



## Guía: Creación de la clase PersonaDao (DAO) en Python

### Introducción

En esta guía crearemos la clase `PersonaDao` que implementa el patrón de diseño **DAO (Data Access Object)**. Esta clase nos permitirá realizar operaciones CRUD (Create, Read, Update, Delete) sobre la tabla `persona` de la base de datos PostgreSQL, encapsulando las sentencias SQL y su ejecución.

---

### Paso 1: Crear el archivo de la clase PersonaDao

 Primero, creamos un archivo llamado `persona_dao.py` dentro de la carpeta de trabajo:

 **Ruta del archivo:** `capa_datos_persona/persona_dao.py`

Este archivo contendrá la implementación de la clase `PersonaDao`.

---

### Paso 2: Definir la clase PersonaDao y sus atributos SQL

## Descripción

Definimos la clase `PersonaDao` y sus atributos de clase con las sentencias SQL necesarias.

### Código trabajado (capa\_datos\_persona/persona\_dao.py):

```
from conexion import Conexion
from persona import Persona
from logger_base import log

class PersonaDAO:
    """
    DAO (Data Access Object)
    CRUD (Create-Read-Update-Delete)
    """
    _SELECCIONAR = 'SELECT * FROM persona ORDER BY id_persona'
    _INSERTAR = 'INSERT INTO persona(nombre, apellido, email) VALUES(%s, %s, %s)'
    _ACTUALIZAR = 'UPDATE persona SET nombre=%s, apellido=%s, email=%s WHERE id_persona=%s'
    _ELIMINAR = 'DELETE FROM persona WHERE id_persona=%s'
```

### Explicación:

- Definimos las sentencias SQL para cada operación CRUD como atributos de clase.

---

## Paso 3: Definir el método `seleccionar`

### Descripción

Creamos un método de clase `seleccionar` que ejecuta la consulta `SELECT` y devuelve una lista de objetos `Persona`.

### Código trabajado (capa\_datos\_persona/persona\_dao.py):

```
@classmethod
def seleccionar(cls):
    with Conexion.obtener_conexion() as conexion:
        with conexion.cursor() as cursor:
            cursor.execute(cls._SELECCIONAR)
            registros = cursor.fetchall()
            personas = []
            for registro in registros:
                persona = Persona(registro[0], registro[1], registro[2], registro[3])
```

```

        personas.append(persona)
    return personas

```

### ✓ Explicación:

- Usamos `with` para gestionar automáticamente la conexión a la base de datos y el cursor.
- Ejecutamos la sentencia SQL de selección.
- Recuperamos los registros con `fetchall()`.
- Creamos un objeto `Persona` por cada registro y los agregamos a una lista.

## Paso 4: Probar el método `seleccionar`

### Descripción

Agregamos una prueba al final del archivo para comprobar que el método `seleccionar` funciona correctamente.

### Código trabajado (`capa_datos_persona/persona_dao.py`):

```

if __name__ == '__main__':
    personas = PersonaDAO.seleccionar()
    for persona in personas:
        log.debug(persona)


```

### ✓ Explicación:

- Si ejecutamos directamente este archivo, se llamará al método `seleccionar`.
- Se mostrarán en consola los objetos `Persona` recuperados, gracias al logger.

## Sección final: Código completo de los archivos trabajados

### Código completo del archivo:

 Ruta y nombre del archivo: `capa_datos_persona/persona_dao.py`

```

from conexion import Conexion

```

```

from persona import Persona

```

```

from logger_base import log

```

```

class PersonaDAO:

```

```

    ...

```

```
DAO (Data Access Object)
CRUD (Create-Read-Update-Delete)
'''
_SELECCIONAR = 'SELECT * FROM persona ORDER BY id_persona'
_INSERTAR = 'INSERT INTO persona(nombre, apellido, email) VALUES(%s, %s, %s)'
_ACTUALIZAR = 'UPDATE persona SET nombre=%s, apellido=%s, email=%s WHERE id_persona=%s'
_ELIMINAR = 'DELETE FROM persona WHERE id_persona=%s'

@classmethod
def seleccionar(cls):
    with Conexion.obtener_conexion() as conexion:
        with conexion.cursor() as cursor:
            cursor.execute(cls._SELECCIONAR)
            registros = cursor.fetchall()
            personas = []
            for registro in registros:
                persona = Persona(registro[0], registro[1], registro[2], registro[3])
                personas.append(persona)
            return personas

if __name__ == '__main__':
    # Seleccionar objetos
    personas = PersonaDAO.seleccionar()
    for persona in personas:
        log.debug(persona)
```

---

## ✅ Conclusión

👏 En esta guía implementamos la clase `PersonaDao`, aplicando el patrón de diseño DAO para encapsular las operaciones de acceso a la base de datos. Además, probamos la funcionalidad de la consulta `SELECT`, verificando que devuelve una lista de objetos `Persona`.

🚀 ¡Con esta implementación profesional, nuestra aplicación está lista para interactuar con la base de datos utilizando programación orientada a objetos y buenas prácticas!

---

💡 Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 🙌

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)