

# Ingeniería de Software II

## Ingeniería de Sistemas y Computación



### Manual Técnico Versión 1

DIG - Grupo 5

Nombres:

Iván Cáseres

Juan Sebastián Montoya C.

David Ricardo Ríos García

Elkin Duván Ramírez

Juan Sebastián Vargas C.



### ÍNDICE

- Prerrequisitos
- Clonación del proyecto
- Creación de entornos virtuales
- Instalación de librerías
- Creación de la Base de datos PostgreSQL
- Migration
- Migrate
- Conexión
- Inicialización

La información de este manual es de uso PRIVADO y EXCLUSIVO de DIG, usado con fines de divulgación académica en la vigencia del curso de Ingeniería de Software 2 de la Universidad Nacional de Colombia sede Bogotá ÚNICAMENTE durante la vigencia del semestre académico 2022-1, y para integrantes del equipo de desarrollo de DIG. Se prohíbe la distribución parcial o total de este contenido.

## Prerrequisitos

Para la ejecución del proyecto DIG, debe tener en cuenta que la implementación se realizará en lenguajes de programación en uso con unas IDE's definidas:

- Python versión 3.8 (Puede obtenerlo desde la [Microsoft store](#) o página oficial [Python](#))
- PostgreSQL [versión 14](#) NOTA: durante la instalación tenga claro el nombre de administrador y la contraseña que configuró, ya que será necesario los accesos como administrador más adelante en el proyecto. Por defecto se aconseja usar por defecto el usuario: postgres , contraseña: 12345678.
- Visual Studio Code (con la instalación de las extensiones de Python, SQL server y PostgreSQL), (Puede obtenerlo desde la [Microsoft store](#) o [página oficial](#)).
- GitHub [Desktop](#) o [Git console](#)

## Clonación del proyecto

Una vez cumpliendo los prerrequisitos, clone el repositorio almacenado en GitHub alojado en la [organización de DIG](#) (Puede usar git console git clone o la GUI de GitHub Desktop).

Tenga en cuenta que por seguridad y derechos de autor, el repositorio al código fuente es PRIVADO y no podrá acceder a él a menos que tenga una cuenta de usuario la cual se le ha añadido previamente al repositorio de la organización. El acceso al repositorio implica la ACEPTACIÓN de la normatividad respecto a la confidencialidad de la información. Acepta la PROHIBICIÓN de compartir completa o parcialmente información de la base de datos y código fuente a terceros o personal no autorizado. Los cambios (Branch, push, commit) SOLAMENTE son realizados por el equipo de desarrollo con autorización previa, se prohíbe ejecutar cualquier comando de modificación o clonación del repositorio sin autorización, también está TOTALMENTE PROHIBIDO la eliminación de ramas, commits, branches, total o parcial de algún repositorio bajo única excepción de qué esté aprobada por el administrador de DIG. Únicamente el administrador del equipo de desarrollo está habilitado para realizar control de acceso a cuentas de usuario, por lo tanto a excepción del administrador, se PROHÍBE dar acceso a usuarios ajenos a DIG o eliminar el acceso a miembros activos del equipo de desarrollo. Acepta que se atiene a la normatividades vigentes de la legislación Colombiana.

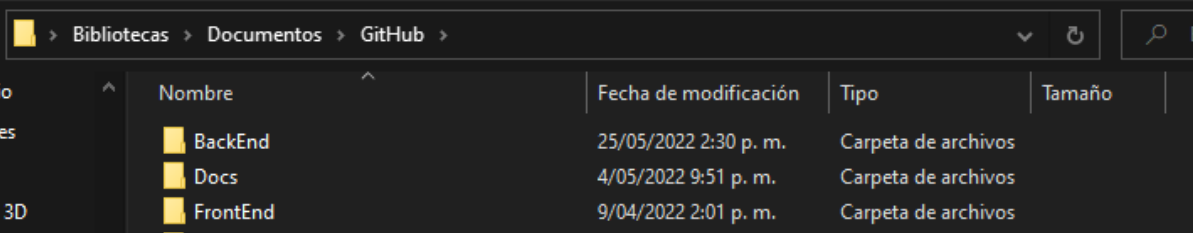
Dentro de la organización encontrará 3 repositorios:

- Backend (Obligatorio): Aquí se lleva todo el desarrollo backend de la aplicación: Base de datos, conectores de base de datos, clases , objetos, clases control las cuales son las encargadas de gestionar peticiones y realizar las operaciones CRUD en la base de datos, así como gestionar

peticiones, permisos y novedades al frontend. [Link de enlace de git para su clonación](#).

- Frontend (Obligatorio): Aquí estarán todas las distintas interfaces de usuario de DIG, es el modelo de vista, el diseño de las GUI están basadas según la paleta de colores previamente establecida en la documentación [Link de enlace de git para su clonación](#).
- Documentación (Opcional): Aquí estará toda la información documental con respecto al desarrollo de la aplicación DIG, así como también todo lo relacionado con documentación de Sprints, directrices y definiciones del proyecto (Product Owner, Product backlog, Paleta de colores, definición de logo, reglas y lógica de negocio), así como también el manual técnico y de usuario en sus distintas versiones de acuerdo con la versión de DIG. [Link de enlace de git para su clonación](#)

Una vez clonados los repositorios, dentro de su biblioteca de archivos debe tener los repositorios contenidos dentro de una misma carpeta, de lo contrario no se tendrá la conexión entre el back y front end de la aplicación.



Nombre	Fecha de modificación	Tipo	Tamaño
BackEnd	25/05/2022 2:30 p. m.	Carpeta de archivos	
Docs	4/05/2022 9:51 p. m.	Carpeta de archivos	
FrontEnd	9/04/2022 2:01 p. m.	Carpeta de archivos	

## Creación de entornos virtuales

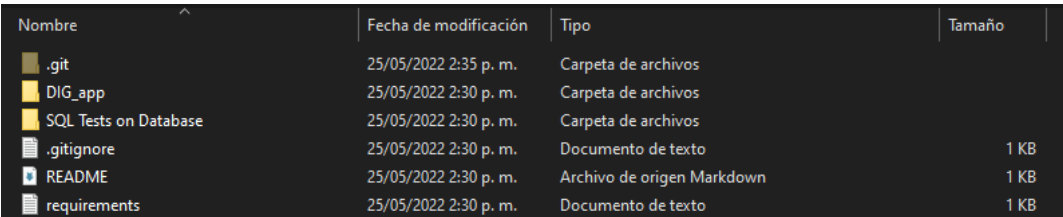
Python desde la versión 3 permite la creación de entornos virtuales, esta creación de entornos virtuales permite aislar las librerías y funciones realizadas de las que están instaladas por defecto en el ambiente de desarrollo de python.

Para poder tener uniformidad y evitar problemas de incompatibilidad, y para el desarrollo fluido y flexible en varios sistemas operativos, se aconseja realizar la creación de entorno virtual y que corra la aplicación bajo el entorno virtual (venv)

- **Creación de entorno virtual en Windows**

Para crear el entorno virtual debe:

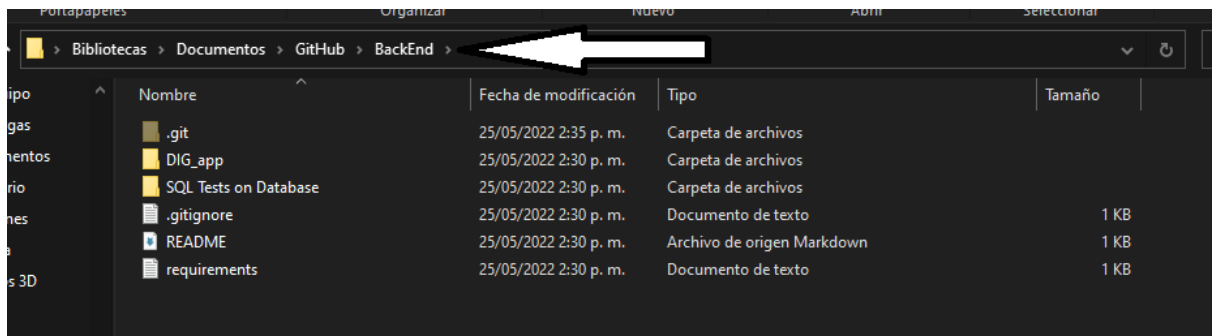
### 1. Ingrese a la carpeta de backend



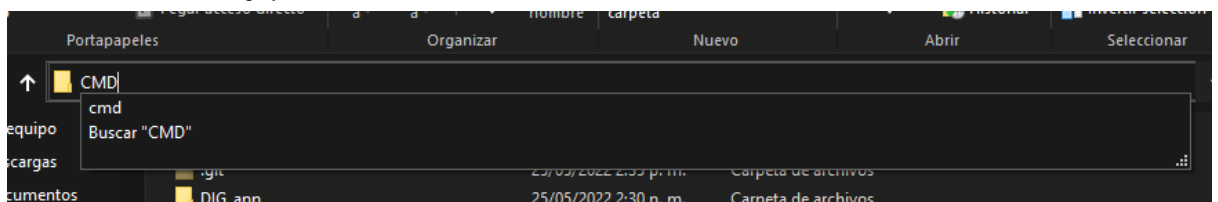
Nombre	Fecha de modificación	Tipo	Tamaño
.git	25/05/2022 2:35 p. m.	Carpeta de archivos	
DIG_app	25/05/2022 2:30 p. m.	Carpeta de archivos	
SQL Tests on Database	25/05/2022 2:30 p. m.	Carpeta de archivos	
.gitignore	25/05/2022 2:30 p. m.	Documento de texto	1 KB
README	25/05/2022 2:30 p. m.	Archivo de origen Markdown	1 KB
requirements	25/05/2022 2:30 p. m.	Documento de texto	1 KB

- 2. Despliegue una ventana de comandos CMD:** Para ello, debe dirigirse a la barra del explorador y escribir CMD y presiona enter, posteriormente se abrirá una consola CMD en la ubicación del repositorio de BackEnd.

2.1 Diríjase a la barra de explorador y haga clic



2.2 Escriba CMD y presione enter



2.3 Aparecerá la ventana de comandos, asegúrese que la ruta mostrada sea la correcta.

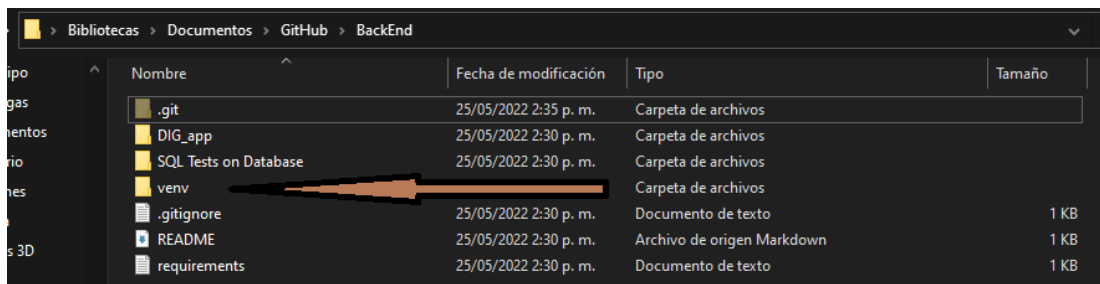
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1706]
(c) Microsoft Corporation. Todos los derechos reservados.
D:\jsmon\Documents\GitHub\BackEnd>
```

- 3. Escriba el siguiente comando:** `python3.8 -m venv venv`

Es importante que si tiene más de una versión instalada de python DEBE especificar `python3.8`, de lo contrario el comando `python -m venv venv` es también válido.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1706]
(c) Microsoft Corporation. Todos los derechos reservados.
D:\jsmon\Documents\GitHub\BackEnd>python3.8 -m venv venv
D:\jsmon\Documents\GitHub\BackEnd>
```

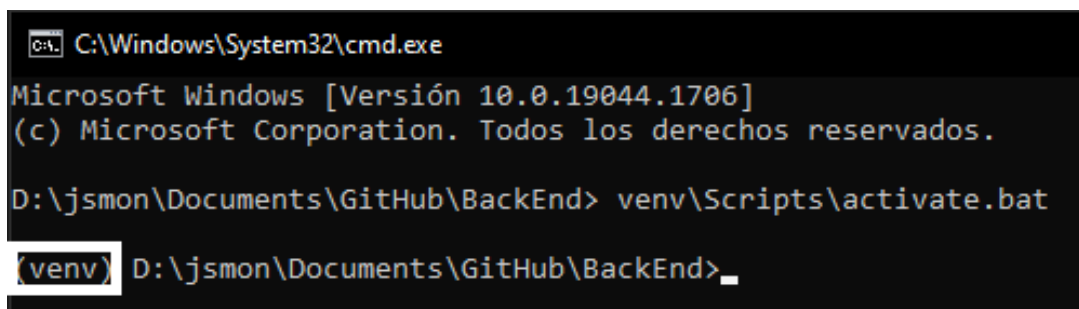
El proceso será exitoso si no aparecen mensajes y si en la carpeta de backend aparece una nueva carpeta llamada venv



#### 4. Activar el entorno virtual:

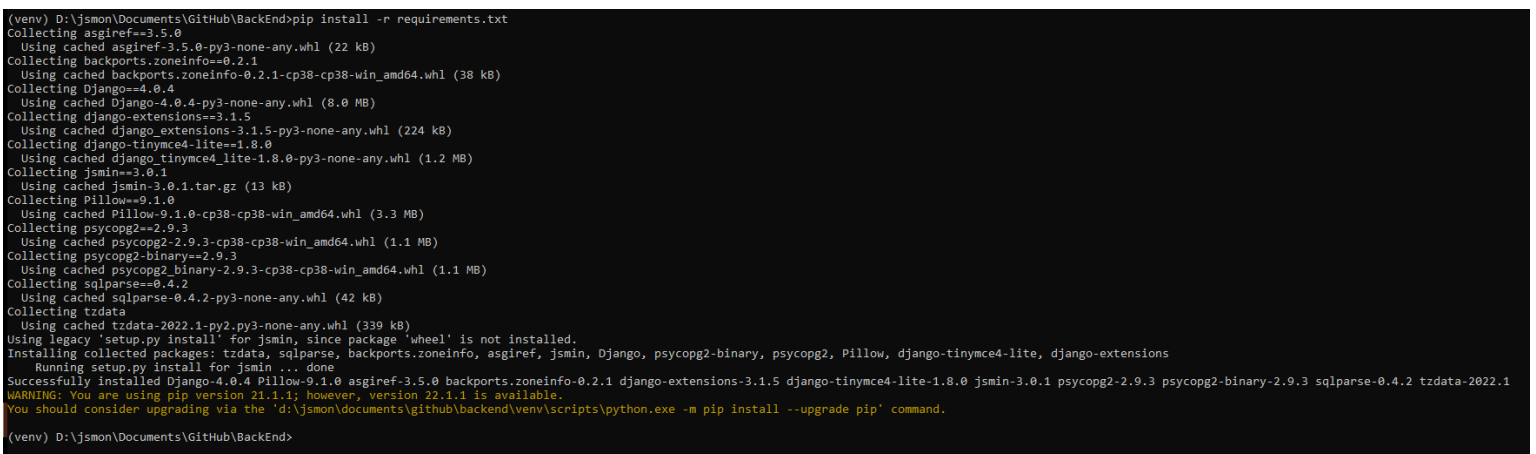
en la consola se digitarán los siguientes comandos: `venv\Scripts\activate.bat`

4.1 Presione enter y espere unos segundos, para comprobar que el entorno está activado en la parte izquierda debe aparecer (venv)



#### Instalación de librerías implementadas

Dentro de la carpeta de Backend, está un archivo de texto requirements.txt el cual contiene las librerías instaladas en el entorno virtual, para instalarlas, en la consola de windows digite: `pip install -r requirements.txt`



Tenga en cuenta que la versión 3.9 de python en adelante no cuenta con soporte para la librería backport.zone-info, por este motivo debe tener la versión 3.8 o inferior.

Nota:

IGNORE la recomendación de actualizar pip, debido a que al hacerlo se genera un error y se pierde pip, y hasta el momento no se tiene una solución a este problema en entornos virtuales y tendrá que eliminar y reinstalar el venv

```
WARNING: You are using pip version 21.1.1; however, version 22.1.1 is available.  
You should consider upgrading via the 'd:\jsmon\documents\github\backend\venv\scripts\python.exe -m pip install --upgrade pip' command.
```

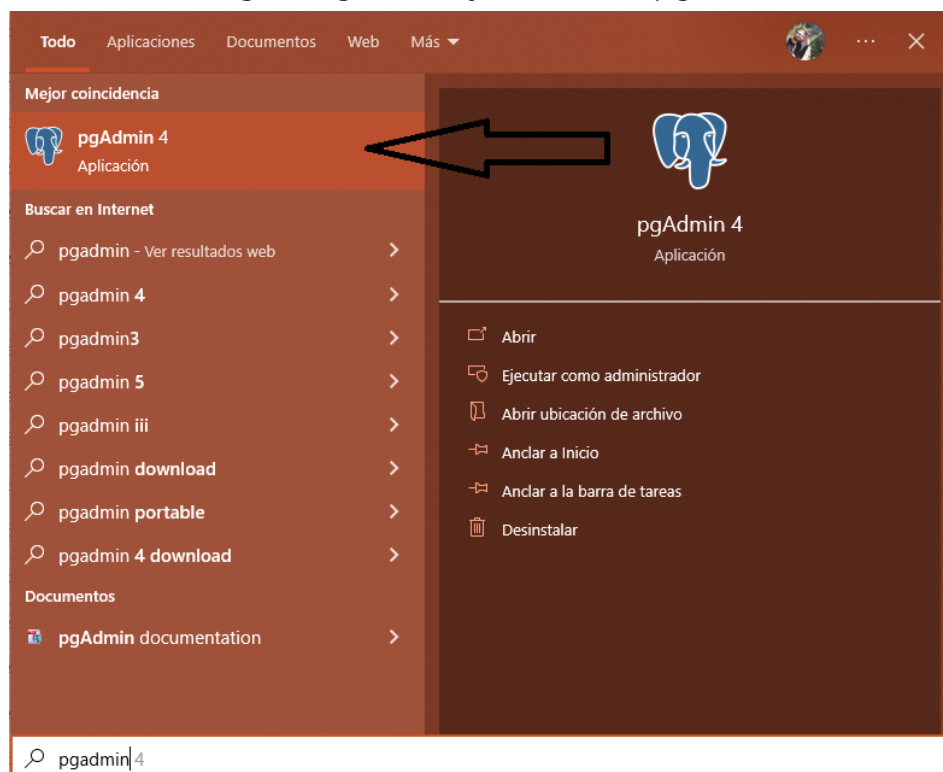
Por Favor IGNORAR ESTE WARNING.

Para comprobar que todas las librerías estén correctamente instaladas, ejecute el comando en el CMD: pip freeze

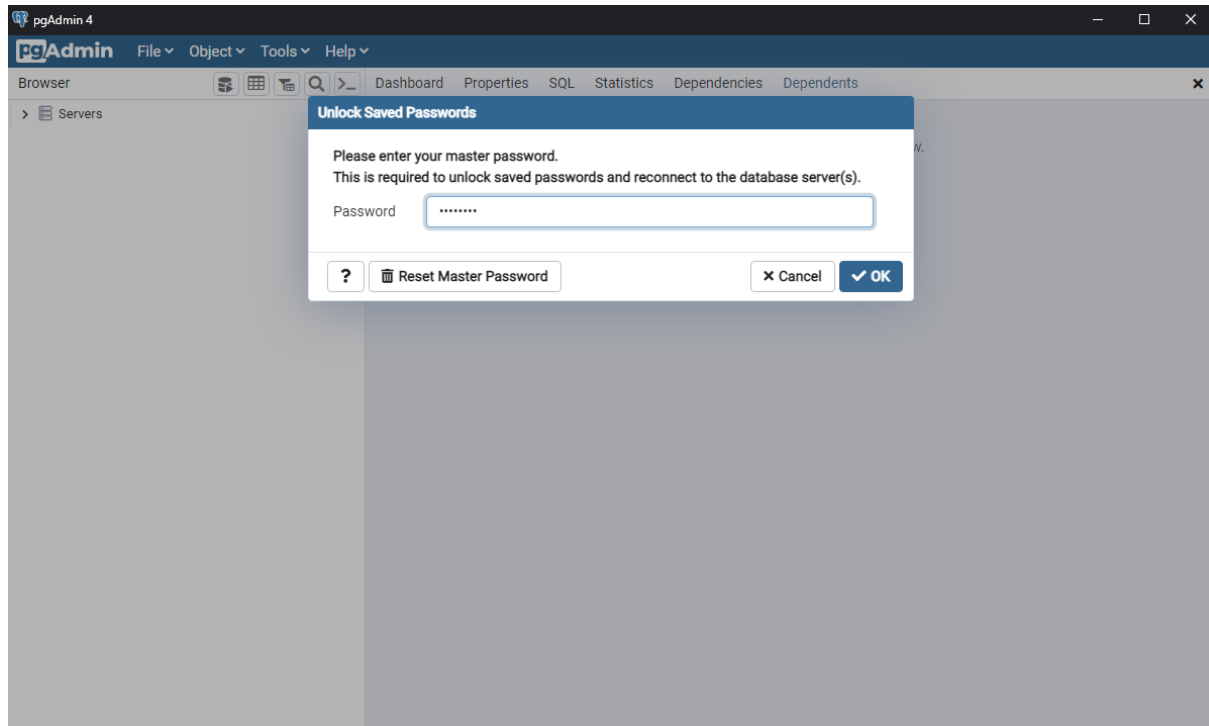
```
(venv) D:\jsmon\Documents\GitHub\BackEnd>pip freeze  
asgiref==3.5.0  
backports.zoneinfo==0.2.1  
Django==4.0.4  
django-extensions==3.1.5  
django-tinymce4-lite==1.8.0  
jsmin==3.0.1  
Pillow==9.1.0  
psycopg2==2.9.3  
psycopg2-binary==2.9.3  
sqlparse==0.4.2  
tzdata==2022.1  
  
(venv) D:\jsmon\Documents\GitHub\BackEnd>
```

## Creación de base de datos en PostgreSQL

Ahora, debe crear la base de datos local por medio de PostgreSQL, para ello en el buscador de windows digite “Pgadmin” y selecciona pgAdmin 4

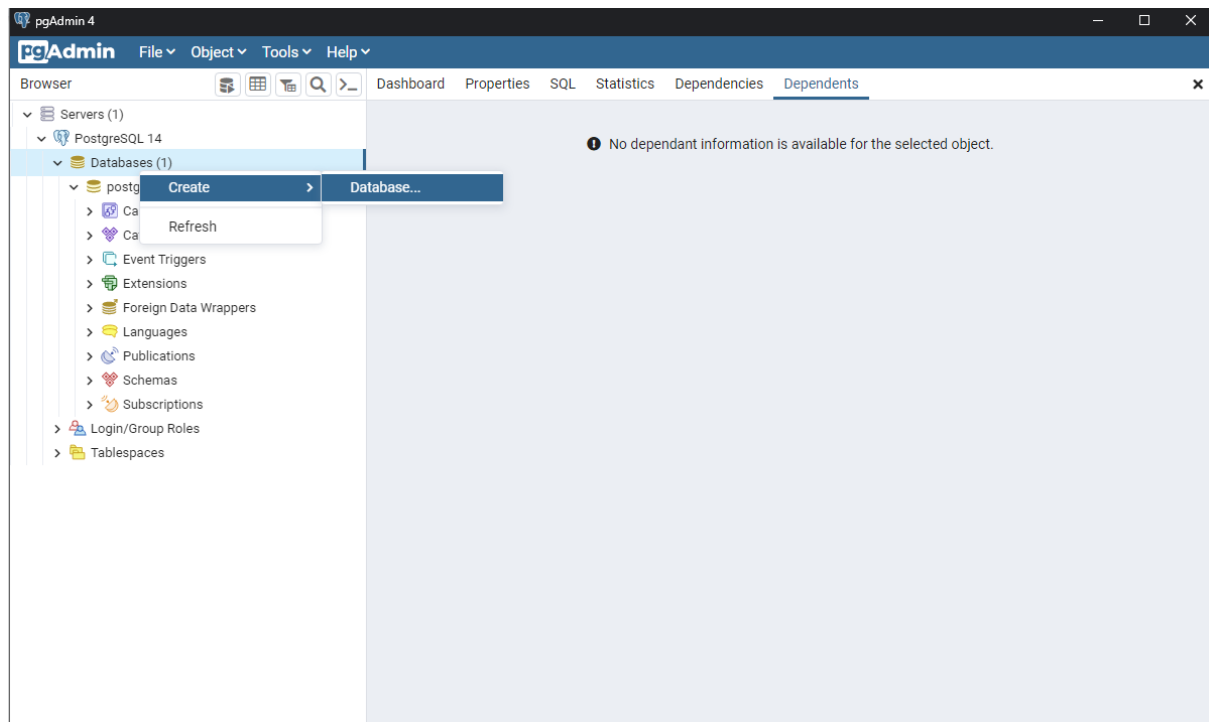


Una vez el programa se haya iniciado, debe digitar la contraseña de administrador que definió al momento de instalar PostgreSQL.

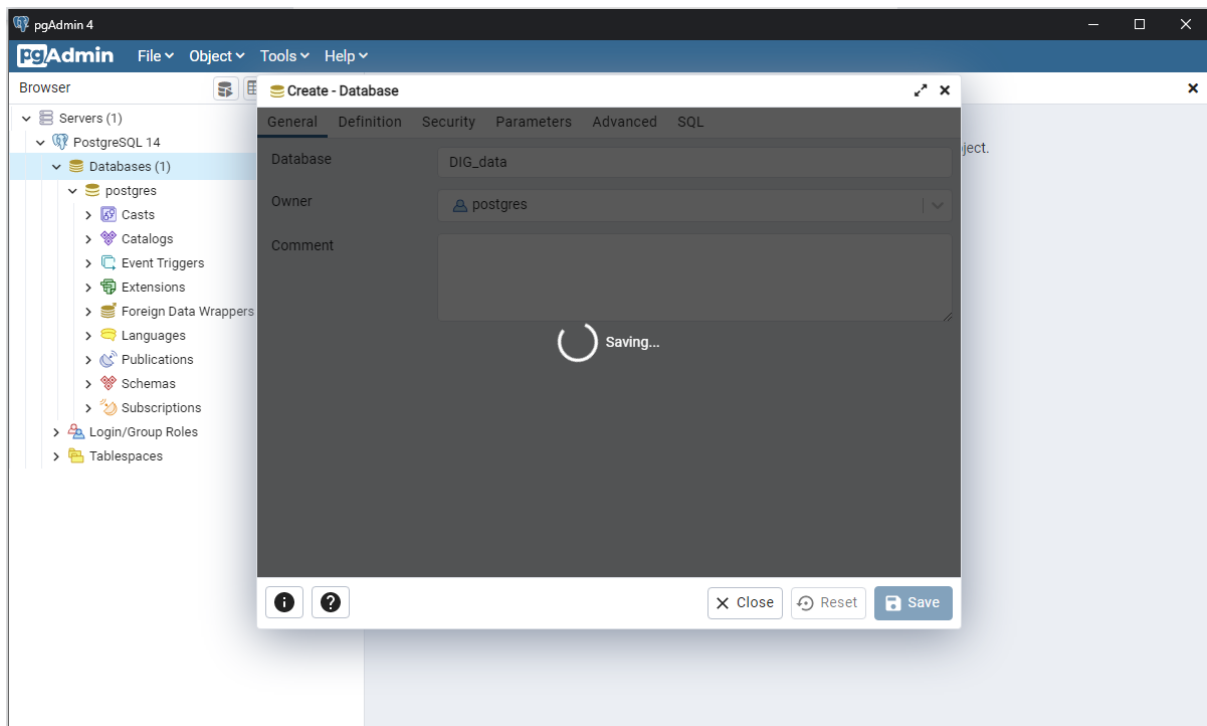
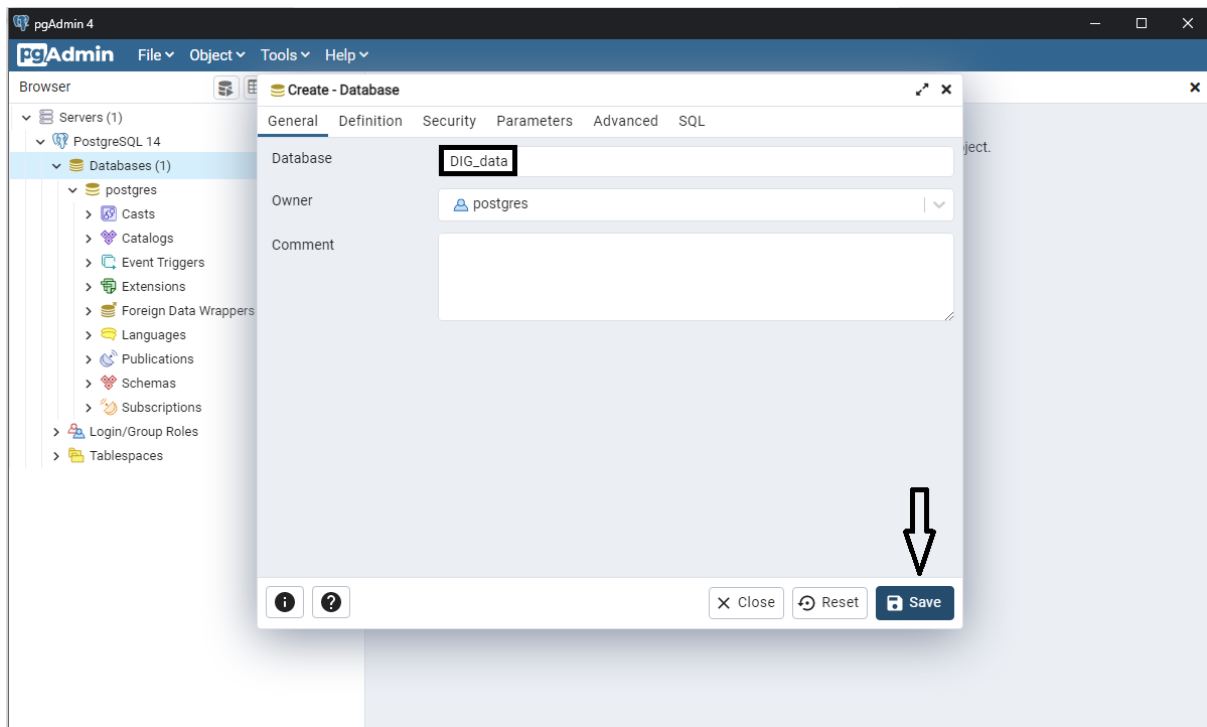


Una vez se haya logueado como admin, seleccione “servers”, “postgresql14”, “databases”.

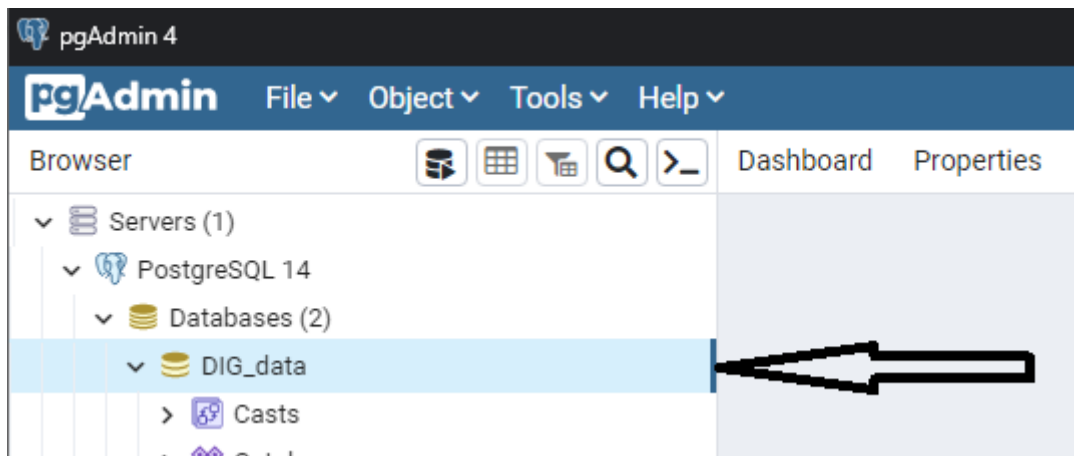
Sobre Databases hace click derecho, create, click izquierdo en “Database”.



Se desplegará una ventana en donde pedirá el nombre de la base de datos, la cual debe ser "DIG\_data", es NECESARIO que sea este el nombre de la base de datos y no otro ya que el programa no funcionará. Luego hace click en save.





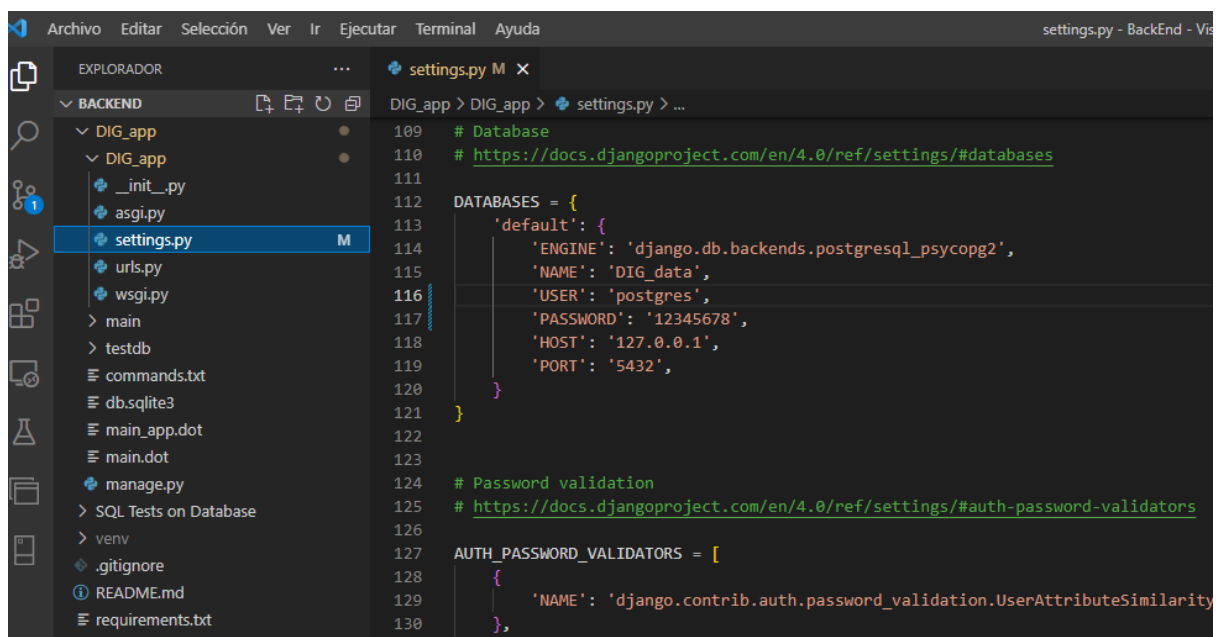


## Migrations

Ahora, en la ventana de comandos CMD en el folder de Backend, digite el siguiente comando: "code ./"

```
(venv) D:\jsmon\Documents\GitHub\BackEnd>code ./
(venv) D:\jsmon\Documents\GitHub\BackEnd>
```

Se desplegará Visual Studio code y cargará el proyecto, ahora es necesario que configure la base de datos en el archivo de configuración para hacer las migraciones, para ello debe dirigirse a DIG\_app, settings.py y modificar las líneas 115 y 116 de código (USER y PASSWORD) colocando su nombre de usuario administrador y su contraseña. Posteriormente presiona ctrl+g para guardar los cambios.



Una vez realizado esto, regrese al CMD, ingresa a la carpeta de DIG APP y digite el siguiente comando: “python manage.py makemigrations” el cual migrará el modelo UML a un modelo ER.

```
(venv) D:\jsmon\Documents\GitHub\BackEnd>cd DIG_app

(venv) D:\jsmon\Documents\GitHub\BackEnd\DIG_app>python manage.py makemigrations
No changes detected

(venv) D:\jsmon\Documents\GitHub\BackEnd\DIG_app>
```

## Migrate

Ahora ejecute el comando “python manage.py migrate” para migrar la base de datos a Django.

```
(venv) D:\jsmon\Documents\GitHub\BackEnd\DIG_app>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... 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 main.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 main.0002_alter_user_city_alter_user_rating_and_more... OK
  Applying main.0003_alter_user_rating_alter_user_telephone_number... OK
  Applying main.0004_report_favorite_consult... OK
  Applying main.0005_alter_consult_business_id... OK
  Applying main.0006_occupationstatus_alter_business_telephone_number_and_more... OK
  Applying main.0007_alter_report_report_support... OK
  Applying main.0008_alter_favorite_business_id... OK
  Applying main.0009_city_alter_business_city_alter_user_city... OK
  Applying main.0010_alter_business_telephone_number... OK
  Applying main.0011_business_cover_picture... OK
  Applying main.0012_alter_business_cover_picture_alter_user_avatar... OK
  Applying main.0013_alter_business_cover_picture_alter_user_avatar... OK
  Applying main.0014_alter_business_cover_picture_alter_user_avatar... OK
  Applying main.0015_alter_report_comments... OK
  Applying sessions.0001_initial... OK

(venv) D:\jsmon\Documents\GitHub\BackEnd\DIG_app>
```

## Conexión

Digite el comando “python manage.py createsuperuser” para crear el usuario desarrollador administrador.

```
(venv) D:\jsmon\Documents\GitHub\BackEnd\DIG_app>python manage.py createsuperuser
Username: jsmontoyaco
Email: jsmontoyaco@mail.com
Password:
Password (again):
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
```

Se le pedirá un nombre de usuario , correo y contraseña de administrador, no olvide sus credenciales.

Por último digite el comando “python manage.py runserver” para ejecutar el servidor.

```
(venv) D:\jsmon\Documents\GitHub\BackEnd\DIG_app>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 25, 2022 - 22:53:44
Django version 4.0.4, using settings 'DIG_app.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Conserve esta ventana abierta mientras ejecuta el programa , así como también postgresQL.

## Inicialización

Para iniciar el proyecto una vez instalado todos los prerequisites y entornos de desarrollo, para iniciar la aplicación solo debe iniciar el entorno de desarrollo, luego con CMD ubicarse en la carpeta de DIG\_app y ejecutar el comando “python manage.py runserver”

Esto inicializará el servidor local, ahora para inicializar la aplicación solo debe poner 127.0.0.1:8000 en la barra de explorador de cualquier explorador web. (Aconsejable usar Google Chrome, Opera o Firefox).

