

Periodo: FEBRERO – JUNIO 2024 Pregrado Cali
Asignatura: 750009C - Desarrollo de Software I
Grupo: 01 y 80
Profesora: Beatriz Florián Gaviria, Ph.D. (Profesora Asociada EISC)
Monitor: Santiago Duque Chacón (Ingeniería de Sistemas, 6to. semestre)

Guía taller práctico 1: Back-end con Django

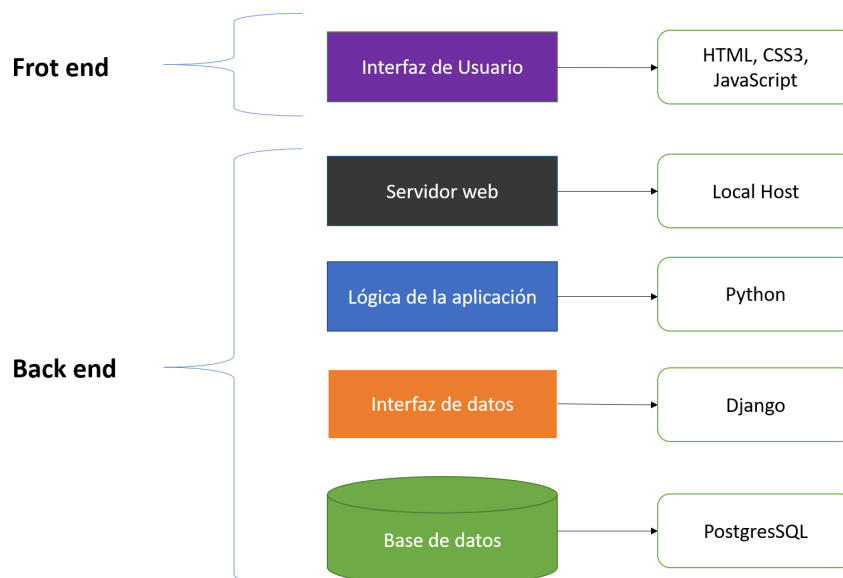
Introducción

Esta guía tiene como objetivo presentar a los estudiantes conceptos básicos y un ejemplo sencillo sobre el trabajo con el framework Django en el proceso de desarrollo de una aplicación desde la capa del Back-End.

De la misma manera, se establecerá el trabajo a realizar a lo largo del semestre con talleres adicionales que se complementarán para dar lugar a una aplicación simple, para que puedan aplicar los distintos conceptos y prácticas enseñadas en el desarrollo del proyecto de este curso.

Arquitectura de la aplicación y tecnologías a usar

Para la aplicación que se buscará desarrollar a lo largo de estos talleres, se seguirá una arquitectura monolítica usando las tecnologías correspondientes a: PostgreSQL como motor de la base de datos, Python con el framework Django en el backend y React en conjunto con la librería Material UI en el frontend. Esta arquitectura se podría ver de manera simplificada así:



Modelo de datos

Para este taller, se realizará el módulo correspondiente a una entidad de Usuario, usando como base el modelo nativo de Django *AbstractBaseUser*, el cuál ya incluye importantes herramientas para la autenticación y seguridad de usuarios en una aplicación. Este modelo tiene campos como 'first_name', 'last_name', y demás que son comúnmente usados en todas las aplicaciones por parte de los módulos correspondientes a los usuarios.

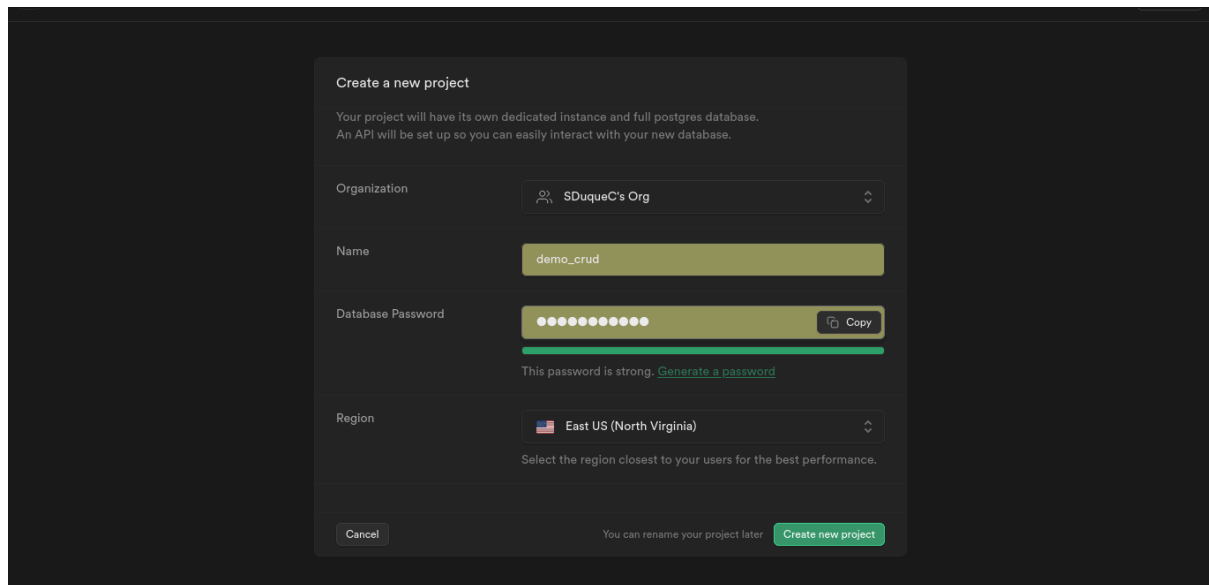
Este modelo fue construido sabiendo la estructura del modelo de Django que estamos usando como base, al cuál se le pueden dar nuevos campos en función de las necesidades del proyecto, siendo una herramienta bastante conveniente que brinda este framework.

Es importante mencionar que otra ventaja de usar esta manera de trabajo es la capacidad de manejar distintos grupos de usuarios en el contexto de la aplicación de una manera sencilla a través de las interfaces que provee y ya tiene implementadas Django con este módulo.

User	
<i>id</i>	<i>int</i>
<i>password</i>	<i>varchar</i>
<i>last_login</i>	<i>timestamp</i>
<i>is_superuser</i>	<i>boolean</i>
<i>username</i>	<i>varchar</i>
<i>firs_name</i>	<i>varchar</i>
<i>last_name</i>	<i>varchar</i>
<i>email</i>	<i>varchar</i>
<i>is_staff</i>	<i>boolean</i>
<i>is_active</i>	<i>boolean</i>
<i>date_joined</i>	<i>timestamp</i>

En este taller, usaremos PostgreSQL a través de la herramienta [Supabase](#), la cual nos permite alojar bases de datos en la nube, simplificando el proceso general de desarrollo.

Tras seguir el proceso de registro de la página, crearemos un nuevo proyecto con el nombre que daremos a la base de datos, además de la contraseña que se usará para la conexión con esta. Esta contraseña debe ser almacenada en un lugar seguro, pues será necesaria más adelante.



Create a new project

Your project will have its own dedicated instance and full postgres database.
An API will be set up so you can easily interact with your new database.

Organization: SDuqueC's Org

Name: demo_crud

Database Password: [masked] [Copy](#)

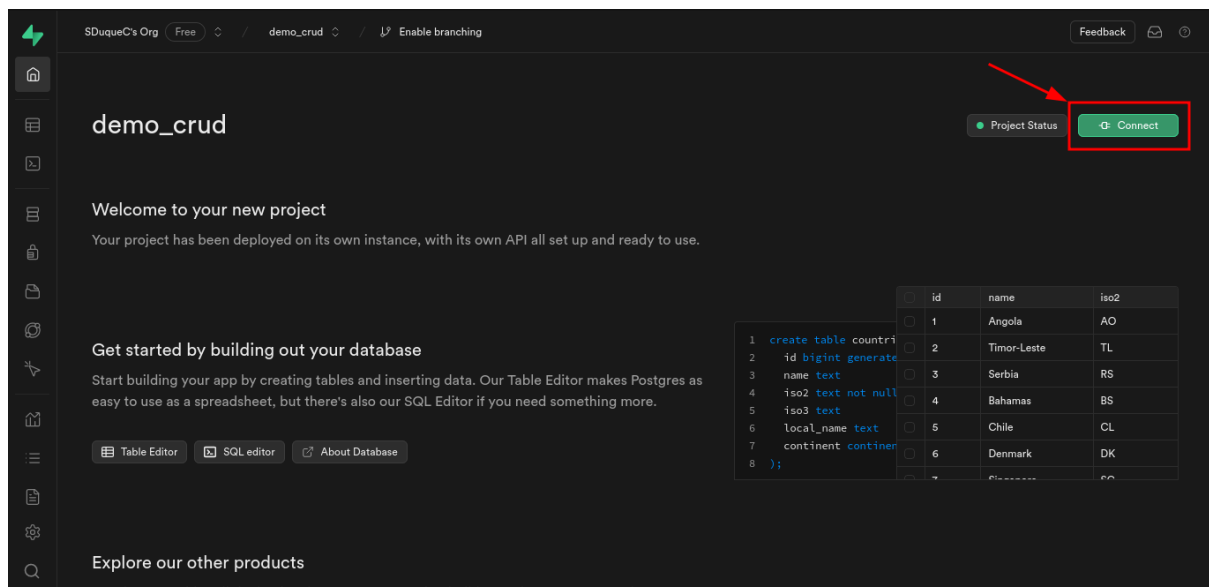
This password is strong. [Generate a password](#)

Region: East US (North Virginia)

Select the region closest to your users for the best performance.

[Cancel](#) [Create new project](#)

Seguido a esto, abriremos las opciones de conexión con el botón 'connect':



SDuqueC's Org Free / demo_crud / Enable branching

demo_crud

Project Status [Connect](#)

Welcome to your new project

Your project has been deployed on its own instance, with its own API all set up and ready to use.

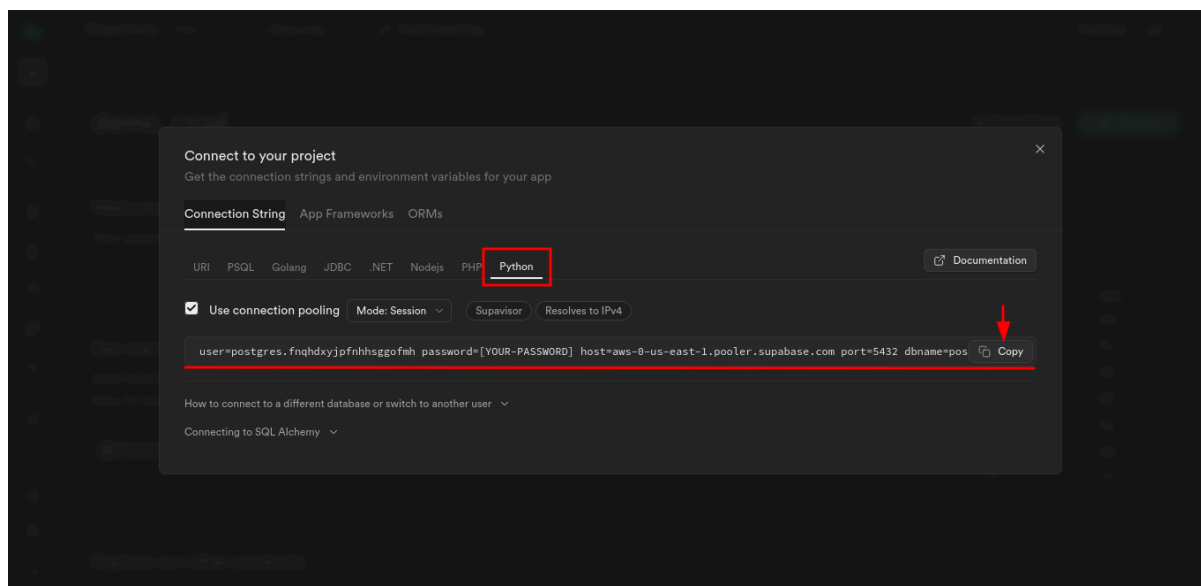
Get started by building out your database

Start building your app by creating tables and inserting data. Our Table Editor makes Postgres as easy to use as a spreadsheet, but there's also our SQL Editor if you need something more.

[Table Editor](#) [SQL editor](#) [About Database](#)

id	name	iso2
1	Angola	AO
2	Timor-Leste	TL
3	Serbia	RS
4	Bahamas	BS
5	Chile	CL
6	Denmark	DK

Y guardaremos la información que nos muestra en el apartado específico para Python, ya que estos datos serán los que usaremos en el proyecto de Django para configurar la conexión con la base de datos: (en el campo password, se debe sustituir [YOUR-PASSWORD] por la contraseña que establecimos al crear el proyecto de Supabase)



Tras seguir estos pasos, estaría lista la base de datos que usaremos en el proyecto.

Estructura del backend

En un proyecto realizado usando Django, es importante reconocer la estructura que siguen los archivos que pertenecen a este. Esta estructura parte del directorio en el que se trabajará, y posteriormente hay 2 tipos de directorios importantes que se generan cuando trabajamos con Django: una carpeta que representa un DjangoProject y carpetas para apps dentro del proyecto Django.

Para tener un mayor entendimiento sobre la diferencia entre los conceptos de Proyecto y App en el contexto de desarrollo con Django, se recomienda el contenido de este [enlace](#).

Instalación

Al abrir una terminal, se verifica si Django ya está instalado con el comando

```
Unset  
python -m django --version
```

En caso de que no lo esté, instalamos Django:

Unset

```
python -m pip install Django
```

Adicionalmente, instalaremos una librería que permite el trabajo con PostgreSQL:

Unset

```
pip install psycopg2-binary
```

Nos dirigimos a la carpeta donde se creará el proyecto e inicializamos un proyecto de Django:

Unset

```
cd /ruta/al/directorio  
django-admin startproject [project_name]
```

Tras iniciar el proyecto de Django, nos dirigimos a la carpeta que se crea, donde verificaremos que se encuentre el archivo manage.py y el directorio específico del proyecto (en este caso, DemoCrud):

Unset

```
cd DemoCrud/  
ls  
DemoCrud  manage.py [Esto es lo que esperamos encontrar]
```

Estando en este directorio, iniciaremos nuestra app demo_crud:

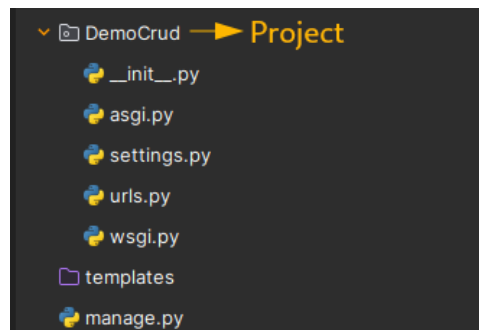
Unset

```
python manage.py startapp demo_crud
```

Tras ejecutar estos comandos, se puede comenzar a trabajar en los archivos específicos para este módulo CRUD.

Archivos del proyecto DemoCrud

En este caso *DemoCrud* es el directorio del DjangoProject, el cual es donde se aloja la configuración y distintos parámetros correspondientes a las conexiones entre las partes del backend, la conexión con la base de datos y las librerías o paquetes que se usarán.



El archivo `__init__.py` no lo modificamos; es el archivo que “marca” un directorio como un paquete de Python.

El archivo `asgi.py` tampoco lo modificaremos; este archivo es usado al desplegar con ASGI.

El archivo `settings.py` es uno de los más importantes en el proyecto, pues es el que contiene la configuración general del proyecto, almacena la información de la conexión con la base de datos, etc.

En este archivo encontraremos lo siguiente:

Python

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]  
  
...  
  
DATABASES = {  
    'default': {  
        'ENGINE': '[YOUR_ENGINE]', # motor de la DB  
        'NAME': '[YOUR_NAME]', # nombre de la DB  
        'USER': '[YOUR_USER]', # usuario de la DB  
        'PASSWORD': '[YOUR_PASSWORD]', # contraseña de la DB
```

```
'HOST' : '[YOUR_HOST]', # host de la DB
'PORT' : '[YOUR_PORT]' # puerto donde se aloja
}
```

Donde en `INSTALLED_APPS` se agrega el nombre de las apps que pertenezcan a nuestro proyecto, así como la librería que acabamos de instalar 'psycopg2', mientras que en `DATABASE` debemos cambiar los campos por la información de conexión que nos dió Supabase, y en el `ENGINE` deberá ir 'django.db.backends.postgresql'.

El archivo `urls.py` también tiene gran relevancia, ya que es donde están las declaraciones de las URL para el proyecto: se puede ver como una “tabla de contenidos” del proyecto.

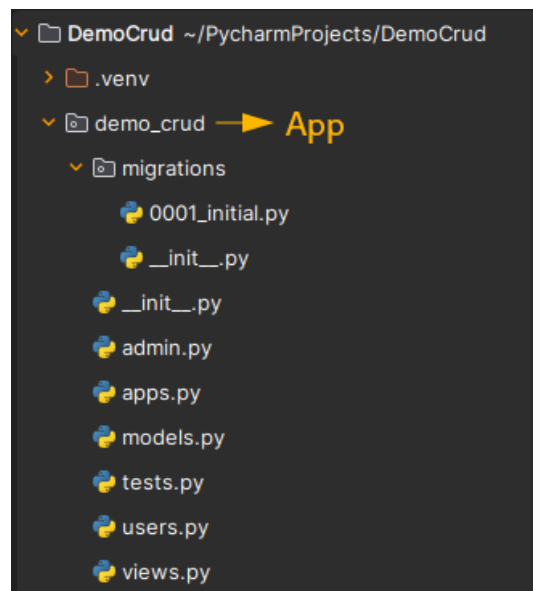
El archivo `wsgi.py` no lo modificaremos; es usado para desplegar con WSGI.

Archivos de la app `demo_crud`

Para este ejemplo tenemos un solo directorio de app, el cual es `demo_crud`, que representa un sub-módulo del proyecto entero, que en este caso será donde tendremos los modelos, operaciones, vistas y migraciones.

El directorio `migrations` es donde se almacenan las migraciones a la base de datos que involucran a los modelos de la app, es decir, los cambios realizados, como la creación de las tablas correspondientes.

El archivo `__init__.py` no se modifica y cumple el mismo propósito que en el proyecto.



El archivo `admin.py` tiene el fin de mostrar los modelos en el panel de administración de Django. Aquí importaremos los modelos que creemos.

El archivo `apps.py` es usado para facilitar el proceso de inclusión de las apps en el proyecto y su configuración.

El archivo `models.py` es donde se establecen los modelos de la app.

El archivo `tests.py` suele utilizarse en la realización de pruebas específicas de la app en que se encuentra.

El archivo `users.py` no es un archivo creado por defecto, en este archivo se definió una clase personalizada de usuario usando `AbstractBaseUser`, que tiene lo siguiente:

```
Python
from django.contrib.auth.models import AbstractBaseUser,
BaseUserManager
from django.db import models

class UserManager(BaseUserManager):
    def create_user(self, email, name, password, phone, is_boss):
        if not email:
            raise ValueError('Users must have an email address')

        user = self.model(
            email=self.normalize_email(email),
            name=name,
            phone=phone,
            is_boss=is_boss
        )

        user.set_password(password)
        user.save(using=self._db)
        return user

class User(AbstractBaseUser):
    # Campos personalizados
    is_boss = models.BooleanField(default=False)

    objects = UserManager()

    USERNAME_FIELD = 'email' # cual va a ser la PK
    REQUIRED_FIELDS = ['name', 'phone']
```


Por último, el archivo `views.py` contiene las vistas, que se refieren a las funciones que toman las peticiones realizadas al backend y del mismo modo devuelven sus respectivas respuestas:

Python

```
from django.shortcuts import render, redirect
from demo_crud.users import *

def ListarUsuarios(request):
    users = User.objects.all()
    return render(request, 'listar_usuarios.html', {'usuarios':
users})

def CrearUsuario(request):
    if request.method == 'POST':
        email = request.POST['email']
        name = request.POST['name']
        password = request.POST['password']
        phone = request.POST['phone']
        is_boss = request.POST['is_boss']
        user = User(email=email, name=name, password=password,
phone=phone, is_boss=is_boss)
        if email.is_valid():
            user.save()
            return redirect('listar_usuarios')
        else:
            return render(request, 'crear_usuario.html')

def EditarUsuario(request, email):
    user = User.objects.get(email=email)
    if request.method == 'GET':
        return render(request, 'editar_usuario.html', {'usuario': user})
    else:
        user.email = request.POST['email']
        user.name = request.POST['name']
        user.password = request.POST['password']
        user.phone = request.POST['phone']
        user.is_boss = request.POST['is_boss']
        user.save()
        return redirect('listar_usuarios')

def EliminarUsuario(request, email):
    user = User.objects.get(email=email)
    if request.method == 'POST':
        user.delete()
        return redirect('listar_usuarios')
```

```
else:  
    return render(request, 'eliminar_usuario.html', {'usuario':  
user})  
# Create your views here.
```

Tras realizar esto, se deben ejecutar los siguientes comandos en el directorio raíz del proyecto (donde está el archivo manage.py) para terminar de configurar el modelo y la base de datos:

```
Unset  
python manage.py makemigrations demo_crud  
python manage.py migrate demo_crud
```

Lo cual realizará de manera efectiva los cambios necesarios en la base de datos.

Conclusiones

En el presente taller, se establecieron algunos de los conceptos básicos que se deben tener en cuenta al trabajar con Django así como se establecieron aspectos referentes al trabajo que se realizará en futuros talleres de manera incremental, logrando complementar las bases para el desarrollo del proyecto de este curso.

Algunos de los conceptos y prácticas abordadas fueron: una manera efectiva de gestionar la base de datos aprovechando un servicio en la nube (Supabase), los pasos para la instalación de Django así como iniciar un proyecto de Django, la diferencia entre un 'Project' y 'App' en el contexto de Django, la estructura que siguen las aplicaciones desarrolladas con Django, el contenido de algunos de estos archivos y qué función tienen dentro de la aplicación.

Enlace al repositorio:

https://github.com/SDuqueC/demo_crud