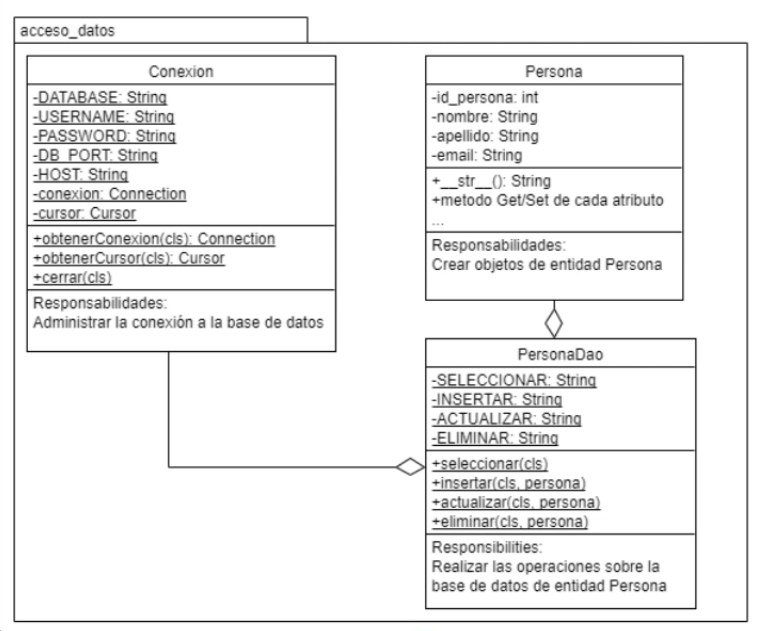
**CREANDO UNA CAPA DE DATOS**

Para lograr unas mejores “practicas” lo que se va a hacer es crear una clase que administre la conexiona la DB.

Las siglas DAO significan Data Access Object

Este es el diagrama de clase del ejercicio propuesto para esta sección.



**USO DE LOGGING**

Loggin es un modulo que maneja distintos “niveles” para enviar información, tanto a consola como a un archivo.

Los niveles pueden medirse según su “intensidad” o “valor”. Este es el orden

| **Level** | **When it’s used** |
| --- | --- |
| DEBUG | Detailed information, typically of interest only when diagnosing problems. |
| INFO | Confirmation that things are working as expected. |
| WARNING | An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected. |
| ERROR | Due to a more serious problem, the software has not been able to perform some function. |
| CRITICAL | A serious error, indicating that the program itself may be unable to continue running. |

La configuración default, solo “Muestra” los niveles con igual “prioridad” o mayor a Warning.

Igualmente esto puede ser modificado de la siguiente manera:

import logging as log

log.basicConfig(level= log.DEBUG)

de este modo se establece como nivel “basico” o “primario” el nivel de debug.

**COFIGURANDO MÁS EL LOGGING**

El modulo logging (as log) puede incluir más configuraciones para realizar más funciones o mostrarse de otro modo.

Estas configuraciones se agregan en el **basicConfig**, este es un ejemplo:

log.basicConfig(level= log.DEBUG, format='%(asctime)s: %(levelname)s [%(filename)s %(lineno)s %(message)s]',  
 datefmt='%I:%M:%S %p',  
 handlers=[  
 log.FileHandler('capa\_datos.log'),  
 log.StreamHandler()  
 ])

como se ve, aquí se agregaron unos cuantos nuevos parámetros.   
También aparece un nuevo concepto. EL de handlers, que son los que se verán afectados por el log.

El parámetro **format** recibe por medio de placeholders la “forma” o formato que tendrá el mensaje.

El parámetro **datefmt=** modifica el formato de la fecha (%(asctime)S) ya que por defecto incluye varios datos como (milisegundos)

Por ultimo en el parámetro **handlers** se incluye una lista [] de los manejadores. En este caso se incluye un archivo FileHandler (que se creara) y StreamHanlder() que es la consola

**CLASE CONEXIÓN**

La clase conexión no es una que pueda reutilizarse para cualquier conexión (aunque podría modificarse). Solo sirve para esta conexión:

class ConexionClase:  
  
 \_DATABASE = "test\_db\_python"  
 \_USERNAME = 'postgres'  
 \_PASSWORD = 'admin'  
 \_DB\_PORT = '5432'  
 \_HOST = 'localhost'  
 \_CONEXION = None  
 \_CURSOR = None  
  
 @classmethod  
 def get\_conexion(cls):  
 if cls.\_CONEXION is None:  
  
 try:  
 conn = psycopg2.connect(user=cls.\_USERNAME,  
 password=cls.\_PASSWORD,  
 host=cls.\_HOST,  
 port=cls.\_DB\_PORT,  
 database=cls.\_DATABASE)  
 cls.\_CONEXION = conn  
 log.debug(f'conexion exitosa: {cls.\_CONEXION}')  
 return cls.\_CONEXION  
 except Exception as e:  
 log.error(f'Ocurrio un error al obtener la conexion a la db {e}')  
 sys.exit()  
 else:  
 return cls.\_CONEXION  
  
 @classmethod  
 def get\_cursor(cls):  
 if cls.\_CURSOR is None:  
  
 try:  
 cursor = cls.get\_conexion().cursor()  
 cls.\_CURSOR = cursor  
 log.info(f'se abrió correctamente el cursor {cls.\_CURSOR}')  
 return cls.\_CURSOR  
 except Exception as e:  
 log.error(f'Ocurrio un error al obtener el cursor {e}')  
 else:  
 return cls.\_CURSOR

como se ve los atributos y métodos son de clase y no de instancia, por lo tanto, no es necesario instanciar la clase para utilizarles.

**PERSONA DAO:**

Las siglas DAO significan Data Access Object y se utilizan para realizar interacciones con la base de datos.