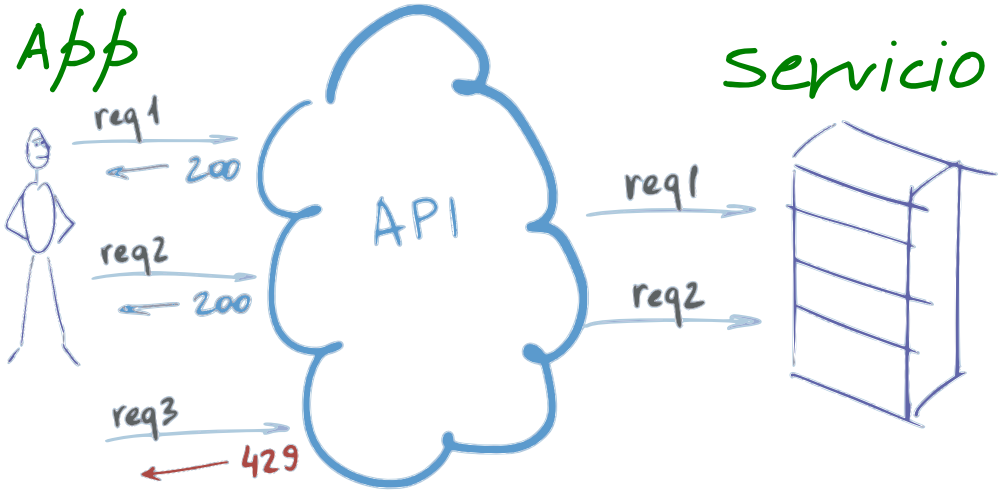
ESTUDIOS SUPERIORES DE BUENOS AIRES
ESBA

Conexión a Bases de Datos

API

Una **API** (application programming interface) es un conjunto de **código**, **definiciones** y **protocolos** que se utiliza para desarrollar e integrar aplicaciones usando distintas tecnologías.

Una API brinda cierto nivel de abstracción que simplifica la complejidad de acceso a un servicio o aplicación, implementando un conjunto de atributos y métodos de los cuales sólo se documentan los parámetros y los valores devueltos, sin necesidad de saber cómo están implementados.



Interfaz DB-API

Para el acceso a bases de datos la API está estandarizada mediante la especificación Database API (DB-API), actualmente en la versión 2.0 (PEP 249: Python Database API Specification v2.0)

El proceso de conexión siempre sigue estos pasos:

- Importar el conector
- Conectarse a la base de datos (función connect del módulo conector)
- Abrir un Cursor (método cursor de la conexión)
- Ejecutar una consulta (método execute del cursor)
- Obtener los datos (método fetch o iterar sobre el cursor)
- Cerrar el cursor (método close del cursor)

PyMySQL

Existen varios conectores para mysql implementados en python. Varios han quedado obsoletos como MySQLdb que era el estándar para la versión 2.

En los ejemplos usaremos el conector PyMySQL.

Cómo este no viene de forma estándar en el paquete de python, habrá que instalarlo mediante el comando **pip3** que se instala por defecto en las nuevas versiones de Python. Primero nos aseguramos que el comando pip3 o pip está disponible, en la ventana de comandos del sistema operativo escribir (con V mayúscula):

```
pip3 -V
```

Si todo está bien saldrá un texto indicando el número de versión y la ruta de instalación. Sino, habrá que buscar en internet cómo instalarlo y eso depende del S.O. y de la versión de python instalada.

Para instalar la librería hay varias formas, pero el comando para su instalación desde el S.O. que recomiendan en el mismo paquete es:

```
python3 -m pip install PyMySQL
```

Conectar a BBDD

Mediante el método .**connect()** de la clase **pymysql** se nos crea un objeto que tendrá los métodos para acceder a la base de datos.

Para conectarnos le debemos proporcionar el nombre del servidor, el nombre del usuario, la password de acceso y el nombre de la base de datos a la que nos queremos conectar.

Si tiene éxito, nos devolverá un objeto de la clase ***Connection*** que tendrá los métodos para gestionar los recursos de MySQL.

```
import pymysql

db = pymysql.connect( host="localhost",  # Servidor (PC local)
                      port= 3306,        # puerto de conexión
                      user="root",       # usuario
                      password="1234",   # clave
                      db="mibase" )      # base de datos
```

Cursores

Un **cursor** es una estructura de datos que se usa para recorrer (y eventualmente procesar) un conjunto de registros de resultado de una consulta **SQL**. El método para crear el cursor se llama precisamente cursor():

```
cursor = db.cursor()
```

Vamos a probar con una consulta para obtener la versión de MySQL instalada.

```
import pymysql
# Abrir la conexión
db = pymysql.connect(host="localhost", user="root",
                     password="1234", db="mibase" )
# Instanciar objeto cursor utilizando el método cursor()
cursor = db.cursor()

# Ejecutar una consulta SQL usando el método execute() method.
cursor.execute("SELECT VERSION()")
# Obtener un registro utilizando el método fetchone()
data = cursor.fetchone()
print ("Versión de MySQL: %s " % data)
```

```
Versión de MySQL: 8.0.35-0ubuntu0.22.04.1
```

Ejemplo

```
import pymysql
# Abrir la conexión
try:
    db = pymysql.connect(host="localhost", user="root", password="1234", db="mibase" )
except Exception as ex:
    print(f"Error al conectar: {ex}")
else:
    # Instanciar objeto cursor utilizando el método cursor()
    cursor = db.cursor()
    # Ejecutar una consulta SQL usando el método execute() method.
    try:
        cursor.execute("SELECT * FROM usuarios")
        # Recorrer los registros de un cursor
        for registro in cursor:
            print (registro)
    except Exception as ex:
        print(f"Error en la consulta: {ex}")

    # Al final, nos desconectamos del server
    db.close()
```