UNIVERSIDAD
NACIONAL
DE COLOMBIA

Ciclo III

Desarrollo de Software

Capa Lógica: Conexión con la BD

18

Jeisson Andrés Vergara Vargas
Departamento de Ingeniería de Sistemas e Industrial
http://colswe.unal.edu.co/~javergarav/
javergarav@unal.edu.co

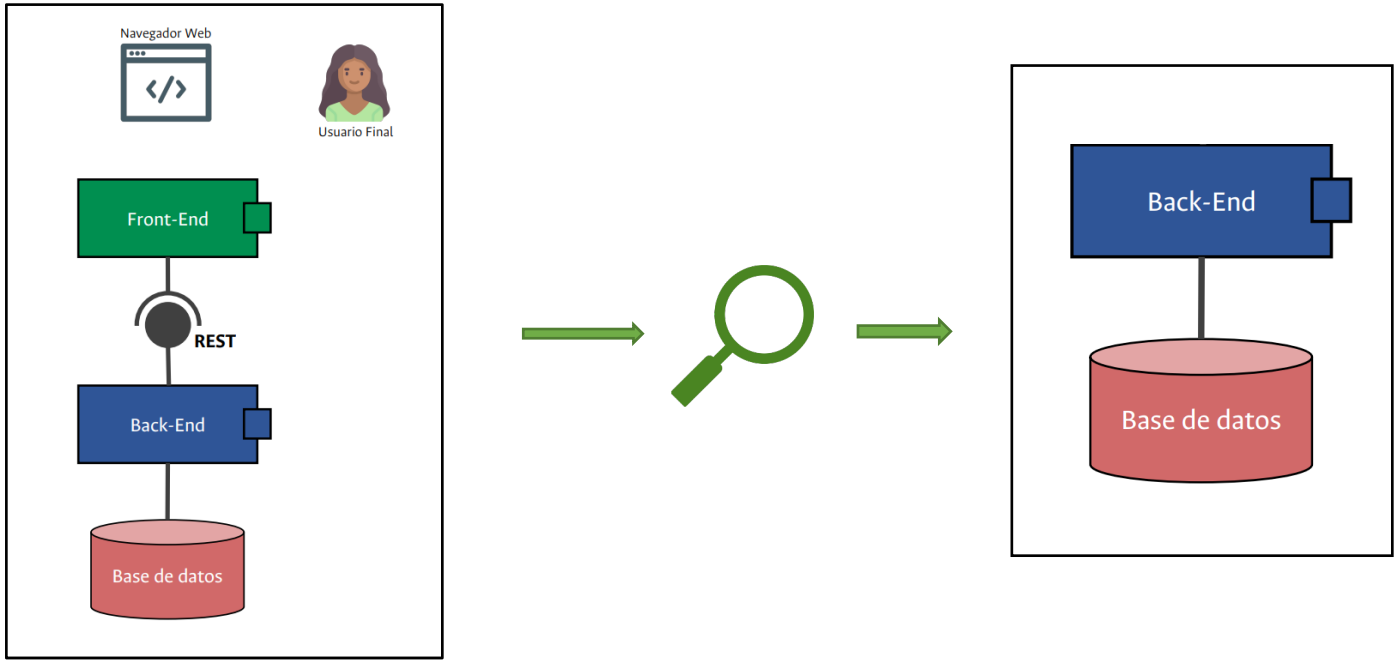
Objetivo de Aprendizaje

Identificar e **Implementar** la conexión entre la capa lógica y la capa de datos.

Conceptos Básicos

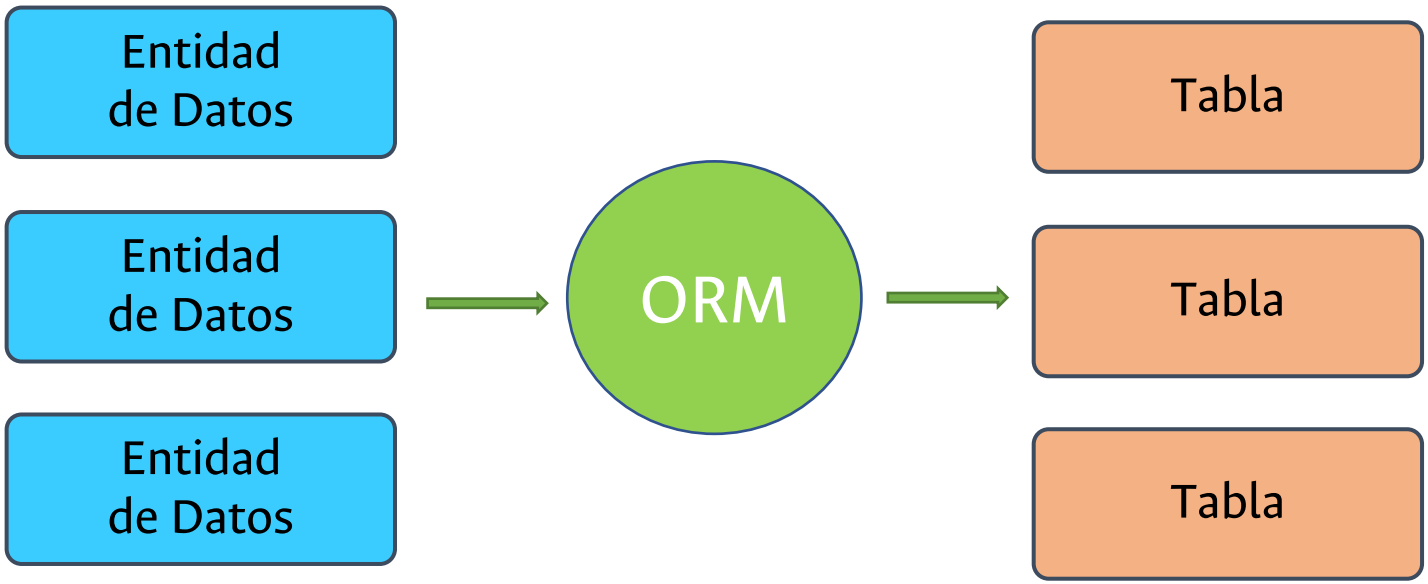
Conexión con la Capa de Datos

Siguiendo la Arquitectura de 3 capas, es necesario implementar una conexión entre el Back-End y la Base de Datos, para asegurar la persistencia de la información.



ORM

Un ORM (Object-Relational Mapping), se encarga de crear y manipular un esquema en una Base de Datos, con el fin de darle persistencia a entidades de datos definidas en el Back-End.

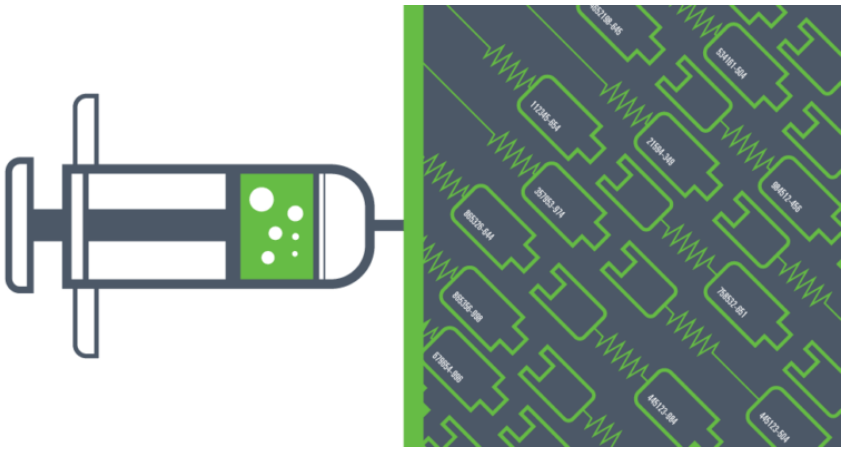


SQLAlchemy

El ORM que se usará es SQLAlchemy, este permitirá crear las entidades de datos y a partir de estas crear el respectivo esquema en la base de datos.

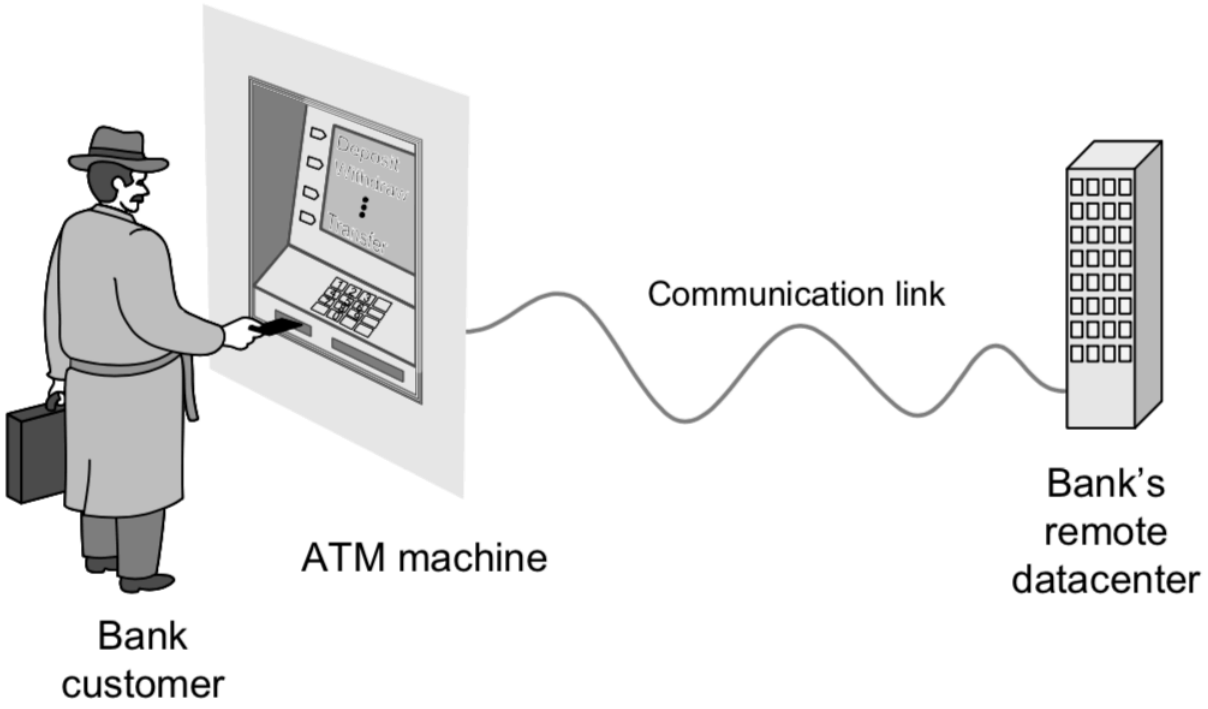
Inyección de Dependencias

La inyección de dependencias es un mecanismo que delega a terceros la creación de requisitos necesarios para la ejecución de una funcionalidad, evitando que la funcionalidad realice este trabajo.



Ejemplo

Software para un «ATM»



Ejemplo

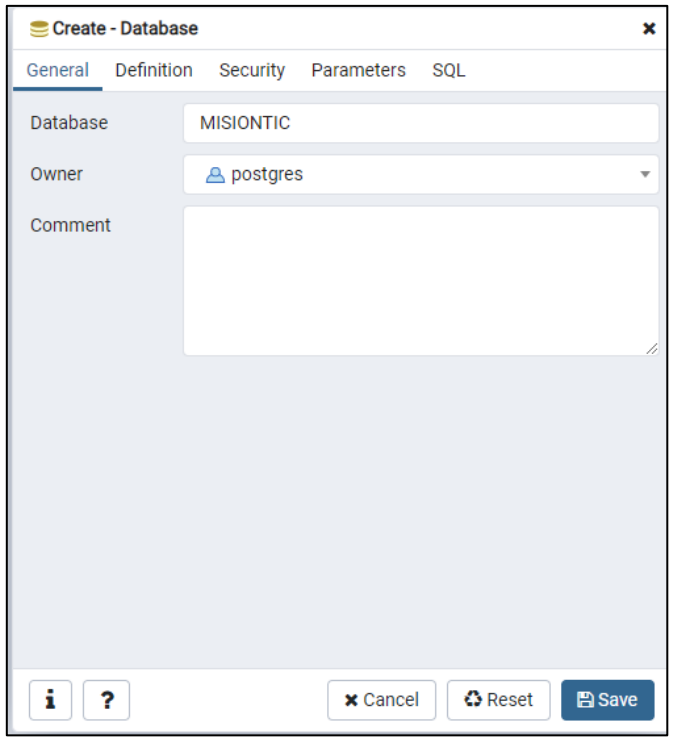
En esta sesión se realizarán las siguientes actividades:

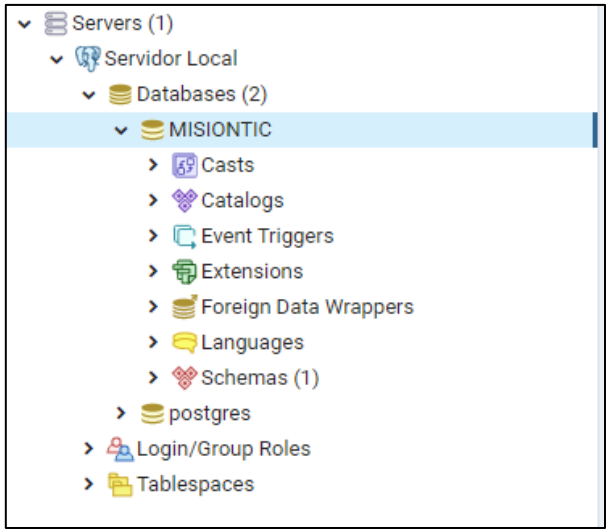
1. Crear una Base de Datos y un Esquema vacío en PostgreSQL.
2. Crear las entidades de datos correspondientes con SQLAlchemy.
3. Crear el mecanismo de conexión con la base de datos.

Creación de Base de Datos y Esquema

Creación de Base de Datos

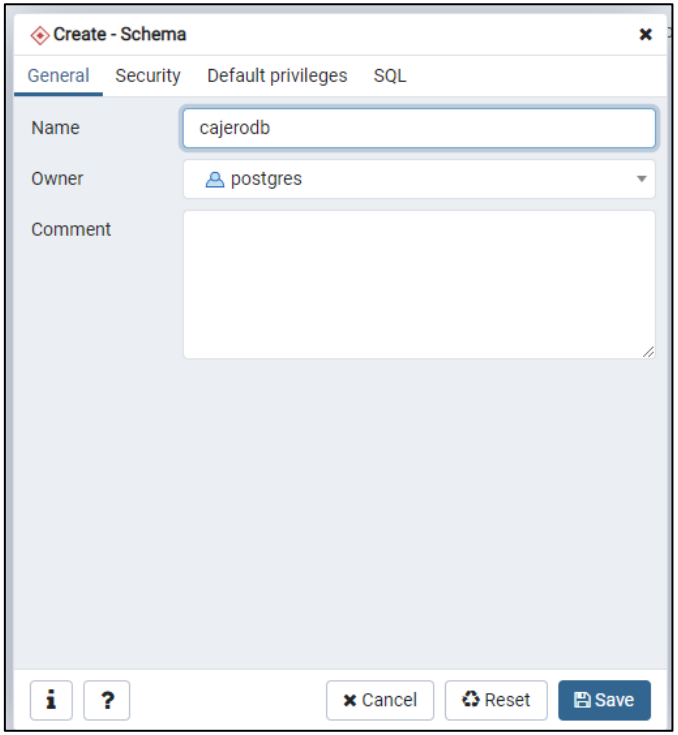
Con ayuda de pgAdmin crear una base de datos llamada MISIONTIC en el servidor local:

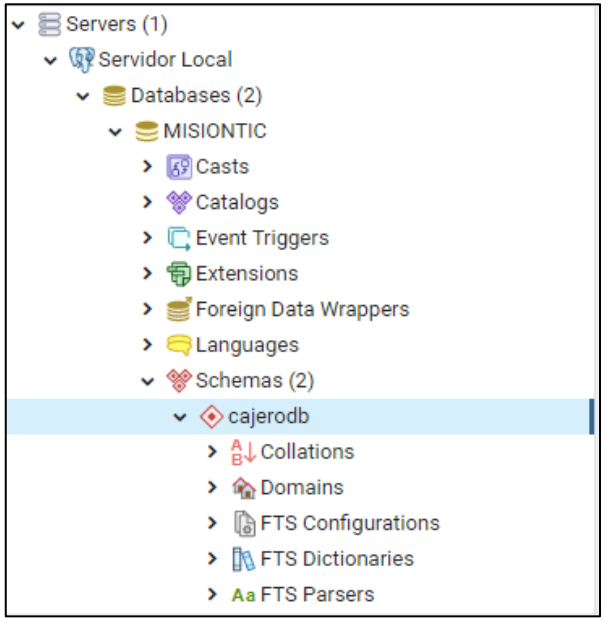




Creación de Esquema Vacío

Con ayuda de pgAdmin crear un esquema llamado cajerodb (es importante que este permanezca vacío:

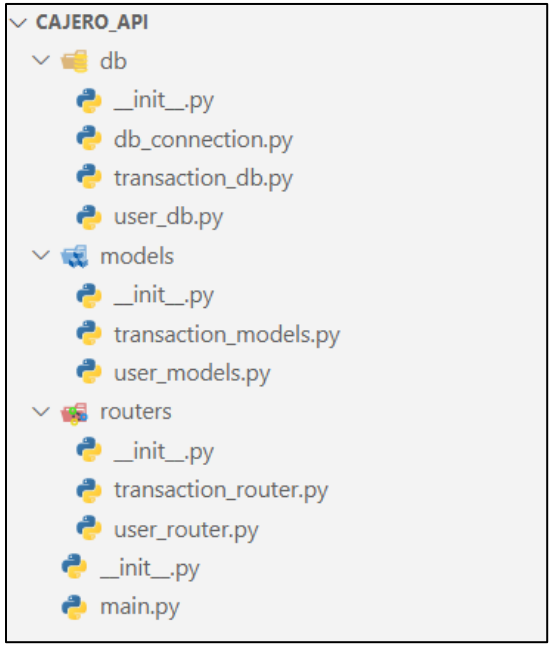




Estructura de la Capa Lógica

Estructura de la Capa Lógica

La estructura de la aplicación será la siguiente:



NOTA: Todos los archivos estarán vacíos, de momento solo es necesario crear la carpeta db y sus archivos.

Creación de la Conexión

Creando la Conexión

Para establecer la conexión con la base de datos, es necesario realizar lo siguiente:

- Importar los paquetes necesarios.
- Crear el motor de base de datos que se usará.
- Establecer una sesión con la base de datos, y crear la funcionalidad para inyectar las dependencias.
- Crear el modelo que se usará como base para la creación de las entidades de datos.

Importar los Paquetes Necesarios

En el archivo db_connection.py, definir el siguiente bloque de código:

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
```

Cuando se realice la ejecución, se instalarán los paquetes correspondientes.

Creación del Motor

En el archivo db_connection.py, defibir el siguiente bloque de código:

```
DATABASE_URL = "postgresql://user:password@host:port/name_db"
engine       = create_engine(DATABASE_URL)
```

Los valores de DATABASE_URL deben ser remplazados por los correspondientes.

Creación del Motor

Una posible configuración para DATABASE_URL es:

- user: *postgres*
- password: *password*
- host: *localhost*
- port: *5432*
- name_db: *MISIONTIC*

Esto quedará así:

```
DATABASE_URL = "postgresql://postgres:password@localhost:5432/MISIONTIC"
```

Creación de Sesión y Dependencias

En el archivo db_connection.py, definir el siguiente bloque de código:

```
SessionLocal = sessionmaker(autocommit=False,
                            autoflush=False,
                            bind=engine)

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

get_db será la encargada de inyectar la dependencia, es decir, SessionLocal().

Base para las Entidades de Datos

En el archivo db_connection.py, definir el siguiente bloque de código:

```
Base = declarative_base()
Base.metadata.schema = "cajerodb"
```

Base permitirá definir entidades de datos a partir de ella, estas entidades se convertirán en tablas.

Creación de Entidades de Datos

Entidad User

En el archivo user_db.py, definir el siguiente bloque de código:

```
from sqlalchemy import Column, Integer, String

from db.db_conection import Base, engine
```

Se importan los módulos necesarios.

Entidad User

En el archivo user_db.py, definir el siguiente bloque de código:

```
class UserInDB(Base):
    __tablename__ = "users"

    username    = Column(String, primary_key=True, unique=True)
    password    = Column(String)
    balance     = Column(Integer)

Base.metadata.create_all(bind=engine)
```

Se crea la entidad de datos y con la ultima línea se hace el proceso de creación de la tabla.

Entidad Transaction

En el archivo transaction_db.py, definir el siguiente bloque de código:

```
from sqlalchemy import Column, ForeignKey,
from sqlalchemy import Integer, String, DateTime
import datetime

from db.db_conection import Base, engine
```

Se importan los módulos necesarios.

Entidad Transaction

En el archivo transaction_db.py, definir el siguiente bloque de código:

```
class TransactionInDB(Base):
    __tablename__ = "transactions"

    id              = Column(Integer, primary_key=True, autoincrement=True)
    username        = Column(String, ForeignKey("users.username"))
    date            = Column(DateTime, default=datetime.datetime.utcnow)
    value           = Column(Integer)
    actual_balance  = Column(Integer)


Base.metadata.create_all(bind=engine)
```

Notar la relación en la columna username.

Referencias

- **[FASTAPI]** Comunidad FastAPI. (2020, noviembre). FastAPI. FastAPI. https://fastapi.tiangolo.com/