

Tutorial de Django

Primeros pasos: (instalación y creación del proyecto)

1. Instalar entorno virtual

```
pip install virtualenv
```

2. crear entorno virtual (pr2 es el nombre que escogí, pero puede ser cualquiera)

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.22000.856]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\migue\OneDrive\Documentos\django documentacion>py -m venv pr2
```

3. Luego de activado ir a la carpeta scripts del entorno y activarlo con el comando activate

```
C:\Users\migue\OneDrive\Documentos\django documentacion>cd pr2/scripts
C:\Users\migue\OneDrive\Documentos\django documentacion\pr2\Scripts>activate
```

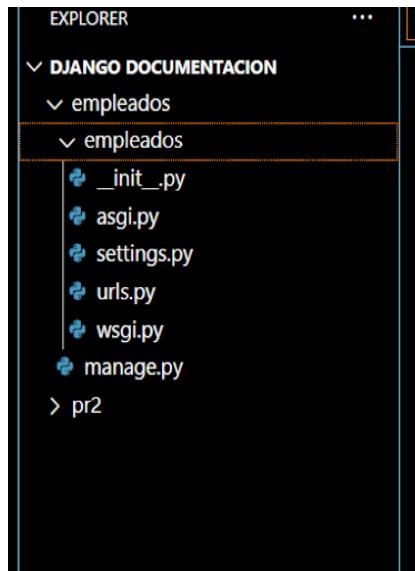
4. Devolvernos a la raiz donde vamos a manejar el proyecto por fuera de los archivos del entorno he instalamos Django con el comando pip install django

```
(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion\pr2\Scripts>cd..
(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion\pr2>cd..
(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion>pip install django
Collecting django
  Using cached Django-4.1-py3-none-any.whl (8.1 MB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)
Collecting tzdata
  Using cached tzdata-2022.2-py2.py3-none-any.whl (336 kB)
Collecting asgiref<4,>=3.5.2
  Using cached asgiref-3.5.2-py3-none-any.whl (22 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.5.2 django-4.1 sqlparse-0.4.2 tzdata-2022.2
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Users\migue\OneDrive\Documentos\django documentacion\pr2\Scripts\python.exe -m pip install --upgrade pip' command.
```

```
(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion>django-admin startproject empleados  
(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion>
```

Configuración inicial: (organización del archivo settings y creación de la carpeta de templete)

- Después de lo anterior se habrán creado algunos archivos dentro de la carpeta de nuestro proyecto (en este caso empleado) estos archivos son:



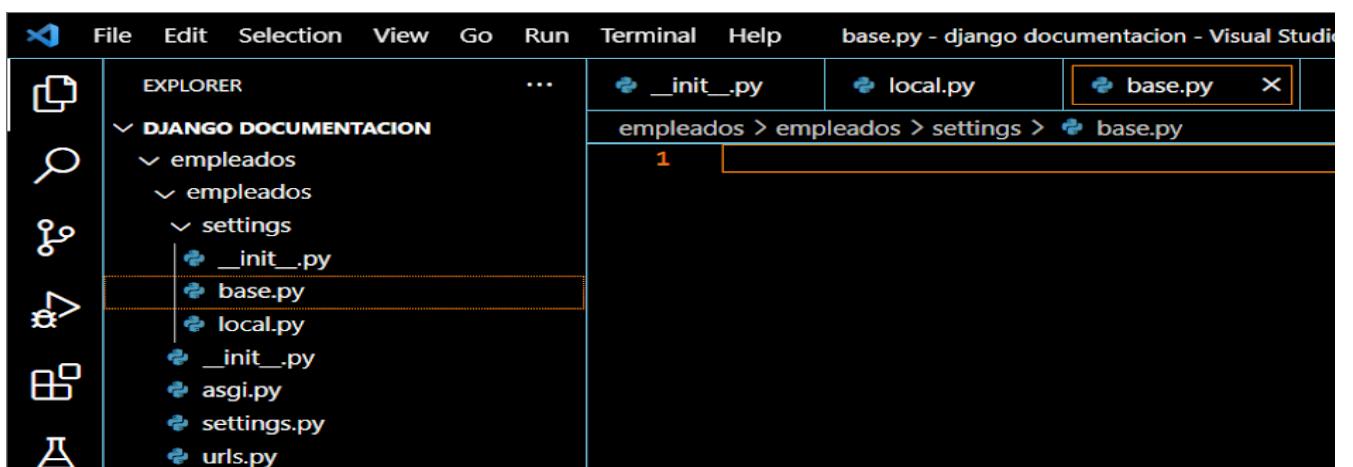
__init__.py: es un archivo sumamente importante porque es quien le dice a Python que en dicha carpeta debe buscar porque hay código Python que necesita para la aplicación del programa(poo)

Settings.py: es donde se encuentran las configuraciones que van a aplicar a todo el aplicativo web, desde aquí diremos que aplicaciones están instaladas, el tipo de base de datos que usaremos, donde buscara los templete y los archivos statics, la seguridad del aplicativo y las herramientas para desplegar nuestro proyecto en la web.

Urls.py: en este archivo se gestionan las direcciones web donde podremos encontrar la información que deseamos mostrar en el aplicativo web en internet

cada una de ellas nos llevará a una vista en la que encontraremos una pieza de información del aplicativo.

- A la altura de nuestro proyecto crearemos una carpeta que contendrá todas nuestras configuraciones, con el objetivo de organizar nuestro trabajo y de dividir las fases de nuestro proyecto sin que cause conflicto si estas son realizadas por diferentes personas, para ello crearemos dentro de la carpeta :
 - Crear Un archivo `__init__.py` para que Django entienda que debe buscar ahí
 - Crear Un archivo `base.py` donde almacenaremos la parte del `settings` que es común a todas las fases del proyecto
 - Crear un archivo `local.py` donde pondremos la configuración relevante para la fase de creación de nuestro proyecto en la que trabajaremos en los instantes iniciales



3. Ahora pasaremos a dividir la información de settings.py a estos nuevos archivos de la siguiente manera

The screenshot shows the Visual Studio Code interface with the following details:

- Terminal:** terminal Help
- File Explorer:** sistemaGestion > sistemaGestion > settings > base.py > ...
- Editor:** The file `base.py` is open. The code defines settings for a Django application, including paths, security keys, installed apps, and middleware. The code is as follows:

```
1  from pathlib import Path
2
3  # Build paths inside the project like this: BASE_DIR / 'subdir'.
4  BASE_DIR = Path(__file__).resolve().parent.parent.parent
5
6  # SECURITY WARNING: keep the secret key used in production secret!
7  SECRET_KEY = 'djang'
8
9  INSTALLED_APPS = [
10     'django.contrib.admin',
11     'django.contrib.auth',
12     'django.contrib.contenttypes',
13     'django.contrib.sessions',
14     'django.contrib.messages',
15     'django.contrib.staticfiles',
16     'pc'
17 ]
18
19 MIDDLEWARE = [
20     'django.middleware.security.SecurityMiddleware',
21     'django.contrib.sessions.middleware.SessionMiddleware',
22     'django.middleware.common.CommonMiddleware',
23     'django.middleware.csrf.CsrfViewMiddleware',
24     'django.contrib.auth.middleware.AuthenticationMiddleware',
25     'django.contrib.messages.middleware.MessageMiddleware',
26     'django.middleware.clickjacking.XFrameOptionsMiddleware',
27 ]
```

The code editor highlights the line `SECRET_KEY = 'djang'` with a blue box.

Bottom Status Bar: base.py - proyecto-de-cistema-de-gestion-de-certificados-y-pqrs - Visual Studio Code



```
base.py - proyecto-de-cistema-de-gestion-de-certificados-y-pqrs - Visual Studio Code
terminal Help
setting.py base.py x
sistemaGestion > sistemaGestion > settings > base.py > ...
30
31     TEMPLATES = [
32         {
33             'BACKEND': 'django.template.backends.django.DjangoTemplates',
34             'DIRS': [BASE_DIR/'templete'],
35             'APP_DIRS': True,
36             'OPTIONS': {
37                 'context_processors': [
38                     'django.template.context_processors.debug',
39                     'django.template.context_processors.request',
40                     'django.contrib.auth.context_processors.auth',
41                     'django.contrib.messages.context_processors.messages',
```

```
62     |     [
63     |     |     'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
64     |     ],
65   ]
66
67
68 # Internationalization
69 # https://docs.djangoproject.com/en/4.1/topics/i18n/
70
71 LANGUAGE_CODE = 'es-Es'
72
73 TIME_ZONE = 'UTC'
74
75 USE_I18N = True
76
77 USE_TZ = True
78
79 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
80
```

4. En local es importante importar base

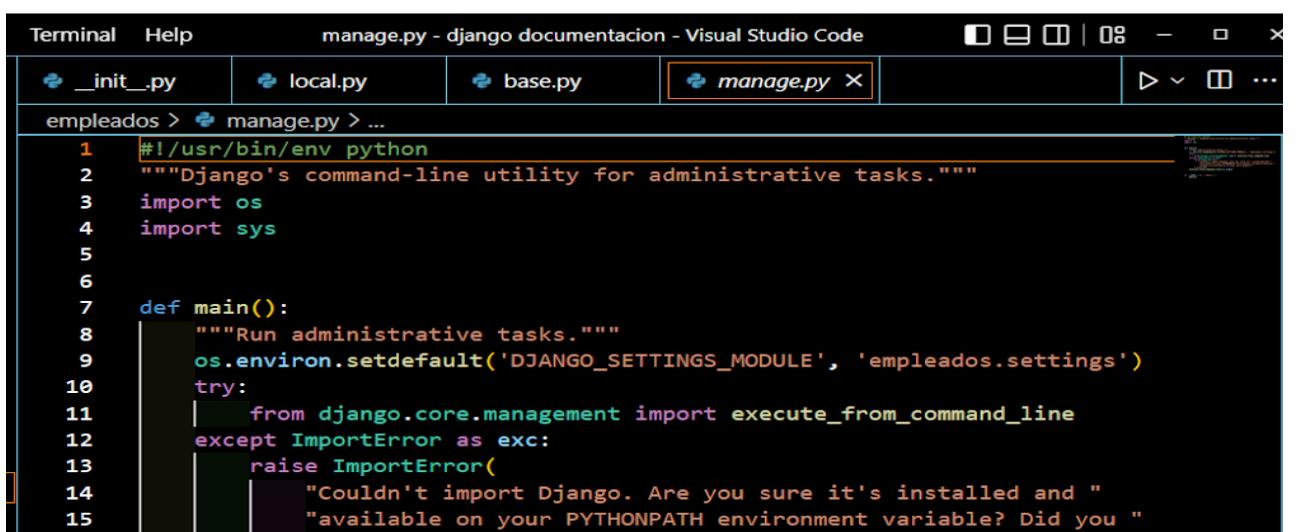
The screenshot shows the Visual Studio Code interface with the following details:

- Terminal:** Help
- File:** local.py - proyecto-de-cistema-de-gestion-de-certificados-y-pqrs - Visual Studio Code
- Code Editor:** The file local.py is open, showing the following code:

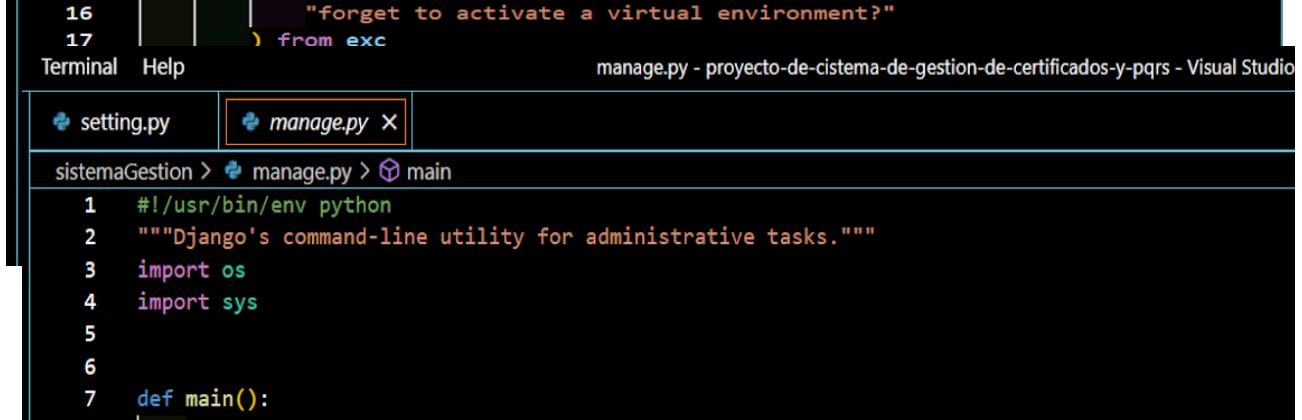
```
1  from .base import *
2
3  # SECURITY WARNING: don't run with debug turned on in production!
4  DEBUG = True
```
- File Explorer:** Shows the project structure: sistemaGestion > sistemaGestion > settings > local.py > ...

5. Luego de estos cambios importantes debemos a nuestro antiguo archivo settings borrarle las para que no quede funcionando y luego indicarle a nuestro archivo manage.py que ahora debe buscar a local.py para lanzar el proyecto

Antes se vería así nuestro manage.py



```
Terminal Help manage.py - django documentacion - Visual Studio Code
__init__.py local.py base.py manage.py
empleados > manage.py > ...
1 #!/usr/bin/env python
2 """Django's command-line utility for administrative tasks."""
3 import os
4 import sys
5
6
7 def main():
8     """Run administrative tasks."""
9     os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'empleados.settings')
10    try:
11        from django.core.management import execute_from_command_line
12    except ImportError as exc:
13        raise ImportError(
14            "Couldn't import Django. Are you sure it's installed and "
15            "available on your PYTHONPATH environment variable? Did you "
16            "forget to activate a virtual environment?"
17        )
18        from exc
```



```
Terminal Help manage.py
setting.py manage.py
sistemaGestion > manage.py > main
1 #!/usr/bin/env python
2 """Django's command-line utility for administrative tasks."""
3 import os
4 import sys
5
6
7 def main():
8
```

Configuración de la carpeta templates:

1. Lo primero que debemos hacer es cambiar el nombre de la carpeta principal del proyecto para evitar conflictos, ya que por defecto django crea dos carpetas con el mismo nombre.
2. Luego a la altura dentro de la carpeta del proyecto donde se encontraba el archivo settings.py que modificamos vamos a crear una carpeta llamada templates
3. Luego vamos a nuestro archivo base.py y nos dirigimos a nuestro apartado templates en el lugar donde dice dirs[] y dentro de los corchetes escribiremos,

```
'DIRS': [BASE_DIR/'templates'],
```

4. Ahora como hemos modificado settings esta esta bajo una carpeta y necesitara buscar mas profundo para encontrar los templates, por ello vamos a donde esta el comando que se encarga de buscar archivos y le damos la instrucción que busque más allá de la siguiente manera

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

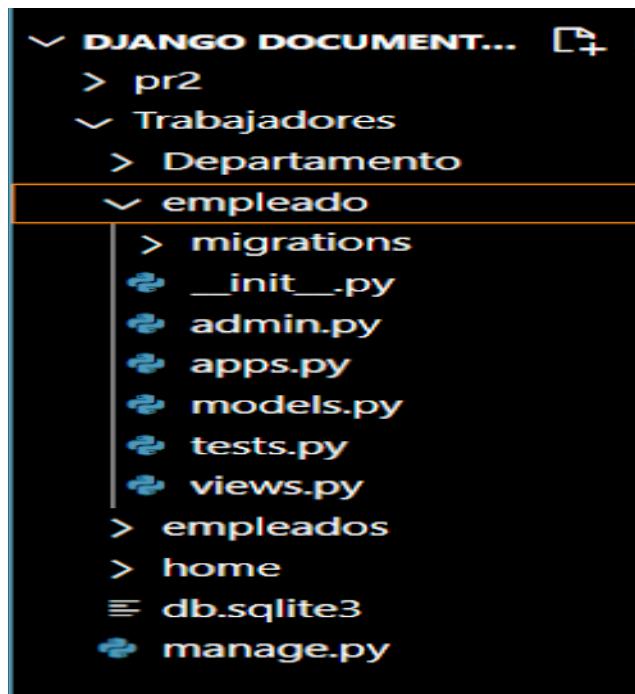
Así se encuentra a una primera vista y añadiremos un parent mas así

```
BASE_DIR = Path(__file__).resolve().parent.parent.parent
```

Instalación de aplicaciones:

Es una buena practica trabajar sobre aplicaciones, nos facilita organizarnos mas y saber que estamos haciendo en el proyecto y diversificar el trabajo, cuando creas una aplicación vienen unos nuevos archivos que si bien puedes crear manualmente no es recomendable.

1. Ir al cmd a la altura del manage.py y ejecutar el comando py manage.py startapp (y seguido de un espacio el nombre de la app que vas a crear)
2. Luego de eso veras que se crea una carpeta con los siguientes archivos



En este caso me he creado una app llamada empleado y viene con los siguientes archivos

Admin.py: en donde cargaremos las instrucciones para esta aplicación en el administrador incorporado con django que veremos más adelante.

Models.py: este archivo es donde crearemos los modelos de la base de datos (tablas y campos) usando código Python que el ORM de Django traducirá a SQL.

Views.py: este archivo es donde pondremos la lógica que llevaran cada una de nuestras páginas web, desde aquí haremos consultas, eliminaciones, ediciones y registros a nuestra base de datos y las mandaremos a nuestras páginas

web entre miles de funcionalidades más.

3. Por ultimo debemos añadir nuestra app a las apps instaladas asi:
 - A. Nos dirigimos a base.py que se encuentra en la carpeta settings que creamos antes
 - B. Nos dirigimos a la instancia installed_apps y allí agregaremos empleado asi

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # apps nuestras
    'empleado'
]
```

Creación de la primera vista:

Para esto nos iremos a el archivo views.py en la app empleado y veremos que ya hay importado un modulo que es

```
from django.shortcuts import render
```

que es el encargado de digitalizar la información en código Python y HTML y lo demás necesario para mostrar una página web completa.

Para comenzar trabajaremos con vistas basadas en clases usando vistas que nos provee Django para hacer esto más fácil que son las vistas genéricas, así:

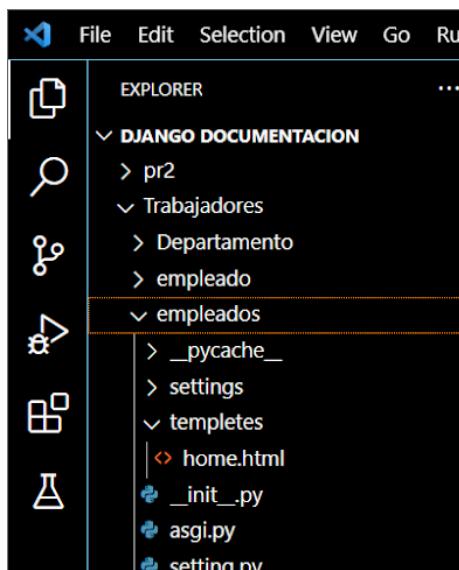
1. Primero debemos importar estas vistas genéricas de los módulos de django

```
from django.views.generic import *
```

2. Luego crearemos una clase para la pagina principal que llamaremos home y utilizaremos la vista genérica templeteview así :

```
class Homview(TemplateView):
    template_name = "home.html"
```

3. Es de notar que también hemos colocado una variable que es reservada de django para desde ahí hacer el cargue del archivo html en este caso es home.html que es donde mostraremos un saludo.
4. Pero primero debemos crear dicho archivo en la carpeta templete que nos hemos creado de antemano con el nombre exactamente igual



5. Podríamos pensar que ya está terminado el trabajo, pero no hay que indicarle a Django donde la queremos poner dentro de la red, por lo cual debemos darle una dirección url propia para esta vista, para que al momento de ir al navegador y poner esta dirección nos muestre esta pagina de home. Para esto haremos los siguientes pasos:

- a. Primero crearemos un archivo dentro de la aplicación que llamaremos urls.py
- b. En el haremos la importación del módulo path que es el que administra las url, podemos tomarlo del archivo urls.py que se encuentra en el proyecto principal

```
from django.urls import path  
y también tomaremos de allá el
```

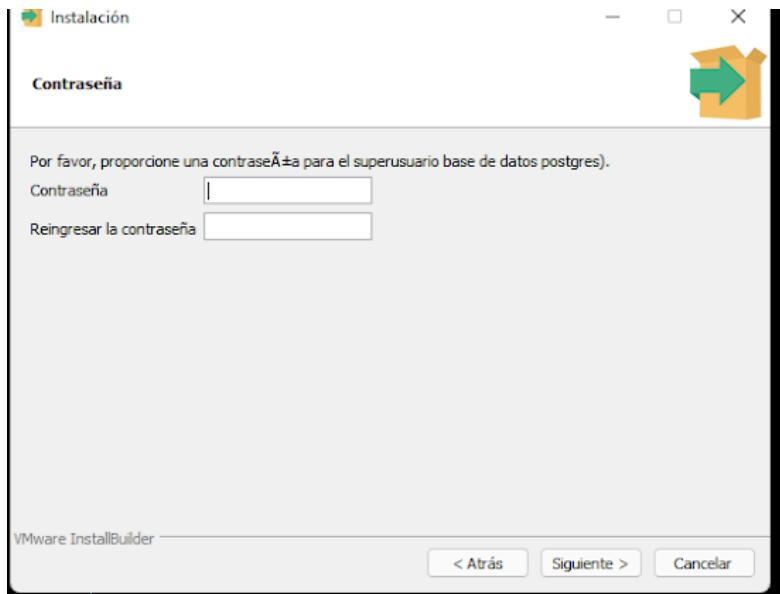
```
urlpatterns = [  
    path('admin/', admin.site.urls),  
]
```

- c. Ahora debemos indicarle a django que debe buscar también en estas direcciones así que vamos al archivo urls.py del principal y le daremos la siguiente instrucción
 - d. `path('empleados/', include('empleado.urls'))`,
aquí le estamos diciendo que en todas las direcciones que comiencen por empleados debe incluir el archivo urls que se encuentra en la carpeta de la app empleados
 - e. pero notamos que include nos muestra en blanco y es porque al lado del path también debemos importar el include así
- ```
from django.urls import path, include
```
- f. ahora si nos iremos a las urls.py de empleados y agregaremos nuestra vista homeview que creamos, para eso remplazaremos el admin que se encuentra ahí y que no necesitaremos, pero primero importaremos nuestro modulo de vistas así
  - g. y ahora si agregaremos la url así
- ```
from .views import *  
path('' ,Homview.as_view()),
```
- aquí estamos indicando que en la dirección empleados en el buscador nos encontraremos con el home por eso entre comillas dejamos un vacío pero como es solo una añadidura de las urls principales y dijimos que todas estas urls empezarían con el nombre de esta app entonces empleados es el nombre que aparecerá en la url y ahora si está completo todo y nos vamos al navegador.
- h. Por último si no lo han hecho iniciaremos el servidor dándole la instrucción en el cmd a la altura de py manage, py manage.py runserver
 - i. Y ya podríamos ir al buscador con la siguiente dirección para ver nuestro home

Base de datos en postgres y creación de modelos:

Al comenzar un proyecto en django por defecto crea una base de datos en sqlite3 la cual va genial para proyectos pequeños y la cual no introduzco aquí porque está en la configuración inicial y no hay que hacer nada para ya usarla, así que añadiré como usar una en postgres, la cual es más robusta y útil para proyectos profesionales de calibre más grande y con alta escalabilidad y seguridad.

1. Primero descargaremos postgres desde el siguiente link:
<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
2. Descargue la versión más actualizada posible, pero evite descargar la última pues puede que aún esté en pruebas y contenga errores.
3. Instale con especial cuidado ya que llegará el momento que nos pide una contraseña la cual es vital para trabajar en postgres y debemos guardarla con mucho cuidado



4. Luego dejaremos el puerto predeterminado 5432
5. Desmarcamos la opción de ir al stat builder del final y damos finalizar
6. Después en las aplicaciones nos vamos al pgadmin que se acaba de instalar y lo abrimos
7. Al abrir nos pide la contraseña que pusimos en la instalación
8. Luego nos vamos al sqlshellen las aplicaciones
9. Ahora ejecutaremos los siguientes pasos: (el nombre de la base de datos, el nombre del usuario y la contraseña son personales y puedes poner lo que quieras pero debes guardar estos datos ya que con ellos enlazaras a django

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:
psql (13.8)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=#
```

aparece entre corchetes

11. en la contraseña del usuario de postgres pondrás la que pusiste en tu instalación y en el pgadmin y saltara una advertencia pero después veremos que ya estamos dentro de postgres

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:
psql (13.8)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=# CREATE DATABASE dbempleado;
CREATE DATABASE
postgres=# CREATE USER maqui;
CREATE ROLE
postgres=# \c dbempleado
Ahora está conectado a la base de datos «dbempleado» con el usuario «postgres».
dbempleado=#
```

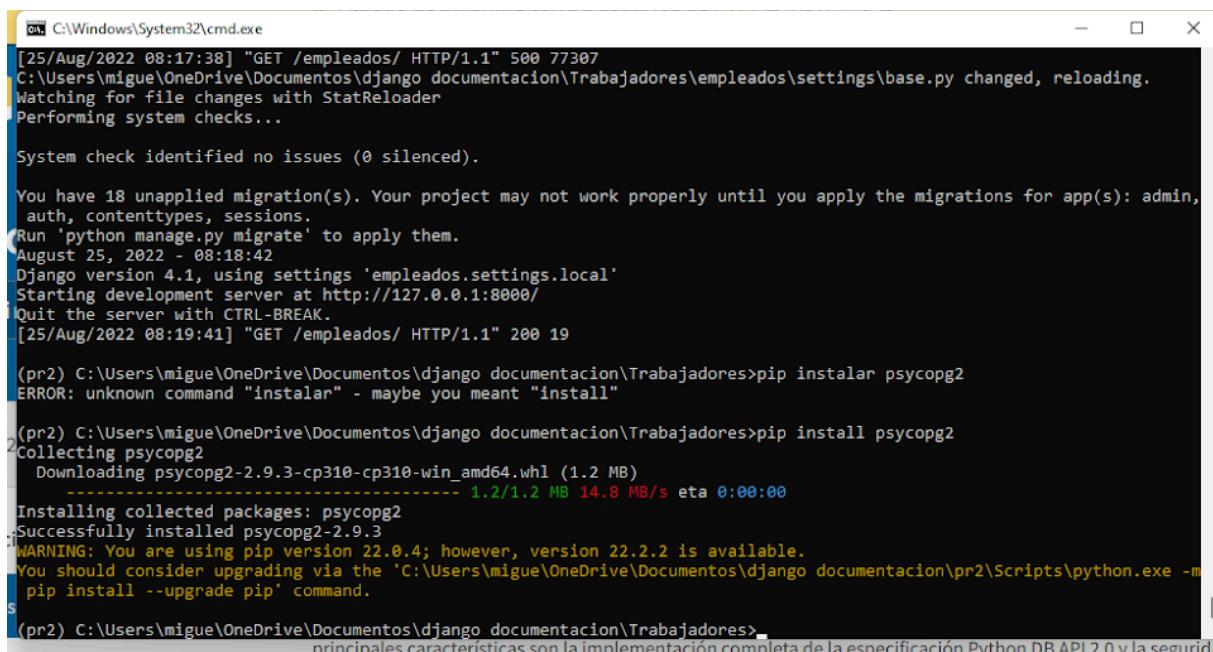
12. a continuación, daremos los comandos para crear la base de datos y el usuario en mi caso es db empleados y el usuario maqui, no olvidar el punto y coma al final siempre de una instrucción
13. luego entraremos en la base de datos que creamos con el comando \c dbempleado (el nombre de la base de datos)
14. por ultimo pondremos una contraseña para este usuario para brindar mas seguridad para este caso la contraseña ira entre comillas simples que será migue para este caso

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:
psql (13.8)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=# CREATE DATABASE dbempleado;
CREATE DATABASE
postgres=# CREATE USER maqui;
CREATE ROLE
postgres=# \c dbempleado
Ahora está conectado a la base de datos «dbempleado» con el usuario «postgres».
dbempleado=# ALTER ROLE maqui WITH PASSWORD 'migue';
ALTER ROLE
dbempleado=#
```

15. ahora debemos instalar una librería que nos ayuda en la conexión con esta base de datos de postgre la cual es psycopg2 que encontraremos en la siguiente pagina
<https://pypi.org/project/psycopg2/>

16. para instalar nos iremos al cmd y lo instalaremos por medio del comando pip en nuestro entorno previamente activado



```
C:\Windows\System32\cmd.exe
[25/Aug/2022 08:17:38] "GET /empleados/ HTTP/1.1" 500 77307
C:\Users\migue\OneDrive\Documentos\django_documentacion\Trabajadores\empleados\settings\base.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 25, 2022 - 08:18:42
Django version 4.1, using settings 'empleados.settings.local'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[25/Aug/2022 08:19:41] "GET /empleados/ HTTP/1.1" 200 19

(pr2) C:\Users\migue\OneDrive\Documentos\django_documentacion\Trabajadores>pip instalar psycopg2
ERROR: unknown command "instalar" - maybe you meant "install"

(pr2) C:\Users\migue\OneDrive\Documentos\django_documentacion\Trabajadores>pip install psycopg2
Collecting psycopg2
  Downloading psycopg2-2.9.3-cp310-cp310-win_amd64.whl (1.2 MB)
    1.2/1.2 MB 14.8 MB/s eta 0:00:00
Installing collected packages: psycopg2
Successfully installed psycopg2-2.9.3
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Users\migue\OneDrive\Documentos\django_documentacion\pr2\Scripts\python.exe -m
pip install --upgrade pip' command.

(pr2) C:\Users\migue\OneDrive\Documentos\django_documentacion\Trabajadores>
```

Tener cuidado con el traductor que pondrá instalar

17. ahora solo falta anclar la base de datos a django en nuestro archivo de configuración que llamamos local.py realizamos cambios a la sección de databases de modo que ahora tenga la información de la conexión a postgre y ya no a sqlite y debe quedar de la siguiente manera: para mi caso que mi db se llama dbempleado, mi usuario es maqui y mi password migue quedaría así (en tu caso solo remplaza esos campos por los tuyos) lo demás quedara siempre igual

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'dbempleado',
        'USER': 'maqui',
        'PASSWORD': 'migue',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

18. y el ultimo paso es ir al cmd y realizar las migraciones por medio del comando py manage.py migrate y ya estaría configurada nuestra base de datos en postgre

crear modelos y tablas de la base de datos

en este apartado veremos como y donde crear las tablas de la base de datos de nuestro proyecto que será en postgre

1. nos dirigiremos al archivo models de nuestras app en este caso empezaremos por una nueva app que llamamos Departamentos ahí crearemos nuestro primer modelo para agregar la tabla departamentos y los campos área y departamento
2. primero crearemos la tabla y en django es muy fácil solo tendremos que crear una clase con el nombre de la tabla y los atributos de la clase serán los campos de esta.
3. Importante que todos deberán heredar la clase models.Model para las tablas y para los campos models seguido del tipo de campo a crear

The screenshot shows the Visual Studio Code interface with the title bar "models.py - django documentacion - Visual Studio Code". The Explorer sidebar on the left shows a project structure under "DJANGO DOCUMENTACION" with files like local.py, models.py (selected), migrations, admin.py, apps.py, tests.py, and models.py. The main editor area displays the following Python code:

```
from django.db import models

# Create your models here.
class Departamento(models.Model):
    area = models.CharField('areas', max_length=50)
    departamento = models.CharField('departamentos', max_length=50)
```

4. El ORM de django traduce esto a SQL, en este caso usamos dos campos de tipo charfield que contendrán caracteres de tipo strings y podrá almacenar hasta 50 caracteres debido al parámetro max_length (nota= no se olviden de agregar su app a instaled apps en base.py)
5. Ahora haremos lo mismo en otro modelo de otra app para probar la relación uno a muchos por medio del campo fk
6. En empleado nos iremos a models.py y crearemos nuestra tabla empleados

The screenshot shows the Visual Studio Code interface with the title bar "models.py - django documentacion - Visual Studio Code". The Explorer sidebar on the left shows a project structure under "DJANGO DOCUMENTACION" with files like pr2, Trabajadores, Departamento, empleado (selected), __pycache__, migrations, __init__.py, admin.py, apps.py, models.py (selected), tests.py, urls.py, views.py, empleados, __pycache__, settings, __pycache__, and __init__.py. The main editor area displays the following Python code:

```
from django.db import models
from Departamento.models import Departamento

# Create your models here.
class Empleados(models.Model):
    job_choices=((('0','Contador'),
    ('1','Administrador'),
    ('2','Economista'),
    ('3','Otros'),
    )
    name = models.CharField('nombres', max_length=70)
    last_name = models.CharField('apellidos', max_length=70)
    job = models.CharField('trabajos', max_length=50,
    choices=job_choices)
    departamento = models.ForeignKey(Departamento, on_delete=models.CASCADE)
```

7. Así quedaría nuestra tabla para añadir empleados, nótese que creamos una variable en la que pasamos una tupla para el campo joba si le damos una selección de un grupo de trabajo ya prestablecido para así ahorrar espacio en la base de datos así solo ocuparíamos un carácter en vez de lo que ocuparía escribir todo el campo, en realidad el max:length debería ser de 1
8. También hay que notar que se uso una relación por medio de

```
departamento = models.ForeignKey(Departamento,
on_delete=models.CASCADE)
```

se coloca que es una llave foránea y dentro de los paramatros primero agregamos a que lo vamos a relacionar, en este caso seria a Departamento y como pertenece a otro archivo debemos importarlo así

```
from Departamento.models import Departamento
```

y es obligatorio usar la forma en que se va a borrar en este caso de tipo cascada que significa que al borrar departamento se borrara la información de este campo en esta tabla también para no tener datos sueltos.

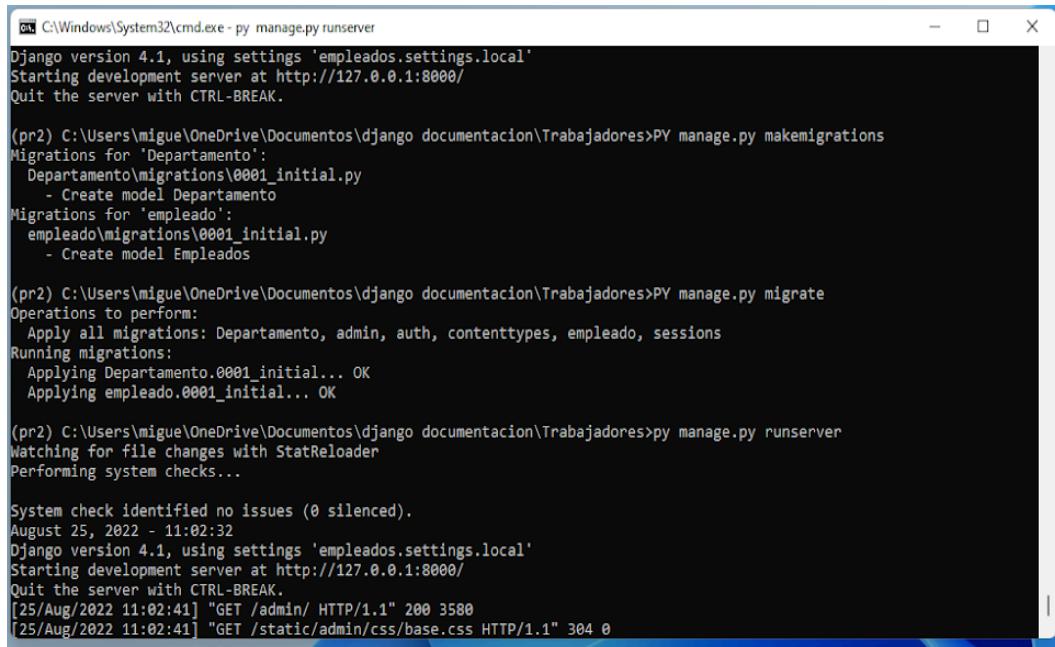
9. Ahora veremos también las relaciones muchos a muchos para esto creamos una tabla en empleado que se llamara Habilidades y como un empleado puede tener muchas habilidades y una habilidad repetirse en varios empleados usaremos este tipo de relación que en django se llama many to many y django automáticamente gestiona la tabla intermedia que nos tocaría obligatoriamente crear en cualquier otro modelo en sql

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda
- Title Bar:** models.py - proyecto django udemy - Visual Studio Code
- Explorer View:**
 - PROYECTO...
 - explorador
 - empleados
 - applications
 - departamento
 - empleados
 - _pycache_
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py (selected)
 - tests.py
 - views.py
 - home
 - espacio1
 - activado.bat
 - instal django.bat
- Code Editor:**

```
4     from applications.departamento.models import Departamento
5     # Create your models here.
6
7     class Habilidades(models.Model):
8         habilidad = models.CharField('Habilidad', max_length=50)
9         class meta:
10             verbose_name='Habilidad'
11             ordering=['habilidad']
12             def __str__(self) -> str:
13                 return self.habilidad
14
15
16     class Empleado(models.Model):
17         """Modelo para la tabla empleado"""
18         job_choices=((0,'Contador'),
19                      ('1','Administrador'),
20                      ('2','Economista'),
21                      ('3','Otro'))
22
23         first_name=models.CharField('nombres',max_length=60)
24         last_name=models.CharField('apellidos', max_length=50)
25         job = models.CharField('trabajos', max_length=50,choices=job_choices)
26         departamento=models.ForeignKey(Departamento,on_delete=models.CASCADE)
27         #imagen = models.ImageField( upload_to='empleado', height_field=None, blank=True,null=True)
28         habilidades = models.ManyToManyField(Habilidades)
29
30         def __str__(self) :
31             return self.first_name +" "+ self.last_name
```

10. Siempre que se realicen cambios en los archivos models de cualquier aplicación debemos hacer las migraciones y para ello tenemos dos comandos que son



```
C:\Windows\System32\cmd.exe - py manage.py runserver
Django version 4.1, using settings 'empleados.settings.local'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion\Trabajadores>PY manage.py makemigrations
Migrations for 'Departamento':
  - Create model Departamento
Migrations for 'Empleado':
  - Create model Empleados

(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion\Trabajadores>PY manage.py migrate
Operations to perform:
  Apply all migrations: Departamento, admin, auth, contenttypes, empleado, sessions
Running migrations:
  Applying Departamento.0001_initial... OK
  Applying empleado.0001_initial... OK

(pr2) C:\Users\migue\OneDrive\Documentos\django documentacion\Trabajadores>py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 25, 2022 - 11:02:32
Django version 4.1, using settings 'empleados.settings.local'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[25/Aug/2022 11:02:41] "GET /admin/ HTTP/1.1" 200 3580
[25/Aug/2022 11:02:41] "GET /static/admin/css/base.css HTTP/1.1" 304 0
```

Makemigrations y migrate en ese orden

crear super usuario

el super usuario nos sirve para entrar al administrador de django desde donde modificaremos las bases de datos y daremos permisos a mas usuarios y otorgaremos permisos, etc.

1. Primero nos iremos a nuestro cmd a la altura de mana.py y con nuestro entorno activado daremos la instrucción py manage.py createsuperuser
2. Lo segundo es seguir los pasos escoger tu usuario, proporcionar un correo valido y una contraseña, recordarlos porque con ellos accederas al administrador como veremos a continuación

```
C:\Windows\System32\cmd.exe
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(pr2) C:\Users\migue\OneDrive\Documentos\django_documentacion\Trabajadores>py manage.py createsuperuser
Username (leave blank to use 'migue'):
Email address: migue2807.maqd@gmail.com
Password:
Password (again):
Superuser created successfully.

(pr2) C:\Users\migue\OneDrive\Documentos\django_documentacion\Trabajadores>
py
```

Ahora py manage.py runserver he iremos a la dirección <http://127.0.0.1:8000/admin>
Y nos saldrá el login

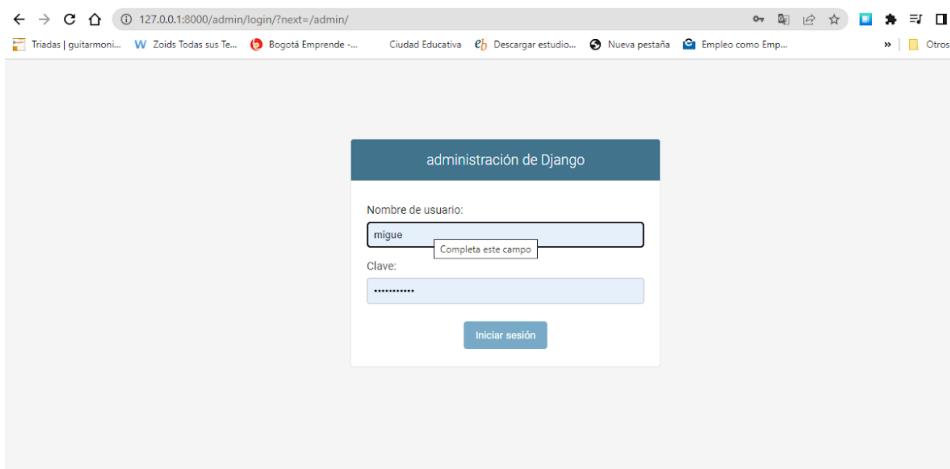
Administrador

Es una de las herramientas mas poderosas que ofrece django en la que podremos hacer infinidad de tareas como, administrar la base de datos, agregar usuarios, dar permisos o restringirlos, etc.

Comenzaremos por ver como se visualizarán las bases de datos que creamos desde el administrador con el super usuario que nos creamos anteriormente.

Para ello iremos a la dirección <http://127.0.0.1:8000/admin/>

Y nos encontraremos algo así



Luego de entrar veremos algo así

The screenshot shows the Django Admin interface for the 'AUTENTICACION Y AUTORIZACION' app. On the left, there are two main sections: 'Grupos' and 'Usuarios'. Each section has two buttons: '+ Agregar' (Add) and 'Cambio' (Change). The 'Grupos' section is currently selected, indicated by a blue header bar. The 'Acciones recientes' (Recent actions) sidebar on the right shows a single entry: 'Mis acciones' (My actions) with the message 'Ninguno disponible' (None available).

AUTENTICACION Y AUTORIZACION	
Grupos	+ Agregar Cambio
Usuarios	+ Agregar Cambio

Acciones recientes

Mis acciones

Ninguno disponible

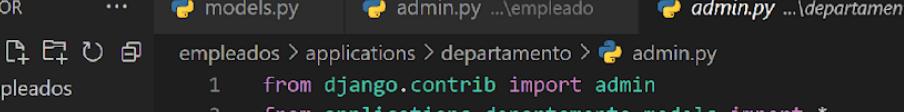
Ahora vamos a agregar funcionalidad a nuestro administrador y lo haremos así:

1. Iremos a nuestras aplicaciones instaladas e iremos a los archivos admin de ambas respectivamente y escribiremos la siguiente línea de código

The screenshot shows the Visual Studio Code interface with the following details:

- Project Explorer:** Shows the project structure:
 - PROYECTO...
 - empleados
 - applications
 - departamento
 - empleado
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
- Editor:** The file `admin.py` is open, showing the following code:

```
from django.contrib import admin
from applications.empleado.models import Empleado,Habilidades
admin.site.register(Empleado)
admin.site.register(Habilidades)
```
- Status Bar:** Shows "admin.py - proyecto django udemy - Visual Studio Code".

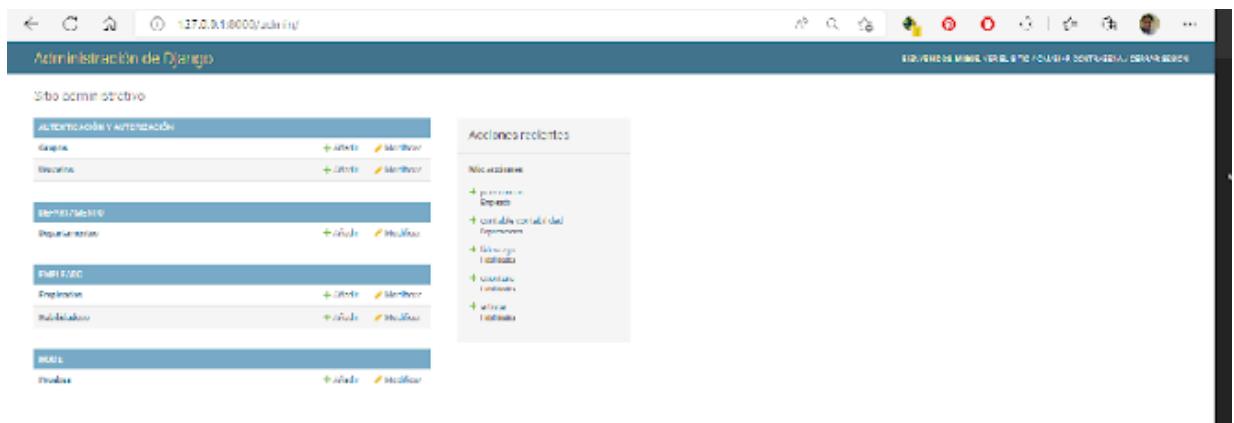


The screenshot shows the PyCharm IDE interface with the following details:

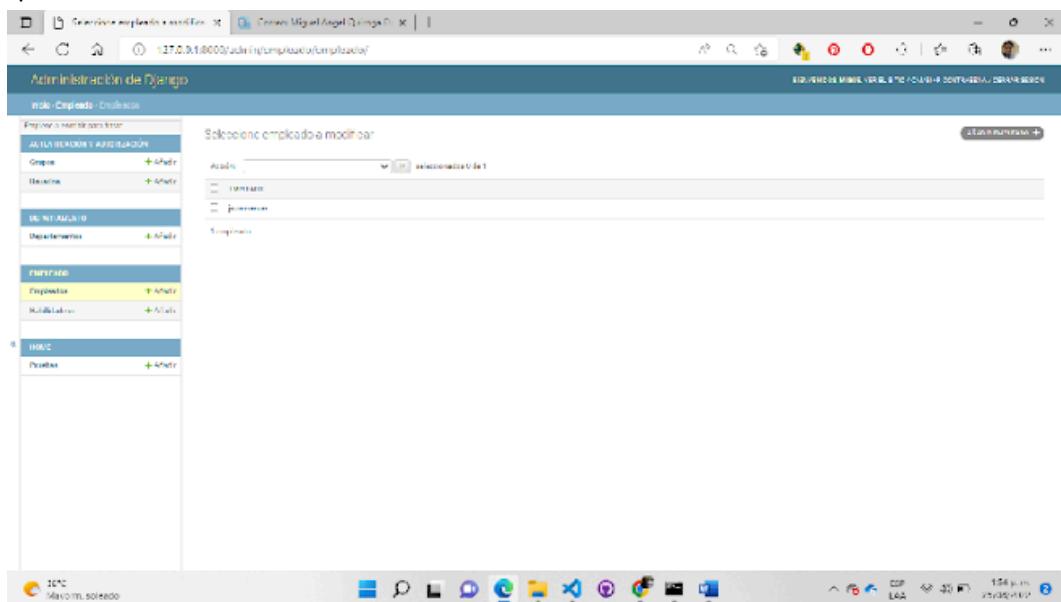
- Toolbar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Project Explorer:** PROYECTO... (containing empleados, applications, departamento, __pycache__, migrations, __init__.py, admin.py, apps.py, models.py, tests.py).
- Code Editor:** The file `admin.py` is open in the editor. The code is as follows:

```
1 from django.contrib import admin
2 from applications.departamento.models import *
3 # Register your models here.
4 admin.site.register(Departamento)
5
6
7
```
- Status Bar:** admin.py - proyecto djan...

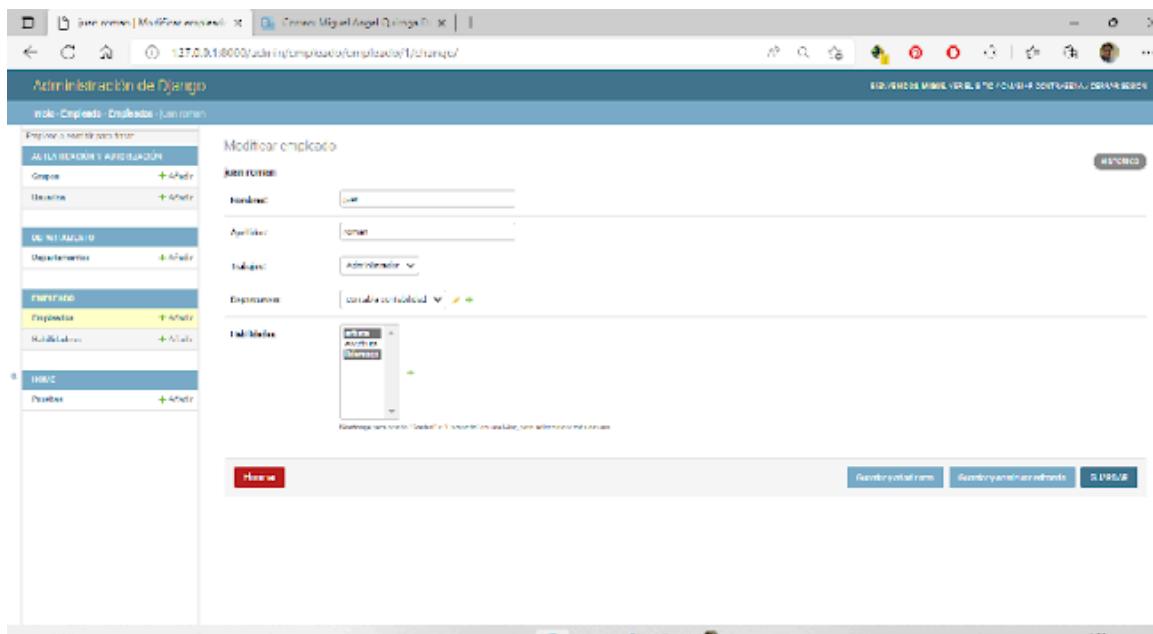
Con esto lograremos que nuestro administrador se vea así ahora



Como vemos ahora aparecen nuestras tablas que creamos separadas por su respectiva aplicación



En esta podemos apreciar las relaciones de uno a muchos con departamento y de muchos a muchos en habilidades donde me permite añadir varias habilidades



2. Ahora si deseamos ver nuestra información de manera mas amplia podemos hacer una especie de tabla para observar más información, para eso iremos a el admin.py y para esta ocasión el de empleado para agregar una clase que tomara herencia de admin y ModelAdmin para dar esta apariencia pondremos de nombre a la clase el nombre del modelo y Admin y luego los argumentos así:

```
class EmpleadoAdmin(admin.ModelAdmin):
```

luego para lograr la tabla nos aprovecharemos de un atributo que es

```
list_display= ('first_name', 'last_name', 'departament', 'job', )
```

tener en cuenta que los campos que colocamos deben coincidir perfectamente con los colocados en el modelo y por ultimo movemos el register debajo de la clase pero sin identacion y agregamos en los parámetros nuestra clase de modo que en total quede así

The screenshot shows a Visual Studio Code interface with several tabs open. The main editor tab contains Python code for defining an `EmployeeAdmin` class:

```
from django.contrib import admin
from applications.empleado.models import empleado,Habilidades

class empleadoAdmin(admin.ModelAdmin):
    list_display = ('first_name','last_name','departamento','job')
admin.site.register(empleado,Habilidades)
admin.site.register(empleado, EmployeeAdmin)
```

To the right of the code editor is a file tree showing the project structure:

- espacio
- applications
- departamento
- empleado
- migrations
- __init__.py
- admin.py
- models.py
- serializers.py
- views.py
- home
- empleados
- simplest-home
- email.html
- home.html
- Touch.html
- changes.html
- manager.html
- espacio.html
- actividad.html
- entrega.html

Below the code editor is a browser window displaying the Django Admin interface for the `Employee` model. The URL is `127.0.0.1:8000/admin/empleado/employee/`. The page title is "Administración de Django". The left sidebar shows the "Empleados" section with "1 EMPLEADO" listed under "DEPARTAMENTO". The main content area is titled "Seleccione empleado a modificar" and lists four employees:

Nombre	Apellido	Departamento	Nombre
antonio	lopez	DEPARTAMENTO	antonio
perez	lopez	DEPARTAMENTO	perez
anton	lopez	DEPARTAMENTO	anton
jose	lopez	DEPARTAMENTO	jose

Y este seria el resultado que obtendríamos

3. Ahora para añadir filtros debemos hacer lo siguiente

- a. Buscador: en la misma clase de EmpleadoAdmin para este caso y luego del atributo list_display añadiremos otro atributo llamado search_fields y una tupla con el nombre del campo a buscar entre comillas y no olvidar la coma después de las comillas

```

    Arriba Editar Eliminar Vie Ir Ejecutar Terminal Ayuda
    admin.py - proyecto_django_ufree - Visual Studio Code
    empleados > applications > empleado > admin.py > empleadoAdmin
    1 from django.contrib import admin
    2
    3 from applications.empleado.models import empleado, habilidades
    4
    5
    6 # admin.site.register(Habilidades)
    7 class empleadoAdmin(admin.ModelAdmin):
    8     list_display = ('first_name', 'last_name', 'departamento', 'job')
    9     search_fields = ('first_name')
   10
   11 admin.site.register(empleado, empleadoAdmin)
  
```

- b. Filtros laterales por campo: mismo procedimiento anterior solo que el atributo ahora será list_filter

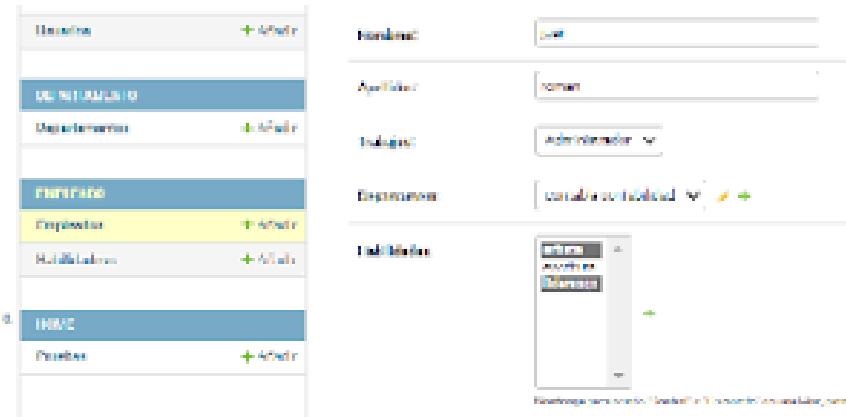
```
list_filter= ('job',)
```

Nombre	Apellido	Departamento	Trabajo
pedro	lopez	contabilidad	Otro
anton	lopez	contabilidad	Administrador
luis	lopez	contabilidad	Administrador

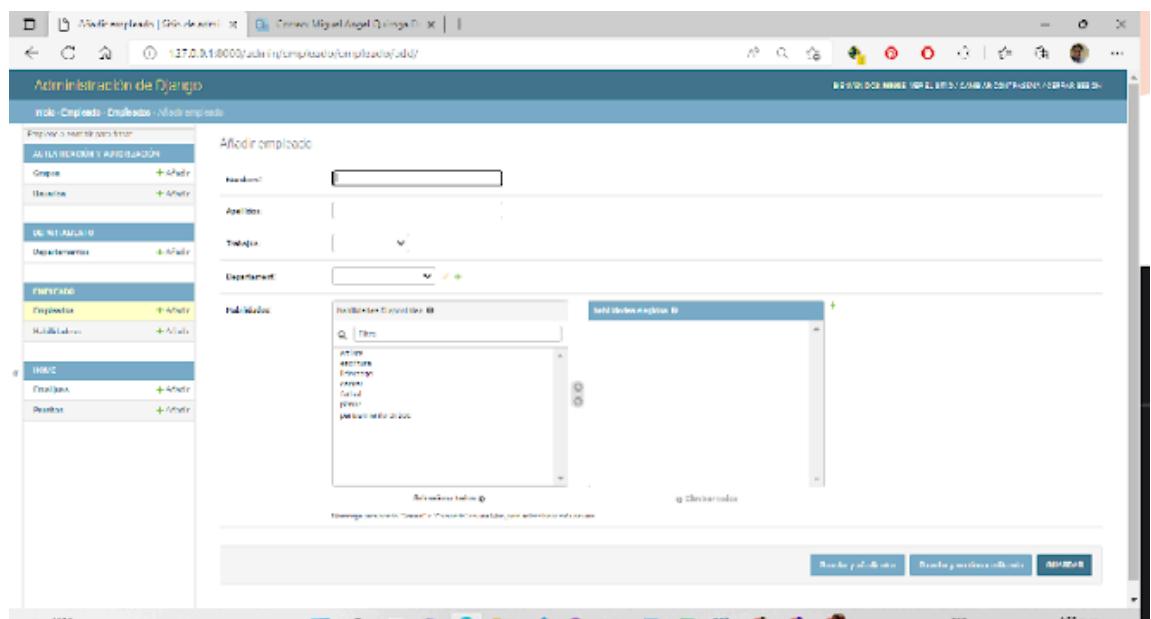
Ahora vemos a la derecha el filtro por trabajos

- c. Ahora las habilidades seria mejor que apareciera un seleccionador mas amigable para los campos de muchos a muchos y para eso usaremos un filtro horizontal

Antes se veía así



Y ahora se verían así



- C. Ahora si queremos agregar una columna mas a la tabla pero que no pertenece a ning n campo en nuestro modelo, lo que haremos es
 - I. A adir el campo a la tupla en list_display el campo que queremos a adir
 - II. Pero como este campo no pertenece al modelo saldr a un error, pero esto lo corregiremos por medio de un m todo(funci n) que llamaremos igual al campo que estamos a adiendo y pasaremos por par metro (self,obj)
 - III. Que retorne lo que queremos mostrar, para nuestro ejemplo uniremos el nombre completo uniendo dos campos

```

    from django.contrib import admin
    from applications.empleado.models import empleado, habilidades

    class empleadoAdmin(admin.ModelAdmin):
        list_display = ('first_name', 'last_name', 'departamento', 'job', 'full_name')
        search_fields = ('first_name')
        list_filter = ('job',)
        filter_horizontal = ('habilidades',)

        # Método para que funcione el campo adicional que no pertenece a los campos en el modelo
        def full_name(self, obj):
            return obj.first_name + ' ' + obj.last_name

    admin.site.register(Habilidades)
    admin.site.register(empleadoAdmin)

```

Nombre	Apellido	Cargo	Departamento	Nombre
manuel	ANALISTA	contable contabilidad	Director	manuel
pedro	analista	contable contabilidad	Director	pedro
erick	analista	contable contabilidad	Administrador	erick
juan	analista	contable contabilidad	Administrador	juan

- d. Añadir editor de texto: para este objetivo ya tendremos que instalar apps de terceros así que aprenderemos como
- El primer paso es ir a la pagina del creador de app en este caso tomaremos la ckeditor para añadir un campo amplio de agregar texto, el siguiente es el link <https://django-ckeditor.readthedocs.io/en/latest/>
 - Lo siguiente es buscar el comando de instalación en el cmd activado para nuestro entorno nos vamos a `pip install django-ckeditor`
 - Luego vamos a nuestras aplicaciones instaladas en el archivo base.py que creamos para la configuración y añadiremos la app ckeditors

```

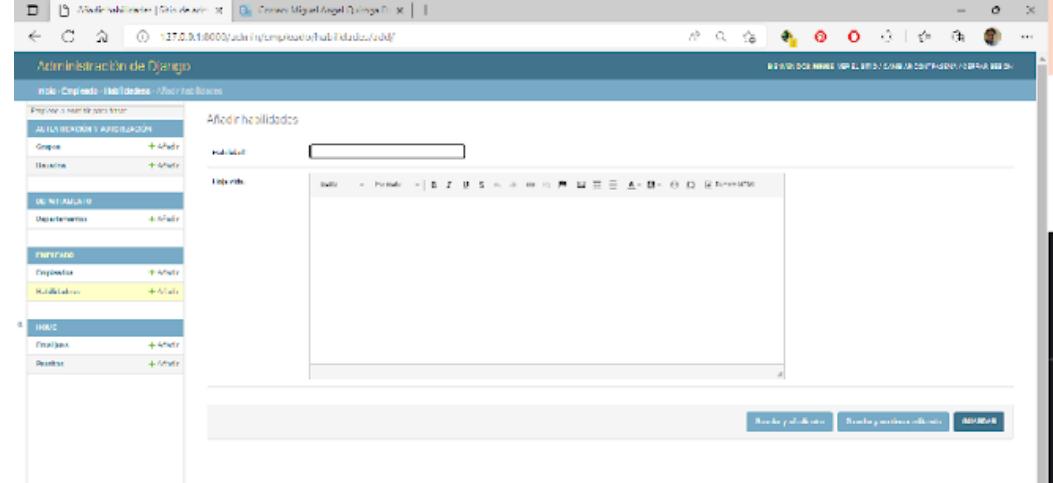
1     from pathlib import Path
2
3     # Build paths inside the project like this: BASE_DIR / 'subdir'.
4     BASE_DIR = Path(__file__).resolve().parent.parent.parent
5
6     # SECURITY WARNING: keep the secret key used in production secret!
7     SECRET_KEY = 'django-insecure-subsetindividuallinksweak0000000000000000'
8
9     # Application definition
10
11    INSTALLED_APPS = [
12        'django.contrib.admin',
13        'django.contrib.auth',
14        'django.contrib.contenttypes',
15        'django.contrib.sessions',
16        'django.contrib.messages',
17        'django.contrib.staticfiles',
18        'apps.departamento',
19        'apps.upload',
20        'applications.departamento',
21        'applications.upload',
22        'applications.home',
23    ]
24
25
26    MIDDLEWARE = [
27        'django.middleware.security.SecurityMiddleware',
28        'django.contrib.sessions.middleware.SessionMiddleware',
29        'django.middleware.common.CommonMiddleware',
30        'django.middleware.csrf.CsrfViewMiddleware',
31        'django.contrib.auth.middleware.AuthenticationMiddleware',
32        'django.contrib.messages.middleware.MessageMiddleware',
33    ]

```

- IV. Y ahora vamos a añadir un campo a nuestro modelo para usar esta app y vamos al campo de habilidad y añadimos un campo que se llame hoja_vida que use esta app de esta manera

hoja_vida=RichTextField()

- V. Hacemos la migración en el cmd como de costumbre cada vez que realizamos un cambio algún modelo por parte de makemigrations y migrate
VI. Luego arrancamos el servidor y encontraremos esto



Ahora se puede apreciar este gran campo de texto donde podemos agregar contenido y todas las funcionalidades

Vistas genéricas:

En este sitio se encontrara todas las formas de vistas genéricas y sus funciones. Este documento contendrá solo algunas de ellas

<https://ccbv.co.uk/>

listview: itera listas y bases de datos útiles para mostrar información en tablas.

Atributos:

- a. Template_name = 'lista.html' con este atributo le decimos a nuestra vista basada en clase donde y como se llama la plantilla html en que mostraremos la información
- b. Model =empleados | aquí ponemos el modelo que iremos a iterar y hace una consulta de esta tabla a la base de datos
- c. context_object_name = 'lista' genera el contexto que le pasara a la página html para poder hacer llamado de los objetos de la base de datos, luego este string colocado aquí lo pondremos en html entre llaves y podremos recorrerlo por medio de un ciclo for