



**UNIVERSIDAD DE LAS AMERICAS FACULTAD
DE INGENIERIA Y CIENCIAS APLICADAS
ING. EN CIBERSEGURIDAD**

ITIZ3201 ADMINISTRACIÓN DE BASE DE DATOS

PROYECTO INTEGRADOR

TEMA

**Proyecto Integrador- Solución de base de datos Courier
contenerizada**

Docente:

Patricio Moreno

Autor:

William Morales- Ana Belen Gavilanes- Jonathan Lema

Fecha:

2024

OBJETIVO PROPUESTO DE LA ACTIVIDAD:

El trabajo por realizar tiene como finalidad implementar una solución de base de datos contenerizada que permita cubrir los siguientes puntos:

- Plantear el despliegue de una solución de base de datos contenerizada
- Efectuar el despliegue de una solución de base de datos contenerizada
- Validar el despliegue de una solución de base de datos contenerizada
- Analizar los riesgos de seguridad a nivel privilegios que puede poseer una base de datos contenerizada
- Plantear una alternativa de solución a los riesgos de seguridad a nivel privilegios que puede poseer una base de datos contenerizada, basándose en un enfoque RBA
- Implementar una solución de seguridad basada en RBA para la operación segura de una base de datos contenerizada.

INDICACIONES:

En grupos de 2 o 3 estudiantes se deberá construir, como habilitador para poder rendir las evaluaciones del tercer progreso y eventualmente del examen de recuperación, una solución de base de datos contenerizada.

Para empezar, se deberá implementar la mencionada solución de base de datos contenerizada, basándose en las buenas prácticas vistas y aplicadas en clase:

- Archivos de configuración deberán residir en un almacenamiento persistente.
- Archivos de la base de datos deberán residir en un almacenamiento persistente
- No se deberá utilizar el puerto de conexión por default
- Se deberá mapear el puerto de conexión de la instancia de base de datos hacia el host donde se ejecuta el contenedor
- Se deberá utilizar un cliente gráfico SQL para acceder y manejar la instancia de base de datos contenerizada.

Posteriormente, se deberá diseñar e implementar en la base de datos contenerizada, un modelo de datos que permita solventar una necesidad de almacenamiento de datos planteada por el docente.

Una vez implementado el modelo de datos, se deberá analizar los riesgos de seguridad que podría existir en el acceso a dichos datos, con el fin de plantear e implementar una solución basada en un enfoque RBA, la cual permitirá que la base de datos contenerizada opere de manera segura, una vez más, en base a las buenas prácticas vistas y aplicadas en clase:

- Deberá existir un rol de base de datos con privilegios de solo lectura
- Deberá existir al menos un rol de base de datos con privilegios de manipular los datos
- Deberá existir un primer usuario de base de datos a quien se le deberá conceder el primer rol.
- Deberá existir un segundo usuario de base de datos a quien se le deberá conceder el segundo rol.
- Deberán generarse evidencias del correcto funcionamiento de la solución de seguridad basada en RBA.

FORMA DE TRABAJO:

El líder de cada uno de los grupos de trabajo remitirá tan pronto como le sea posible un correo al docente solicitando se indique la necesidad de almacenamiento que se deberá implementar en la base de datos contenerizada.

Cada uno de los grupos procederá a implementar una solución de base de datos contenerizada en base a las indicaciones dadas y considerando además los siguientes puntos:

- La solución de base de datos contenerizada deber implementarse en una máquina virtual con Linux
- La solución de base de datos contenerizada debe implementarse utilizando la última versión de PostgreSQL.
- El cliente gráfico SQL para acceder a la base de datos contenerizada deberá instalarse en el computador host (este cliente se usará en la defensa del proyecto).
- El diseño de la base de datos implementado deberá estar acompañado de su diagrama de diseño físico de la base de datos, y del diccionario de datos.

- El contenido de la base de datos contenerizada deberá ser generado aleatoriamente utilizando Python y el módulo Faker (<https://faker.readthedocs.io/en/master/>)

ESPECIFICACIONES DE ENTREGA:

Se deberá crear un proyecto de GitHub, conteniendo la estructura de carpetas adecuada para la solución de base de datos contenerizada implementada, incluyendo la documentación de como instalar y ejecutar la solución.

Cada grupo deberá generar un PDF formal con el siguiente contenido:

1. Descripción de la Solución Implementada

La solución se implementó en una máquina virtual con Linux, utilizando la última versión de PostgreSQL. El contenedor de la base de datos se creó utilizando Dockerfile, y se mapeó el puerto 5432 del contenedor al puerto 5432 del host.

Modelo de datos

El modelo de datos implementado se basa en las siguientes tablas:

Tabla	Descripción
Cliente	Almacena información básica sobre los clientes, como su nombre, apellido, dirección y número de teléfono.
Destinatario	Almacena información sobre los destinatarios de las encomiendas, como su nombre, apellido, dirección y número de teléfono.
Encomienda	Almacena información sobre las encomiendas, como su número de seguimiento, fecha de creación, fecha de entrega, etc.
Envío	Almacena información sobre los envíos, como su tipo, peso, dimensiones, etc.
Estado encomienda	Almacena el estado actual de la encomienda.
Repartidor	Almacena información sobre los repartidores, como su nombre, apellido, número de identificación, etc.
Ruta	Almacena información sobre las rutas de reparto, como su nombre, origen, destino, etc.
Ticket	Almacena información sobre los tickets de servicio, como su número, fecha de creación, etc.
Tipo envío	Almacena información sobre los tipos de envío, como su nombre, descripción, etc.

Diagrama de diseño lógico de la base de datos bdcourier con power designer

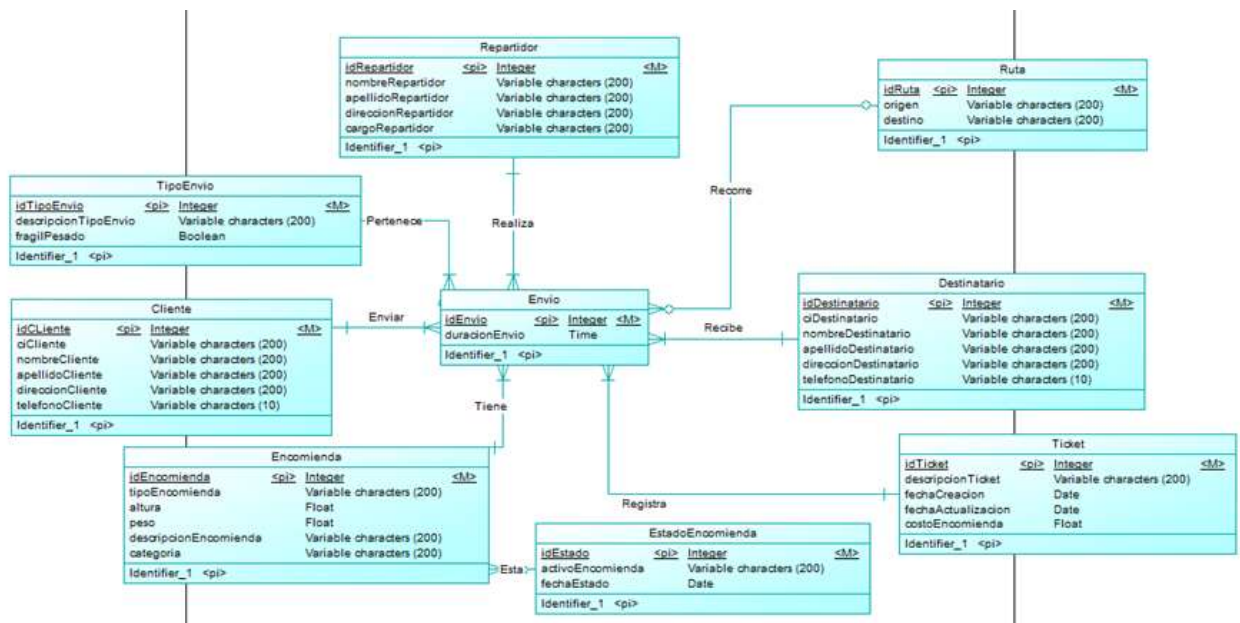
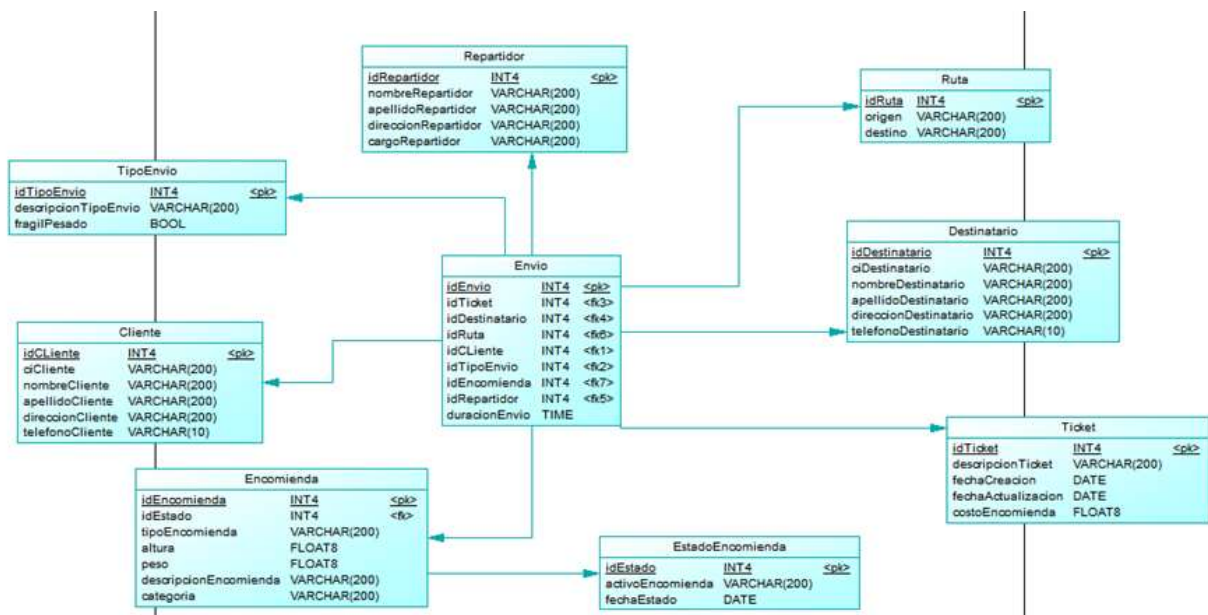


Diagrama de diseño físico de la base de datos bdcourier con power designer



Análisis de riesgos de seguridad

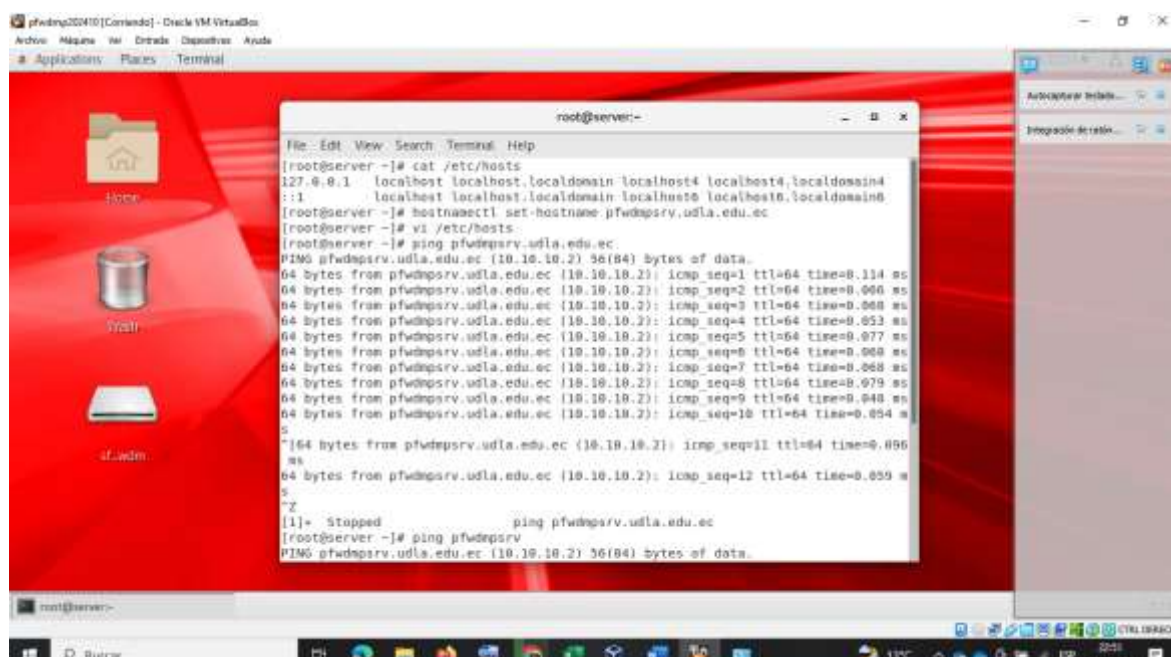
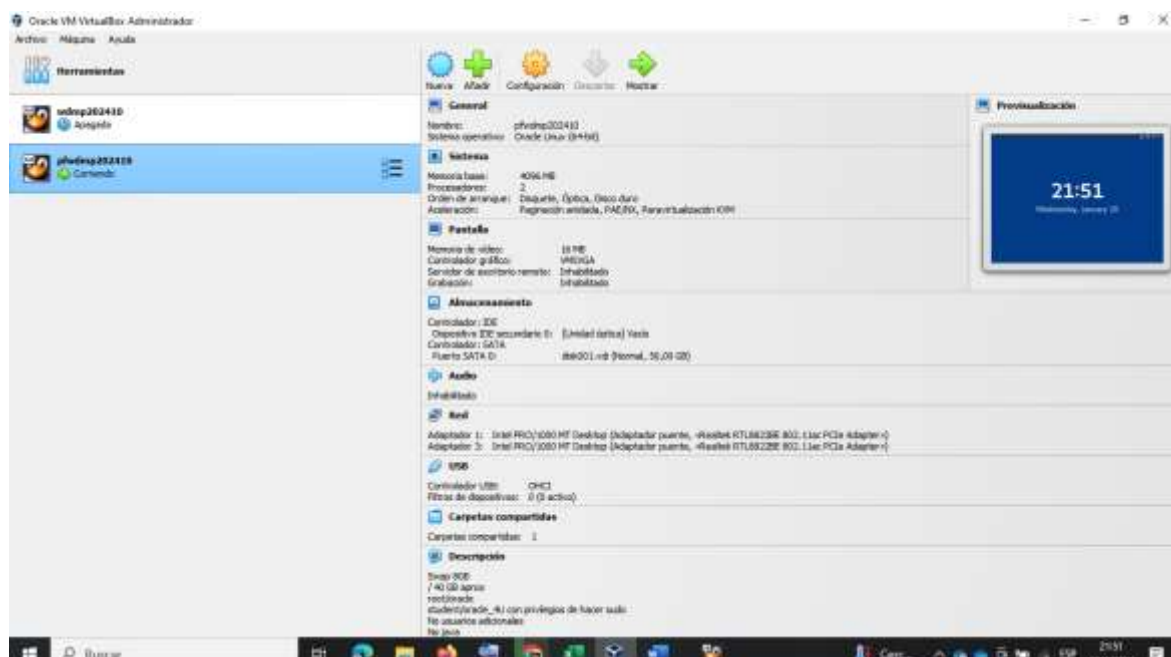
El análisis de riesgos de seguridad identificó los siguientes riesgos potenciales:

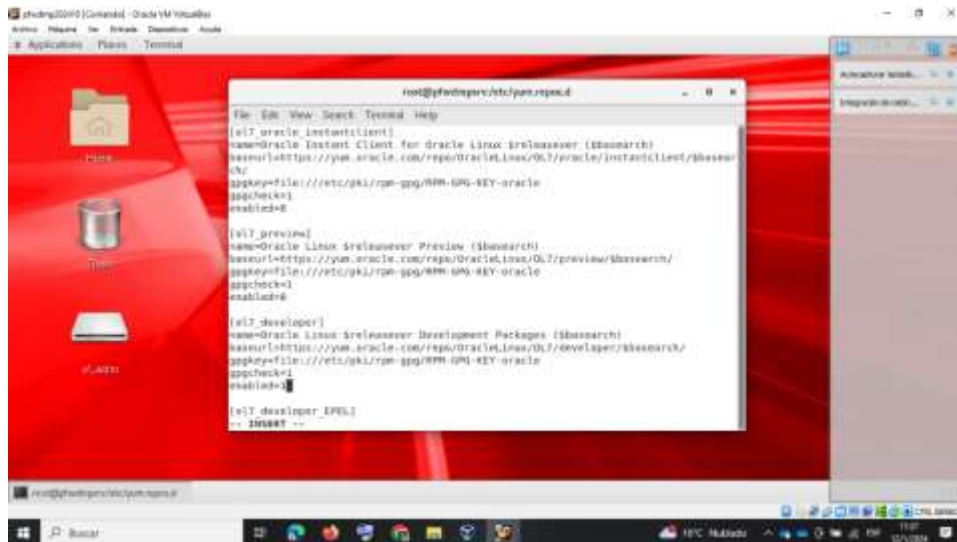
- Un usuario malintencionado podría acceder a los datos de los clientes, destinatarios, encomiendas, datos de clientes, etc.
- Un usuario malintencionado podría modificar o eliminar los datos de los clientes, destinatarios, encomiendas, etc.
- Un usuario malintencionado podría crear nuevas encomiendas o tickets.

Para mitigar estos riesgos, se implementó una solución de seguridad basada en RBA.

Desarrollo:

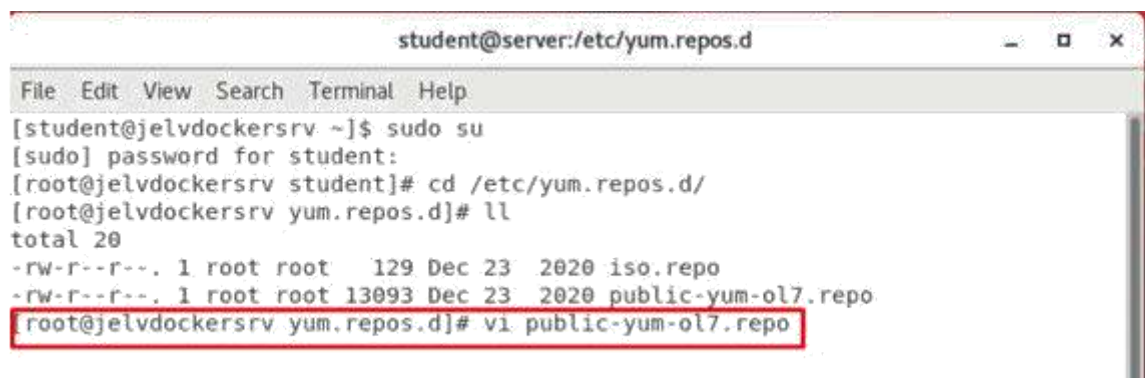
1. Creación de la máquina virtual Oracle





2. Instalación de Docker en OL 7.6

- Vamos a ingresar el comando: `cd /etc/yum.repos.d/` para ir al repositorio que nos va a servir para instalar Docker
- Ingresamos el comando: **vi public-yum-ol7.repo** para realizar modificaciones para la instalación de **Docker**



- Vamos a colocar **1** en las líneas de **enabled** que se encuentran con valor de **0**:

```
[ol7_developer]
name=Oracle Linux $releasever Development Packages ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL7/developer/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1

[ol7_developer_EPEL]
name=Oracle Linux $releasever Development Packages ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL7/developer_EPEL/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
-- INSERT --
```

- Ingresamos el comando **yum repoinfo** para refrescar el repositorio de paquetes
- Ingresamos el comando: **yum install device-mapper-persistent-data lvm2** para realizar la instalación




```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help

=====
Package                               Arch      Version                               Repository  Size
=====
Updating:
lvm2                                  x86_64    7:2.02.187-6.0.5.el7_9.5            ol7_latest  1.3 M
yum-utils                             noarch    1.1.31-54.0.1.el7_8                 ol7_latest  122 k
Updating for dependencies:
device-mapper                         x86_64    7:1.02.170-6.0.5.el7_9.5            ol7_latest  297 k
device-mapper-event                  x86_64    7:1.02.170-6.0.5.el7_9.5            ol7_latest  192 k
device-mapper-event-libs             x86_64    7:1.02.170-6.0.5.el7_9.5            ol7_latest  192 k
device-mapper-libs                   x86_64    7:1.02.170-6.0.5.el7_9.5            ol7_latest  325 k
lvm2-libs                            x86_64    7:2.02.187-6.0.5.el7_9.5            ol7_latest  1.1 M
lvm2-python-libs                     x86_64    7:2.02.187-6.0.5.el7_9.5            ol7_latest  189 k

Transaction Summary
=====
Upgrade 2 Packages (+6 Dependent packages)

Total download size: 3.7 M
Is this ok [y/d/N]: Exiting on user command
Your transaction was saved, rerun it with:
yum load-transaction /tmp/yum_save tx.2023-04-25.11-46.f0cfAp.yumtx
[root@jelvdockersrv yum.repos.d]#

```

```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help

Verifying : 7:device-mapper-libs-1.02.170-6.0.5.el7_9.5.x86_64 7/16
Verifying : 7:device-mapper-event-libs-1.02.170-6.0.5.el7_9.5.x86_64 8/16
Verifying : 7:lvm2-2.02.180-8.el7.x86_64 9/16
Verifying : 7:lvm2-libs-2.02.180-8.el7.x86_64 10/16
Verifying : 7:device-mapper-1.02.149-8.el7.x86_64 11/16
Verifying : yum-utils-1.1.31-50.0.1.el7.noarch 12/16
Verifying : 7:lvm2-python-libs-2.02.180-8.el7.x86_64 13/16
Verifying : 7:device-mapper-libs-1.02.149-8.el7.x86_64 14/16
Verifying : 7:device-mapper-event-1.02.149-8.el7.x86_64 15/16
Verifying : 7:device-mapper-event-libs-1.02.149-8.el7.x86_64 16/16

Updated:
lvm2.x86_64 7:2.02.187-6.0.5.el7_9.5 yum-utils.noarch 0:1.1.31-54.0.1.el7_8

Dependency Updated:
device-mapper.x86_64 7:1.02.170-6.0.5.el7_9.5
device-mapper-event.x86_64 7:1.02.170-6.0.5.el7_9.5
device-mapper-event-libs.x86_64 7:1.02.170-6.0.5.el7_9.5
device-mapper-libs.x86_64 7:1.02.170-6.0.5.el7_9.5
lvm2-libs.x86_64 7:2.02.187-6.0.5.el7_9.5
lvm2-python-libs.x86_64 7:2.02.187-6.0.5.el7_9.5

Complete!
[root@jelvdockersrv yum.repos.d]#

```

-Vamos a añadir el repositorio en donde se descarga Docker con el comando: **yum-config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>**


```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help
[root@jelvdockersrv yum.repos.d]# yum-config-manager --add-repo https://download
.docker.com/linux/centos/docker-ce.repo

```

```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help
[root@jelvdockersrv yum.repos.d]# yum-config-manager --add-repo https://download
.docker.com/linux/centos/docker-ce.repo
Loaded plugins: langpacks
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yu
m.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@jelvdockersrv yum.repos.d]#

```

- Vamos a proceder a instalar **Docker** con el comando: **yum install Docker**:

```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help
[root@jelvdockersrv yum.repos.d]# yum-config-manager --add-repo https://download
.docker.com/linux/centos/docker-ce.repo
Loaded plugins: langpacks
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yu
m.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@jelvdockersrv yum.repos.d]# ll
total 24
-rw-r--r--. 1 root root 1919 Apr 19 07:29 docker-ce.repo
-rw-r--r--. 1 root root 129 Dec 23 2020 iso.repo
-rw-r--r--. 1 root root 13093 Apr 25 11:01 public-yum-ol7.repo
[root@jelvdockersrv yum.repos.d]# yum install docker

```

- Verificamos que se realizó la instalación:

```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help
Verifying : docker-buildx-plugin-0.10.4-1.el7.x86_64 4/10
Verifying : 3:docker-ce-23.0.4-1.el7.x86_64 5/10
Verifying : slirp4netns-0.4.3-4.el7_8.x86_64 6/10
Verifying : 2:container-selinux-2.119.2-1.911c772.el7_8.noarch 7/10
Verifying : docker-ce-rootless-extras-23.0.4-1.el7.x86_64 8/10
Verifying : containerd.io-1.6.20-3.1.el7.x86_64 9/10
Verifying : docker-compose-plugin-2.17.2-1.el7.x86_64 10/10

Installed:
docker-ce.x86_64 3:23.0.4-1.el7

Dependency Installed:
container-selinux.noarch 2:2.119.2-1.911c772.el7_8
containerd.io.x86_64 0:1.6.20-3.1.el7
docker-buildx-plugin.x86_64 0:0.10.4-1.el7
docker-ce-cli.x86_64 1:23.0.4-1.el7
docker-ce-rootless-extras.x86_64 0:23.0.4-1.el7
docker-compose-plugin.x86_64 0:2.17.2-1.el7
fuse-overlayfs.x86_64 0:0.7.2-6.el7_8
fuse3-libs.x86_64 0:3.6.1-4.el7
slirp4netns.x86_64 0:0.4.3-4.el7_8

Complete!
[root@jelvdockersrv yum.repos.d]#

```

- habilita el servicio con el comando: **systemctl enable Docker**
- Iniciamos el servicio de Docker con el comando: **systemctl start docker**
- Comprobamos que el servicio esta levantado con el comando: **systemctl status docker**

```

student@server:/etc/yum.repos.d
File Edit View Search Terminal Help
[root@jelvdockersrv yum.repos.d]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor prese
t: disabled)
   Active: active (running) since Tue 2023-04-25 12:46:58 -05; 50s ago
     Docs: https://docs.docker.com
   Main PID: 16759 (dockerd)
      Tasks: 8
     Memory: 30.6M
    CGroup: /system.slice/docker.service
            └─16759 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/con...

Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com systemd[1]: Started Docker Application...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Apr 25 12:46:58 jelvdockersrv.lpmc.com dockerd[16759]: time="2023-04-25T12:46...
Hint: Some lines were ellipsized, use -l to show in full.
[root@jelvdockersrv yum.repos.d]#

```

3. Instalar Postgresql en contenedor de Docker

- Verificar que no hay contenedores corriendo en docker con el comando: `docker ps`

```
root@projectcoursier:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

- Realizar la descarga de la imagen de PostgreSQL con el comand: docker pull postgres

```
[root@projectocourier ~]# docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
2f44b7a888fa: Pull complete
6d49150dabe2: Pull complete
18d6a86d0fbf: Pull complete
4c9385c30bce: Pull complete
550091272acc: Pull complete
2720859ac49e: Pull complete
b8091cf53545: Pull complete
f3ca5fbd89cd: Pull complete
22fbbce47a56: Downloading [=====>] 50.57MB/108.6MB
b3b5e3b65b95: Download complete
917e5b76e085: Download complete
7f21ce9572c6: Download complete
4ea3941c8572: Download complete
848fee03c034: Download complete
```

- Con el comando **docker images** verificamos que ya se encuentra descargada la imagen de **postgres**:

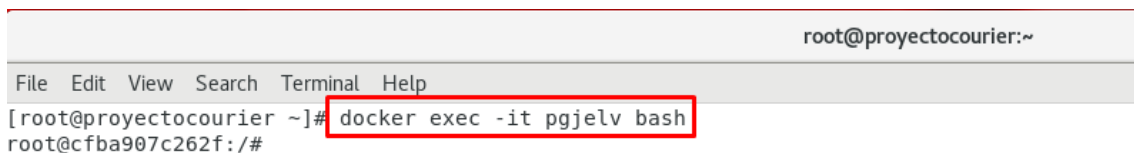
```
[root@proyectocourier ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
postgres      <none>    bdc467e80232  7 days ago    425MB
postgres      latest    75b7bff7c3ad  7 days ago    425MB
```

- Ejecución de un contenedor de PostgreSQL bajo el nombre **pgjelv** que permita mapear el puerto por default de PostgreSQL al puerto 4444 del host, que posea el archivo de configuración y el directorio de datos por fuera del contenedor: **docker run --name pgjelv -e POSTGRES_PASSWORD=postgres -d -p 5432:4444 postgres**



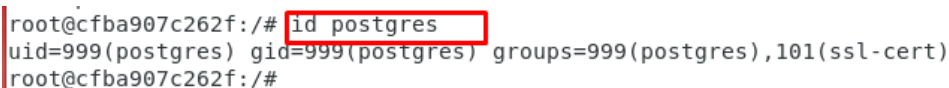
```
root@proyectocourier ~# docker run --name pgjelv -e POSTGRES_PASSWORD=postgres -d -p 5432:4444 postgres
cfba907c262f884a6e58d3a948c166c45d173835f2462918c3e997438c7d81
```

- Vamos a conectarnos dentro del contenedor que se está ejecutando de Postgres con el comando **docker exec -it pgjelv bash**



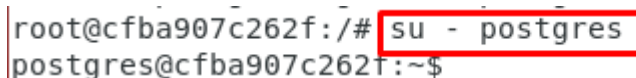
```
root@proyectocourier ~# docker exec -it pgjelv bash
root@cfba907c262f:/#
```

- Observamos el **id** con el que estamos conectados:



```
root@cfba907c262f:/# id postgres
uid=999(postgres) gid=999(postgres) groups=999(postgres),101(ssl-cert)
root@cfba907c262f:/#
```

- Nos conectamos con el **id postgres**:



```
root@cfba907c262f:/# su - postgres
postgres@cfba907c262f:~$
```

- Ingresamos el comando **psql** que es el utilitario para ingresar a **postgres**:

```
postgres@cfba907c262f:~$ psql
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=#
```

- Con el comando **\l** observamos las bases de datos existentes:

```
root@cfba907c262f:/# su - postgres
postgres@cfba907c262f:~$ psql
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=# \l
```

Name	Owner	Encoding	Locale Provider	Collate	Ctype	ICU Locale	ICU Rules	Access privileges
postgres	postgres	UTF8	libc	en_US.utf8	en_US.utf8			
template0	postgres	UTF8	libc	en_US.utf8	en_US.utf8			=c/postgres +
template1	postgres	UTF8	libc	en_US.utf8	en_US.utf8			=c/postgres +

```
(3 rows)

postgres=#
```

- Observamos como estamos conectados con el comando **\conninfo**:

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
postgres=#
```

- Salimos del contenedor con el comando **exit**:

```
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
postgres=# exit
postgres@cfba907c262f:~$ exit
logout
root@cfba907c262f:/# exit
exit
[root@proyectocourier ~]#
```

- Detenemos la ejecución del contenedor **pgjelv**:

```
[root@proyectocourier ~]#
[root@proyectocourier ~]# docker stop pgjelv
pgjelv
[root@proyectocourier ~]#
```

- Vamos a crear el directorio donde se va a almacenar la base de datos por fuera del contenedor, detenemos la ejecución del contenedor **pgjelv**.
- Creamos el directorio con el comando **mkdir -p /docker-storage/postgres16/datadir2**

```

root@proyectocourier:/
File Edit View Search Terminal Help
[root@proyectocourier /]# mkdir -p /docker-storage/postgres16/datadir2
[root@proyectocourier /]#

```

- Vamos a correr un contenedor de postgres en el directorio creado con el nombre de pgjelv2 con el comando: **docker run --name pgjelv2 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres16/datadir2:/var/lib/postgresql/data -d postgres**

```

root@proyectocourier:/
File Edit View Search Terminal Help
[root@proyectocourier /]# docker run --name pgjelv2 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres16/datadir2:/var/lib/postgresql/data -d postgres
0eb7061aeab05865ab57776ebd99d4e83b3a35dc822a2600e918e10021d47
[root@proyectocourier /]#

```

- Con el comando **docker -ps** observamos que se encuentra corriendo el contenedor **pgjelv2**:

```

root@proyectocourier:/
File Edit View Search Terminal Help
[root@proyectocourier /]# docker run --name pgjelv2 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres16/datadir2:/var/lib/postgresql/data -d postgres
0eb7061aeab05865ab57776ebd99d4e83b3a35dc822a2600e918e10021d47
[root@proyectocourier /]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
0eb7061aeab0   postgres  "docker-entrypoint.sh"   41 seconds ago Up 30 seconds 5432/tcp      pgjelv2
[root@proyectocourier /]#

```

```

root@proyectocourier:/
File Edit View Search Terminal Help
[root@proyectocourier /]# docker run --name pgjelv2 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres16/datadir2:/var/lib/postgresql/data -d postgres
0eb7061aeab05865ab57776ebd99d4e83b3a35dc822a2600e918e10021d47
[root@proyectocourier /]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
0eb7061aeab0   postgres  "docker-entrypoint.sh"   41 seconds ago Up 30 seconds 5432/tcp      pgjelv2
[root@proyectocourier /]#

```

- Vamos a ejecutar el comando **docker exec -it pgjelv2 bash** para ingresar al bash del contenedor:

```

root@proyectocourier:/
File Edit View Search Terminal Help
[root@proyectocourier /]# docker exec -it pgjelv2 bash
root@0eb7061aeab0:/#

```

- Ingresamos el comando: **su - postgres**

```

root@0eb7061aeab0:/# su - postgres
postgres@0eb7061aeab0:~$

```

- Ingresamos con el comando **psql**:

```
postgres@0eb7061aeab0:~$ psql
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=#
```

- Nos salimos del contenedor con el comando exit:

```
postgres@0eb7061aeab0:~$ psql
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=# exit
postgres@0eb7061aeab0:~$ exit
logout
root@0eb7061aeab0:/# exit
exit
[root@proyectocourier /]#
[root@proyectocourier /]#
[root@proyectocourier /]#
```

- Accedemos al directorio: **cd /docker-storage/postgres16/datadir2**:

A terminal window titled 'root@proyectocourier:/docker-storage/postgres16/datadir2'. The prompt is '[root@proyectocourier /]#'. The command 'cd /docker-storage/postgres16/datadir2' has been entered and executed, resulting in a new prompt '[root@proyectocourier datadir2]#'.

```
root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier /]# cd /docker-storage/postgres16/datadir2
[root@proyectocourier datadir2]#
```

4. Cambiar de puerto por defecto de Potsgresql a puerto 4444

- Vamos a editar el archivo **postgresql.conf** con el comando: **nano postgresql.conf**:

A terminal window titled 'root@proyectocourier:/docker-storage/postgres16/datadir2'. The prompt is '[root@proyectocourier datadir2]#'. The command 'nano postgresql.conf' has been entered and is highlighted by a red box.

```
root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# nano postgresql.conf
```



```

root@projectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
GNU nano 2.9.1 File: postgresql.conf
-----
PostgreSQL configuration file
-----
This file consists of lines of the form:
# name = value
(The "=" is optional.) Whitespace may be used. Comments are introduced with
# anywhere on a line. The complete list of parameter names and allowed
values can be found in the PostgreSQL documentation.
The commented-out settings shown in this file represent the default values.
Re-commenting a setting is NOT sufficient to revert it to the default value;
you need to reload the server.
This file is read on server startup and when the server receives a SIGHUP
signal. If you edit the file on a running system, you have to SIGHUP the
server for the changes to take effect, run "pg_ctl reload", or execute
"SELECT pg_reload_conf()". Some parameters, which are marked below,
# Get Help      # WriteOut      # Read File      # Read 822 lines
# Exit          # Justify       # Where Is      # Prev Page
#               #               #               # Next Page
#               #               #               # Cut Text
#               #               #               # UnCut Text
#               #               #               # Cur Pos
#               #               #               # To Spell

```

- Cambiamos el puerto al puerto 4444:

```

root@projectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
GNU nano 2.9.1 File: postgresql.conf Modified
-----
# - Connection Settings -
listen_addresses = '*'
# comma-separated list of addresses;
# defaults to 'localhost'; use '*' for all
# (change requires restart)
port = 4444
# (change requires restart)
max_connections = 100
# (change requires restart)
reserved_connections = 0
# (change requires restart)
#superuser_reserved_connections = 3
# (change requires restart)
#unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
# (change requires restart)
#unix_socket_group = ''
# (change requires restart)
#unix_socket_permissions = 0777
# begin with 0 to use octal notation
# (change requires restart)
#bonjour = off
# advertise server via Bonjour
# (change requires restart)
#bonjour_name = ''
# defaults to the computer name
# Get Help      # WriteOut      # Read File      # Prev Page      # Cut Text      # Cur Pos
# Exit          # Justify       # Where Is      # Next Page      # UnCut Text     # To Spell

```

- Vamos a modificar el archivo: **nano pg_hba.conf**:

```

root@projectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@projectocourier datadir2]# nano pg_hba.conf

```

```

root@projectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
GNU nano 2.9.1 File: pg_hba.conf
-----
PostgreSQL Client Authentication Configuration File
-----
Refer to the "Client Authentication" section in the PostgreSQL
documentation for a complete description of this file. A short
synopsis follows.
-----
Authentication Records
-----
This file controls: which hosts are allowed to connect, how clients
are authenticated, which PostgreSQL user names they can use, which
databases they can access. Records take one of these forms:
# local      DATABASE USER METHOD [OPTIONS]
# host       DATABASE USER ADDRESS METHOD [OPTIONS]
# hostssl    DATABASE USER ADDRESS METHOD [OPTIONS]
# hostnossl  DATABASE USER ADDRESS METHOD [OPTIONS]
# hostgssenc DATABASE USER ADDRESS METHOD [OPTIONS]
# Get Help      # WriteOut      # Read File      # Read 128 lines
# Exit          # Justify       # Where Is      # Prev Page
#               #               #               # Next Page
#               #               #               # Cut Text
#               #               #               # UnCut Text
#               #               #               # Cur Pos
#               #               #               # To Spell

```


- Observamos que se encuentre correctamente la última línea y guardamos el archivo:

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
GNU nano 2.3.1 File: pg_hba.conf
# use another authentication method.

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 trust
# IPv6 local connections:
host all all ::1/128 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all trust
host replication all 127.0.0.1/32 trust
host replication all ::1/128 trust
host all all all scram-sha-256
Get Help WriteOut Read File Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell

```

- Detenemos la ejecución del contenedor y volvemos a ejecutarlo:

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# docker stop pgjelv2
pgjelv2
[root@proyectocourier datadir2]# docker start pgjelv2
pgjelv2
[root@proyectocourier datadir2]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
0eb7061aeab8   postgres  "docker-entrypoint.s..."  14 minutes ago Up 12 seconds  5432/Tcp       pgjelv2
[root@proyectocourier datadir2]#

```

- Ingresamos al **bash** del contenedor y como usuario **postgres** con el comando:
docker exec -it pgjelv2 bash

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# docker exec -it pgjelv2 bash
root@0eb7061aeab8:~#

```

- Nos va a aparecer un error por el puerto

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# docker exec -it pgjelv2 bash
root@0eb7061aeab8:~# su - postgres
postgres@0eb7061aeab8:~$ psql
psql: error: connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: No such file or directory
Is the server running locally and accepting connections on that socket?
postgres@0eb7061aeab8:~$

```

- Vamos a cambiar al nuevo puerto 4444, ya no nos aparecerá el error del puerto:

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# docker exec -it pgjelv2 bash
root@0eb7061aeab0:/# su - postgres
postgres@0eb7061aeab0:~$ psql
psql: error: connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: No such file or directory
Is the server running locally and accepting connections on that socket?
postgres@0eb7061aeab0:~$ psql -p 4444
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.
postgres=#

```

- Ingresamos el comando **show port;** para ver el puerto cambiado:

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# docker exec -it pgjelv2 bash
root@0eb7061aeab0:/# su - postgres
postgres@0eb7061aeab0:~$ psql
psql: error: connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: No such file or directory
Is the server running locally and accepting connections on that socket?
postgres@0eb7061aeab0:~$ psql -p 4444
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.
postgres=# show port;
port
-----
4444
(1 row)
postgres=#

```

- Ingresamos exit:

```

postgres=#
postgres=# exit
postgres@0eb7061aeab0:~$

```

- Nos conectamos como host con el comando **psql -h 0eb7061aeab0 -p 4444** al puerto **4444**: (con el container id 0eb7061aeab0)

```

postgres@0eb7061aeab0:~$ psql -h 0eb7061aeab0 -p 4444
Password for user postgres:
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=#

```

- Observamos que nos estamos conectando por TCP/IP y ya no por socket con el comando **\conninfo**:

```

root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
postgres@0eb7061aeab0:~$ psql -h 0eb7061aeab0 -p 4444
Password for user postgres:
psql: error: connection to server at "0eb7061aeab0" (172.17.0.2), port 4444 failed: FATAL: password authentication failed for user "postgres"

postgres@0eb7061aeab0:~$ psql -h 0eb7061aeab0 -p 4444
Password for user postgres:
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=# \conninfo
You are connected to database "postgres" as user "postgres" on host "0eb7061aeab0" (address "172.17.0.2") at port "4444".
postgres=#

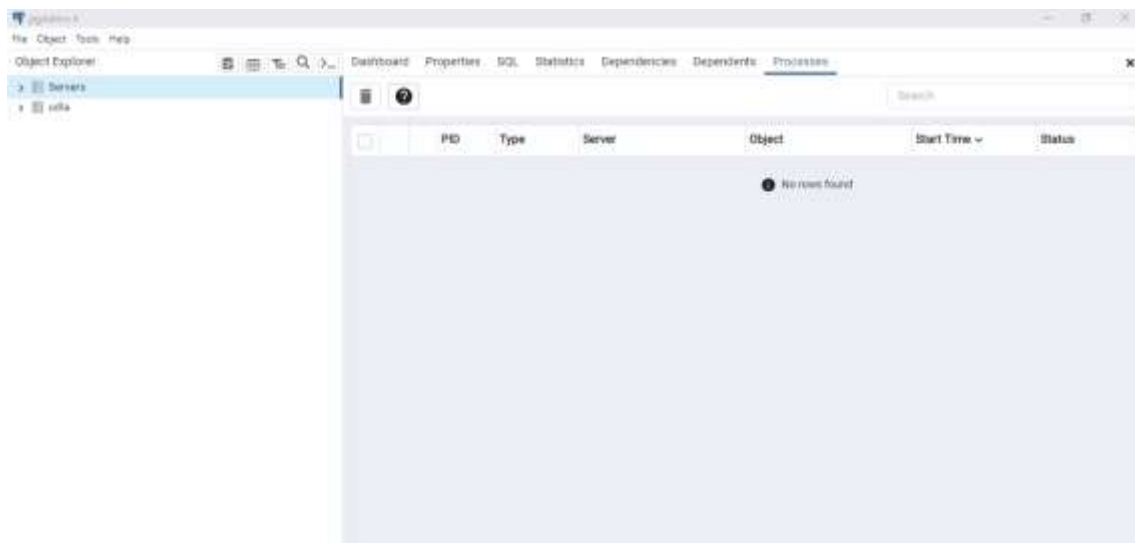
```

- Vamos a detener la ejecución del contenedor para correr con el puerto 15532:4444:
- **docker run --name pgjelv2 -p 15532:4444 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres16/datadir2:/var/lib/postgresql/data -d postgres**

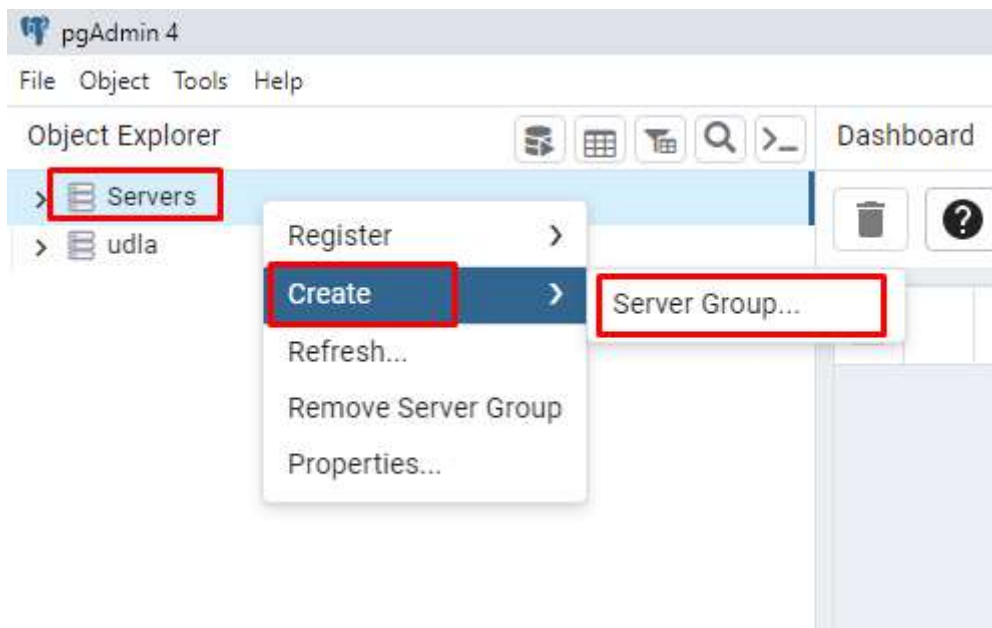
```
[root@projectocourier datadir2]# docker stop pgjelv2
pgjelv2
[root@projectocourier datadir2]# docker rm pgjelv2
pgjelv2
[root@projectocourier datadir2]# docker run --name pgjelv2 -p 15532:4444 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres15/datadir1:/var/lib/postgresql/data -d postgres
docker: Error response from daemon: Conflict. The container name "/pgjelv2" is already in use by container "cfba907c262f884a6e58d3a94ac164ec45dff73835f346291bc3e86743bcfdai". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[root@projectocourier datadir2]# docker run --name pgjelv2 -p 15532:4444 -e POSTGRES_PASSWORD=postgres -v /docker-storage/postgres16/datadir2:/var/lib/postgresql/data -d postgres
93f649b360985a0f6e06e61d89b0f3a383c041ef06b44ec9eb1fa52399770ff9
[root@projectocourier datadir2]#
```

5. Conexión de contenedor de Postgresql con PGAdmin 4 en Windows:

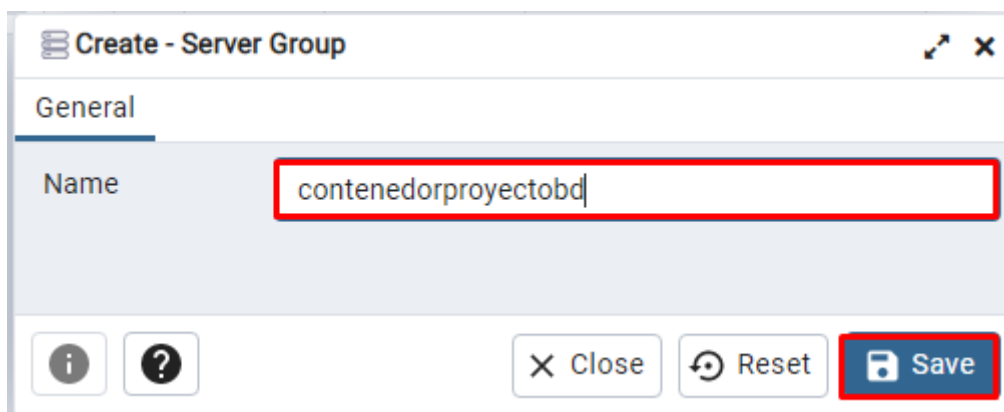
- Abrir PGAdmin 4:



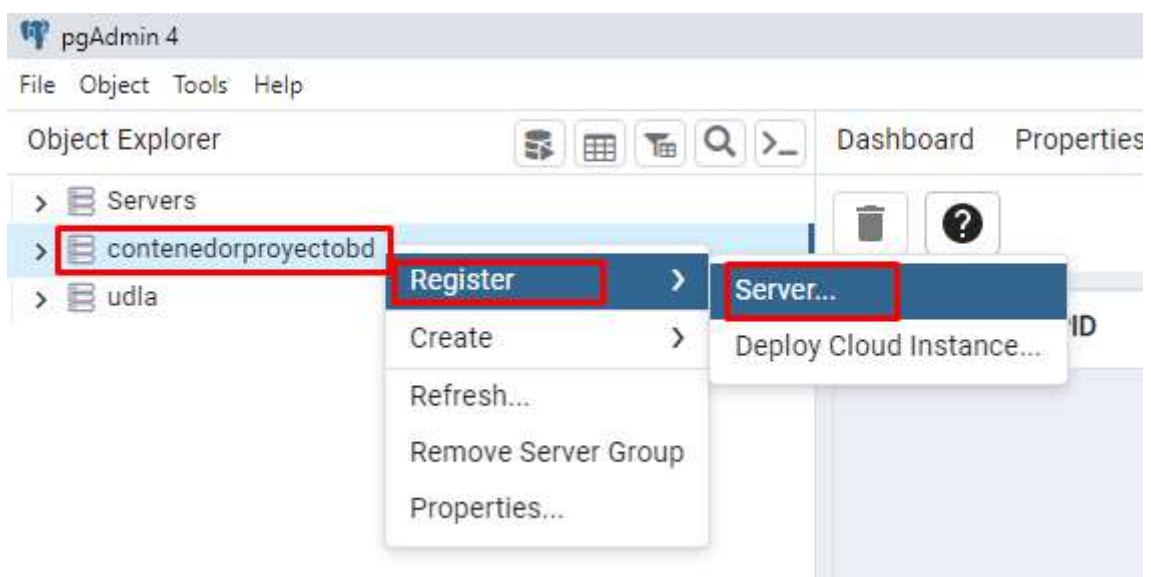
- Crear un nuevo "Server Group", damos clic en la opción **Servers -> Create -> Server Group**



- Ingresamos el nombre del Server Group: "**contenedorproyectobd**" y clic en el botón "Save"



- Vamos a registrar un nuevo servidor en el Server Group "contenedorproyectobd", damos clic derecho en "contenedorproyectobd" -> Register -> Server



- Ingresamos el nombre del servidor: **proyectobd**

The screenshot shows a 'Register - Server' dialog box with the following fields and controls:

- Name:** A text input field containing 'proyectobd'.
- Server group:** A dropdown menu showing 'contenedorproyectobd'.
- Background:** A button with an 'X' icon.
- Foreground:** A button with an 'X' icon.
- Connect now?:** A toggle switch that is currently turned on.
- Comments:** A large text area for entering comments.
- Buttons:** At the bottom, there are 'Close', 'Reset', and 'Save' buttons.

- Damos clic en la pestaña "Connections" e ingresamos:

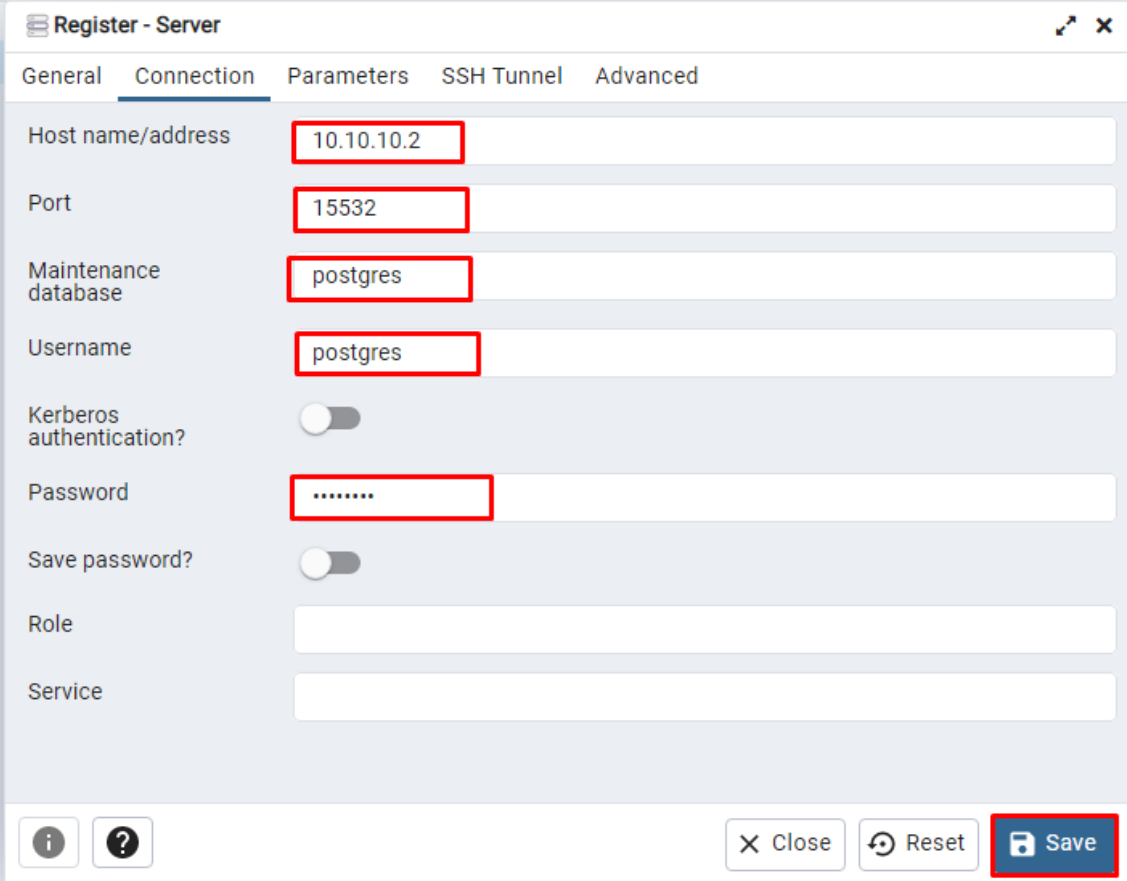
Host name/address: 10.10.10.2

Port: 15532

Maintenance database: postgres

Username: postgres

Password: postgres



Register - Server

General Connection Parameters SSH Tunnel Advanced

Host name/address 10.10.10.2

Port 15532

Maintenance database postgres

Username postgres

Kerberos authentication? ☐

Password

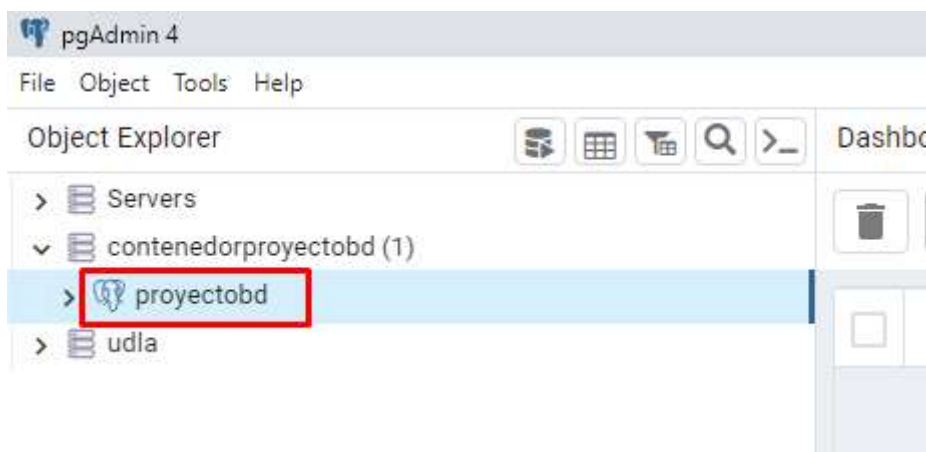
Save password? ☐

Role

Service

Close Reset Save

- Observamos que nos parece el servidor ya registrado y establecido conexión con el contenedor de Postgresql:



- Observamos le estatus del firewall de la maquina virtual: **systemctl status firewalld**

```

root@projectocourier:/docker-storage/postgres16/datadir2
[root@projectocourier datadir2]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: active (running) since Fri 2024-01-12 13:14:46 -05; 4s ago
     Docs: man:firewalld(1)
   Main PID: 28225 (firewalld)
      Tasks: 2
     Memory: 25.9M
    CGroup: /system.slice/firewalld.service
            └─28225 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Jan 12 13:14:45 projectocourier.lpmh.com systemd[1]: Starting firewalld - dynamic firewall daemon...
Jan 12 13:14:46 projectocourier.lpmh.com systemd[1]: Started firewalld - dynamic firewall daemon.
Jan 12 13:14:48 projectocourier.lpmh.com firewalld[28225]: WARNING: iptables not usable, disabling IPv6 firewall.
[root@projectocourier datadir2]#

```

- Vamos a bajar el firewall de la máquina virtual:

```

[root@projectocourier datadir2]# systemctl stop firewalld
[root@projectocourier datadir2]# systemctl disable firewalld
[root@projectocourier datadir2]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead) since Fri 2024-01-12 13:33:16 -05; 9s ago
     Docs: man:firewalld(1)
   Main PID: 28225 (code=exited, status=0/SUCCESS)

Jan 12 13:14:45 projectocourier.lpmh.com systemd[1]: Starting firewalld - dynamic firewall daemon...
Jan 12 13:14:46 projectocourier.lpmh.com systemd[1]: Started firewalld - dynamic firewall daemon.
Jan 12 13:14:48 projectocourier.lpmh.com firewalld[28225]: WARNING: iptables not usable, disabling IPv6 firewall.
Jan 12 13:33:14 projectocourier.lpmh.com systemd[1]: Stopping firewalld - dynamic firewall daemon...
Jan 12 13:33:16 projectocourier.lpmh.com systemd[1]: Stopped firewalld - dynamic firewall daemon.
[root@projectocourier datadir2]#

```

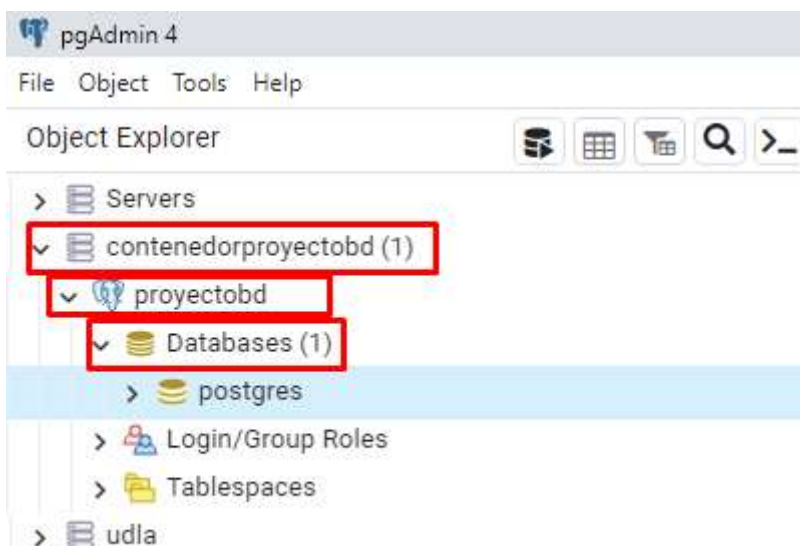
- Observamos la conexión activa: **netstat -ant**

```

[root@projectocourier datadir2]#
[root@projectocourier datadir2]# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:15532           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
[root@projectocourier datadir2]#

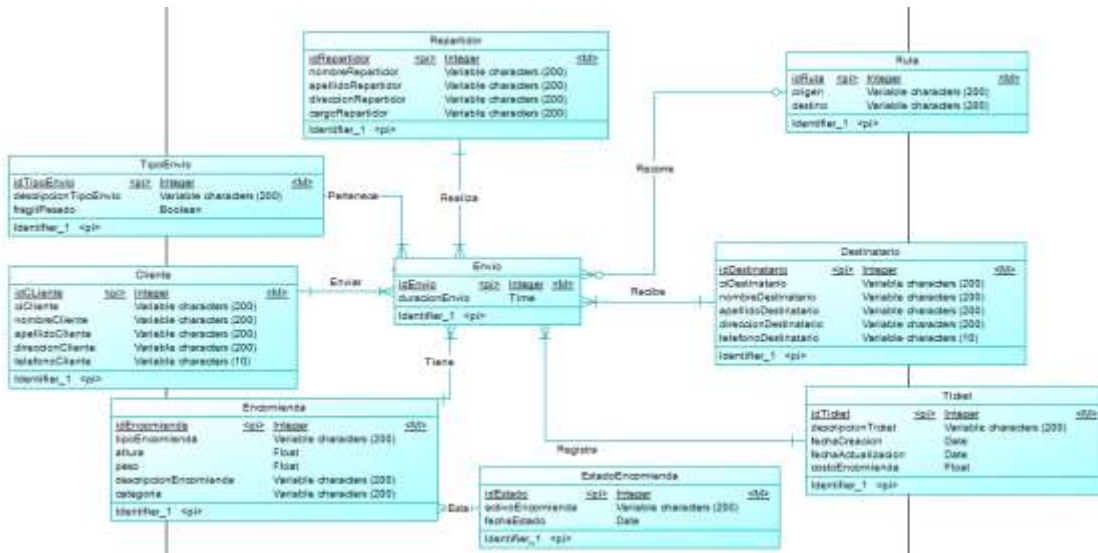
```

- Validamos que se realizó la conexión correcta con PGADMIN:

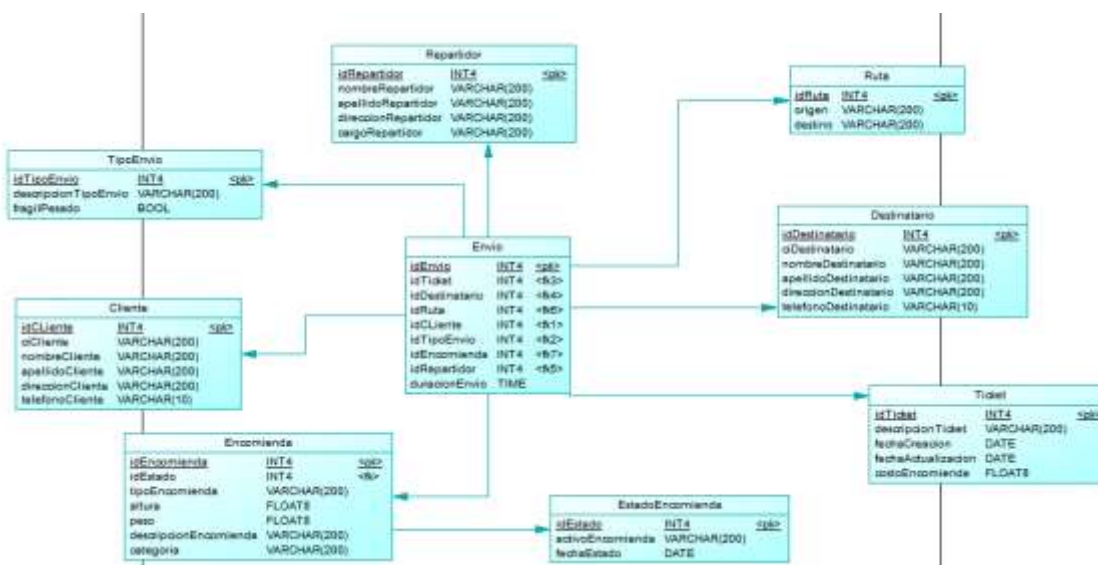


6. Modelado de la Base de Datos bdcourrier

Modelo lógico:

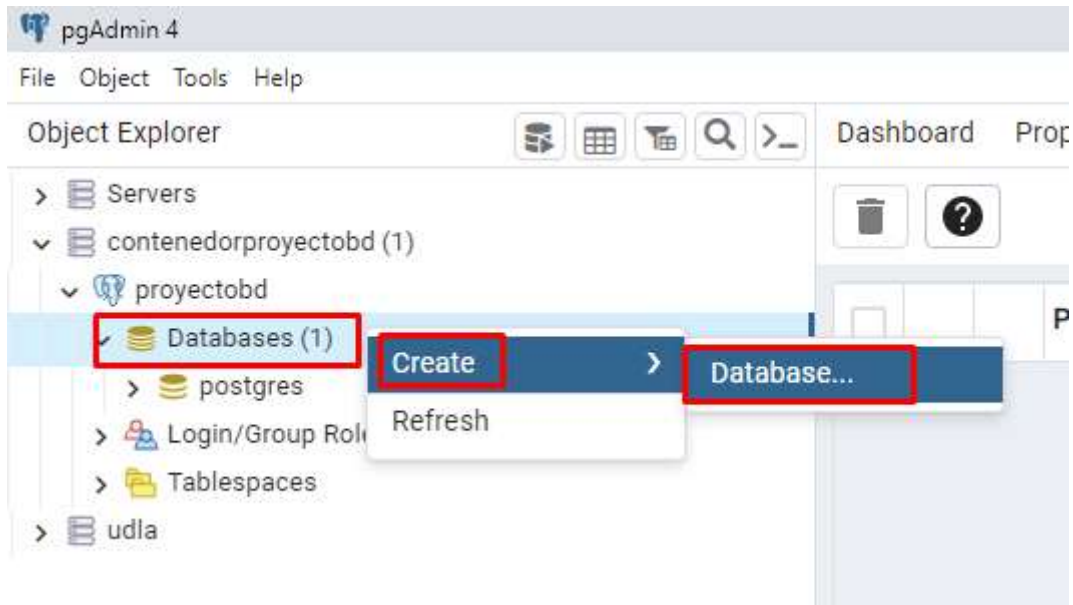


Modelo físico:

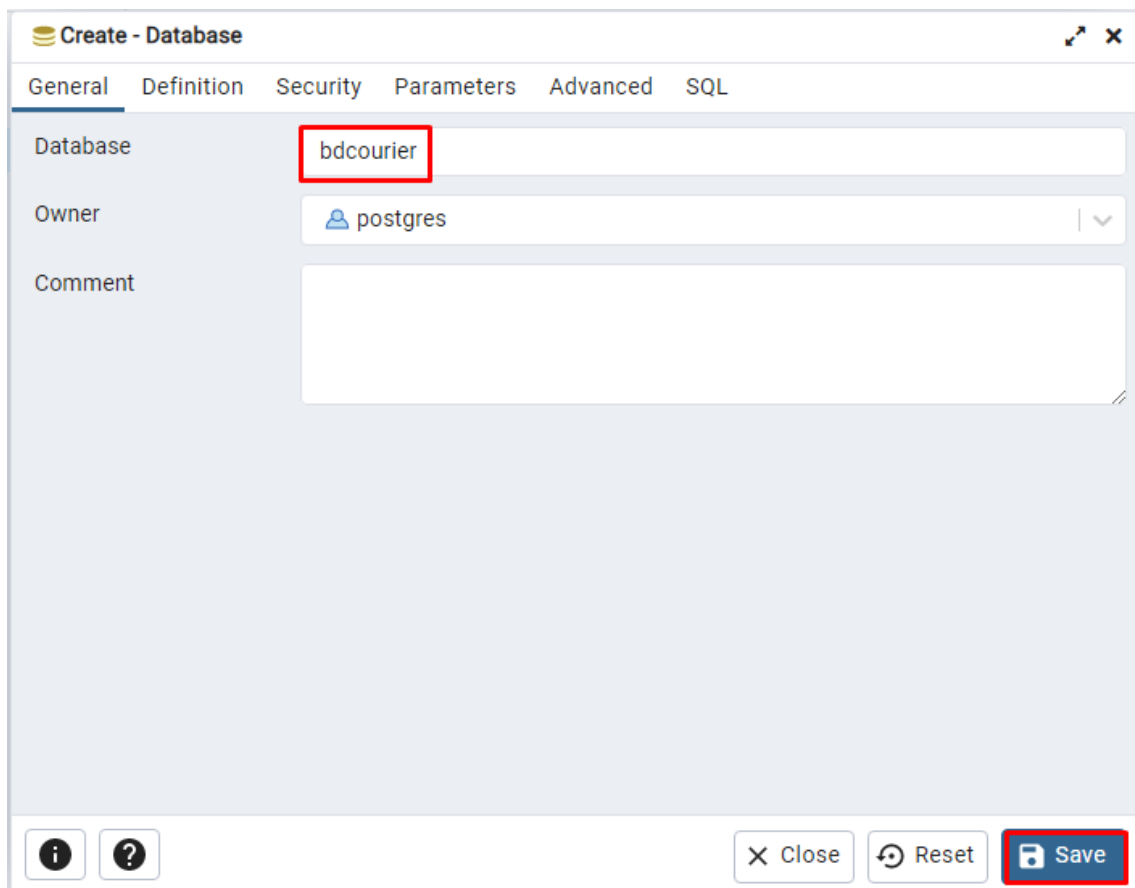


7. Creación de Base de Datos bdcourier en PGAdmin 4

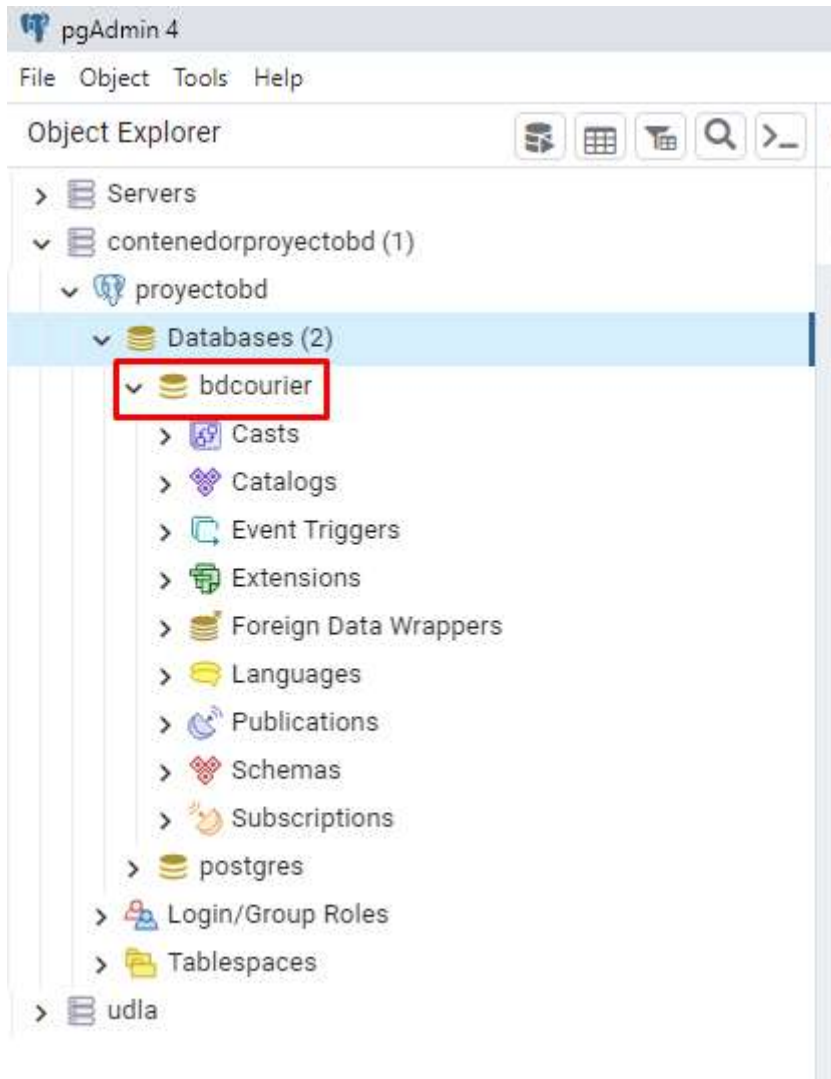
- Desde PGAdmin 4 ir a Databases -> Create -> Database



- Ingresar el nombre de la base de datos "**bdcourier**" y dar clic en "**Save**":



- Observamos que ya se encuentra creada la base da datos "bdcourier" en PGAdmin 4:



- Desde el contenedor de postgresql también observamos que se encuentra creada la base de datos.

Nos conectamos al contenedor: (Password: postgres)

```

root@projectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
[root@projectocourier datadir2]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED     STATUS    PORTS                               NAMES
93f649b36098   postgres  "docker-entrypoint.sh"   45 minutes ago    Up 45 minutes    5432/tcp, 0.0.0.0:15532->4444/tcp    pgjelv2
[root@projectocourier datadir2]# docker exec -it pgjelv2 bash
root@93f649b36098:/# su - postgres
postgres@93f649b36098:~$ psql -h 0.0.0.0 -p 5432
Password for user postgres:
psql (16.1 (Debian 16.1-1.pgdg120+1))
Type "help" for help.

postgres=#

```

- Observamos las bases de datos creadas en el contenedor de Postgresql con el comando: `\l`
- Y observamos la base de datos "bdcourier":

```

root@projectocourier:/docker-storage/postgres15/datadir2
File Edit View Search Terminal Help
[root@projectocourier datadir2]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
93f649b36898   postgres  "docker-entrypoint.s..." 45 minutes ago Up 45 minutes 5432/tcp, 0.0.0.0:15532->4444/tcp pgjelv2
[root@projectocourier datadir2]# docker exec -it pgjelv2 bash
root@93f649b36898:/# su - postgres
postgres@93f649b36898:~$ psql -h 93f649b36898 -p 4444
Password for user postgres:
psql [16.1 (Debian 16.1-1.pgdg12H+1)]
Type "help" for help.

postgres=# \l

```

Name	Owner	Encoding	Locale Provider	List of databases Collate	Ctype	ICU Locale	ICU Rules	Access privileges
bdcourier	postgres	UTF8	libc	en_US.utf8	en_US.utf8			
postgres	postgres	UTF8	libc	en_US.utf8	en_US.utf8			
template0	postgres	UTF8	libc	en_US.utf8	en_US.utf8			=c/postgres postgres=Ct/postgres
template1	postgres	UTF8	libc	en_US.utf8	en_US.utf8			=c/postgres postgres=Ct/postgres

```

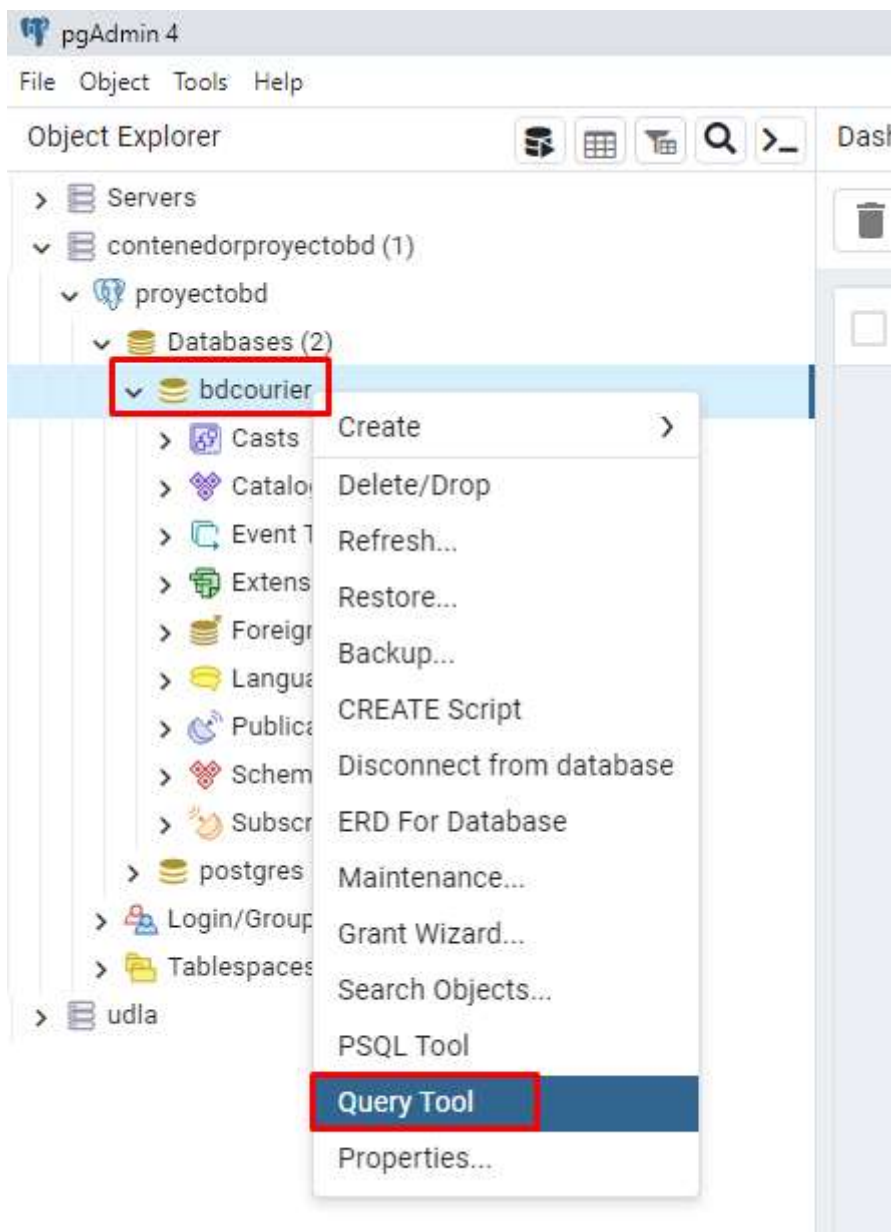
(4 rows)

postgres=#

```

8. Crear tablas en base de datos "bdcourier"

- Desde PGAdmin 4 vamos a ingresar en "bdcourier" -> "Query Tool"

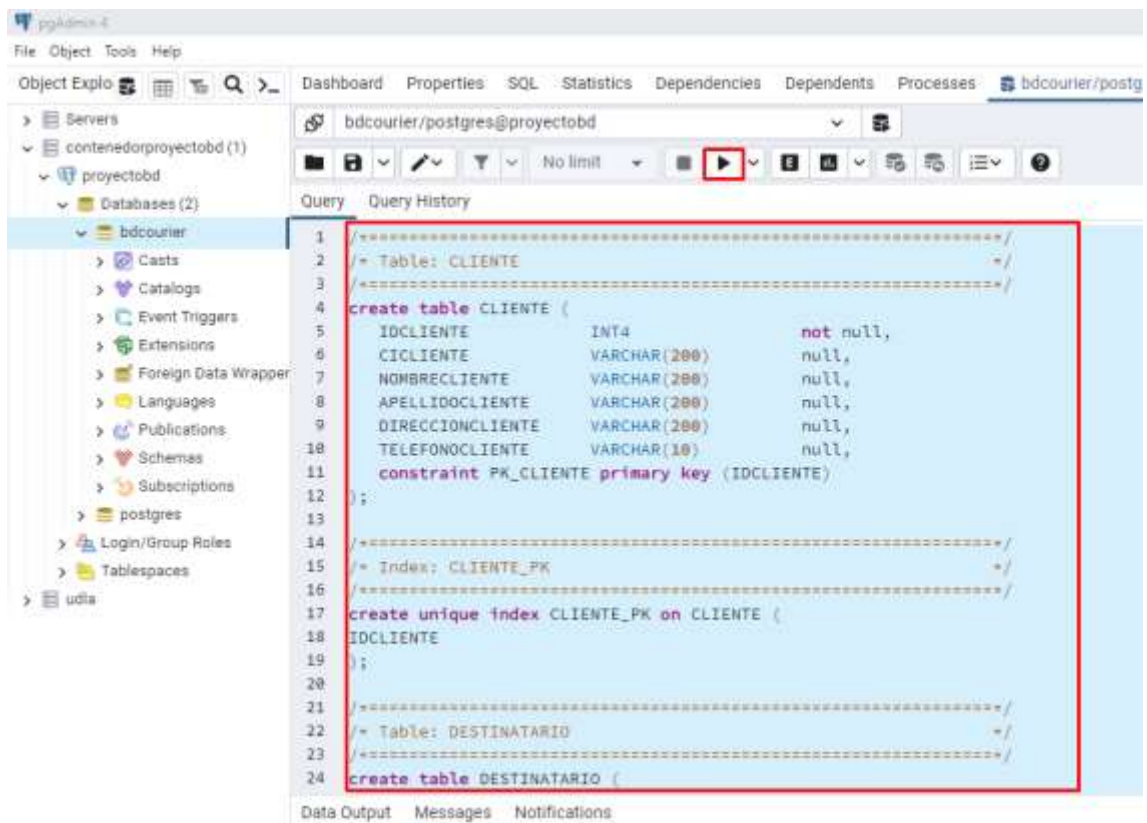


- Ingresamos el script de la base de datos modelada:

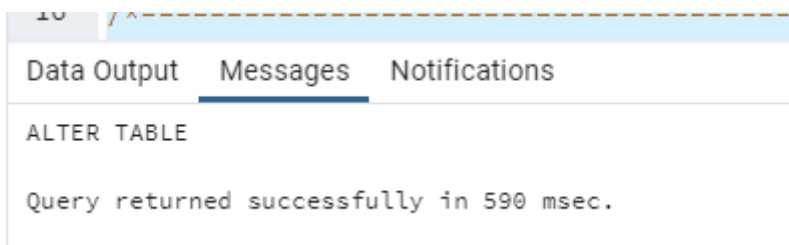
```

58
59  /*=====*/
60  /* Table: CLIENTE */
61  /*=====*/
62  create table CLIENTE (
63      IDCLIENTE          INT4          not null,
64      CICLIENTE          VARCHAR(200)  null,
65      NOMBRECLIENTE     VARCHAR(200)  null,
66      APELLIDOCIENTE    VARCHAR(200)  null,
67      DIRECCIONCLIENTE VARCHAR(200)  null,
68      TELEFONOCIENTE    VARCHAR(10)   null,
69      constraint PK_CLIENTE primary key (IDCLIENTE)
70  );
71
72  /*=====*/
73  /* Index: CLIENTE_PK */
74  /*=====*/
75  create unique index CLIENTE_PK on CLIENTE (
76      IDCLIENTE
77  );
78
79  /*=====*/
80  /* Table: DESTINATARIO */
81  /*=====*/
82  create table DESTINATARIO (
83      IDDESTINATARIO     INT4          not null,
84      CIDESTINATARIO     VARCHAR(200)  null,
85      NOMBREDESTINATARIO VARCHAR(200)  null,
86      APELLIDODESTINATARIO VARCHAR(200) null,
87      DIRECCIONDESTINATARIO VARCHAR(200) null,
88      TELEFONODESTINATARIO VARCHAR(10)  null,
89      constraint PK_DESTINATARIO primary key (IDDESTINATARIO)
90  );
91
92  /*=====*/
93  /* Index: DESTINATARIO_PK */
94  /*=====*/
95  create unique index DESTINATARIO_PK on DESTINATARIO (
  
```

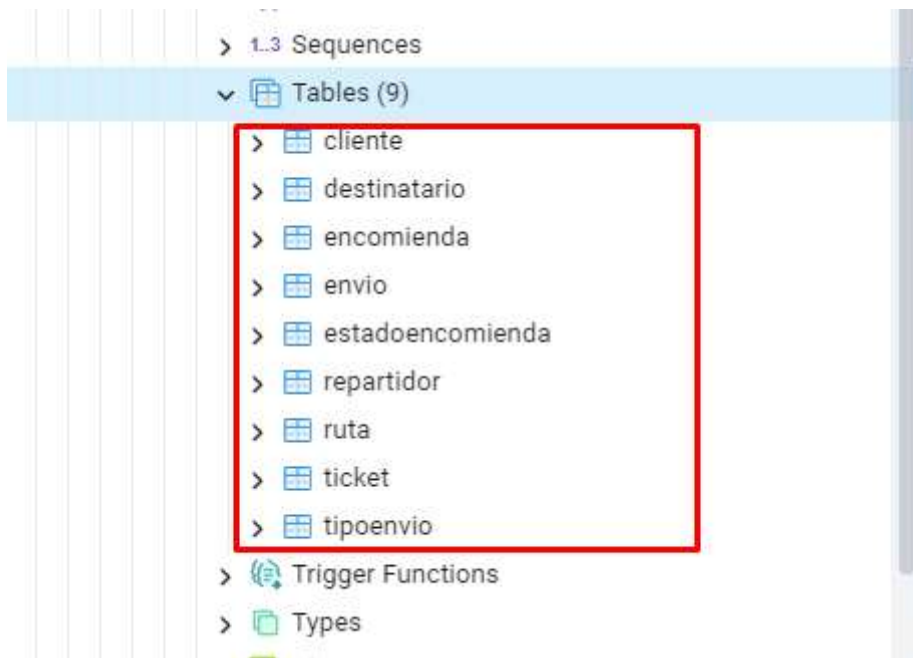
- Damos clic en el botón "Play" para que se ejecuten las sentencias SQL de creación e las tablas del modelado dentro de la base de datos "bdcourier":



- Observamos que se a ejecutado correctamente el query:



- Visualizamos que se han creado correctamente las tablas en la base de datos "bdcourier":

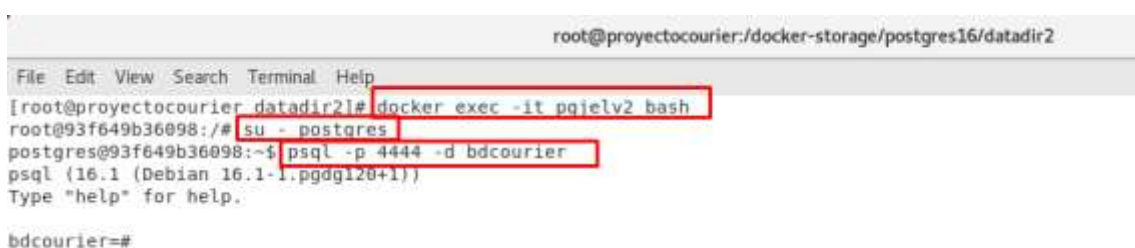


- Conexión desde PgAdmin instalado en Windows y desde el cual se pueda consultar la tabla cliente:
- Salimos de la conexión actual con exit:

```
postgres=# exit
postgres@93f649b36098:~$ exit
logout
root@93f649b36098:/# exit
exit
[root@proyectocourier datadir2]#
[root@proyectocourier datadir2]#
```

- **Ingresamos desde la máquina virtual como usuario postgres por el puerto 4444 y a la base de datos bdcourier:**

```
docker exec -it pgjelv2 bash
su - postgres
psql -p 4444 -d bdcourier
```



- Realizamos un select para consultar los datos de la base de datos bdcourier y validar la conexión:


```

root@proyectocourier:/docker-storage/postgres15/datadir2
File Edit View Search Terminal Help
[root@proyectocourier datadir2]# docker exec -it pgjelv2 bash
root@93f649b36098:/# su - postgres
postgres@93f649b36098:~$ psql -p 4444 -d bdcourier
psql (15.1 (Debian 15.1-1.pgdg120+1))
Type "help" for help.

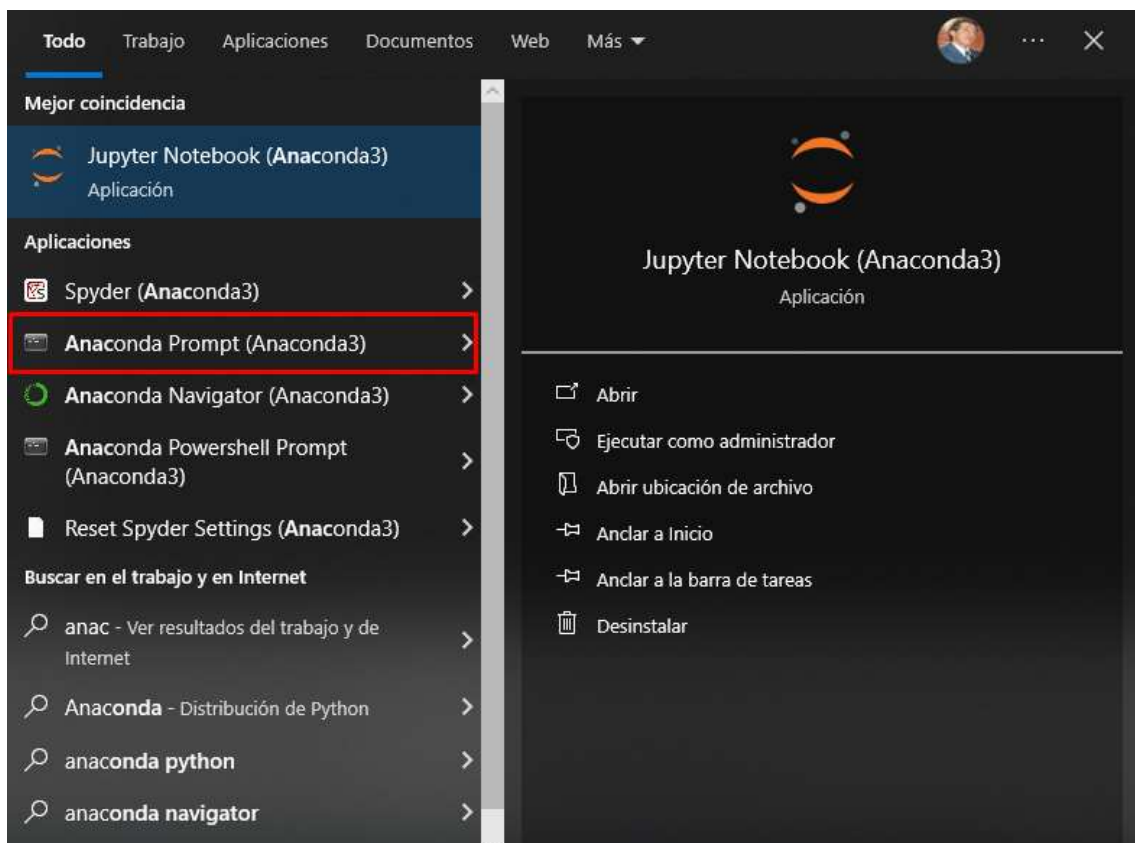
bdcourier=# select * from cliente;
 idcliente | ticliente | nombrecliente | apellidocliente | direccioncliente | telefonocliente
-----
(0 rows)

bdcourier=#

```

9. Poblar la Base de datos "bdcourier" con Faker

- Abrimos la consola de Anaconda:



- En la consola de Anaconda ingresamos el comando para instalar faker: **pip install faker**

```

Anaconda Prompt (Anaconda3)

(base) C:\Users\jona>pip install faker

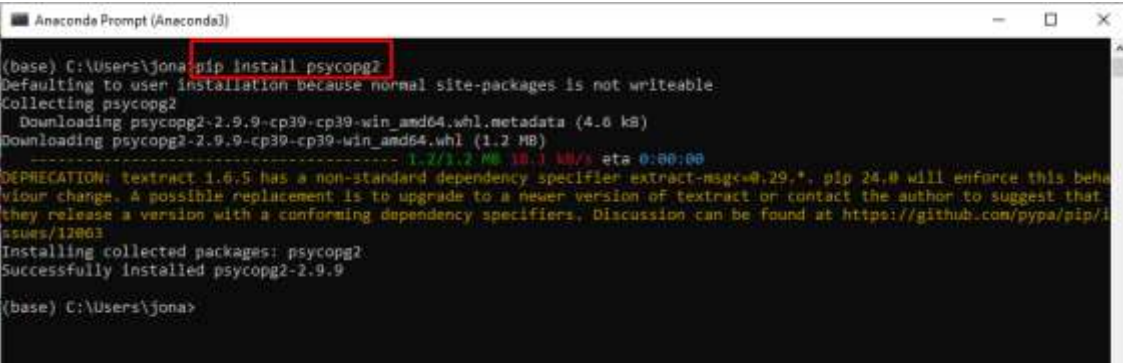
```

- Esperamos a que se realice la descarga e instalación del módulo "Faker" en Python

```
(base) C:\Users\jona>pip install faker
Defaulting to user installation because normal site-packages is not writeable
Collecting faker
  Downloading Faker-22.2.0-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: python-dateutil>=2.4 in c:\programdata\anaconda3\lib\site-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.4->faker) (1.12.0)
Downloading Faker-22.2.0-py3-none-any.whl (1.7 MB)
----- 1.7/1.7 MB 21.8 kB/s eta 0:00:00
DEPRECATION: textextract 1.6.5 has a non-standard dependency specifier extract-msg<0.29.*. pip 24.0 will enforce this behaviour change. A possible replacement is to upgrade to a newer version of textextract or contact the author to suggest that they release a version with a conforming dependency specifiers. Discussion can be found at https://github.com/pypa/pip/issues/12063
Installing collected packages: faker
  WARNING: The script faker.exe is installed in 'C:\Users\jona\AppData\Roaming\Python\Python39\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed faker-22.2.0

(base) C:\Users\jona>
```

- En la consola de Anaconda ingresamos el comando para instalar faker: **pip install faker psycpg2**



```
Anaconda Prompt (Anaconda3)
(base) C:\Users\jona>pip install psycpg2
Defaulting to user installation because normal site-packages is not writeable
Collecting psycpg2
  Downloading psycpg2-2.9.9-cp39-cp39-win_amd64.whl.metadata (4.6 kB)
  Downloading psycpg2-2.9.9-cp39-cp39-win_amd64.whl (1.2 MB)
----- 1.2/1.2 MB 10.1 kB/s eta 0:00:00
DEPRECATION: textextract 1.6.5 has a non-standard dependency specifier extract-msg<0.29.*. pip 24.0 will enforce this behaviour change. A possible replacement is to upgrade to a newer version of textextract or contact the author to suggest that they release a version with a conforming dependency specifiers. Discussion can be found at https://github.com/pypa/pip/issues/12063
Installing collected packages: psycpg2
Successfully installed psycpg2-2.9.9

(base) C:\Users\jona>
```

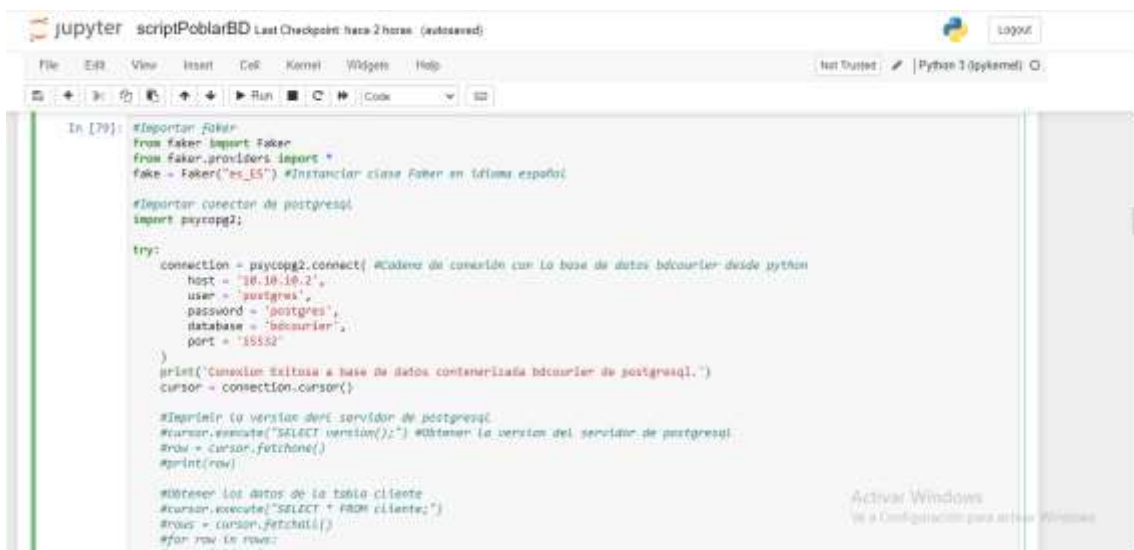
- Abrimos Jupyter Notebook



- Crear un script llamado "scriptPoblarBD.py"

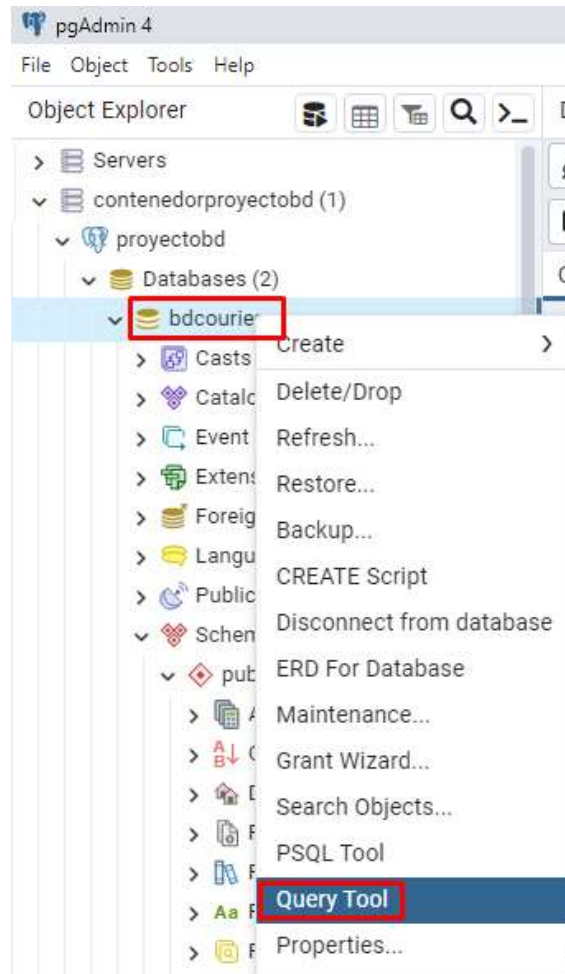


- Creamos el script para la inserción de datos en las tablas utilizando la librería de Python Faker:



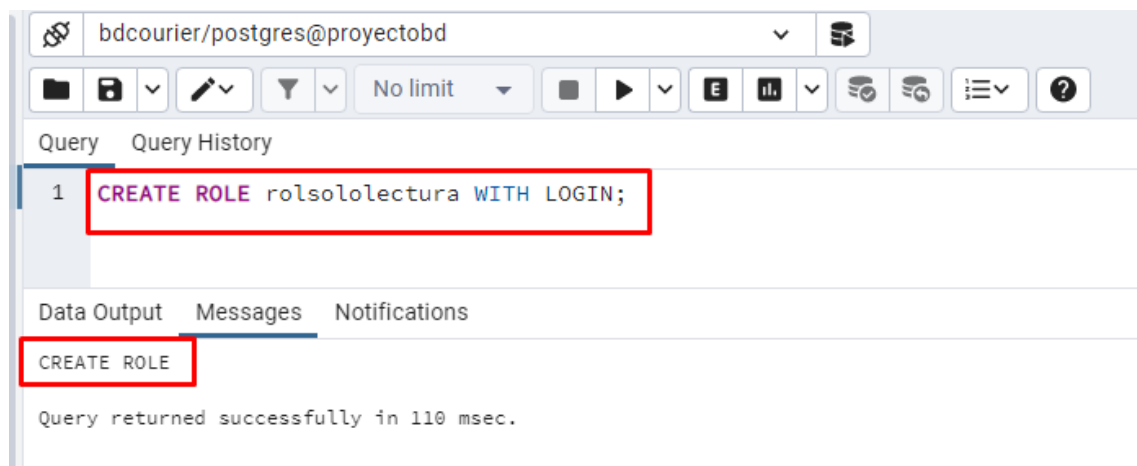
10. Creación de usuarios roles

- Desde Pgadmin ingresamos a "bdcourier" -> "Query Tool" para ingresar las sentencias sql para crear los roles y usuarios



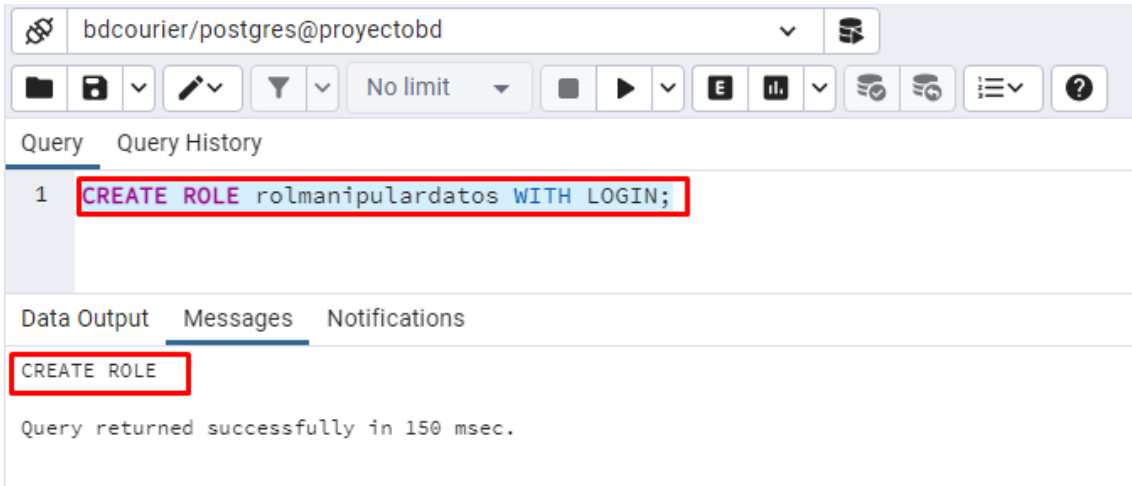
- Creamos el primer role llamado "rolsololectura":

CREATE ROLE rolsololectura WITH LOGIN;



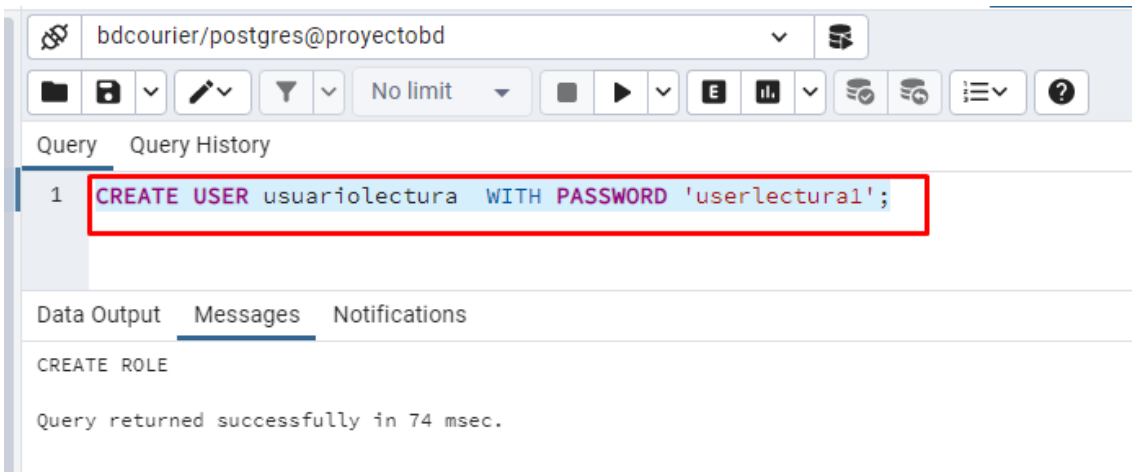
- Creamos el segundo rol llamado "rolmanipulardatos":

CREATE ROLE rolmanipulardatos WITH LOGIN;



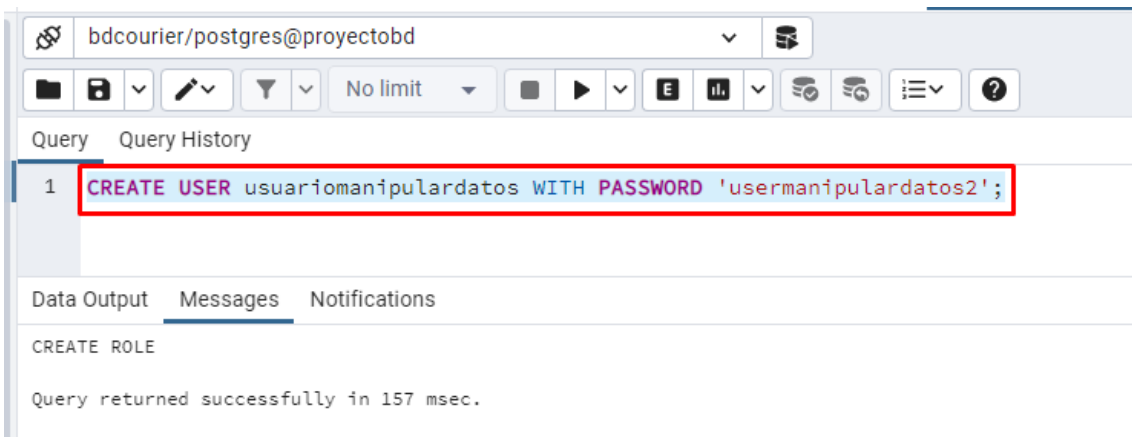
The screenshot shows the bdcourier PostgreSQL interface. The connection is 'bdcourier/postgres@proyectobd'. The query editor contains the SQL command: `CREATE ROLE rolmanipulardatos WITH LOGIN;`. The 'Messages' tab is selected, displaying the output: `CREATE ROLE` and `Query returned successfully in 150 msec.`

- Creamos el primer usuario llamado "usuariolectura":
CREATE USER usuariolectura WITH PASSWORD 'userlectura1';



The screenshot shows the bdcourier PostgreSQL interface. The connection is 'bdcourier/postgres@proyectobd'. The query editor contains the SQL command: `CREATE USER usuariolectura WITH PASSWORD 'userlectura1';`. The 'Messages' tab is selected, displaying the output: `CREATE ROLE` and `Query returned successfully in 74 msec.`

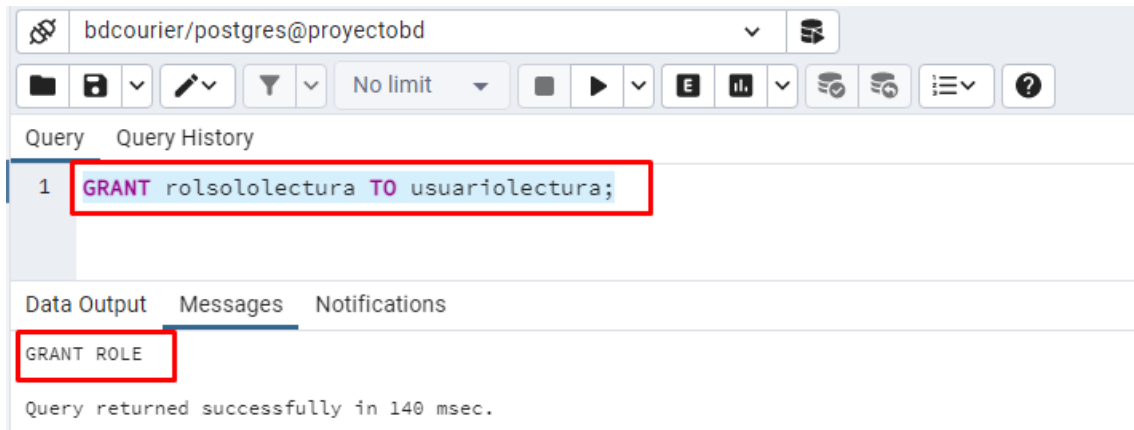
- Creamos el segundo usuario llamado "usariomanipulardatos":
CREATE USER usariomanipulardatos WITH PASSWORD 'usermanipulardatos2';



The screenshot shows the bdcourier PostgreSQL interface. The connection is 'bdcourier/postgres@proyectobd'. The query editor contains the SQL command: `CREATE USER usariomanipulardatos WITH PASSWORD 'usermanipulardatos2';`. The 'Messages' tab is selected, displaying the output: `CREATE ROLE` and `Query returned successfully in 157 msec.`

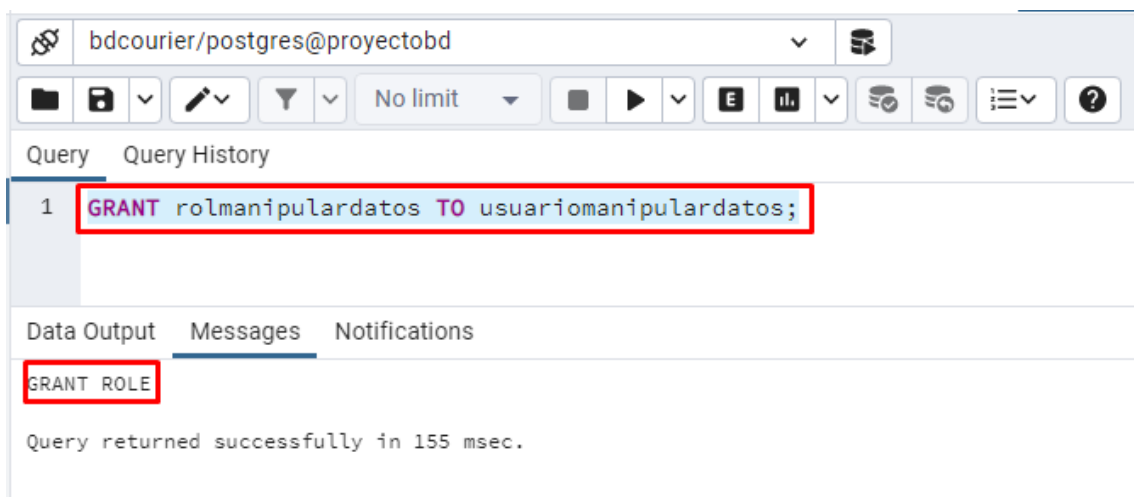
- Concedemos el primer rol "**rolsololectura**" al primer usuario "**usuariolectura**":

GRANT rolsololectura TO usuariolectura;



- Concedemos el segundo rol "**rolmanipulardatos**" al segundo usuario "**usuariomanipulardatos**":

GRANT rolmanipulardatos TO usuariomanipulardatos;



- Asignamos los privilegios de solo lectura al rol "**rolsololectura**":

DO \$\$

DECLARE

r RECORD;

BEGIN

FOR r IN SELECT tablename FROM pg_tables WHERE schemaname = 'public'

LOOP

EXECUTE 'GRANT SELECT ON ' || quote_ident(r.tablename) || ' TO rolsololectura;

END LOOP;

END

\$\$;

```

1 DO $$
2 DECLARE
3     r RECORD;
4 BEGIN
5     FOR r IN SELECT tablename FROM pg_tables WHERE schemaname = 'public'
6     LOOP
7         EXECUTE 'GRANT SELECT ON ' || quote_ident(r.tablename) || ' TO rolsololectura';
8     END LOOP;
9 END
10 $$;
11

```

Data Output Messages Notifications

DO

Query returned successfully in 334 msec.

- Asignamos los privilegios de solo manipulación de datos al rol "rolmanipulardatos":

```

DO $$
DECLARE
    r RECORD;
BEGIN
    FOR r IN SELECT tablename FROM pg_tables WHERE schemaname = 'public'
    LOOP
        EXECUTE 'GRANT SELECT, INSERT, UPDATE, DELETE ON ' || quote_ident(r.tablename) || '
TO rolmanipulardatos;
    END LOOP;
END
$$;

```

```

DO $$
DECLARE
    r RECORD;
BEGIN
    FOR r IN SELECT sequence_name FROM information_schema.sequences WHERE
sequence_schema = 'public'
    LOOP
        EXECUTE 'GRANT USAGE, SELECT ON SEQUENCE public.' ||
quote_ident(r.sequence_name) || ' TO rolmanipulardatos;
    END LOOP;
END
$$;

```



```

1 DO $$
2 DECLARE
3   r RECORD;
4 BEGIN
5   FOR r IN SELECT tablename FROM pg_tables WHERE schemaname = 'public'
6   LOOP
7     EXECUTE 'GRANT SELECT, INSERT, UPDATE, DELETE ON ' || quote_ident(r.tablename) || ' TO rolmanipulardatos';
8   END LOOP;
9 END
10 $$;
11
12 DO $$
13 DECLARE
14   r RECORD;
15 BEGIN
16   FOR r IN SELECT sequence_name FROM information_schema.sequences WHERE sequence_schema = 'public'
17   LOOP
18     EXECUTE 'GRANT USAGE, SELECT ON SEQUENCE public.' || quote_ident(r.sequence_name) || ' TO rolmanipulardatos';
19   END LOOP;
20 END
21 $$;

```

Query returned successfully in 199 msec.

- Observamos los usuarios y roles existentes en la base de datos:

\du+

```

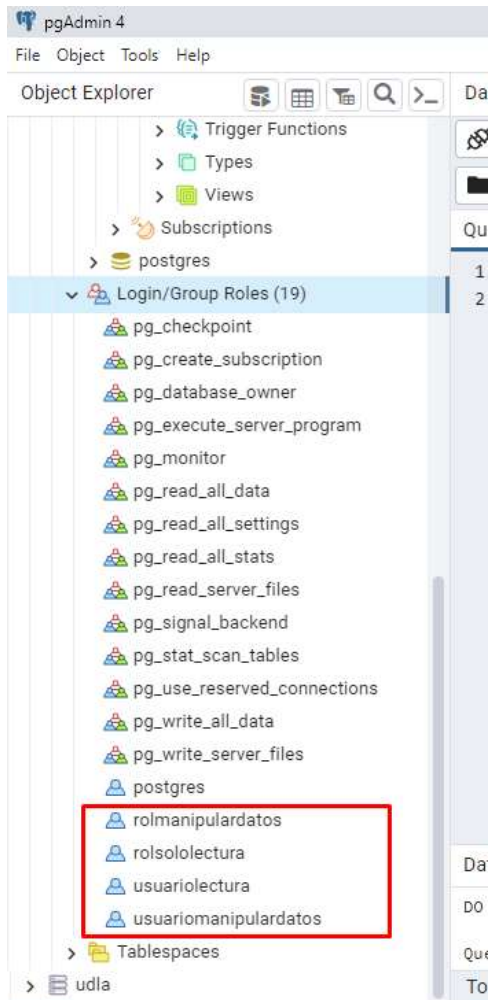
root@proyectocourier:/docker-storage/postgres16/datadir2
File Edit View Search Terminal Help
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=#
bdcourier=# \du+

```

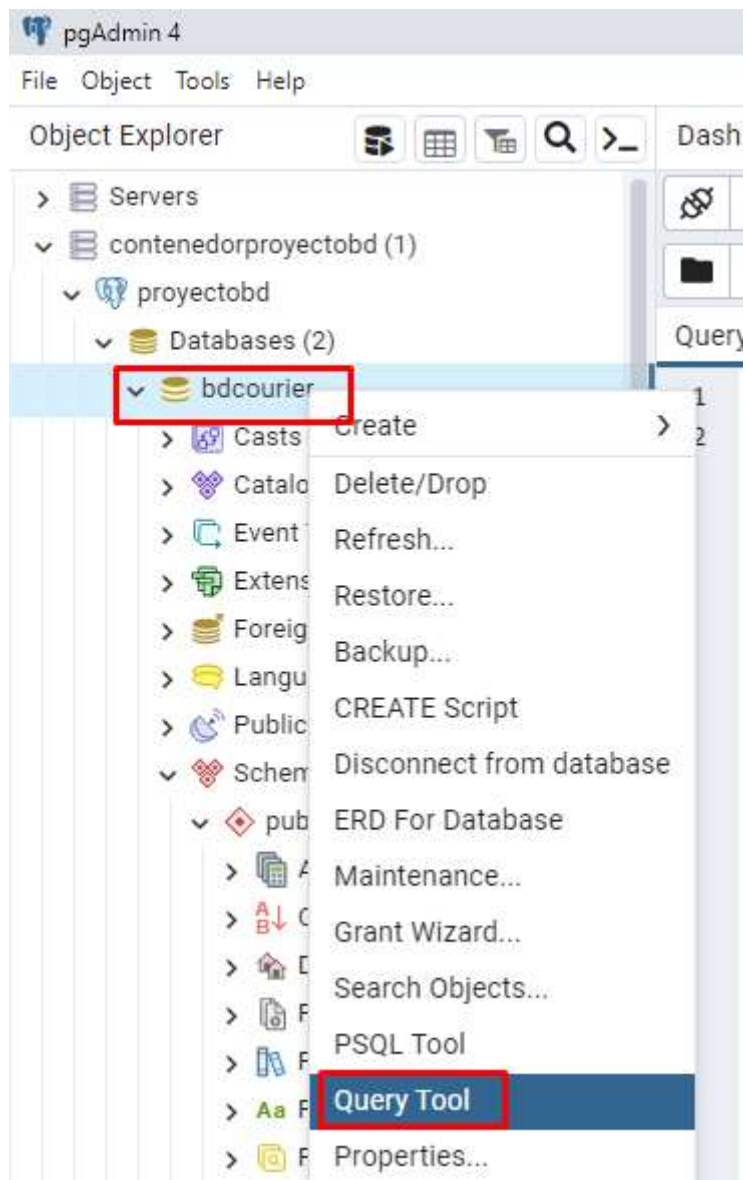
Role name	List of roles	Attributes	Description
postgres			Superuser, Create role, Create DB, Replication, Bypass RLS
rolmanipulardatos			
rolsololectura			
usuariolectura			
usariomanipulardatos			

bdcourier=#

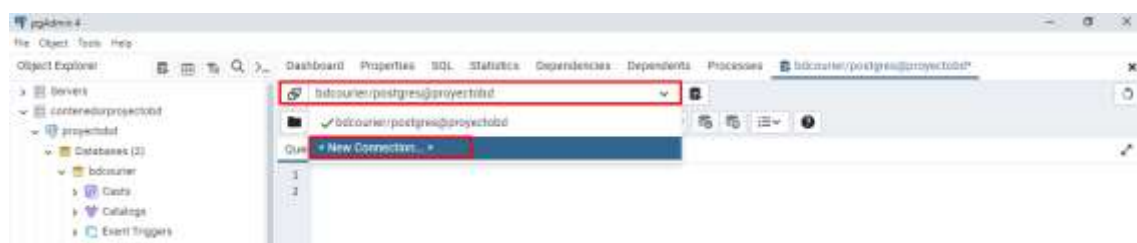
- Desde PGAdmin 4 también podemos observar los usuarios y roles creados ingresando a "Login/Group Roles"



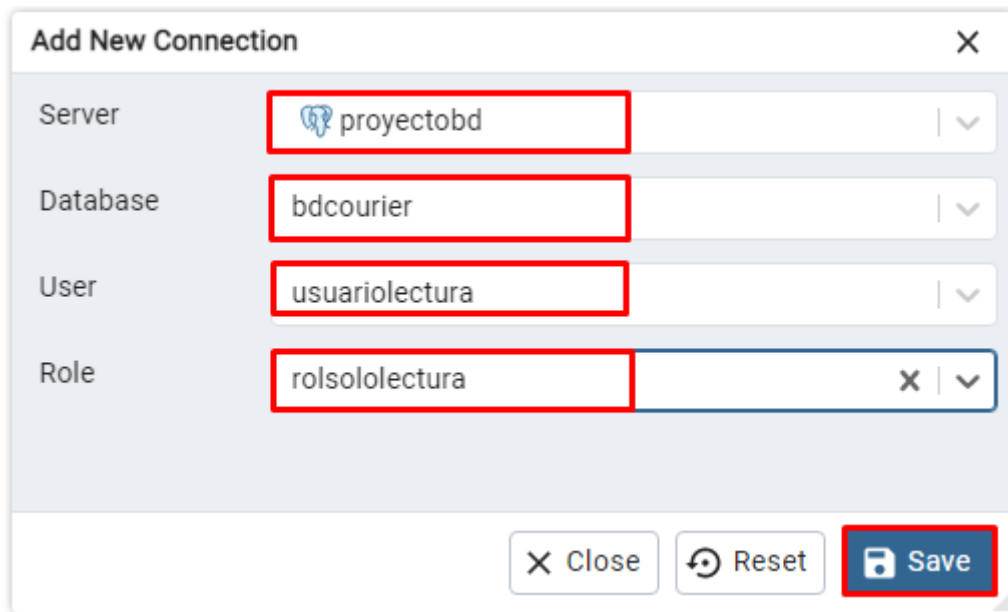
- Nos vamos a conectar como usuario "usuariolectura":



- Damos clic en la opción de la conexión actual y seleccionamos " New connection >"



- Ingresamos los parámetros para la nueva conexión:



Add New Connection

Server: proyectobd

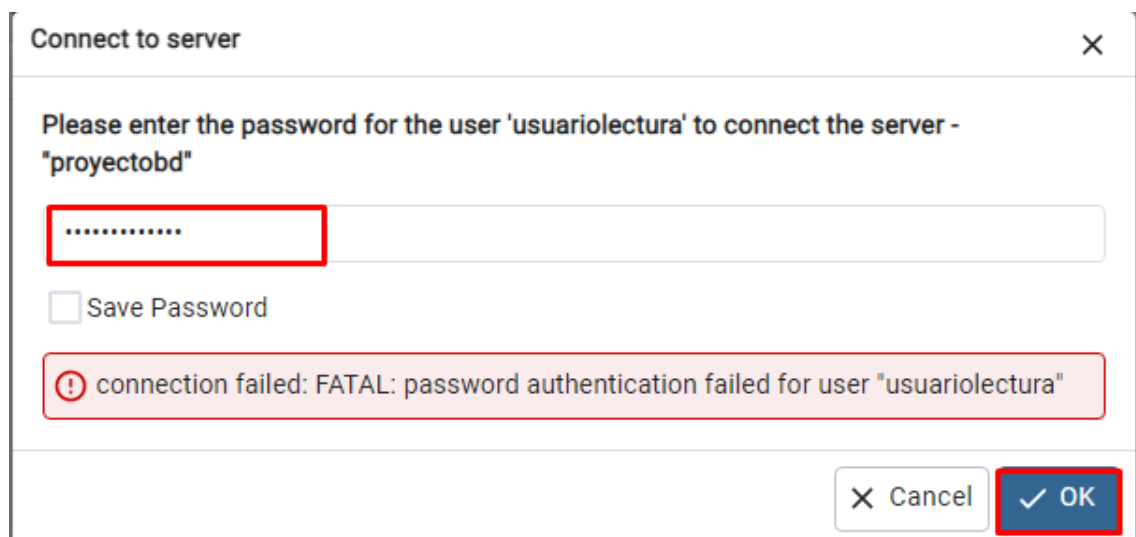
Database: bdcourier

User: usuariolectura

Role: rolsololectura

Close Reset Save

Ingresamos el password para la conexión: "userlectura1"



Connect to server

Please enter the password for the user 'usuariolectura' to connect the server - "proyectobd"

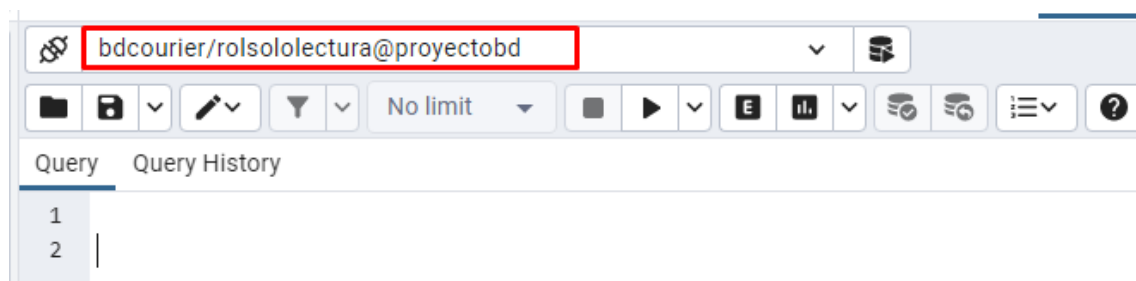
.....

☐ Save Password

connection failed: FATAL: password authentication failed for user "usuariolectura"

Cancel OK

Observamos que se realizó la conexión correctamente:



bdcourier/rolsololectura@proyectobd

Query Query History

1

2

- Realizamos una consulta a la base de datos "bdcourier" y se observa que si es posible:

bdcourier/rolsololectura@proyectobd

</

- Intentamos realizar un INSERT INTO en la tabla "cliente" y no se nos permite ya que el rol "rolsololectura" asignado al usuario "usuariolectura" solo permite la lectura de los datos de la base de datos:

insert into cliente

(ccliente,nombrecliente,apellidocliente,direccioncliente,telefonocliente) values ('1723593596','Jonathan','Lema','La Mena','0996523254');

The screenshot shows the same database query tool interface. The query editor now contains the SQL statement: `insert into cliente (ccliente,nombrecliente,apellidocliente,direccioncliente,telefonocliente) values ('1723593596',`. The 'Data Output' tab is still active, but it displays an error message: `ERROR: permission denied for table cliente`. Below the error message, the 'SQL state' is shown as `42501`.

- Intentamos realizar un UPDATE en la tabla "cliente" y no se nos permite:

UPDATE cliente

SET nombrecliente = 'Pedro'

WHERE cicliente = '8314519454';

The screenshot shows a database client interface with a toolbar at the top containing icons for database operations and a dropdown menu showing 'bdcourier/rolsololectura@proyectobd'. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query in a text editor. The query is highlighted with a red box and consists of three lines: 'UPDATE cliente', 'SET nombrecliente = 'Pedro'', and 'WHERE cicliente = '8314519454';'. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying an error message: 'ERROR: permission denied for table cliente'. Below the error message, the text 'SQL state: 42501' is visible.

```
1 UPDATE cliente
2 SET nombrecliente = 'Pedro'
3 WHERE cicliente = '8314519454';
4
5
```

ERROR: permission denied for table cliente

SQL state: 42501

- Intentamos eliminar un registro en la tabla "cliente" y no se nos permite:

DELETE FROM cliente WHERE cicliente = '8314519454';

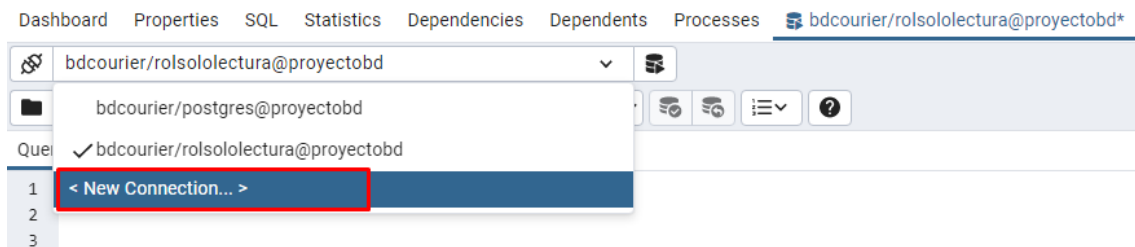
The screenshot shows a database client interface with a toolbar at the top containing icons for database operations and a dropdown menu showing 'bdcourier/rolsololectura@proyectobd'. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query in a text editor. The query is highlighted with a red box and consists of one line: 'DELETE FROM cliente WHERE cicliente = '8314519454';'. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying an error message: 'ERROR: permission denied for table cliente'. Below the error message, the text 'SQL state: 42501' is visible.

```
1 DELETE FROM cliente WHERE cicliente = '8314519454';
2
3
```

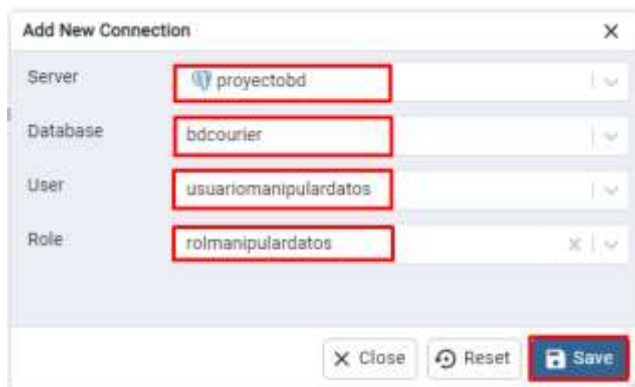
ERROR: permission denied for table cliente

SQL state: 42501

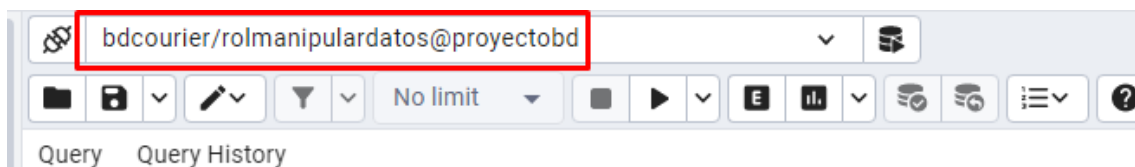
- Vamos a conectarnos a la base de datos "bdcourier" con el usuario "usuariomanipulardatos" dando clic en la opción de la conexión actual y seleccionamos "New connection >":



- Ingresamos los parámetros para la nueva conexión:



- Ingresamos el password para la conexión: "**usermanipulardatos2**"
- Observamos que se realizó la conexión correctamente:



- Realizamos una consulta a la base de datos "bdcourier" y se observa que si es posible:

bdcourier/rolmanipulardatos@proyectobd

Query Query History

```
1 select * from repartidor;
```

Data Output Messages Notifications

	idrepartidor [PK] integer	nombrerepartidor character varying	apellidorepartidor character varying	direccionrepartidor character varying	cargorepartidor character varying
1	11	Cayetana	Caro	Alameda Guadalupe Ponce 925 Piso 0	Repartidor 27y
2	12	Victoriano	Ferrández	Avenida Amor Montenegro 525	Repartidor 28C
3	13	Perlita	Guzmán	Cuesta de Valentina Simó 80	Repartidor 36o
4	14	Fito	Bermúdez	Ronda de Maxi Villegas 8	Repartidor 51I
5	15	Bernardino	Escobar	Via Jesusa Taboada 18 Piso 1	Repartidor 73j
6	16	Ana Sofía	Morales	Pasaje de Marisela Valcárcel 41	Repartidor 99W
7	17	Nacho	Patiño	Camino de Rosa Mateos 39	Repartidor 97c
8	18	Ale	Palomares	Callejón de Ruth Pereira 70	Repartidor 64U
9	19	Luciana	Batlle	Plaza Magdalena Vall 3	Repartidor 62d
10	20	Xiomara	Tello	Urbanización de Natalio Pujol 175 Apt. 06	Repartidor 53Q

- Intentamos realizar un INSERT INTO en la tabla repartidor y si se nos permite:

INSERT INTO

repartidor(nombrerepartidor,apellidorepartidor,direccionrepartidor,cargorepartidor)
VALUES ('Jorge','Mena','La tola','Repartido 10a');

bdcourier/rolmanipulardatos@proyectobd

Query Query History

```
1 INSERT INTO repartidor(nombrerepartidor,apellidorepartidor,direccionrepartidor,cargorepartidor)
```

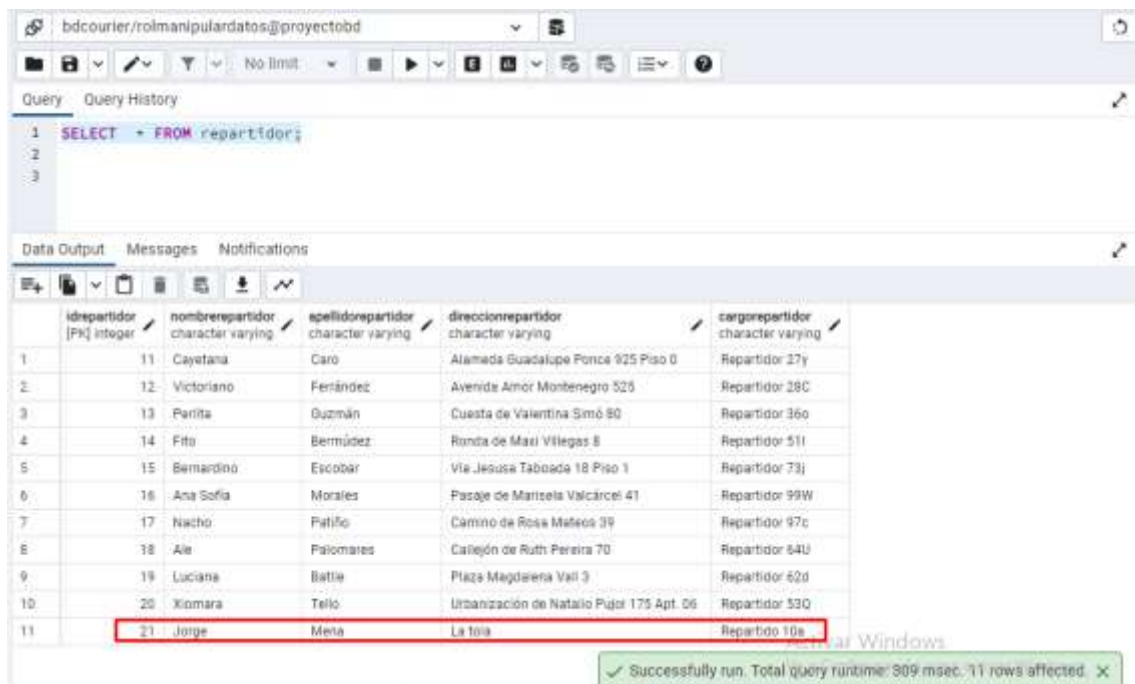
```
2 VALUES ('Jorge','Mena','La tola','Repartido 10a');
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 251 msec.

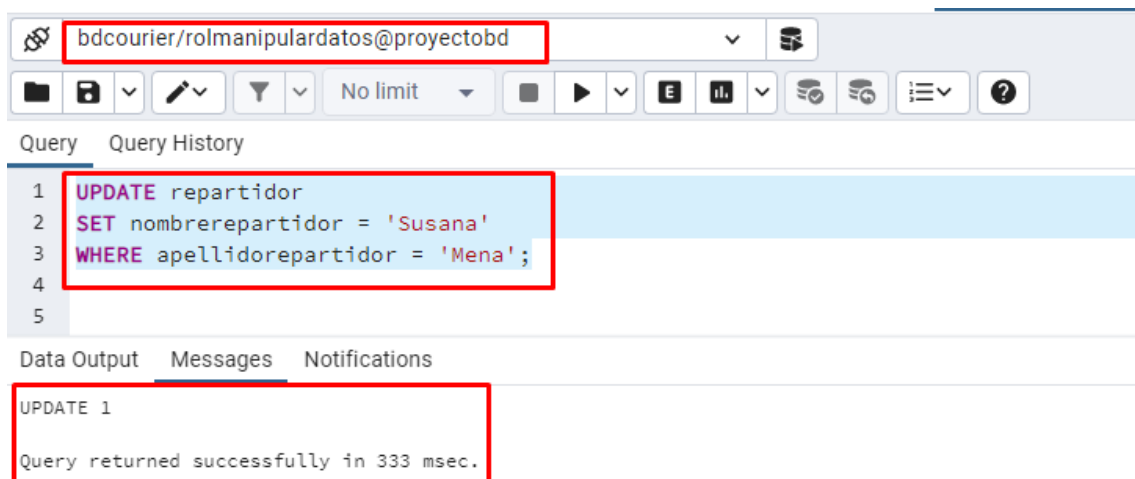
- Observamos que si se ingresó el registro correctamente en la tabla repartidor:
SELECT * FROM repartidor;



	idrepartidor [PK] integer	nombrerepartidor character varying	apellidorepartidor character varying	direccionrepartidor character varying	cargorepartidor character varying
1	11	Cayetana	Caro	Alameda Guadalupe Ponce 925 Piso 0	Repartidor 27y
2	12	Victoriano	Fernández	Avenida Amor Montenegro 525	Repartidor 28C
3	13	Parlita	Guzmán	Cuesta de Valentina Simó 80	Repartidor 36o
4	14	Fito	Bermúdez	Ronda de Maxi Villegas 8	Repartidor 51i
5	15	Bernardino	Escobar	Via Jesusa Taboada 18 Piso 1	Repartidor 73j
6	16	Ana Sofia	Morales	Paseo de Mariela Valcárcel 41	Repartidor 99W
7	17	Nacho	Patifo	Camino de Rosa Mateos 39	Repartidor 97c
8	18	Ale	Palomares	Callejón de Ruth Pereira 70	Repartidor 84U
9	19	Luciana	Battie	Plaza Magdalena Vall 3	Repartidor 62d
10	20	Xiomara	Tello	Urbanización de Natalio Pujol 175 Apt. 06	Repartidor 53Q
11	21	Jorge	Mena	La 10a	Repartido 10a

Successfully run. Total query runtime: 809 msec. 11 rows affected.

- Intentamos realizar un UPDATE en la tabla "repartidor" y si se nos permite:
UPDATE repartidor
SET nombrerepartidor = 'Susana'
WHERE apellidorepartidor = Mena;



```

1 UPDATE repartidor
2 SET nombrerepartidor = 'Susana'
3 WHERE apellidorepartidor = 'Mena';
4
5

```

UPDATE 1

Query returned successfully in 333 msec.

- Observamos que se actualizo el registro correctamente:

bdcourier/rolmanipulardatos@proyectobd

Query Query History

```
1 select * from repartidor;
```

Data Output Messages Notifications

	idrepartidor (FK) integer	nombrerepartidor character varying	apellidorepartidor character varying	direccionrepartidor character varying	cargorepartidor character varying
1	11	Cayetana	Caio	Alameda Guadalupe Ponce 925 Piso 0	Repartidor 27y
2	12	Victoriano	Ferrández	Avenida Amor Montenegro 525	Repartidor 28C
3	13	Perlita	Guzmán	Cuesta de Valentina Simó 80	Repartidor 36o
4	14	Fito	Bermúdez	Ronda de Maxi Villegas 8	Repartidor 51l
5	15	Bernardino	Escobar	Via Jesusa Taboada 18 Piso 1	Repartidor 73j
6	16	Ana Sofia	Morales	Pasaje de Mariela Valcarcel 41	Repartidor 99W
7	17	Nachó	Patillo	Caminó de Rosa Mateos 39	Repartidor 97c
8	18	Ale	Palomares	Callejón de Ruth Pereira 70	Repartidor 64U
9	19	Luciana	Battie	Plaza Magdalena Vial 3	Repartidor 62d
10	20	Xiomara	Tello	Urbanización de Natalio Pujol 175 Apt. 06	Repartidor 53Q
11	21	Susana	Mena	La tola	Repartido 10a

Successfully run. Total query runtime: 155 msec. 11 rows affected.

- Intentamos eliminar el registro actualizado recientemente:

DELETE FROM repartidor WHERE nombrerepartidor = 'Susana' ;

bdcourier/rolmanipulardatos@proyectobd

Query Query History

```
1 DELETE FROM repartidor WHERE nombrerepartidor = 'Susana';
```

Data Output Messages Notifications

DELETE 1

Query returned successfully in 237 msec.

- Observamos que se eliminó correctamente el registro:

select * from repartidor;

The screenshot shows a database query interface. The query entered is `select * from repartidor;`. The results are displayed in a table with 10 rows and 5 columns: `idrepartidor` (PK integer), `nombrerepartidor` (character varying), `apellidorepartidor` (character varying), `direccionrepartidor` (character varying), and `cargorepartidor` (character varying). A red box highlights the query and the table. A green status bar at the bottom indicates: "Successfully run, Total query runtime: 162 msec, 10 rows affected."

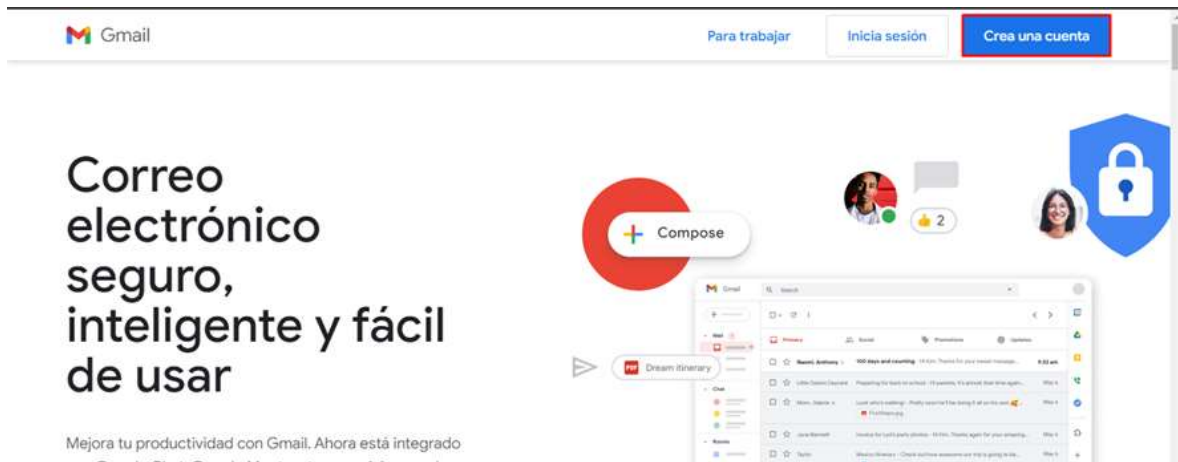
	idrepartidor (PK) integer	nombrerepartidor character varying	apellidorepartidor character varying	direccionrepartidor character varying	cargorepartidor character varying
1	11	Cayetana	Caro	Alameda Guadalupe Ponce 925 Piso 0	Repartidor 27y
2	12	Victoriano	Fernández	Avenida Amor Montenegro 525	Repartidor 28C
3	13	Perlita	Guzmán	Cuesta de Valentina Simó 80	Repartidor 36o
4	14	Fito	Bermúdez	Ronda de Maxi Villegas 8	Repartidor 51i
5	15	Bernardino	Escobar	Via Jesusa Taboada 18 Piso 1	Repartidor 73j
6	16	Ana Sofia	Morales	Pasaje de Mariela Valcárcel 41	Repartidor 99W
7	17	Nacho	Patillo	Camino de Rosa Mateos 39	Repartidor 97c
8	18	Ale	Palomares	Callejón de Ruth Pereira 70	Repartidor 64U
9	19	Luciana	Battie	Plaza Magdalena Vail 3	Repartidor 62d
10	20	Kiomara	Tello	Urbanización de Natalio Pujol 175 Apt. 06	Repartidor 53Q

11.Creación de Github

- Crear correo de Gmail para crear cuenta en Github
- Accedemos a Gmail

The screenshot shows a Google search for "correo gmail". The search results include links to Gmail, a Wikipedia article about Gmail, and a Google Support page. A red box highlights the first search result: "Gmail: Correo electrónico sin coste, privado y seguro". To the right, there is a large advertisement for Gmail with the Google logo and text describing the service.

- Dar clic en el botón **“Crear una cuenta”**



- Ingresar nombre del proyecto: **“proyectobasecourierg3”**

A screenshot of the Google account creation form. The title is 'Crea una Cuenta de Google' with the instruction 'Ingresa tu nombre'. There are two input fields: 'Nombre' and 'Apellido (opcional)'. The 'Nombre' field contains the text 'proyectobasecourierg3' and is highlighted with a red rectangle. Below the fields is a blue 'Siguiente' button, also highlighted with a red rectangle.

- Ingresar información para la cuenta de Gmail y dar clic en “Siguiente”:

The screenshot shows the 'Información básica' (Basic Information) step of a Google account creation process. At the top is the Google logo. Below it, the title 'Información básica' is followed by the instruction 'Completa tu fecha de nacimiento y género.' (Complete your birth date and gender). There are three input fields for birth date: 'Día' (Day) with '01', 'Mes' (Month) with 'Enero' (January), and 'Año' (Year) with '2000'. Below these is a 'Género' (Gender) dropdown menu with 'Prefiero no decirlo' (I prefer not to say) selected. A link 'Por qué solicitamos la fecha de nacimiento y el género' (Why we request your birth date and gender) is visible. At the bottom right is a blue button labeled 'Siguiente' (Next).

- Ingresamos el nombre del correo de Gmail para el proyecto **“projectobasecourierg3”** y dar clic en el botón **“Siguiente”**:

The screenshot shows the 'Elige tu dirección de Gmail' (Choose your Gmail address) step. At the top is the Google logo. Below it, the title 'Elige tu dirección de Gmail' is followed by the instruction 'Elige una dirección de Google o crea una nueva.' (Choose a Google address or create a new one). There are three radio button options: 'projectobasecourierg51@gmail.com', 'projectobasecourierg@gmail.com', and 'Crear tu propia dirección de Gmail' (Create your own Gmail address), which is selected. Below the options is a text input field for creating a new address, containing 'projectobasecourierg3' followed by a dropdown menu showing '@gmail.com'. A link 'Puedes usar letras, números y signos de puntuación' (You can use letters, numbers, and punctuation) is visible. At the bottom right is a blue button labeled 'Siguiente' (Next).

- Ingresar el password para el correo del proyecto: Password: **bdcourier**



Google

Crea una contraseña segura

Crea una contraseña segura con letras, números y símbolos combinados.

Contraseña

bdcourier

Confirmación

bdcourier

☒ Mostrar contraseña

Siguiente

- Ingresar un número telefónico para validar la cuenta de correo de Gmail creada para el proyecto:



Google

Demuestra que no eres un robot

Recibir un código de verificación en tu teléfono

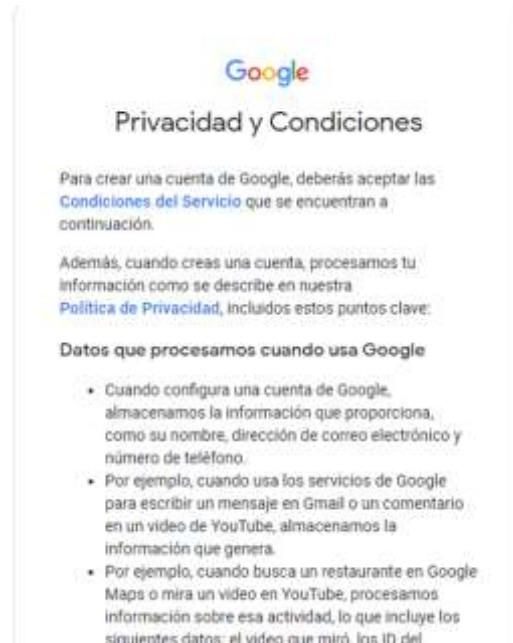
Número de teléfono

0996863282

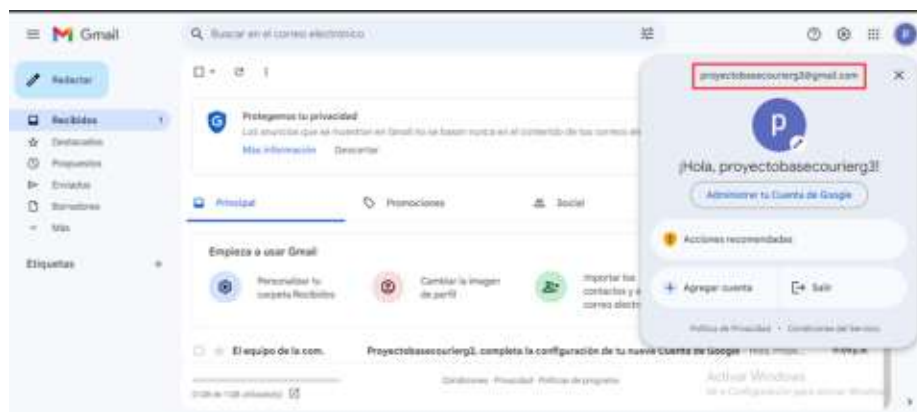
Google verificará este número mediante SMS (es posible que se apliquen cargos).

Siguiente

- Se aceptan los términos y condiciones de Google para la creación del correo electrónico en gmail:

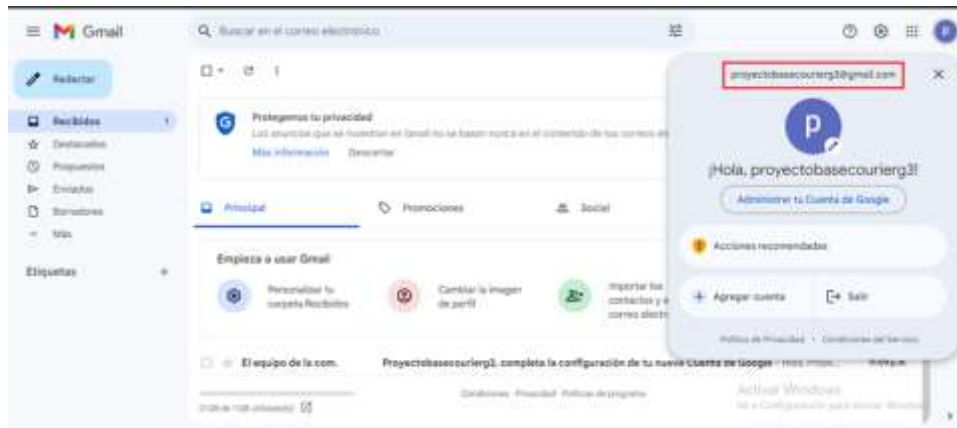


- Observamos que se ha creado correctamente el correo electrónico para el proyecto:

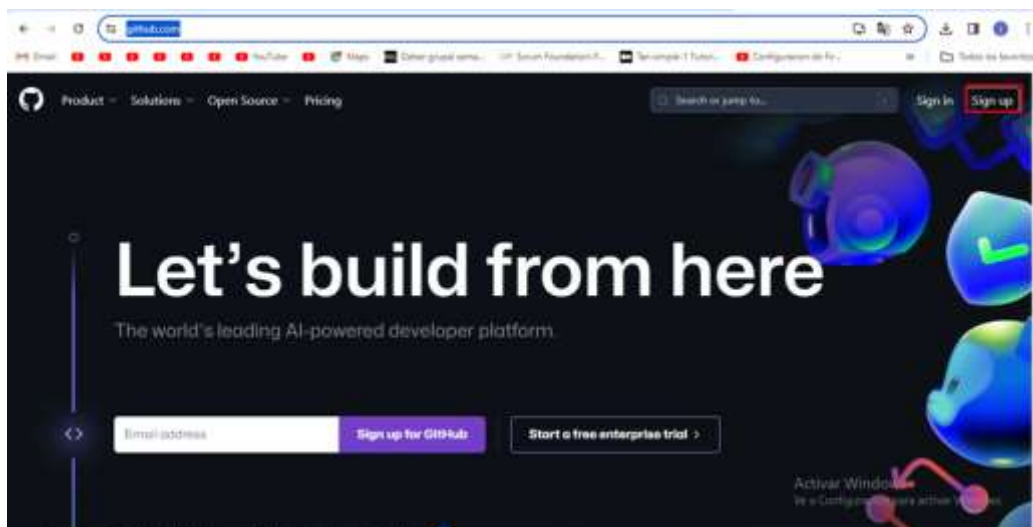


Vamos a crear la cuenta de Github para el proyecto.

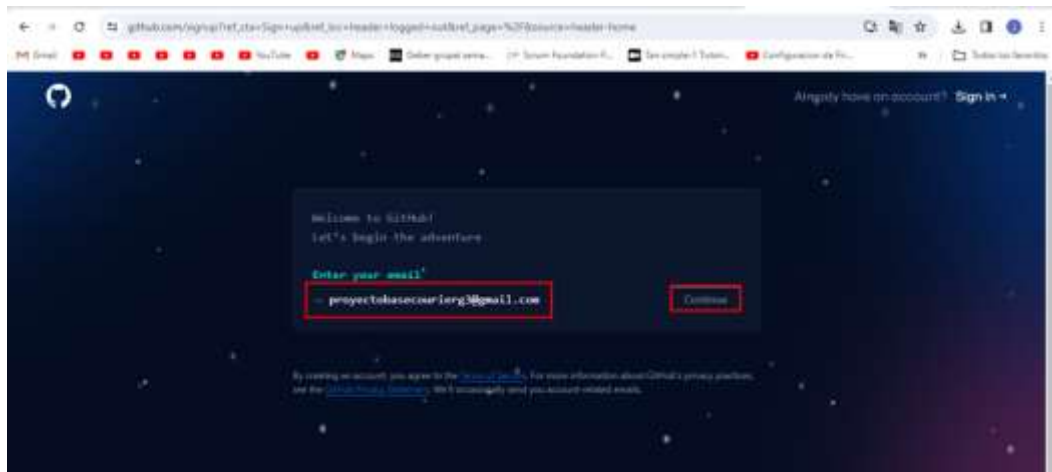
- Ingresamos a la página oficial de Github con el link: <https://github.com/>



- En la página web oficial de Github damos clic en el botón “Sign up” para crear la cuenta de Github para el proyecto:

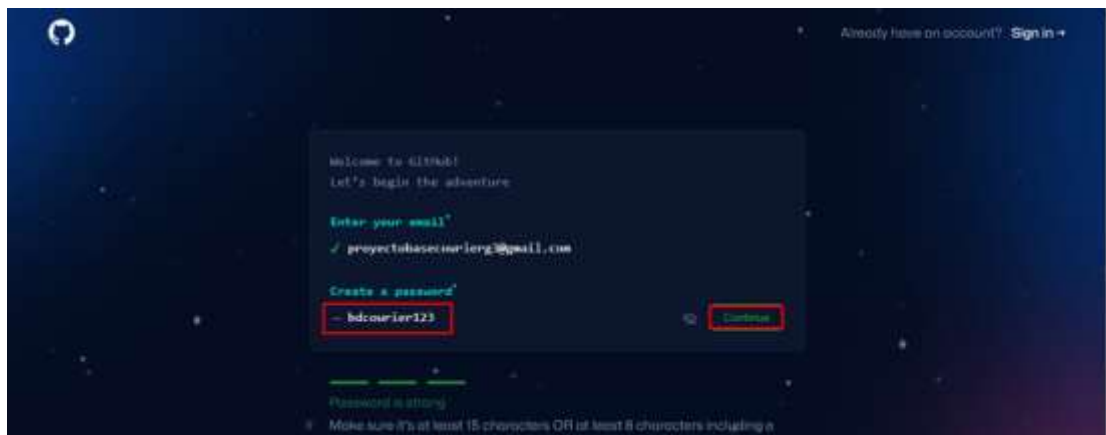


- Ingresar el correo de Gmail creado para el proyecto en el input de “Enter your *email” en la página web de Github y dar clic en el botón “Continue”:



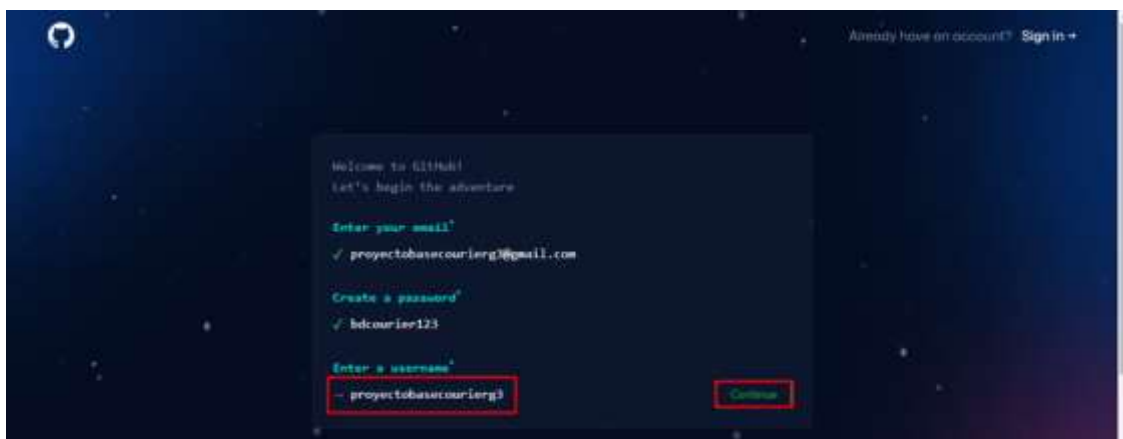
- Crear el password para la cuenta de Github del proyecto:

Password: bdcourier123

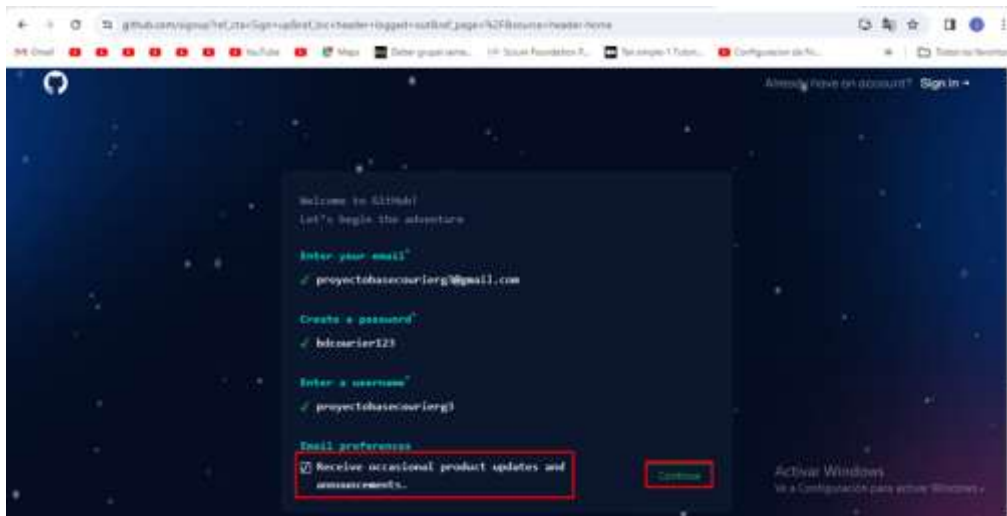


Ingresamos un username:

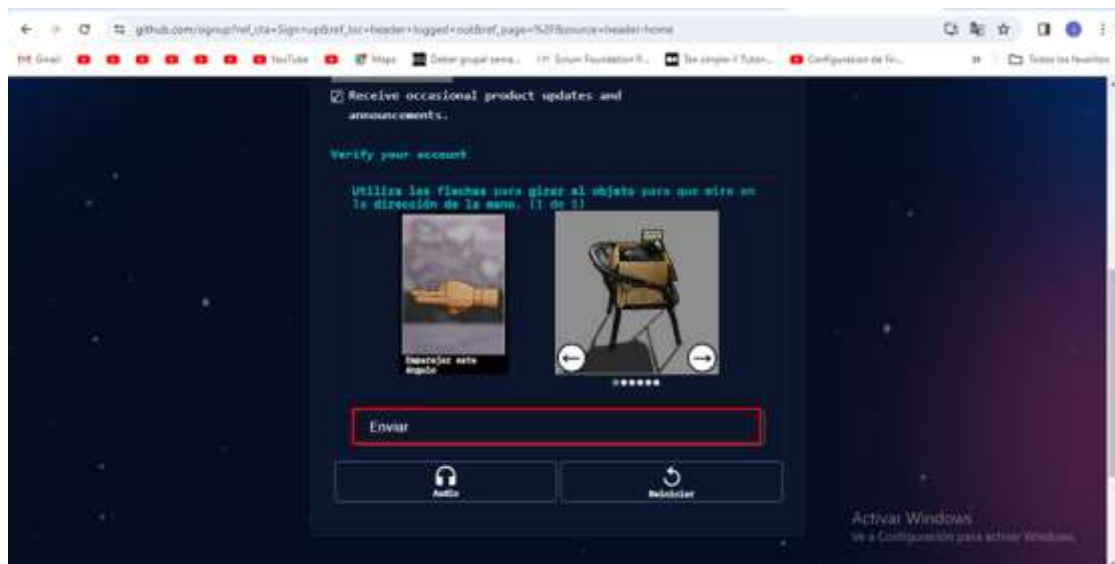
Username: projectobasecourier3



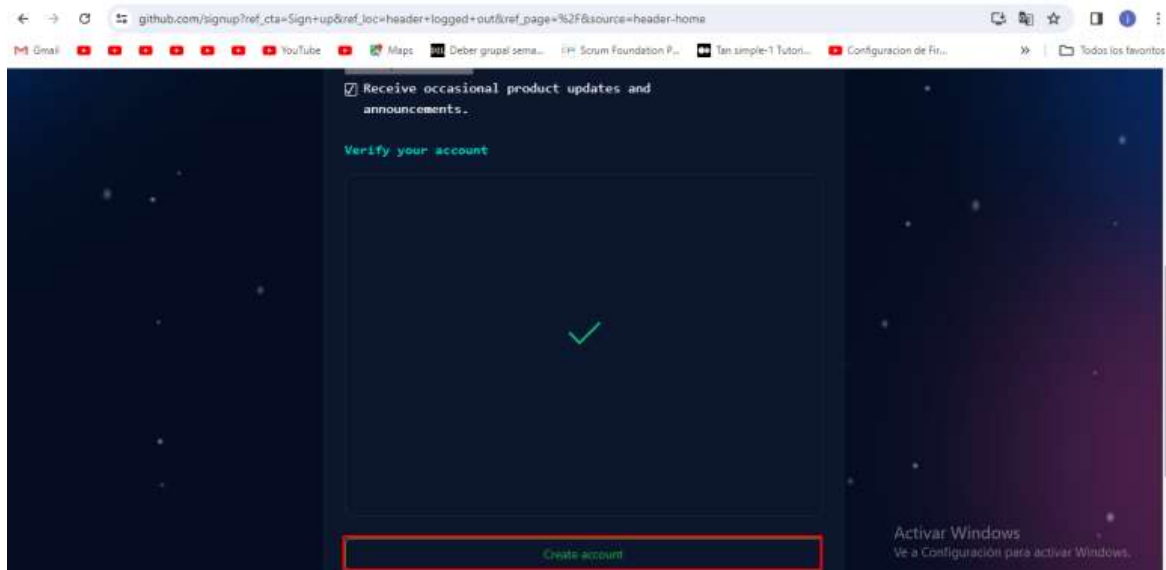
- Dar clic en check “Email preferences” y dar clic en botón “Continue”:



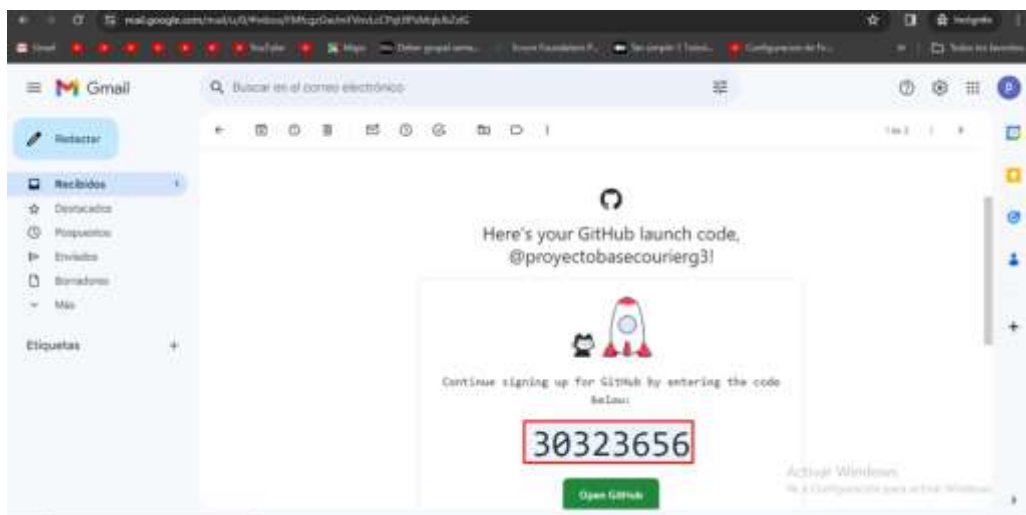
- Verificar la cuenta de Github:



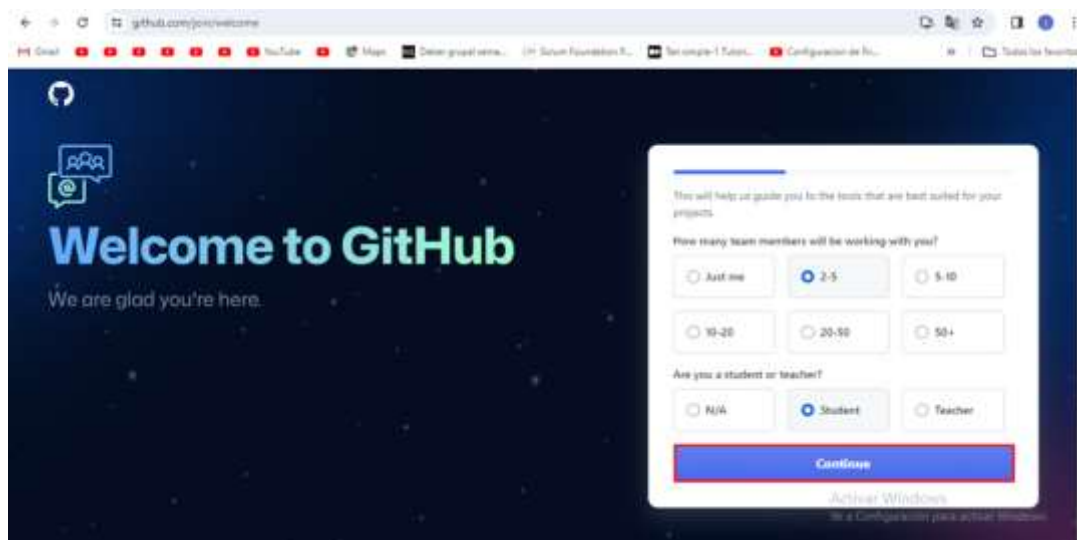
- Dar clic en el botón “Create account”:



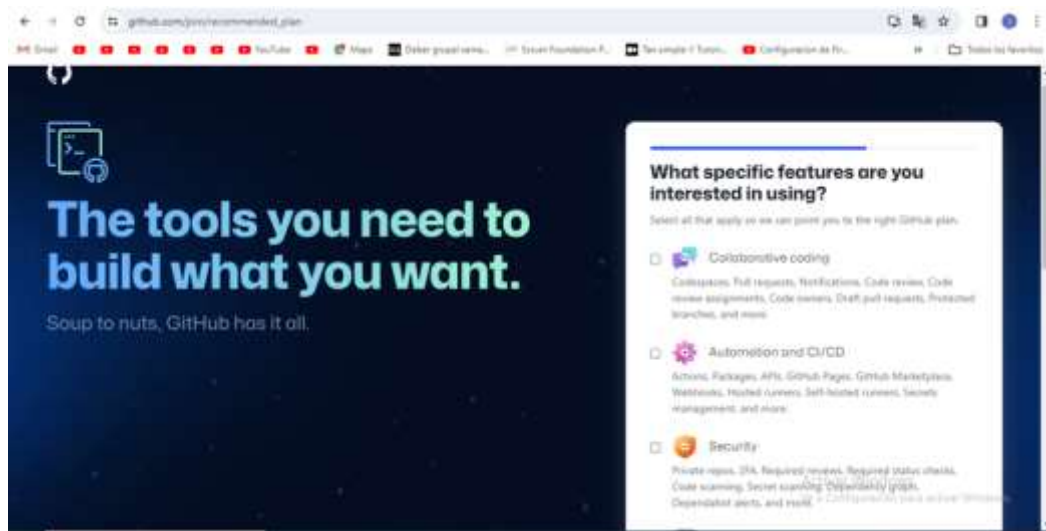
- Ingresar el código recibido en el correo: [“proyectobasecourierg3@gmail.com”](mailto:proyectobasecourierg3@gmail.com)



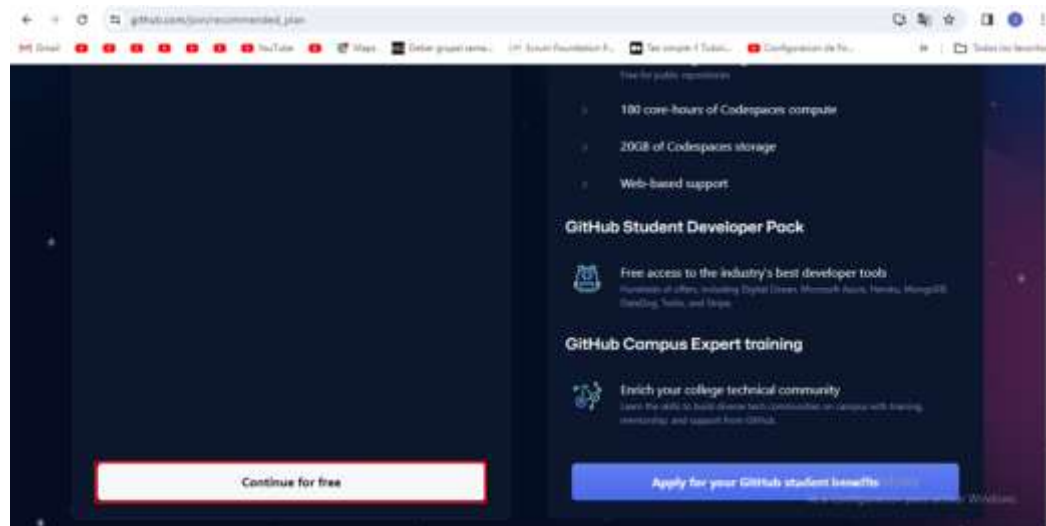
- Damos clic en el botón **“Continue”**:



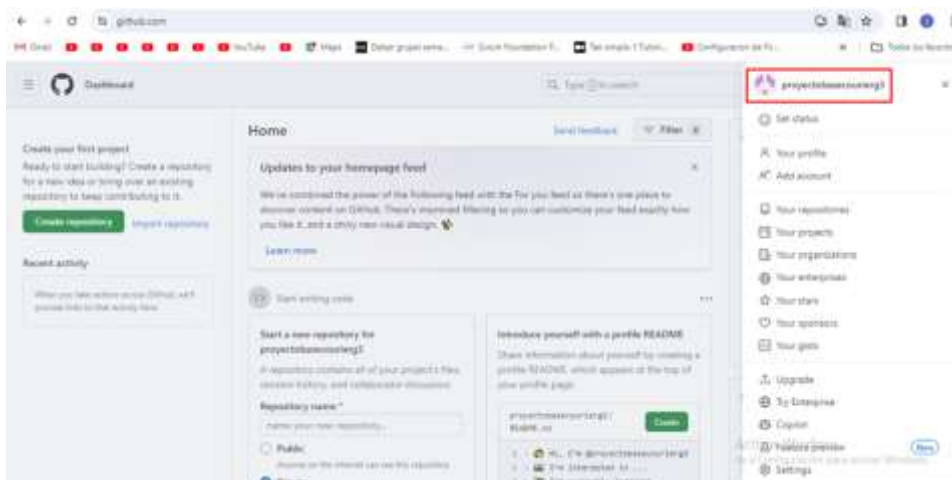
- Ingresamos la configuración básica de Github:



- Seleccionamos una cuenta free de Github para el proyecto:

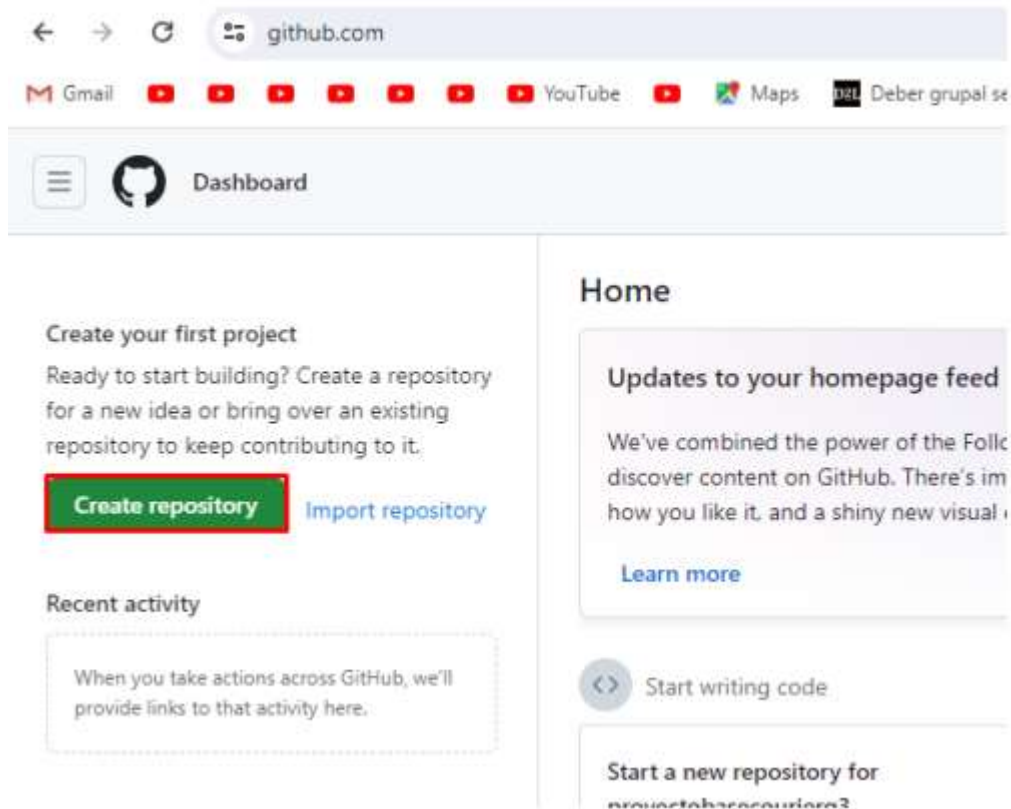


- Observamos que se creó la cuenta de Github para el proyecto exitosamente:

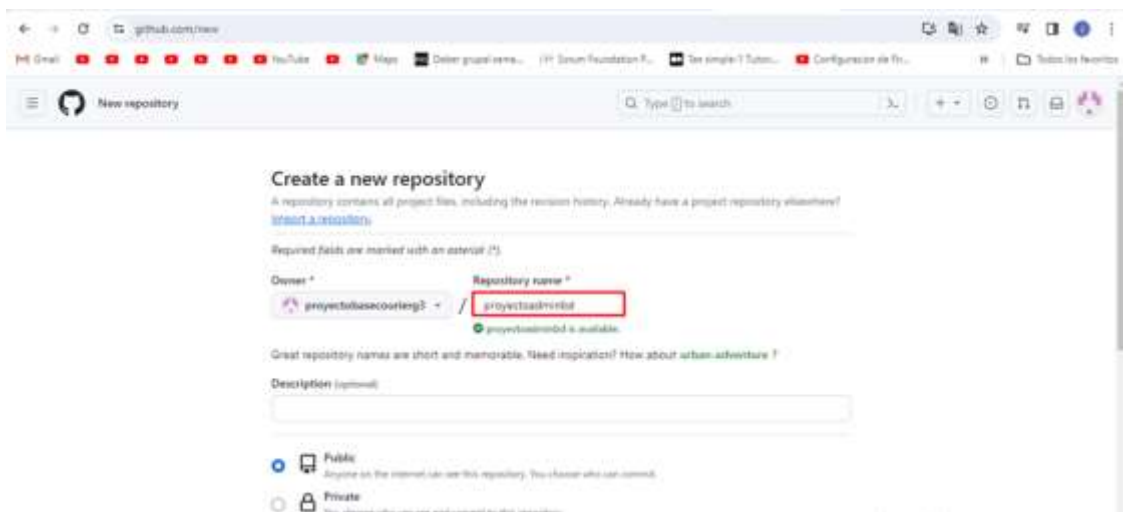


Crear un proyecto nuevo en Github

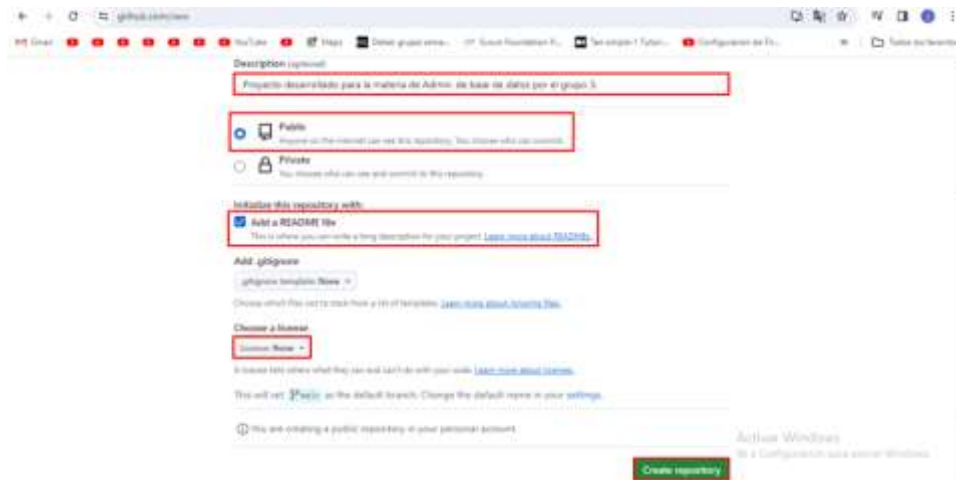
- Damos clic en el botón “Create repository” de la página web de github del proyecto



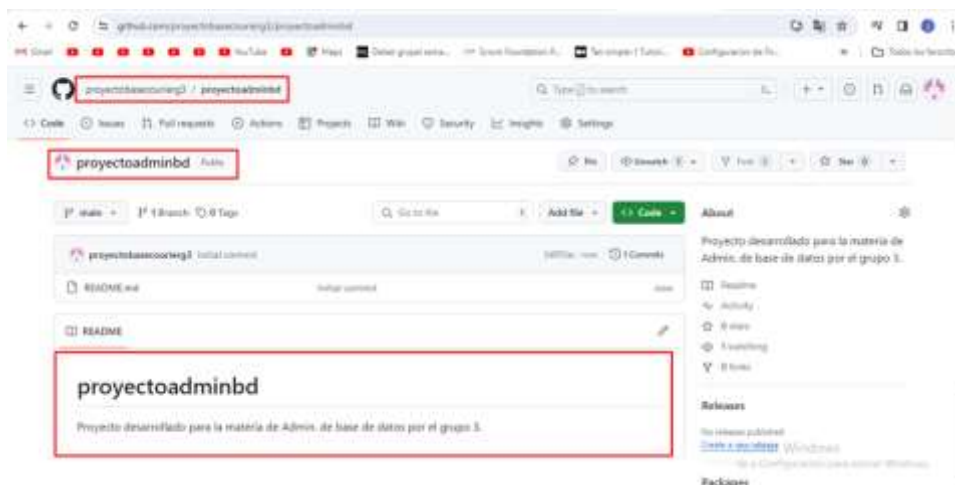
- Colocamos el nombre del repositorio: “**proyectoadminbd**”



- Ingresamos la siguiente configuración para la creación del repositorio nuevo:

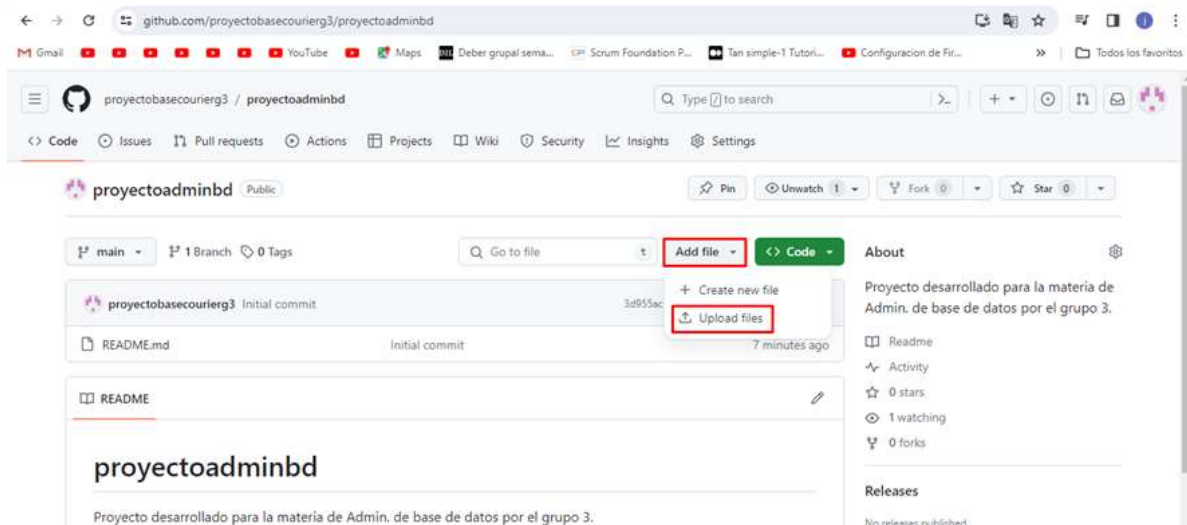


- Observamos que se a creado el repositorio “**proyectoadminbd**” correctamente:



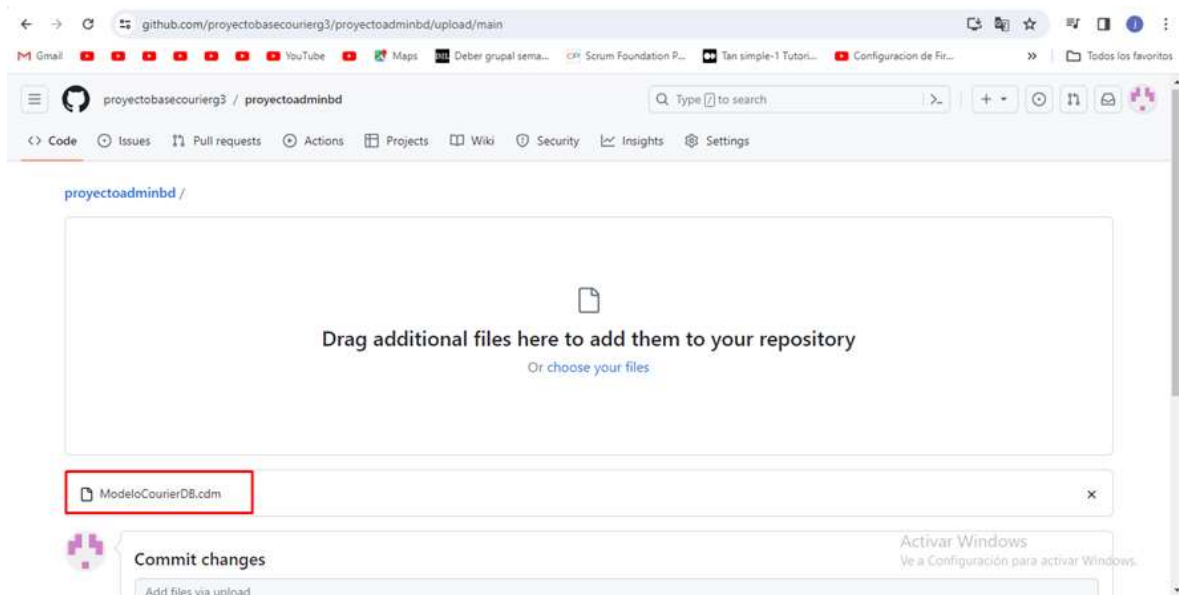
Subir los archivos del proyecto a repositorio de Github

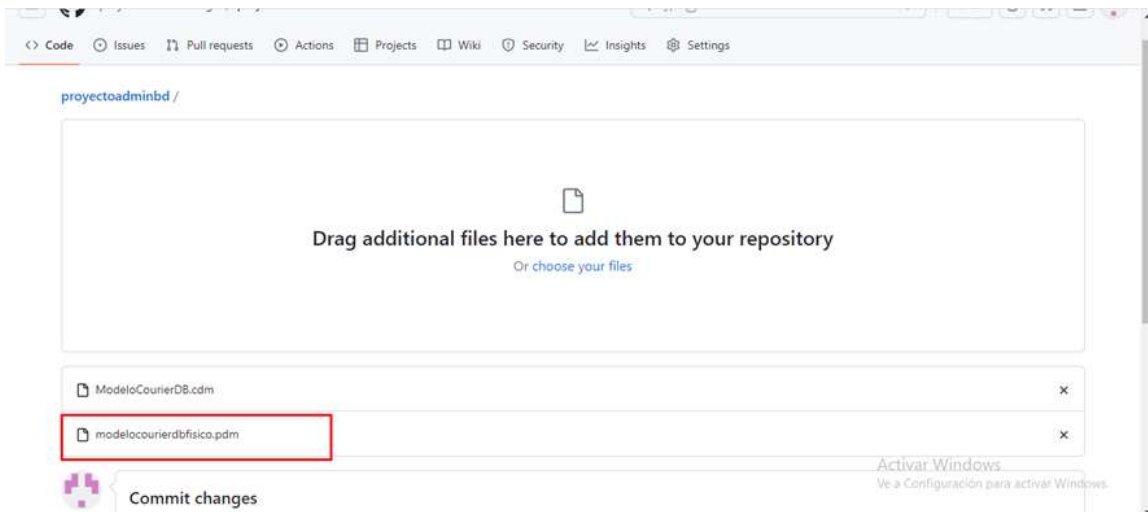
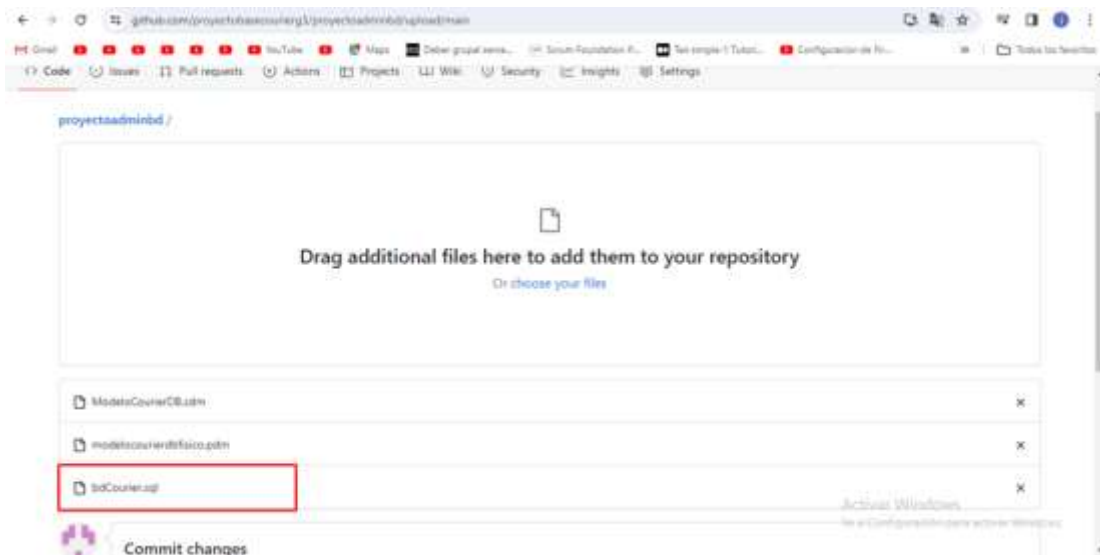
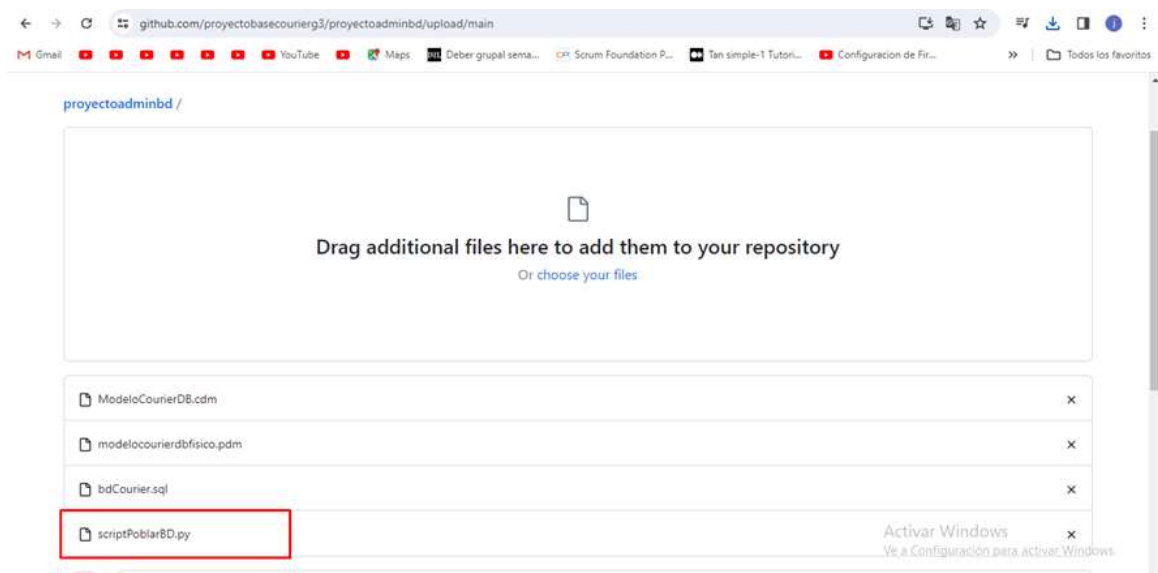
Damos clic en el botón “Add files” -> “Upload files”



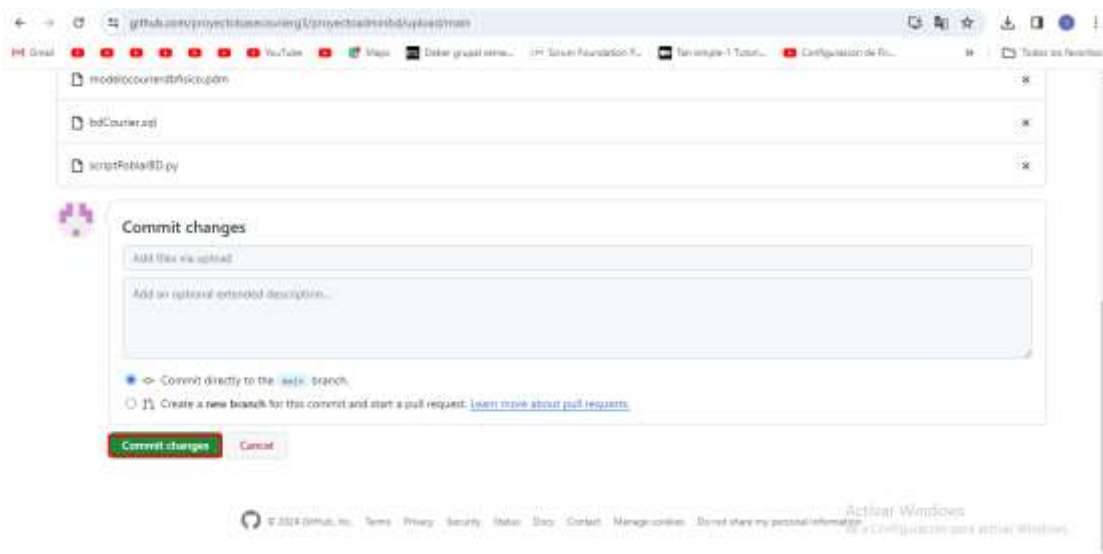
Subimos los archivos del proyecto al repositorio:

Modelo lógico de base de datos:

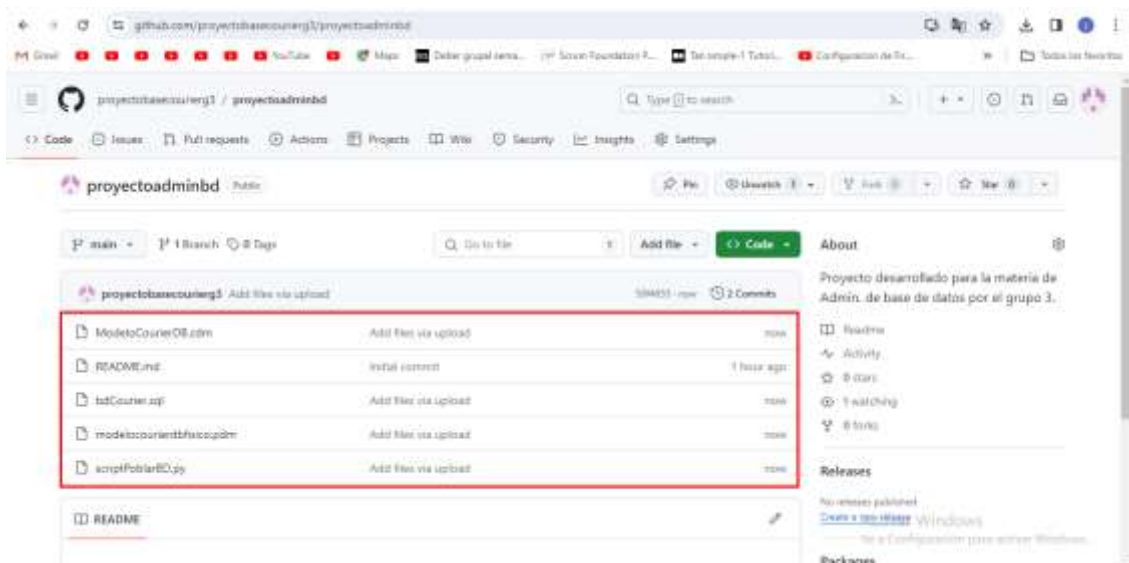


Modelo físico de base de datos:**Script de base de datos:****Documento .pdf de proyecto:****Script para poblar base de datos con Faker:**

Dar clic en el botón "Commit changes" para guardar los cambios



Se observan los archivos del proyecto subidos al repositorio en Github



2. Diccionario de Datos

El diccionario de datos que has creado describe las tablas y campos de una base de datos para un servicio de courier. La base de datos consta de las siguientes tablas:

Tabla: CLIENTE

Esta tabla almacena la información de los clientes de la empresa de Courier.

Campo	Descripción	Tipo de dato	Explicación	Nulo
IDCLIENTE	Identificador único de cada cliente	SERIAL	Autonumerado	No
CICLIENTE	Cédula de identidad del cliente	VARCHAR(200)	Opcional	Sí
NOMBRECLIENTE	Nombre del cliente	VARCHAR(200)	Obligatorio	No
APELLIDOCIENTE	Apellido del cliente	VARCHAR(200)	Obligatorio	No
DIRECCIONCLIENTE	Dirección del cliente	VARCHAR(200)	Obligatorio	No
TELEFONOCIENTE	Número de teléfono del cliente	VARCHAR(10)	Opcional	Sí

Tabla: DESTINATARIO

Esta tabla almacena la información de los destinatarios de las encomiendas.

Campo	Descripción	Tipo de dato	Explicación	Nulo
IDDESTINATARIO	Identificador único de cada destinatario	SERIAL	Autonumerado	No
CIDESTINATARIO	Cédula de identidad del destinatario	VARCHAR(200)	Opcional	Sí
NOMBREDESTINATARIO	Nombre del destinatario	VARCHAR(200)	Obligatorio	No
APELLIDODESTINATARIO	Apellido del destinatario	VARCHAR(200)	Obligatorio	No
DIRECCIONDESTINATARIO	Dirección del destinatario	VARCHAR(200)	Obligatorio	No
TELEFONODESTINATARIO	Número de teléfono del destinatario	VARCHAR(10)	Opcional	Sí

Tabla: ENCOMIENDA

Esta tabla almacena la información de las encomiendas que maneja la empresa.

Campo	Descripción	Tipo de dato	Explicación	Nulo
IDENCOMIENDA	Identificador único de cada encomienda	SERIAL	Autonumerado	No
IDESTADO	Identificador del estado actual de la encomienda	INT4	Relacionado con la tabla ESTADOENCOMIENDA	No
TIPOENCOMIENDA	Tipo de encomienda	VARCHAR(200)	Obligatorio	No
ALTURA	Altura de la encomienda en centímetros	FLOAT8	Obligatorio	No

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

PESO	Peso de la encomienda en kilogramos	FLOAT8	Obligatorio	No
DESCRIPCIONENCOMIENDA	Descripción adicional de la encomienda	VARCHAR(200)	Opcional	Sí
CATEGORIA	Categoría de la encomienda	VARCHAR(200)	Obligatorio	No

Tabla: ENVIO

Esta tabla almacena la información del envío de las encomiendas.

Campo	Descripción	Tipo de dato	Explicación	Nulo
IDENVIO	Identificador único de cada envío	SERIAL	Autonumerado	No
IDENTICKET	Identificador del ticket asociado al envío	INT4	Relacionado con la tabla TICKET	No
IDDESTINATARIO	Identificador del destinatario	INT4	Relacionado con la tabla DESTINATARIO	No
IDRUTA	Identificador de la ruta asignada al envío	INT4	Relacionado con la tabla RUTA	Sí
IDCLIENTE	Identificador del cliente que solicita el envío	INT4	Relacionado con la tabla CLIENTE	No
IDTIPOENVIO	Identificador del tipo de envío	INT4	Relacionado con la tabla TIPOENVIO	No
IDENCOMIENDA	Identificador de la encomienda enviada	INT4	Relacionado con la tabla ENCOMIENDA	No
IDREPARTIDOR	Identificador del repartidor asignado al envío	INT4	Relacionado con la tabla REPARTIDOR	No
DURACIONENVIO	Duración estimada del envío	TIME	Opcional	Sí

Tabla: ESTADO ENCOMIENDA

Esta tabla almacena en estado en el que se encuentra el despacho de la encomienda.

Campo	Descripción	Tipo de dato	Explicación	Nulo
IDESTADO	Identificador único de cada estado	SERIAL	Autonumerado	No
ACTIVOENCOMIENDA	Descripción del estado actual de la encomienda	VARCHAR(200)	Obligatorio	No
FECHAESTADO	Fecha y hora en que se registró el estado actual	DATE	Obligatorio	No

Tabla: REPARTIDOR

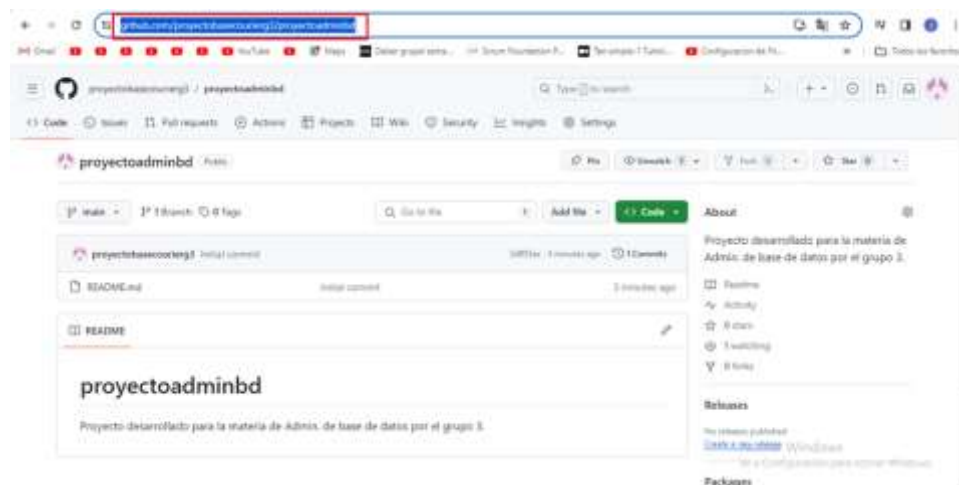
Esta tabla almacena la información de los repartidores de la empresa.

Campo	Descripción	Tipo de dato	Explicación	Nulo
IDREPARTIDOR	Identificador único de cada repartidor	SERIAL	Autonumerado	No
NOMBRE REPARTIDOR	Nombre del repartido			

3. Instrucciones para acceder al repositorio en GitHub

Para acceder al repositorio de Github ingresamos la url del proyecto en el navegador:

Repositorio: <https://github.com/proyectobasecourierg3/proyectoadminbd>



4. Informe de cumplimiento de los siguientes criterios de desempeño:

1.- Criterio

Definición de los Objetivos de Trabajo

Porcentaje de Cumplimiento

100%

Evidencia(s)

- Realizar reuniones para abordar los temas de implementación la ejecución del proyecto
- Cumplir con los horarios establecidos previamente para realizar el proyecto
- Implementar una solución con una base de datos contenerizada

2.- Criterio

Definición de Cronograma

Porcentaje de Cumplimiento

100%

Evidencia(s)

21/12/2023	Modelado de la base de datos
12/1/2024	Implementación de la base de datos contenerizada
12/1/2024	Cambio de puerto
12/1/2024	Conexión entre pgAdmin con la base contenerizada de Postgres
13/1/2024	Creación de la base desde pg admin con el script que ya anteriormente se modelo
13/1/2024	Desarrollo de script para poblar la base de datos utilizando la librería Faker
13/1/2024	Creación de roles y usuarios
15/1/2024	Desarrollo del documento formal
15/1/2024	Crear cuenta de gib hub
15/1/2024	Subir de archivos al repositorio

3.- Criterio

Definición de Roles

Porcentaje de Cumplimiento

100%

Evidencia(s)

Modelador de Base de Datos:

- Es el encargado de diseñar y crear modelos de base de datos que representen la estructura y relación de los datos en un sistema.

Administrador de Base de Datos:

- Es el encargado de la gestión y administración de bases de datos, asegurando su rendimiento, seguridad y disponibilidad.

Desarrollador:

- Es el encargado de escribir el código para construir aplicaciones o sistemas según los requisitos del proyecto.

4.- Criterio

Asignación de roles

Porcentaje de Cumplimiento

100%

Evidencia(s)

William Morales - Modelador de Base de Datos

Belén Gavilanes – Administrador de Base de datos

Jonathan Lema - Desarrollador

5.- Criterio

Asignación de Responsabilidades

Porcentaje de Cumplimiento

100%

Evidencia(s)

William Morales - Modelador de Base de Datos:

- Diseñar y crear el modelo de base de datos que reflejen los requisitos del sistema.

Belén Gavilanes – Administrador de Base de Datos:

- Gestionar y administrar del sistema de datos.

Jonathan Lema - Desarrollador:

- Escribir código de alta calidad que cumpla con los requisitos del proyecto.

6.- Criterio

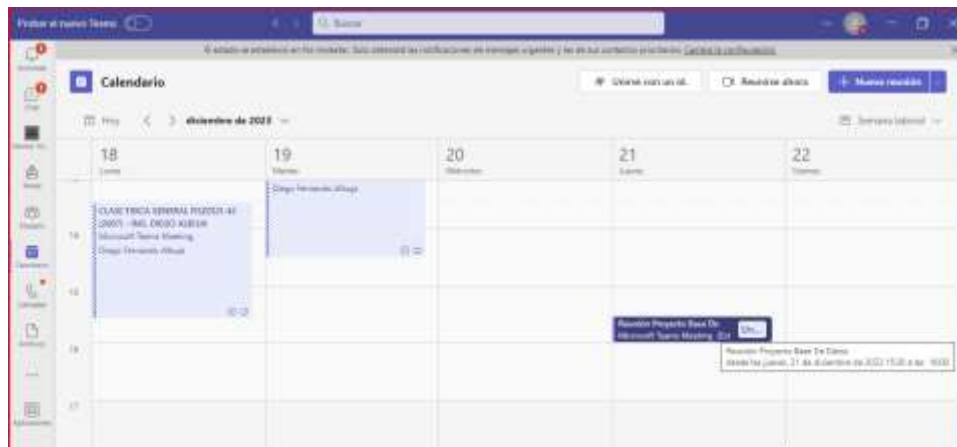
Cronograma de reuniones de trabajo

Porcentaje de Cumplimiento

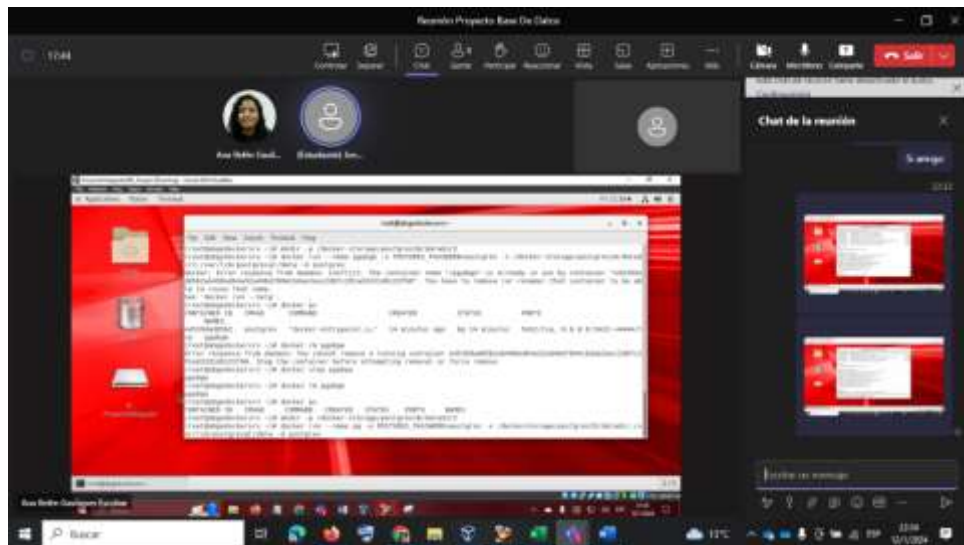
100%

Evidencia(s)

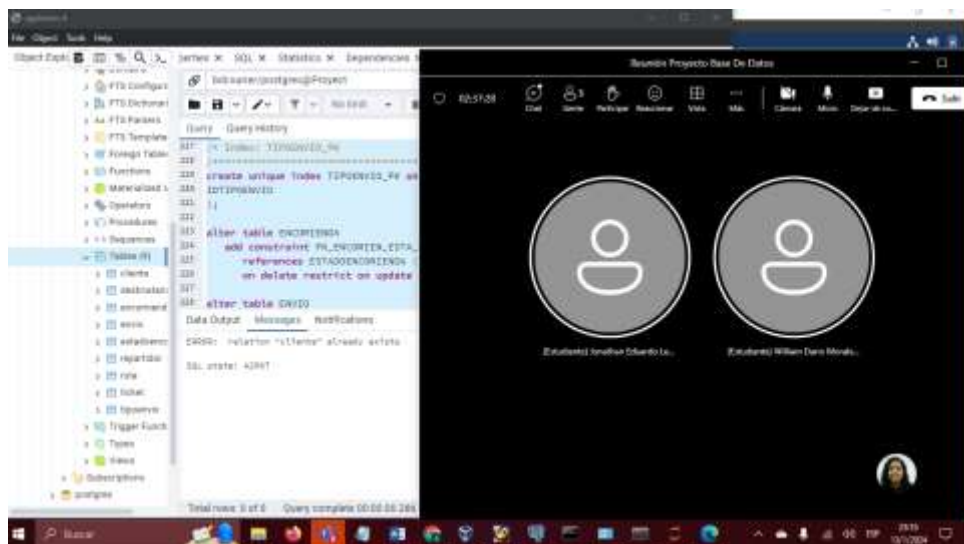
Primera reunión el 21 de diciembre



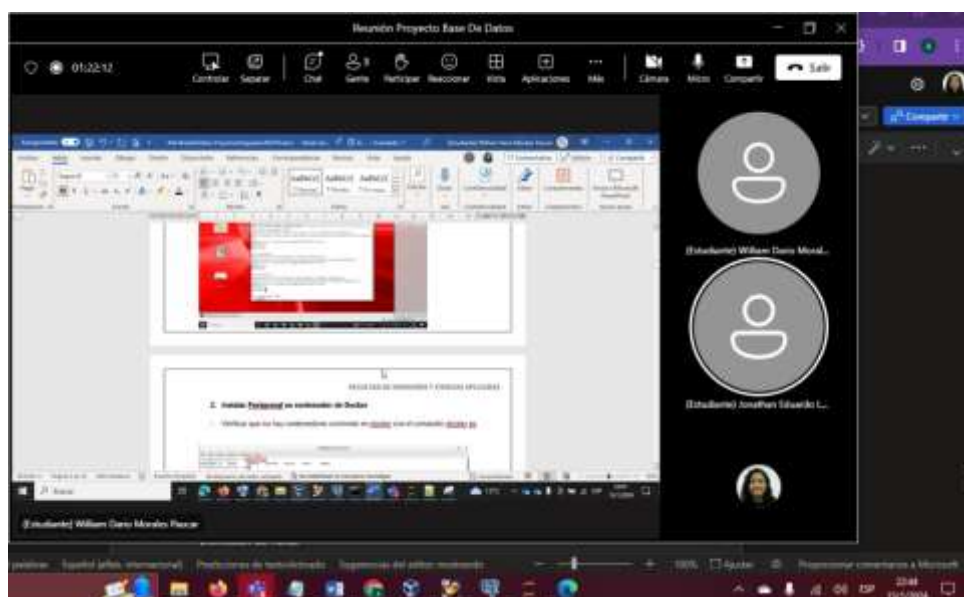
Segunda reunión el 12 de enero



Tercera reunión el 13 de enero



Cuarta reunión el 15 de enero



7.- Criterio

Ideas aportadas para la implementación de la solución por cada participante

Porcentaje de Cumplimiento

100%

Evidencia(s)

8.- Criterio

Aporte individual para la consecución de los Objetivos de Trabajo

Porcentaje de Cumplimiento

Evidencia(s)

Nos aseguramos de comprender claramente los objetivos del proyecto. Si hay alguna ambigüedad, buscamos aclaraciones para garantizar un entendimiento completo mediante búsquedas en internet o comunicándonos con el docente.

Cada miembro tuvo un compromiso personal con los objetivos del equipo. Demuéstranos interés y entusiasmo por lograr el éxito de la correcta implementación y ejecución del proyecto.

9.- Criterio

Aporte individual para la resolución de posibles conflictos

Porcentaje de Cumplimiento

Evidencia(s)

Para la realización de este proyecto tuvimos una comunicación abierta y honesta, establecimos un canal donde todos los miembros del equipo nos sentimos cómodos expresando nuestras opiniones y preocupaciones sin temor a represalias, estando conscientes de que en algún tema uno tiene más conocimiento que otro, pero cada miembro fue paciente y ante cualquier problema nos reuníamos y buscábamos la solución.

Cada miembro del grupo aportó una idea para poder dar solución al presente proyecto, como, por ejemplo: Cuando tuvimos una dificultad que a uno de nuestros compañeros no se podía conectar la base de datos contenerizada al pg admin debido a un error de conexión en virtual box, ante este problema nos reunimos y le dimos solución en conjunto

