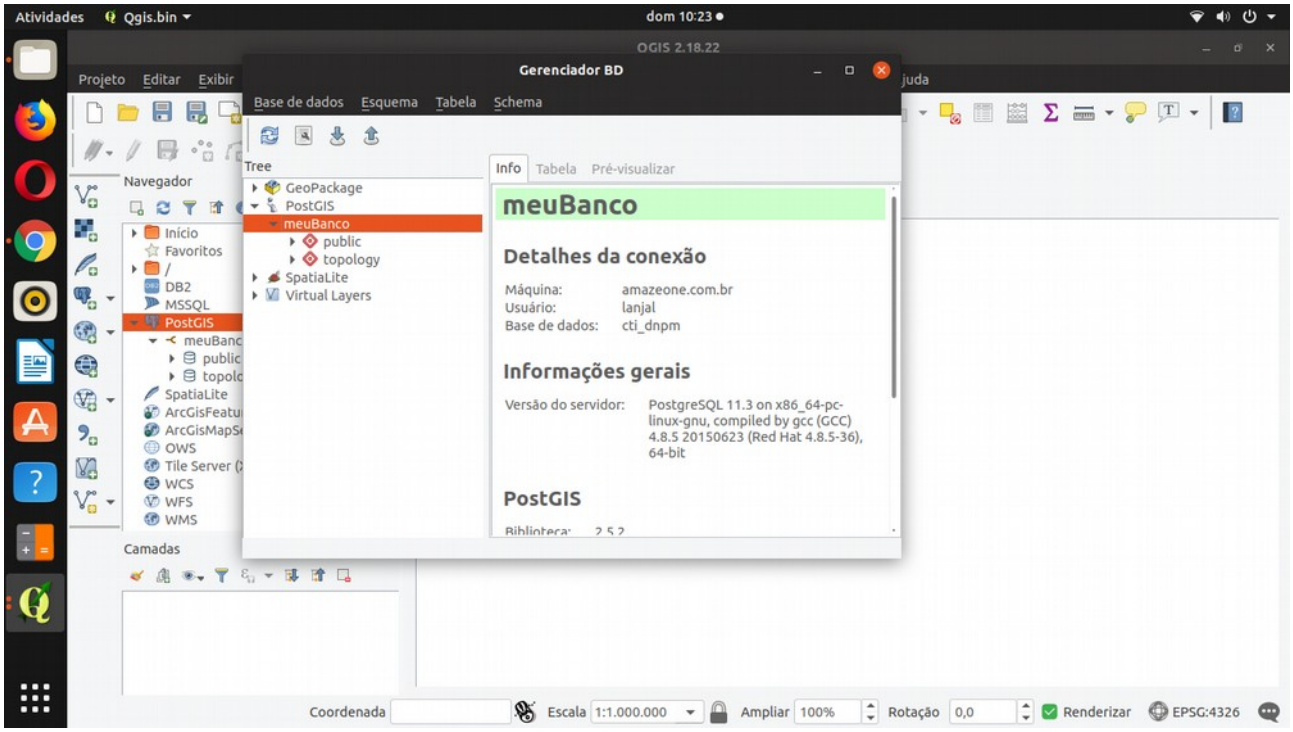
Uma entrada **public** e uma **topology** irá aparecer.



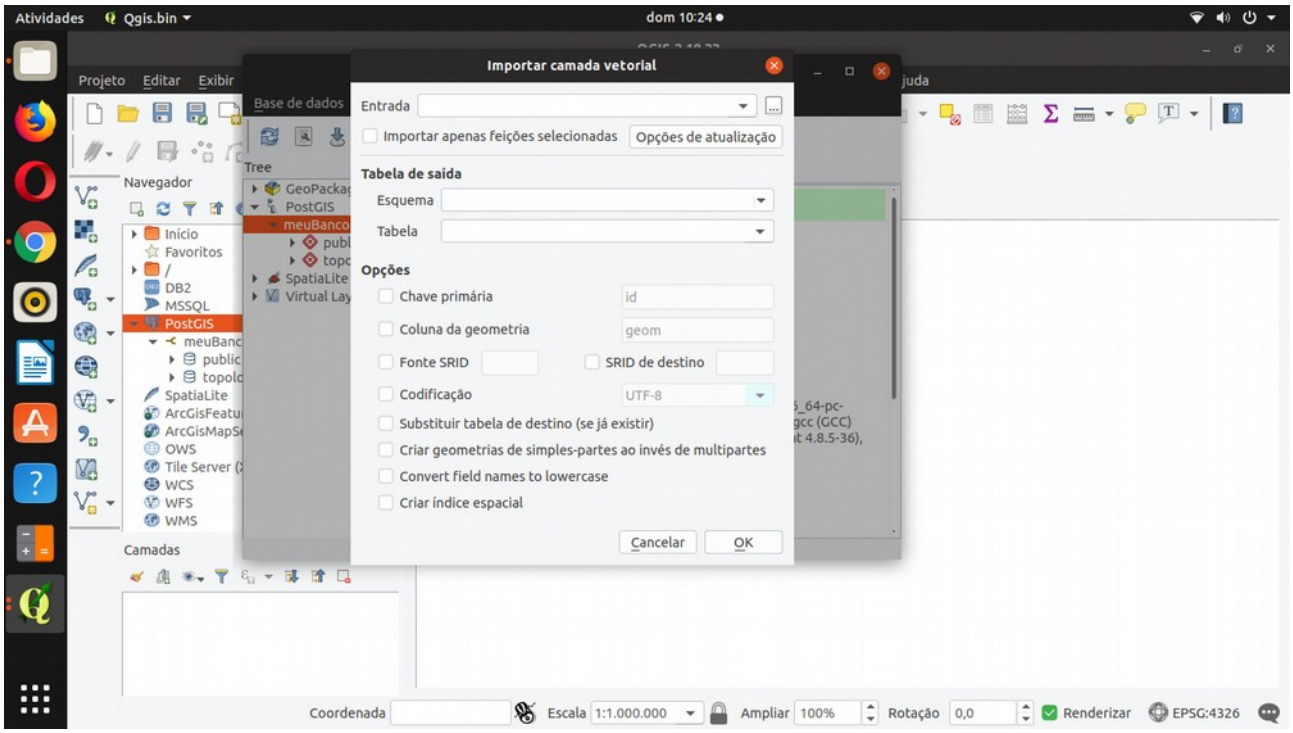
Clique em Banco de dados > Gerenciar BD > Gerenciador BD



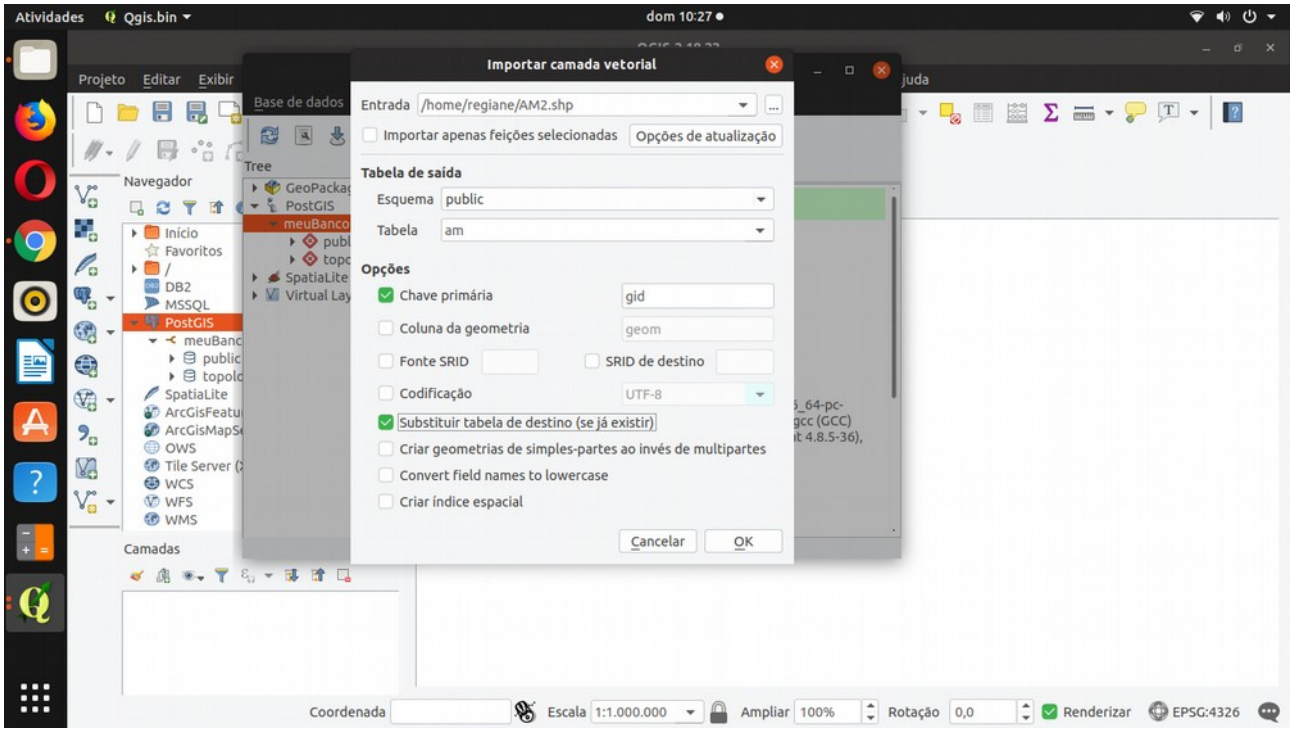
Clique na seta Postgis > meuBanco e aguarde conectar:



Clique Tabela > Importar camada/arquivo:



Selecione o arquivo AM2.shp usando ... e preencha conforme abaixo:



Clique em OK e aguarde (demora uns 25 minutos) até a mensagem de importação bem sucedida aparecer.

Os outros formatos também podem ser carregados no banco Postgis pelo QGIS da mesma forma que fizemos acima.
Usando QGIS é bem mais lento que usando linha de comando mas funciona bem também.

Modo 3

A maioria dos outros programas desktop também se conectam com o Postgis. Dentre eles podemos citar uDig, GRASS, Kosmo, OpenJUMP, TerraView, SAGA GIS.
O ArcGis e o MapInfo Profissional também se conectam com o Postgis mas você precisa pagar por licenças adicionais para ter a possibilidade de editar os dados diretamente no banco de dados.

Segue abaixo alguns links de como se conectar ao Postgis usando alguns aplicativos desktop GIS.

AutoCAD - https://knowledge.autodesk.com/pt-br/support/autocad-map-3d/learn-explore/caas/CloudHelp/cloudhelp/2018/PTB/MAP3D-Use/files/GUID-4476AD90-FD33-41EA-9DC7-C8EAE04B9C7F-htm.html
ArcGIS - https://desktop.arcgis.com/en/arcmap/10.3/manage-data/gdbs-in-postgresql/connect-postgresql.htm
MapInfo - http://support.pitneybowes.com/VFP05_KnowledgeWithSidebarHowTo?id=kA180000000CuaUCAS&popup=false&lang=en_US
uDig - http://www.clickgeo.com.br/integracao-udig-postgis/
kosmo - http://www.clickgeo.com.br/integracao-do-kosmo-gis-com-base-de-dados-postgis/
GRASS - https://grasswiki.osgeo.org/wiki/PostGIS

Na próxima parte vamos ver como extrair os objetos do banco de dados diretamente.

Acessando dados GIS do Postgis diretamente (Parte 3)

Finalmente vamos agora ver dois exemplos de como acessar dados geográficos do Postgis diretamente no formato KML e também no formato geoJSON.

KML

O formato kml é usado pela Google nos seus aplicativos. O GoogleEarth é um deles. Vamos ver como rapidamente carregar os dados usando um servidor web e php que irá ler o banco de dados Postgis e formatar a informação em KML para o GoogleEarth.

Primeiro vamos criar uma função no nosso banco de dados PostgreSQL que fará todo o trabalho de conversão (backend) dos dados em um arquivo do tipo kml:

Crie o seguinte arquivo kml.sql com o conteúdo abaixo:

```
create or replace function filtra_xml(text)
returns text as $$
  select replace(replace(replace($1, '&', '&amp;'), '<', '&lt;'), '>', '&gt;');
$$ language sql immutable strict;

create or replace function envelope_kml(text)
returns text as $$
  select
    '<kml xmlns="http://www.opengis.net/kml/2.2"><Document>'
    || $1 || '</Document></kml>';
$$ language sql immutable strict;

-- geometria mais cinco atributos
create or replace function concatena_kml(text, geometry, anyelement,text,
anyelement,text, anyelement,text, anyelement,text, anyelement,text,text)
returns text as $$
  select $1 || '<Placemark><Style><LineStyle><color>' || $12 ||
  '</color><width>1</width></LineStyle><PolyStyle><color>'||$13 ||
  '</color></PolyStyle></Style><name>' || filtra_xml(cast($3 as text))
  || '</name><description><![CDATA[<table><tr><td>' || $4 || '</td><th>' ||
filtra_xml(cast($5 as text))
  || '</th></tr><tr><td>' || $6 || '</td><th>' || filtra_xml(cast($7 as text))
  || '</th></tr><tr><td>' || $8 || '</td><th>' || filtra_xml(cast($9 as text))
  || '</th></tr><tr><td>' || $10 || '</td><th>' || filtra_xml(cast($11 as text))
  || '</th></tr></table>]]></description>' || st_askml($2) || '</Placemark>';
$$ language sql immutable strict;

create aggregate dcmnt_kml(geometry, anyelement,text, anyelement,text,
anyelement,text, anyelement,text, anyelement,text,text) (
    sfunc = concatena_kml,
    finalfunc = envelope_kml,
    stype = text,
    initcond = ''
);
```

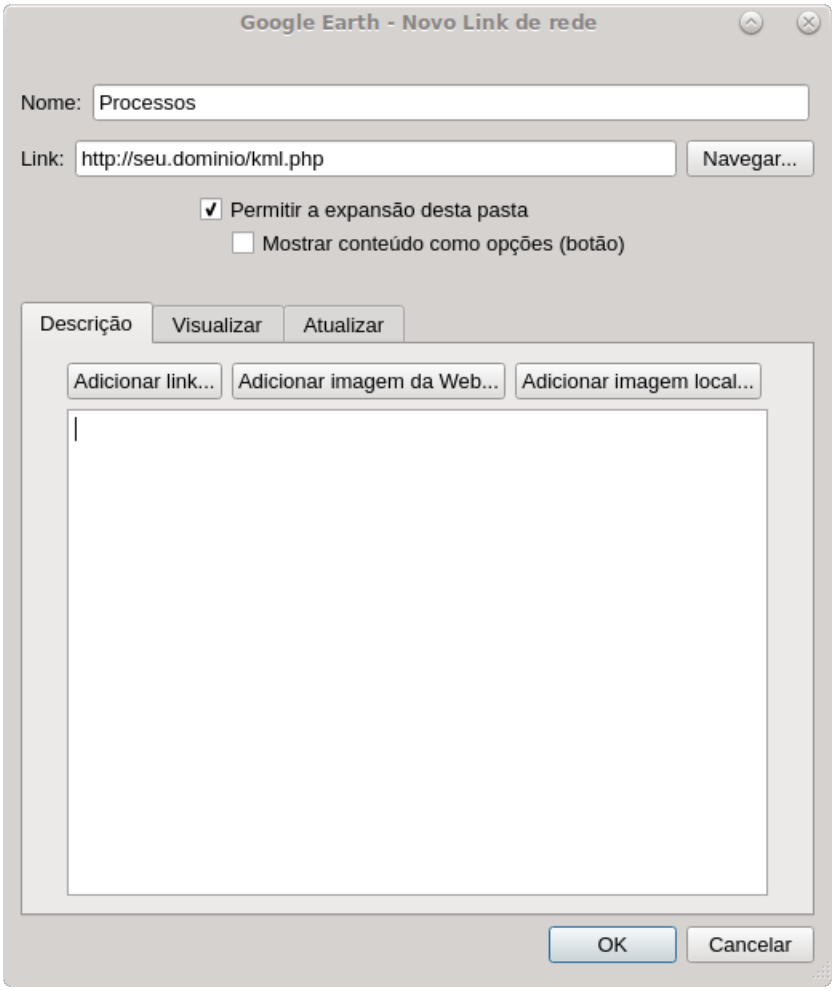
Execute o seguinte comando para carregar estas funções no seu banco de dados:

```
psql -d meubanco -f kml.sql
```

Em seguida usamos a seguinte página kml.php em um servidor web como o frontend para os dados:

```
<?php
$con= pg_connect(" host=seuhost port=5432 dbname=meubanco user=usuárioquesólê
password=segredo  ");
$sql="select dcmnt_kml(geom,processo,'Nome',nome,'Substância',subs,'Último
Evento',ult_evento,'Fase',fase,'ff0000dd','8ca0a0a0') from am;";
$res = pg_exec($con, $sql);
for ($i=0; $i<pg_numrows($res); $i++) {
    $kml = pg_result($res, 0, 0);
}
echo $kml;
pg_close($con);
?>
```

No GoogleEarth crie o link usando Adicionar > Link de rede (substituir **seu.dominio** por um domínio válido ou IP de seu servidor):



Daí o objeto será carregado automaticamente no GoogleEarth. Simples né?

geoJSON

Agora vamos carregar o dado diretamente em um mapa Leaflet usando javaScript (com php AJAX).

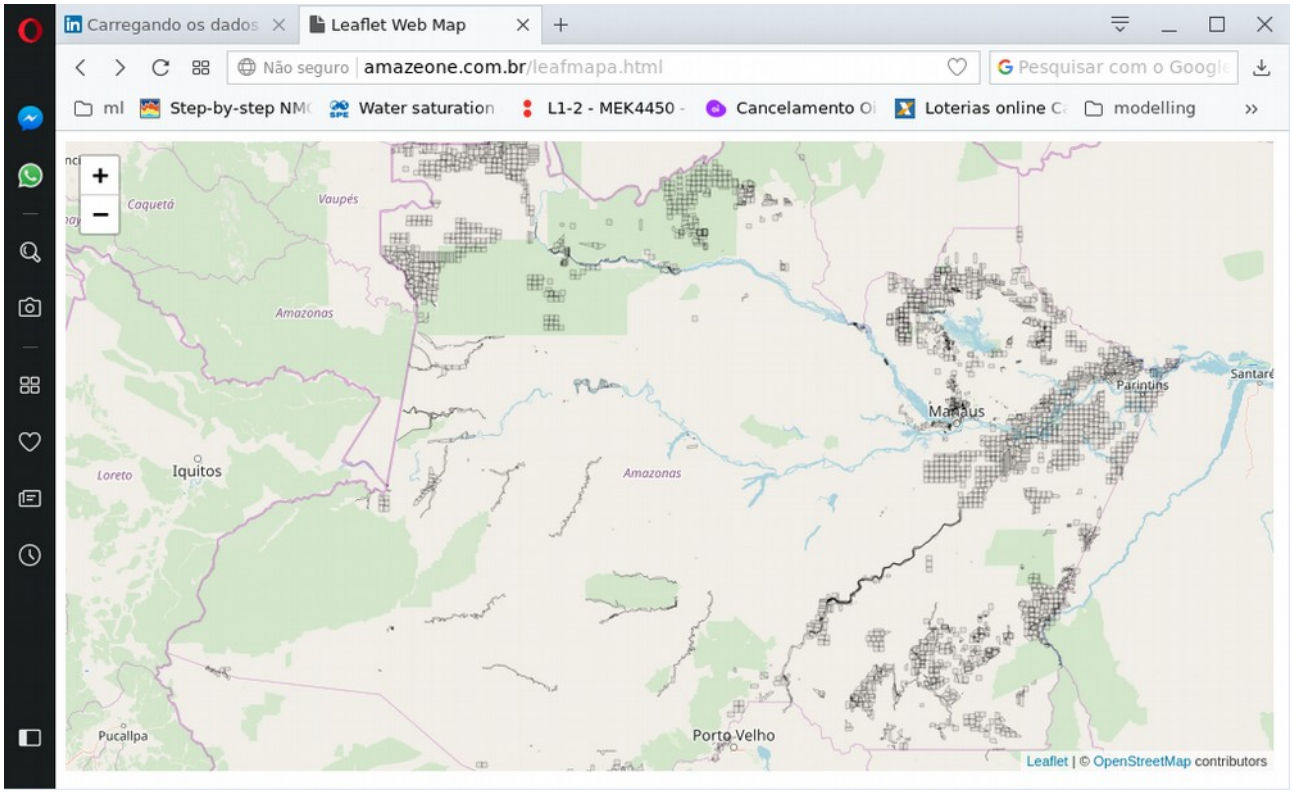
Primeiro crie o arquivo gjson.php no seu servidor web:

```
<?php
$ta=$_GET['tabela'];
$co=$_GET['colunas'];
$sql="SELECT row_to_json(fc)
 FROM ( SELECT 'FeatureCollection' As type, array_to_json(array_agg(f)) As
features
 FROM (SELECT 'Feature' As type
    , ST_AsGeoJSON(st_flipcoordinates(lg.geom))::json As geometry
    , row_to_json(($co)) As properties
   FROM $ta As lg   ) As f )  As fc;";
$db = pg_connect("host=seuHost dbname=meubanco user=usuárioquesólê
password=segredo");
$ret = pg_query($db, $sql);
$rows=pg_fetch_result($ret,0,0);
print $rows;
$db = NULL;
?>
```

Depois crie um mapa Leaflet leafmapa.html conforme abaixo:

```
<!DOCTYPE html>
<html>
<head>
<title>Leaflet Web Map</title>
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.5.1/dist/leaflet.css"/>
<script src="https://unpkg.com/leaflet@1.5.1/dist/leaflet.js"></script>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js">
</script>
<style>#map { width: 960px; height:500px; }</style>
</head>
<body>
  <div id="map"></div>
  <script>
  var grProcessos2= L.layerGroup();
var map = L.map('map',{center: [-3.7, -65],zoom: 6,layers:[grProcessos2] });
L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', { attribution: '&copy;
<a href="http://osm.org/copyright">OpenStreetMap</a>
contributors' }).addTo(map);
processos2(grProcessos2);
function processos2(grProcessos2) {
$.getJSON("gjson.php?
tabela=am&colunas=processo,nome,subs,ult_evento,fase",function(data) {
        for (var i = 0; i < data.features.length; i++) {
          var polig = new L.polygon(data.features[i].geometry.coordinates,
          {color:'black',weight:0.2,fillOpacity:0.05}).addTo(grProcessos2);
      }
});
}
  </script>
</body>
</html>
```

Ao acessar a página **seu.domino/leafmapa.html** você verá:



Finalizamos aqui nossa série. Vimos com montar um banco de dados Postgis, como carregar dados nele e agora vimos dois exemplo simples de como acessar os dados do Postgis diretamente.
As possibilidades são infinitas, basta somente usar a criatividade e um pouco de SQL.