# Ayudantía 3
# Web y SQL

Ayudantes:
Consuelo Inostroza, Amelia González, Tomás Jackson, Vicente San Martín
Ricardo Oviedo, Daniela Torrent, Mateo Andrade

# Objetivos:

Esta ayudantía está enfocada en la Entrega 2. Hoy aprenderemos:

- Cómo trabajar y conectarse al **servidor.**
- **HTML:** Para crear la visualización de la página.
- **Python:** Para poder hacer la conexión a la base de datos y hacer consultas **SQL** a esta.
- Usaremos la librería **psycopg2** de python para poder trabajar con **postgres.**
- **IMPORTANTE:** ¿Cómo evitamos **inyecciones** en SQL?

# Uso del servidor

¿Como conectarse al servidor?

Conociendo nuestro número de grupo, en el terminal haremos lo siguiente:

```
ssh grupoXX@codd.ing.puc.cl
```

# Observaciones

- Nuestra contraseña por defecto será: grupoXX
- Para cambiar la contraseña:
  - Comando passwd
    - Pedirá la clave actual
    - Pedirá la clave nueva dos veces

# Subir archivos en el server

Primera opción:

Github:

1. Ir a repo.new o apretar +”new repository” en Github
2. Hacer el repositorio PRIVADO.
3. Agregar al resto del grupo:

   Settings -> Manage Access -> Invite Colaborator

# Sin branches

Trabajar asincrónicamente:

1. Hacer pull antes de hacer cambios

```
git pull
```

2. Luego se crean los commits directo a main

```
git add index.php
git commit -m "arregla el bug"
```

3. E inmediatamente se suben al origin (GitHub)

```
git push
```

# Con branches

Trabajar asincrónicamente:

1. Hacer pull y luego crear una rama de Git para implementar algo

```
# arreglar-error es un ejemplo, puede ser cualquier nombre
git checkout -b arreglar-error
# o también
git switch -c arreglar-error
```

2. Se hacen commits en esa rama y no en main

```
git add index.php
git commit -m "arregla el bug"
```

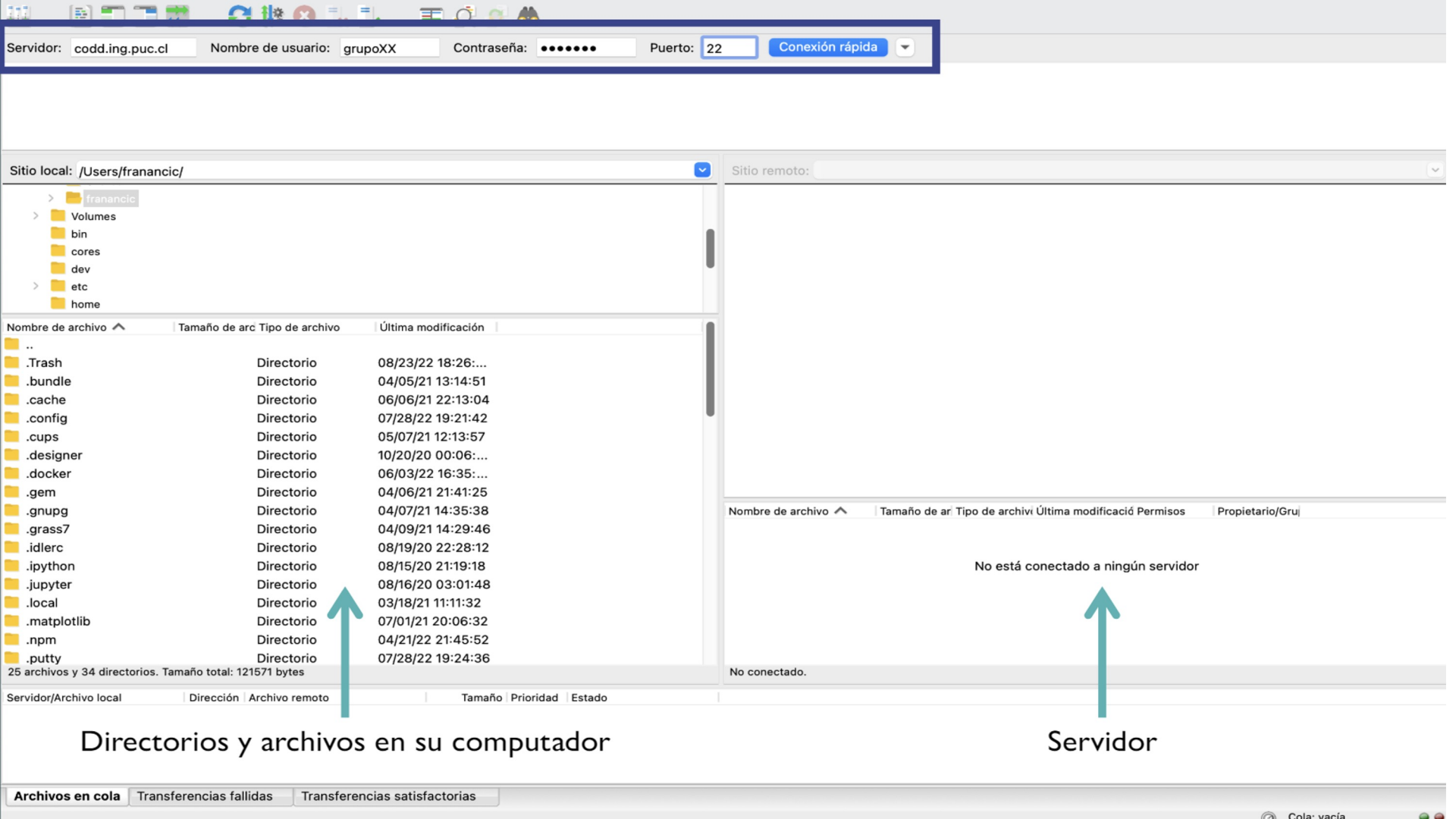
3. Se hacen Pull Requests en GitHub y se unen los cambios

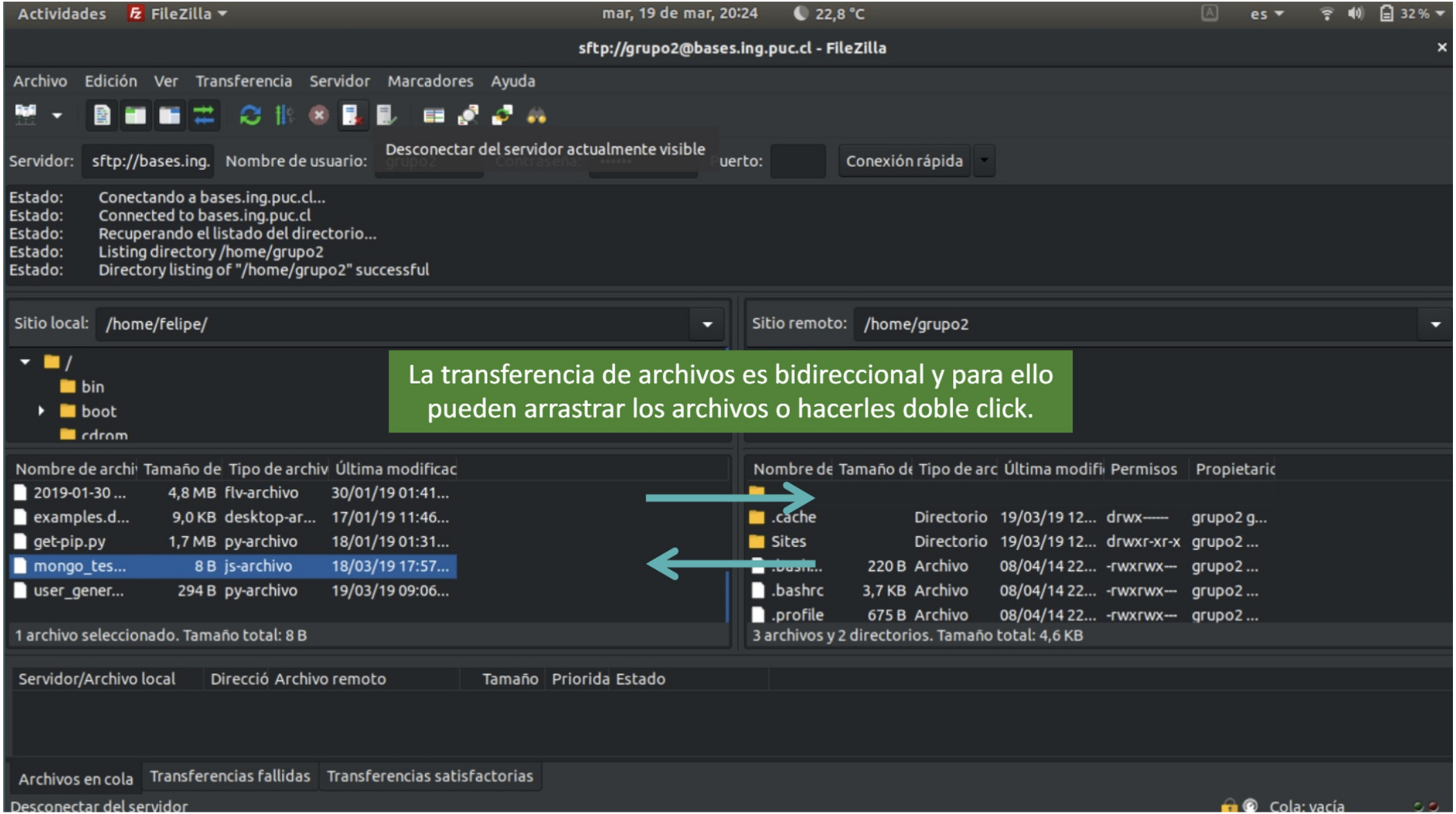
```
git push -u origin arreglar-error
# ir a https://github.com/<usuario>/<repo>/pull/new/arreglar-error
# y ahí unir los cambios
```

4. Volver a main y hacer pull

```
git checkout main
# o
git switch main
# y luego
git pull
```

Filezilla


Servidor: codd.ing.puc.cl
Nombre de usuario: grupoXX
Contraseña: •••••••
Puerto: 22
Conexión rápida
Sitio local: /Users/franancic/
Sitio remoto:
franancic
Volumes
bin
cores
dev
etc
home
Nombre de archivo
Tamaño de arc
Tipo de archivo
Última modificación
..
.Trash Directorio 08/23/22 18:26:...
.bundle Directorio 04/05/21 13:14:51
.cache Directorio 06/06/21 22:13:04
.config Directorio 07/28/22 19:21:42
.cups Directorio 05/07/21 12:13:57
.designer Directorio 10/20/20 00:06:...
.docker Directorio 06/03/22 16:35:...
.gem Directorio 04/06/21 21:41:25
.gnupg Directorio 04/07/21 14:35:38
.grass7 Directorio 04/09/21 14:29:46
.idlerc Directorio 08/19/20 22:28:12
.ipython Directorio 08/15/20 21:19:18
.jupyter Directorio 08/16/20 03:01:48
.local Directorio 03/18/21 11:11:32
.matplotlib Directorio 07/01/21 20:06:32
.npm Directorio 04/21/22 21:45:52
.putty Directorio 07/28/22 19:24:36
25 archivos y 34 directorios. Tamaño total: 121571 bytes
Nombre de archivo
Tamaño de ar
Tipo de archivi
Última modificació
Permisos
Propietario/Gru
No está conectado a ningún servidor
No conectado.
Servidor/Archivo local
Dirección
Archivo remoto
Tamaño
Prioridad
Estado
Directorios y archivos en su computador
Servidor
Archivos en cola
Transferencias fallidas
Transferencias satisfactorias
Cola: vacía


Actividades
FileZilla
mar, 19 de mar, 20:24
22,8 °C
es
32 %
sftp://grupo2@bases.ing.puc.cl - FileZilla
Archivo Edición Ver Transferencia Servidor Marcadores Ayuda
Servidor: sftp://bases.ing.
Nombre de usuario:
Desconectar del servidor actualmente visible
Puerto:
Conexión rápida
Estado: Conectando a bases.ing.puc.cl...
Estado: Connected to bases.ing.puc.cl
Estado: Recuperando el listado del directorio...
Estado: Listing directory /home/grupo2
Estado: Directory listing of "/home/grupo2" successful
Sitio local: /home/felipe/
Sitio remoto: /home/grupo2
/
bin
boot
cdrom
La transferencia de archivos es bidireccional y para ello pueden arrastrar los archivos o hacerles doble click.
Nombre de archi Tamaño de Tipo de archiv Última modificac
2019-01-30 ... 4,8 MB flv-archivo 30/01/19 01:41...
examples.d... 9,0 KB desktop-ar... 17/01/19 11:46...
get-pip.py 1,7 MB py-archivo 18/01/19 01:31...
mongo_tes... 8 B js-archivo 18/03/19 17:57...
user_gener... 294 B py-archivo 19/03/19 09:06...
1 archivo seleccionado. Tamaño total: 8 B
Nombre de Tamaño de Tipo de arc Última modifi Permisos Propietaric
.cache Directorio 19/03/19 12... drwx------ grupo2 g...
Sites Directorio 19/03/19 12... drwxr-xr-x grupo2 ...
220 B Archivo 08/04/14 22... -rwxrwx--- grupo2 ...
.bashrc 3,7 KB Archivo 08/04/14 22... -rwxrwx--- grupo2 ...
.profile 675 B Archivo 08/04/14 22... -rwxrwx--- grupo2 ...
3 archivos y 2 directorios. Tamaño total: 4,6 KB
Servidor/Archivo local Direcció Archivo remoto Tamaño Priorida Estado
Archivos en cola
Transferencias fallidas
Transferencias satisfactorias
Desconectar del servidor
Cola: vacía

# ¿Cómo utilizamos Postgresql en el server?

1. Luego de ingresar al servidor, hacemos el comando **psql.**
2. Ingresar contraseña (por defecto: grupoXX)
3. Cambiamos la contraseña:

**ALTER USER <grupoXX> ENCRYPTED PASSWORD ‘newpassword’;**

1. Ingresar a la base de datos correspondiente:

**\c grupoXXeN**

Para esta entrega N=2.

# ¿Cómo trabajamos en postgres?

- **Creación** de tablas:

CREATE TABLE users (User_id INT (serial) PRIMARY KEY, username VARCHAR (15) UNIQUE NOT NULL, ...);

- Poblamos tablas (**manualmente**):

INSERT INTO users (User_id, username, ...) VALUES (1, ‘sdawda’, ...);

- **Para poblarlas con un CSV:**

**\COPY users (columns_name) FROM ‘relative/path/to/file.csv’ DELIMITER ‘,’ CSV HEADER;**

# Comandos útiles

- \l lista de base de datos.
- \c <db> Ingresar a la base de datos db.
- \dt lista de las tablas de esa base de datos
- \d <table_name> descripción de la tabla <table_name>
- \? todos los comandos de psql
- \q salir (CTRL + D también funciona)

# HTML

Hyper Text Markup Language

- Es el lenguaje de marcado estándar utilizado para crear páginas web.
- Será la representación visual de la página.
- Se puede editar desde cualquier editor de texto.
- Cómo funciona:

Marca:

<marca>Contenido</marca>

Marca viene a ser el tipo de marca y contenido lo que está dentro de esta.

# Tipos de marca

- html: marca que describe el documento HTML (todas las otras marcas van dentro de él)
- head: Marca que contiene el encabezado (información sobre el documento) acá pueden agregar stylesheet.css
- title: Marca que contiene el título de la página.
- body: Marca que contiene el cuerpo del documento. Dentro de ella agregamos todos los elementos que queremos ver en nuestra visualización.
- h1: Marca que contiene un título.

# Tipos de marca

- table: Marca que define una tabla. las siguientes marcas van dentro de esta tabla:
  - tr: contiene la fila de una tabla.
  - td: contiene una celda de una tabla.
  - th: contiene el encabezado de la fila de una tabla.

# Python y psycopg2

Importamos la librería (nos sirve para trabajar con postgres):

```
import psycopg2
```

# Conexión con pyscopg2

Objeto para conectar:

```
try:
    conn = psycopg2.connect(
        database="database",
        user="user",
        host="localhost",
        port=5432,
        password="pass")
except:
    print("No me pude conectar")
```

# Conexión con pyscopg2

- conn.commit(): Luego de haber realizado todos nuestros cambios en la base de datos, este código hará que nuestros cambios sean permanentes en ella.

- conn.close(): Cerramos la conexión a la base de datos.

# Inserción

```
query = "INSERT INTO Peliculas VALUES (" + \
        titulo + ", " + ano + \
        ", " + director + ")";
cur = conn.cursor()
try:
    cur.execute(query)
except psycopg2.Error as e:
    print(e.pgcode)
```

# Cursores (fetchone)

-Puntero que va recorriendo los resultados.

-Iremos iterando de fila en fila y las iremos imprimiendo una por una:

```
import psycopg2

try:
    conn = psycopg2.connect(database="dbname",
            user="dbuser", host="localhost",
            password="dbpass")
    cur = conn.cursor()
    cur.execute("SELECT * FROM R")
    row = cur.fetchone()
    while row:
        print(row)
        row = cur.fetchone()
except:
    print("Hubo algún problema")
```

# Otra forma (fetchall)

```
try:
    conn = psycopg2.connect(database="dbname",
            user="dbuser", host="localhost",
            password="dbpass")
    cur = conn.cursor()
    cur.execute("SELECT * FROM R")
    rows = cur.fetchall()
    for row in rows:
        print(row)
except:
    print("Hubo algún problema")
```

# Pasar parámetros (en orden)

Ingresar argumentos en orden y sin declararlos:

```
cur.execute("""
    INSERT INTO users (id, fecha_primer_inicio_sesion, username)
    VALUES (%s, %s, %s);
    """,
    (10, datetime.date(2020, 10, 18), "dawda"))
```

# Pasar parámetros (con nombres)

Si queremos declarar y nombrar a los argumentos que ingresemos:

```
cur.execute("""
    INSERT INTO users (id, fecha_primer_inicio_sesion, username)
    VALUES (%(id)s, %(fecha_inicio)s, %(usario)s);
    """,
    {'id': 10, 'fecha_inicio': datetime.date(2020, 10, 18),
'usuario': "dawda"})
```

# OJO

Si no declaramos argumentos como en la anterior viñeta y si solo insertamos un argumento en el execute, entonces no nos olvidemos de cerrar bien el paréntesis y colocar una ‘,’ luego del argumento para insertar:

```
cur.execute("INSERT INTO some_table VALUES (%s)", ("adawd"))  # INCORRECTO
cur.execute("INSERT INTO some_table VALUES (%s)", ("adawd",)) # CORRECTO
```

# Inyección SQL

**NUNCA HACER ESTO: (No concatenar)**

```
SQL = "INSERT INTO usuarios (name) VALUES ('%s')"
data = ("Jaime", )
cur.execute(SQL % data)
```

**No usar comillas ni usar el operador %.**

**Otro error:**

```
#OTRO ERROR:
query = "SELECT * FROM users WHERE name = " + nombre
cursor.execute(query)
```

# Forma correcta

Para evitar inyecciones, cambiamos lo siguiente en el anterior código. No utilizamos las comillas en el parámetro y no usamos el operador %.

```
SQL = "INSERT INTO usuarios (name) VALUES (%s)"
data = ("Jaime", )
cur.execute(SQL, data)
```

# HTML y Python (archivo html)

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario</title>
</head>
<body>
    <h2>Formulario de Consulta</h2>
    <form action="consulta.py" method="post">
        Nombre: <input type="text" name="nombre"><br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

# HTML y Python (archivo .py) (Posible Forma)

```
import cgi
import psycopg2
conn = psycopg2.connect(dbname="grupoXXeN, user="grupoXX",
password="contraseña", host="localhost", port=5432)
cursor = conn.cursor()
form = cgi.FieldStorage()
nombre = form.getvalue('nombre')
cursor.execute("SELECT * FROM tabla WHERE nombre = %s",
(nombre,))
resultados = cursor.fetchall()
cursor.close()
conn.close()
```