

Exercise # 8

Making a location aware app in Django (Python) and PostgreSQL

One of the easier ways for making a location aware application is the use of Python based web development tool Django. One barely needs anything but the framework, readily prepares a database for you. The only elements that are required by the user, are proper installation and customization. Keeping this in view, we have prepared a small tutorial for you, that uses your virtual machine to create the application. This application can be accessed by an external browser to say the least.

You'll be using the following tools to develop your nearby shops web application:

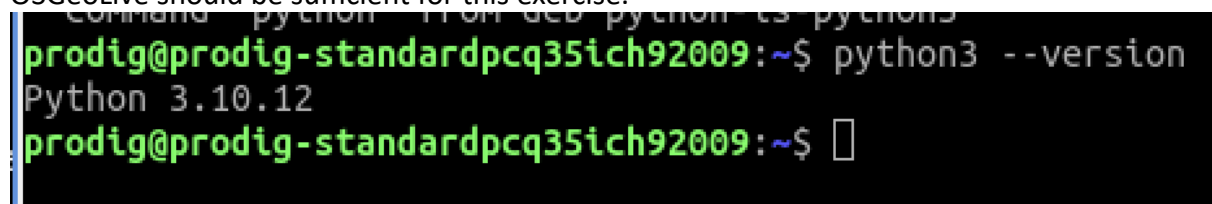
1. The Python programming language
2. The Django web framework
3. The PostgreSQL database for persisting data
4. The PostGIS extension for supporting spatial features in the PostgreSQL database.
5. pip for installing dependencies.
6. The venv module for managing a virtual environment.
7. Virtual for installing PostgreSQL and PostGIS

Almost all the above comes preinstalled and preconfigures with your virtual machines. In case that is not the case, these instructions will make it convenient for you.

To start you can type the following command in your terminal. This will print the python version of your system for you.

```
python3 -- version
```

Your installed version should be python 3.11 or above. Whatever comes packaged with the OSGeoLive should be sufficient for this exercise.

A terminal window with a black background and green text. The prompt is 'prodig@prodig-standardpcq35ich92009:~\$'. The command 'python3 --version' has been entered, and the output 'Python 3.10.12' is displayed on the next line. The prompt is now 'prodig@prodig-standardpcq35ich92009:~\$' followed by a cursor.

```
prodig@prodig-standardpcq35ich92009:~$ python3 --version
Python 3.10.12
prodig@prodig-standardpcq35ich92009:~$
```

You may also want to install the development level updates of python too. For that type of the following lines of code in your windows.

```
sudo apt-get install python3-dev
```

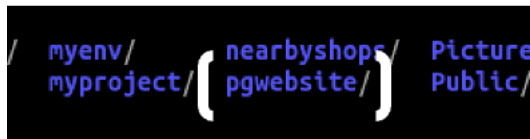
This step eases the process of installing, additional libraries such as psycopg2 for your website. This library will be required for us to connect our database to our website.

In python you have the possibility of creating a virtual environment. This is small environment, inside your virtual machine. That isolates all your libraries and funtions from the rest of the installation. It is more like a small container for your application.

By using the following commands, you now create a python virtual environment. It is totally different from the virtual environment that you created previously, in your virtualbox.

```
python3 -m venv pgwebsite
```

This will create a folder in your immediate directory (the directory that you are in) called pgwebsite.



Once, the virtual environment is created, you may now activate it by typing the following command.

```
source pgwebsite/bin/activate
```

After, you have typed the above line of code, you will be seeing a change of your command prompt header as following. This specifies that you are in new virtual environment now, the name of the virtual environment will be before your username and in paranthesis.

```
prodig@prodig-standardpcq35ich92009:~$ source pgwebsite/bin/activate
(pgwebsite) prodig@prodig-standardpcq35ich92009:~$
```

Now we will install Django, a python framework we need to create your website. This will not be a systemwide installation but specific to your virtual environment. This can be done using the following commands. You must have noticed; we have not used the sudo command here for that purpose too.

```
pip install django
```

You will get the following output on your screen.

```
(pgwebsite) prodig@prodig-standardpcq35ich92009:~$ pip install django
Collecting django
  Using cached Django-4.2.4-py3-none-any.whl (8.0 MB)
Collecting asgiref<4,>=3.6.0
  Using cached asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting sqlparse>=0.3.1
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Collecting typing-extensions>=4
  Using cached typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Installing collected packages: typing-extensions, sqlparse, asgiref, django
```

Now since, our Django framework is installed, we can proceed with installation of our project files. The following commands will create the basic structure for our website, which we will add out templates for application (website).

The application name we are using is nearbyshops. The purpose for this website, will be to create a database, of nearby stores, and shops in our area.

```
django-admin startproject nearbyshops
```

If you type the following command, you will be in your project directory.

```
cd nearbyshops
```

You will get to see the following directories. These directories are the ones we will be editing files, in and creating new files in too.

```
(pgwebsite) prodig@prodig-standardpcq35ich92009:~/nearbyshops$ ls
manage.py*  nearbyshops/
(pgwebsite) prodig@prodig-standardpcq35ich92009:~/nearbyshops$
```

Once, your nearbyshop application is ready. It is time to edit the settings file for your website. This can be done by changing your directory to the nearbyshops folder and opening the settings, file. For that effect use, the following commands.

```
cd nearbyshops/nearbyshops
```

```
nano settings.py
```

and add a new set of code in the DATABASE section. You will need to comment out (commenting out is done by add a # signe to any line of code). By that the software will not ready the code and the line will not have to be deleted as well.

This is how the original file looks like,

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends|sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

This is how it will be after commenting these lines of text out.

```
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

#DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': BASE_DIR / 'db.sqlite3',
#     }
#}
```

Once, you have done so now go to your PGAdmin 4 and create a new database, gis or any name that you want. Then add the following code, with information specific to your database in it. Now copy and paste the following just below this commented out section

```
DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'gis',
        'USER': 'user',
        'PASSWORD': 'user',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}
```

The window will now look as follows.

```
#DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': BASE_DIR / 'db.sqlite3',
#     }
#}

DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'gis',
        'USER': 'user',
        'PASSWORD': 'user',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}
```

Once, you are done. You can do some other changes. This is to add a reference to an installed gis management library in your installed apps section in the top section of your settings file.

add the following line to the installed apps section of your settings.py file.

django.contrib.gis,

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'django.contrib.gis',  
]
```

The file section will now look as follows.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'django.contrib.gis',  
]
```

Now we will need to install a library that helps us connect our postgres database to our website. This is called psycopg2, this library will help our website read the information we enter and add it our database.

pip install psycopg2-binary

Now we will change into the the higher level nearbyshops directory where our manage.py file is and we will create our mapping application called, shops.

python manage.py startapp shops.

This will create a second directory for you called shops.

```
(pgwebsite) prodig@prodig-standardpcq35ich92009:~/nearbyshops$ ls  
manage.py* nearbyshops/ shops/  
(pgwebsite) prodig@prodig-standardpcq35ich92009:~/nearbyshops$
```

Add this line to your settings.py file
'shops',

```

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.gis',
    'shops',
]

```

Now we need to start getting our website in shape. For that change your directory to the shops folder by
cd shops

and the type

nano model.py

```

GNU nano 6.2
from django.db import models

# Create your models here.

```

The redline of code that you see, paste the following text under that line.

```
from django.contrib.gis.db import models
```

```

class Shop(models.Model):
    name = models.CharField(max_length=100)
    location = models.PointField()
    address = models.CharField(max_length=100)
    city = models.CharField(max_length=50)

```

This sets up your website, to ask the user for the above fields, and when the user is going fill this information, it will add that to your database.

Now we will need to run the manage.py file again to make migrations. The process of making migrations in Django are of excellent use. They are going to add the above fields for

you to your database. The above should run without an error if your username, password, and database name are correct.

You will need to change to the directory with manage.py file in it by typing.

```
cd ..
```

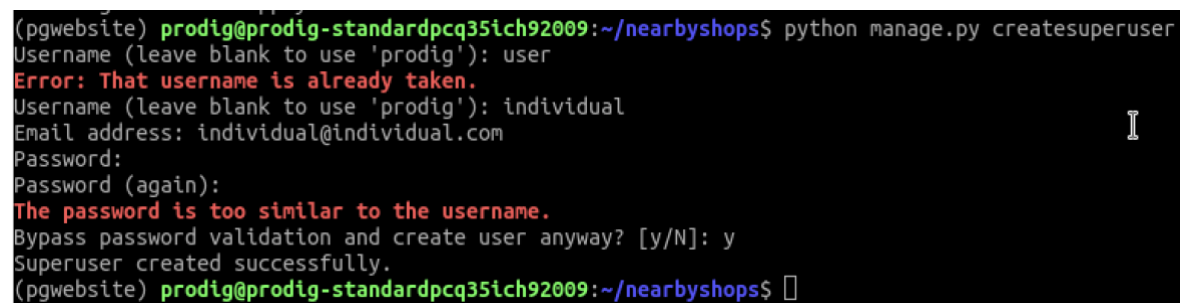
And then run the following command.

```
python manage.py makemigrations
```

```
python manage.py migrate
```

You might want to add a main user to your application, that will be the only one with a password to add new data to your application. This can be done, using the following command. For you, I recommend picking passwords that are easy to remember.

```
python manage.py createsuperuser
```



```
(pgwebsite) prodig@prodig-standardpcq351ch92009:~/nearbyshops$ python manage.py createsuperuser
Username (leave blank to use 'prodig'): user
Error: That username is already taken.
Username (leave blank to use 'prodig'): individual
Email address: individual@individual.com
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(pgwebsite) prodig@prodig-standardpcq351ch92009:~/nearbyshops$
```

Or you may pick as follows

```
Username (leave blank to use 'prodig'): user
Email address: user@user.com
Password: user
Password (again): user
```

Now change to the shops directory

```
cd shops
```

and type

```
nano admin.py
```

and add the following code

```
from django.contrib.gis.admin import OSMGeoAdmin
from .models import Shop
```

```
@admin.register(Shop)
class ShopAdmin(OSMGeoAdmin):
    list_display = ('name', 'location')
```

This will create your admin page and will allow the user to see the administrative page of your application.

You can now give it a try and run your server by typing the following command.

```
cd ..
```

```
python manage.py runserver 0:8000
```

```
(pgwebsite) prodig@prodig-standardpcq35ich92009:~/nearbyshops$ python manage.py runserver 0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 31, 2023 - 02:09:37
Django version 4.2.4, using settings 'nearbyshops.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

If you go to your browser now on your linux machine, and type the above given IP

0.0.0.0:8000

(8000 is the port we are using for running our Django application)

you will be able to see the following.

DisallowedHost at /

Invalid HTTP_HOST header: '0.0.0.0:8000'. You may need to add '0.0.0.0' to ALLOWED_HOSTS.

Request Method:	GET
Request URL:	http://0.0.0.0:8000/
Django Version:	4.2.4
Exception Type:	DisallowedHost
Exception Value:	Invalid HTTP_HOST header: '0.0.0.0:8000'. You may need to add '0.0.0.0' to ALLOWED_HOSTS.
Exception Location:	/home/prodig/pgwebsite/lib/python3.10/site-packages/django/http/request.py, line 150, in get_host
Python Executable:	/home/prodig/pgwebsite/bin/python
Python Version:	3.10.12
Python Path:	['/home/prodig/nearbyshops', '/usr/lib/python310.zip', '/usr/lib/python3.10', '/usr/lib/python3.10/lib-dynload', '/home/prodig/pgwebsite/lib/python3.10/site-packages']
Server time:	Thu, 31 Aug 2023 02:12:44 +0000

Traceback

[Switch to copy-and-paste view](#)

```
/home/prodig/pgwebsite/lib/python3.10/site-packages/django/core/handlers/exception.py, line 55, in inner
    response = get_response(request)
```

You will need to add your ip to allowed hosts, for that you need to change the settings.py file as follows.

cd nearbyshops

nano settings.py

go to the allowed hosts section and add the following line,

['yourip', 'localhost']

In my case the line was as follows

['192.168.64.3', 'localhost']

```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-ocsd&n^f-ph_-bz^uvxn9o_)utw4k0+l_z#c34lwfu2ut#9=##'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []
```

The changed file will look like this,

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['192.168.64.3', 'localhost']

Application definition
```

Now run

python manage.py runserver 0:8000

This shall make, the application now run smoothly. Try typing your virtualmachine IP in your host browser.

And append :8000 to it.

In my case it was

<http://192.168.64.3:8000/>

You will see the following page,

django

View [release notes](#) for Django 4.2



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

Now let us go to the admin page by typing

Youripaddress:8000/admin

In my case it was

<http://192.168.64.3:8000/admin>

You will see the following page,

A screenshot of the Django administration login page. It features a dark blue header with the text "Django administration" and a small icon. Below the header, there are two input fields labeled "Username:" and "Password:". A blue "Log in" button is positioned below the password field. The entire form is centered on a light grey background.

Login to your website and you will see the following page.

A screenshot of the Django administration site interface. At the top, there is a dark blue header with the text "Django administration" on the left and "WELCOME, INDIVIDUAL. VIEW SITE / CHANGE PASSWORD / LOG OUT" on the right. Below the header, the page is divided into two main sections. The left section, titled "Site administration", contains two sub-sections: "AUTHENTICATION AND AUTHORIZATION" and "SHOPS". Each sub-section has a table with columns for the name and actions (Add and Change). The right section, titled "Recent actions", contains a table with columns for "My actions" and "None available".

Here you can add users, and more shops.

Go to the shops, button, and press it and you will find the following.

Django administration WELCOME, INDIVIDUAL. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Shops > Shops > Add shop

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

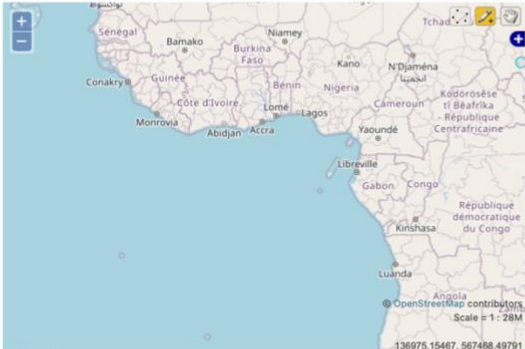
Users [+ Add](#)

SHOPS

Shops [+ Add](#)

Add shop

Name:

Location: 

Delete all Features

Address:

City:

[SAVE](#) [Save and add another](#) [Save and continue editing](#)

Zoom into NUST and add the shop C1 or any place of your choice.

Django administration WELCOME, INDIVIDUAL. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Shops > Shops > Add shop

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)


Users [+ Add](#)

SHOPS

Shops [+ Add](#)

Add shop

Name:

Location: 

Delete all Features

Address:

City:

[SAVE](#) [Save and add another](#) [Save and continue editing](#)

Add shop

Name:

C1

Location:



Delete all Features

Address:

NUST H12

City:

Islamabad

SAVE

Save and add another

Save and continue editing

Now click the save button.

Select shop to change

ADD SHOP +

Action: Go 0 of 1 selected

<input type="checkbox"/>	NAME	LOCATION
<input type="checkbox"/>	C1	SRID=4326;POINT (72.99022435125767 33.64649708186242)

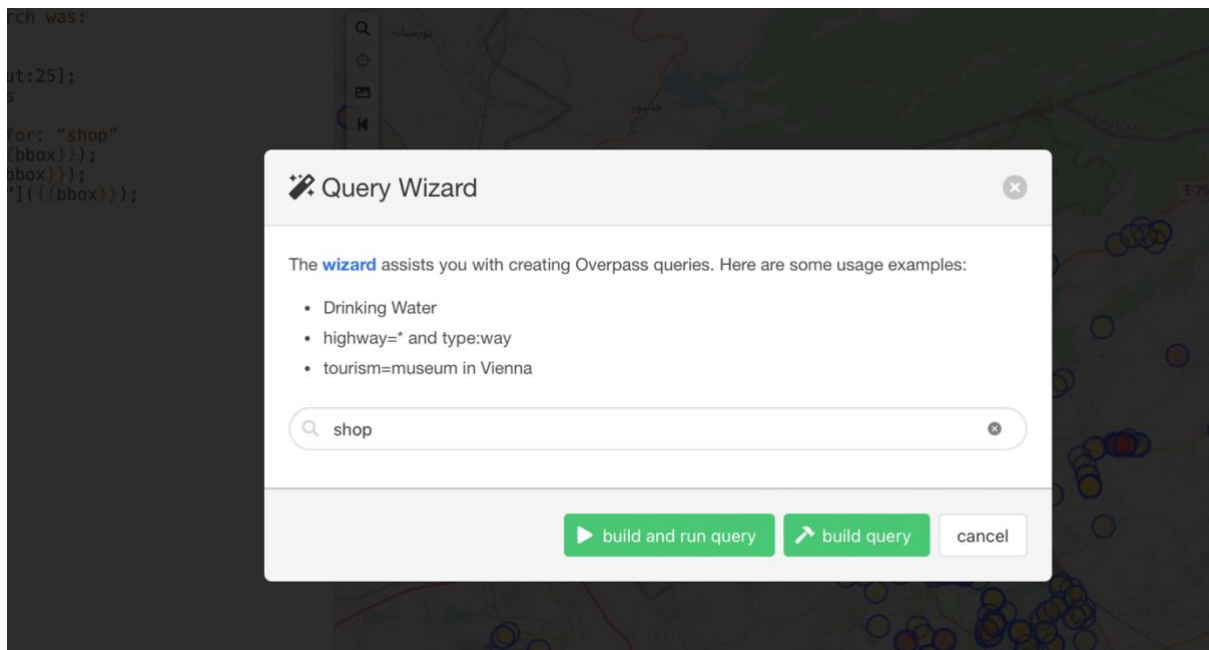
1 shop

You can also import bulk data into your database on many shops in your area of interest. For that go to the website.

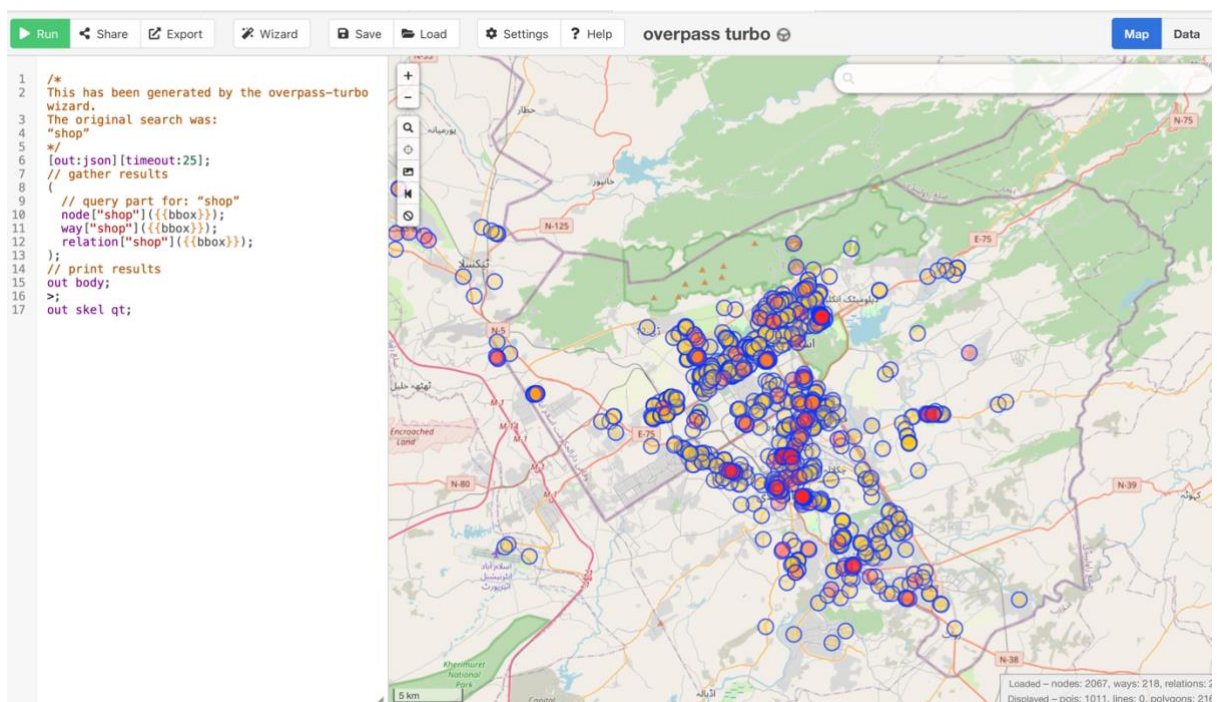
<http://overpass-turbo.eu/>

Zoom to a region of interest and click on the wizard tab

You will have the following window open



Type shop and run your query.



You can work on importing this data into your database