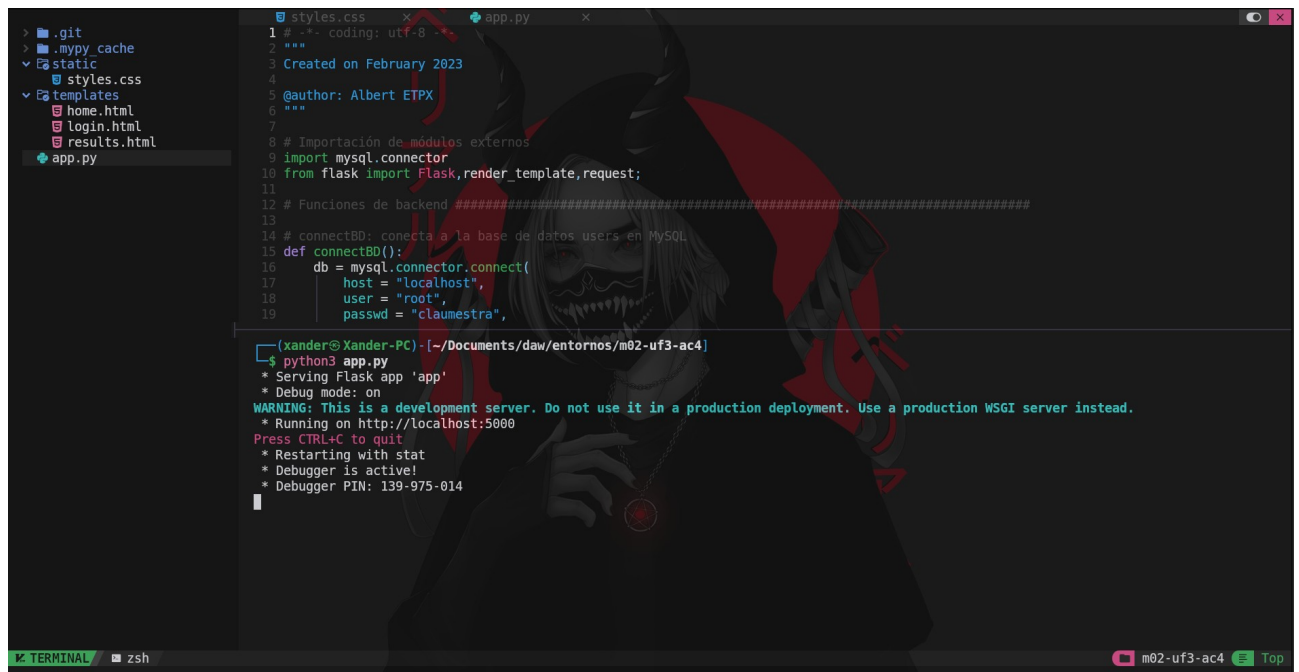


## Tarea 1:

a) Haz un fork del siguiente repositorio en tu cuenta de GitHub. A continuación, clónalo en tu máquina local.

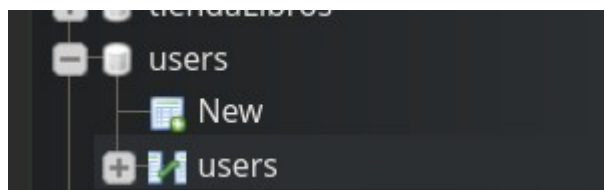
<https://github.com/albertetpx/m02-uf3-ac4.git>



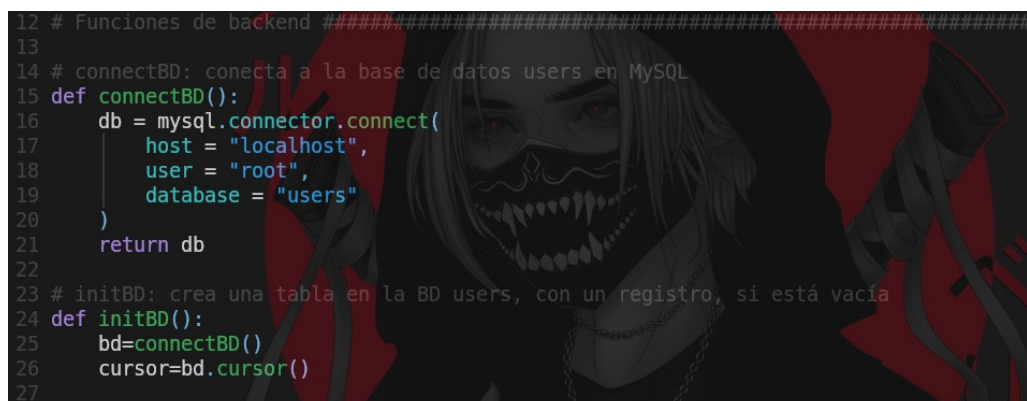
```
styles.css x app.py x
1 # -*- coding: utf-8 -*-
2 ***
3 Created on February 2023
4
5 @author: Albert ETPX
6 ***
7
8 # Importación de módulos externos
9 import mysql.connector
10 from flask import Flask, render_template, request;
11
12 # Funciones de backend #####
13
14 # connectBD: conecta a la base de datos users en MySQL
15 def connectBD():
16     db = mysql.connector.connect(
17         host = "localhost",
18         user = "root",
19         passwd = "claumestra",
20     )
21     return db
22
23 # initBD: crea una tabla en la BD users, con un registro, si está vacía
24 def initBD():
25     bd=connectBD()
26     cursor=bd.cursor()
27
28 (xander@Xander-PC) - [~/Documents/daw/entornos/m02-uf3-ac4]
$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://localhost:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 139-975-014
```

b) Realiza las siguientes operaciones para poder desplegar la aplicación web que se adjunta (formulario.rar).

-A LA APLICACIÓN FLASK: cambia los parámetros de conexión a la base de datos (en concreto, el usuario y la contraseña) para que pueda conectar a tu servidor MySQL.



-AL SERVIDOR MYSQL: crea la base de datos users. No hace falta que crees ninguna tabla; será creada por la propia aplicación web (observa la función



```
12 # Funciones de backend #####
13
14 # connectBD: conecta a la base de datos users en MySQL
15 def connectBD():
16     db = mysql.connector.connect(
17         host = "localhost",
18         user = "root",
19         database = "users"
20     )
21     return db
22
23 # initBD: crea una tabla en la BD users, con un registro, si está vacía
24 def initBD():
25     bd=connectBD()
26     cursor=bd.cursor()
27
```

c) Ejecuta la aplicación (app.py) y compruebe que arranca sin errores. Abra el navegador en `http://localhost:5000`, y compruebe que:

- El usuario `usr01` con contraseña `admin` puede hacer **login correcto** y consultar sus datos.
- El usuario `user01` con contraseña `1234` hace un **login incorrecto**.

Aplicació web amb base de dades (M02-UF3-AC4)

LOGIN CORRECTO

User	Name	Surname 1	Surname 2	Age	Genre
user01	Ramón	Sigüenza	López	35	H

ETPX 2022-2023

d)Prueba a autenticar el usuario `usr01` y la contraseña ‘ `OR 1=1;` (valor exacto).

Estamos accediendo realizando una inyeccion SQL

Log in to application

user01

' OR 1=1;

Enviar

LOGIN CORRECTO

User	Name	Surname 1	Surname 2	Age	Genre
user01	Ramón	Sigüenza	López	35	H

e) Explica qué pasa, y por qué estamos ante una situación de inyección de código.

```
query=f"SELECT user,name,surname1,surname2,age,genre FROM users WHERE user='{user}'\nAND password='{password}'"
```

Estamos manipulando la condición del WHERE de tal manera que se cumpla la condición, siguiendo los puntos:

- El Password = " (Espacio vacío) En la condición se toma como una condición falsa, ya que hay texto en el campo de Password, pero como el OR pide que al menos se cumpla una condición toma como True condición, por eso se pone 1=1.
- Por otro lado el uso de las comillas se hace para que se cierren las comillas simples y se pueda realizar la comparación de OR 1=1;

```
AND password='' OR 1=1;
```

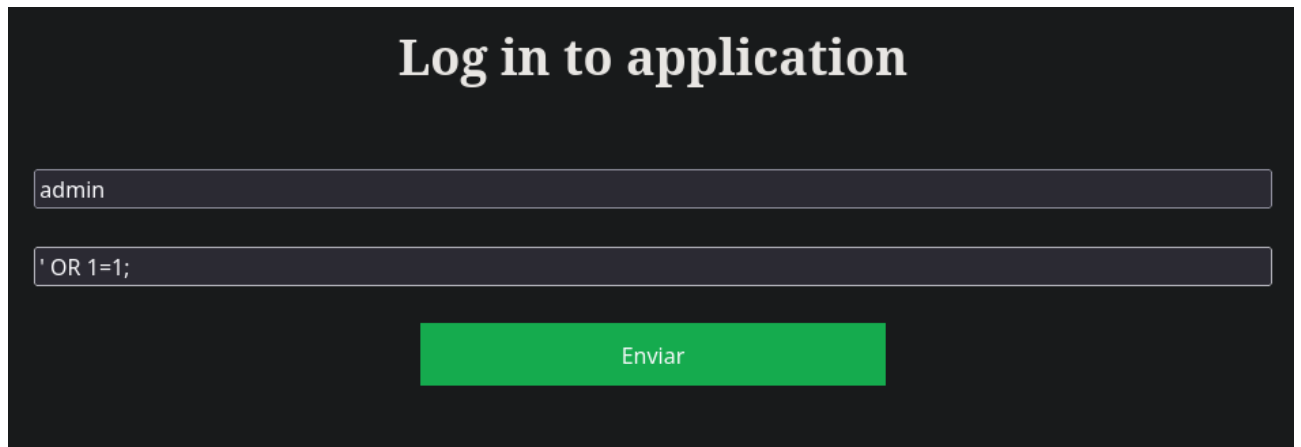
f) Reimplementa la función checkUser para que utilice una sentencia parametrizada que evite la situación de inyección de código:

Por ejemplo:

```
query = """Update employee set Salary = %s where id = %s"""  
values = (8000, 5)  
cursor.execute(query, values)
```

```
def checkUser(user,password):  
    bd=connectBD()  
    cursor=bd.cursor()  
  
    """  
    ?Para evitar las inyecciones de código podemos parametrizar los comandos, es decir,  
    ?debemos evitar es concatenar directamente en la consulta a realizar, ya que da salida a inyecciones de código.  
    """  
  
    query = "SELECT user, name, surname1, surname2, age, genre FROM users WHERE user = %s AND password = %s"  
  
    print(query)  
    cursor.execute(query, (user, password))  
    userData = cursor.fetchall()  
    bd.close()  
  
    if userData == []:  
        return False  
    else:  
        return userData[0]
```

g) Comprueba que, ahora, el formulario de login ya no es vulnerable a la inyección de código.



Log in to application

admin

' OR 1=1;

Enviar



LOGIN INCORRECTO

h) Explica por qué la instrucción parametrizada resuelve la vulnerabilidad.

Esta es capaz de resolver esta vulnerabilidad en cierto grado, ya que los parámetros son tomados directamente como valores por parte de SQL, no como parte del comando, corrigiendo y logrando aliviar una gran posibilidad de inyecciones de código.

## Tarea 2:

a) Crea otra plantilla (signin.html), siguiendo la estructura de login.html. Esta página deberá contener un formulario de registro de usuario, donde se pueda dar de alta a un usuario con: nombre de usuario, contraseña, nombre, apellido1, apellido2, edad y salario.

```
<form action="{{url_for('signin')}}" method="POST" class="formulario">
  <h2>Register to application</h2>

  <input type="text" name="usuario" placeholder="usuario" />
  <input type="password" name="contrasena" placeholder="contraseña" />
  <input type="text" name="name" placeholder="name" />
  <input type="text" name="surname1" placeholder="surname1" />
  <input type="text" name="surname2" placeholder="surname2" />
  <input type="text" name="age" placeholder="age" />
  <input type="text" name="genre" placeholder="genre" />

  <input type="submit" value="Enviar">
</form>
```

b) Modifica la ruta “/signin” en la aplicación flask para que muestre la plantilla signin.html que acabas de crear.

```
@app.route("/register", methods=('GET', 'POST'))
def register():
    return render_template("register.html")
```

c) Crear una nueva ruta (“/newUser”) en la aplicación flask para recibir y procesar los datos del formulario de registro. Tendrás que inspirarte en la ruta “/results” ya existente. Desde esta ruta, llama a la función createUser.

```
@app.route("/register", methods=('GET', 'POST'))
def register():

    if request.method == 'POST':

        formData = request.form;

        user = formData.get('usuario')
        password = formData.get('contrasena')
        name = formData.get('name')
        surname1 = formData.get('surname1')
        surname2 = formData.get('surname2')
        age = formData.get('age')
        genre = formData.get('genre')

        print(formData)

        statusUser = createUser(user, password, name, surname1, surname2, age, genre)

        if (not(statusUser == True)):
            return "Usuario no creado"

    return render_template("register.html")
```

d) Asocia la acción del formulario de registro (atributo action) a la nueva ruta que acabas de crear. Observa cómo se hace en el formulario de login.

```
<form action="{{url_for('register')}}" method="POST" class="formulario">
  <h2>Register to application</h2>
```

e) Implementa la función createUser para que se escriban los datos del nuevo usuario en la base de datos. Utiliza sentencias parametrizadas.

```
# createUser: crea un nuevo usuario en la BD
def createUser(user,password,name,surname1,surname2,age,genre):










    query = """
        INSERT INTO users(user, password, name, surname1, surname2, age, genre)
        VALUES (%s,%s,%s,%s,%s,%s,%s);
        """

    bd = connectBD()
    cursor = bd.cursor()

    try:
        cursor.execute(query, (user, password, name, surname1, surname2, age, genre))
        bd.commit();

        return True;
    except Exception as e:
        bd.rollback();
        return False

    finally:
        bd.close()
```

<div><div><div></div><div></div><div></div></div><div></div></div>														user	password	name	surname1	surname2	age	genre
<input type="checkbox"/>	 Edit	 Copy	 Delete	albaro	albaro69	albaro	florez	salamaleko69	23	H										
<input type="checkbox"/>	 Edit	 Copy	 Delete	pipe	pipe	pipe	pipe	pipe	10	H										
<input type="checkbox"/>	 Edit	 Copy	 Delete	user01	admin	Ramón	Sigüenza	López	35	H										

f) Comprueba el correcto funcionamiento de la aplicación.







## Aplicació web amb base de dades (M02-UF3-AC4)

### Register to application

kris
....
kristian
gutierrez
gutierrezzzz
19
h

Enviar

ETPX 2022-2023

				user	password	name	surname1	surname2	age	genre	
<input type="checkbox"/>		Edit	 Copy	 Delete	kris	kris	kristian	gutierrez	gutierrezzzz	19	H
<input type="checkbox"/>		Edit	 Copy	 Delete	user01	admin	Ramón	Sigüenza	López	35	H

g)Sube el código completo de la aplicación a un repositorio de tu cuenta de GitHub e incluye el enlace en el PDF que entregas.