

Explicación práctica - clase 3 - autorización

Configuración de los logs de la BD (opcional, solo es para obtener más información).

Se puede configurar los mensajes (loggers) para el momento del desarrollo.

```
#admin/src/web/__init__.py
import logging

logging.basicConfig()
logging.getLogger("sqlalchemy.engine").setLevel(logging.INFO)
```

```
INFO:sqlalchemy.engine.Engine:select pg_catalog.version()
INFO:sqlalchemy.engine.Engine:[raw sql] {}
INFO:sqlalchemy.engine.Engine:select current_schema()
INFO:sqlalchemy.engine.Engine:[raw sql] {}
INFO:sqlalchemy.engine.Engine:show standard_conforming_strings
INFO:sqlalchemy.engine.Engine:[raw sql] {}
INFO:sqlalchemy.engine.Engine:BEGIN (implicit)
INFO:sqlalchemy.engine.Engine:SELECT issues.id AS issues_id, issues.email AS issues_email, issues.title AS issues_title, issue
s.description AS issues_description, issues.status AS issues_status, issues.user_id AS issues_user_id, issues.inserted_at AS i
ssues_inserted_at, issues.updated_at AS issues_updated_at
FROM issues
INFO:sqlalchemy.engine.Engine:[generated in 0.00014s] {}
INFO:sqlalchemy.engine.Engine:SELECT users.id AS users_id, users.email AS users_email, users.password AS users_password, users
.inserted_at AS users_inserted_at, users.updated_at AS users_updated_at
FROM users
WHERE users.id = %(pk_1)s
INFO:sqlalchemy.engine.Engine:[generated in 0.00011s] {'pk_1': 1}
INFO:sqlalchemy.engine.Engine:SELECT users.id AS users_id, users.email AS users_email, users.password AS users_password, users
.inserted_at AS users_inserted_at, users.updated_at AS users_updated_at
FROM users
WHERE users.id = %(pk_1)s
INFO:sqlalchemy.engine.Engine:[cached since 0.002145s ago] {'pk_1': 2}
INFO:sqlalchemy.engine.Engine:SELECT users.id AS users_id, users.email AS users_email, users.password AS users_password, users
.inserted_at AS users_inserted_at, users.updated_at AS users_updated_at
FROM users
WHERE users.id = %(pk_1)s
INFO:sqlalchemy.engine.Engine:[cached since 0.003272s ago] {'pk_1': 3}
INFO:sqlalchemy.engine.Engine:ROLLBACK
```

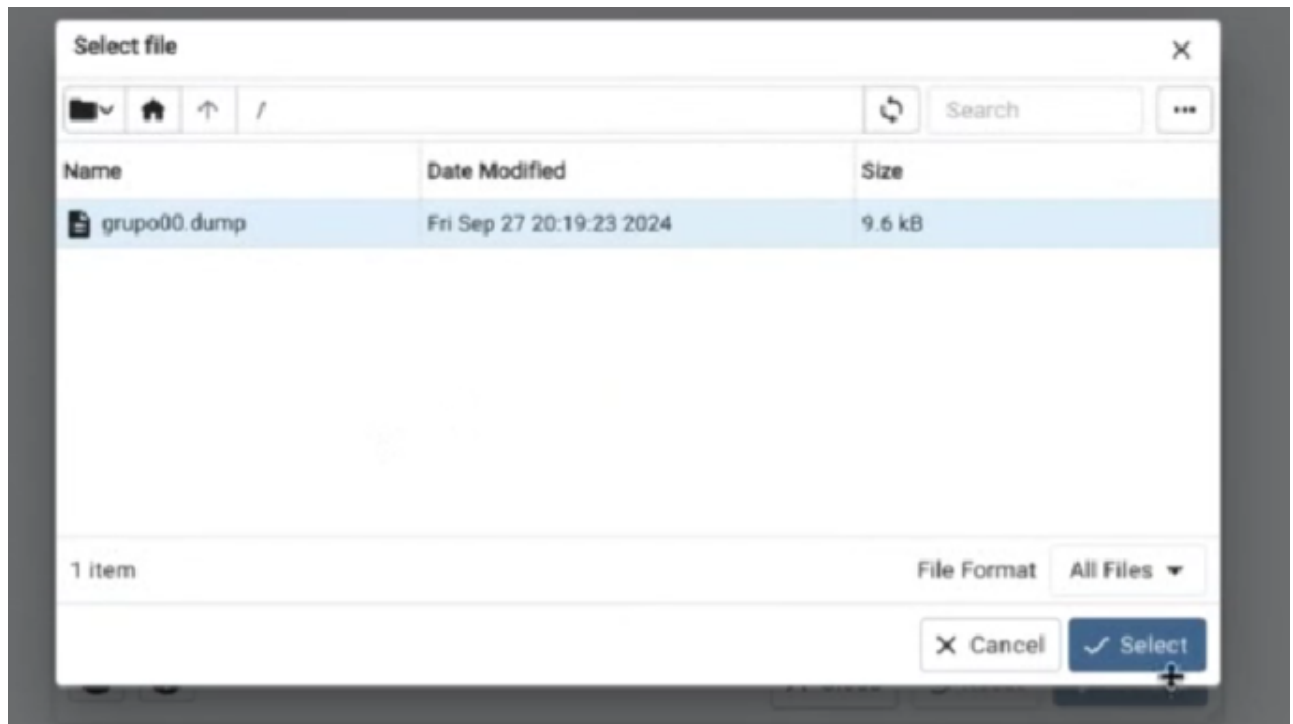
Ejemplo de consulta a la base de datos con el logger

Cargar en pgadmin la base de datos local

Es necesario averiguar cómo crear **.dumps** para levantar la base de datos en el servidor.

Pasos:

- Acceder a pgadmin>servers>databases>grupoXX y click derecho sobre el elemento. Click a **Backup** (para generar un dump), Click a **Restore** (para subir un dump -esto es lo que tenemos que hacer para la base de datos de producción-).
- Aparecerá una nueva ventana. Click a subir un archivo, seleccionar **Upload** en las opciones que se muestran.
- Seleccionar file. Subirlo. Una vez hecho cerrar la ventanita. Veremos en pantalla esto y presionamos **Select**:



- En la nueva ventana presionamos la opción **Restore** que confirma la subida del archivo .dump

Consejos: ya subir cosas a producción xd

Configuración variables de entorno en producción (Vault) - (min 19:00)

```
#src.core.config.py

class ProductionConfig(Config):
    SQLALCHEMY_DATABASE_URI = environ.get("DATABASE_URL")
```

Chequeo de permisos