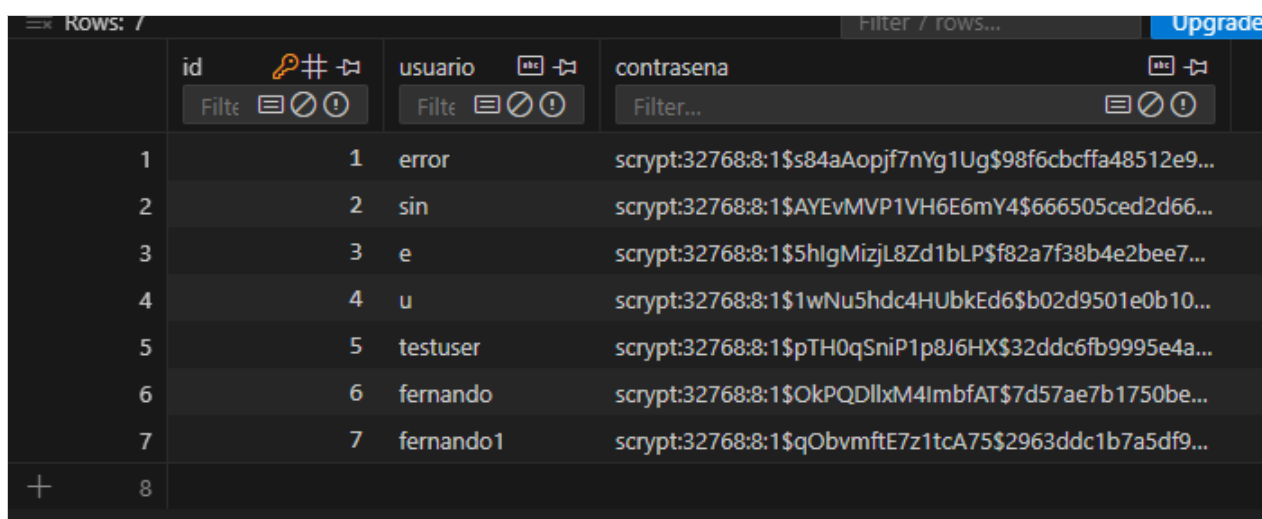


# PFO 2: Sistema de Gestión de Tareas (API Flask + SQLite)

Este proyecto implementa una API REST básica con Flask para registrar usuarios, iniciar sesión y mostrar una vista de tareas protegida. Incluye autenticación básica y almacenamiento de usuarios con contraseñas hasheadas usando SQLite.

## Funcionalidades del servidor

- Registro de usuarios ( POST /registro )
- Inicio de sesión ( POST /login )
- Visualización de bienvenida ( GET /tareas )
- Contraseñas protegidas (hashed)
- Verifica credenciales y permite acceso a las tareas
- Base de datos persistente (SQLite)
- Ejemplos de registro guardados en la base de datos



	id	usuario	contrasena
1	1	error	scrypt:32768:8:1\$s84aAopjf7nYg1Ug\$98f6cbcffa48512e9...
2	2	sin	scrypt:32768:8:1\$AYEvMVP1VH6E6mY4\$666505ced2d66...
3	3	e	scrypt:32768:8:1\$5hlgMizjL8Zd1bLP\$f82a7f38b4e2bee7...
4	4	u	scrypt:32768:8:1\$1wNu5hdc4HUbKEd6\$b02d9501e0b10...
5	5	testuser	scrypt:32768:8:1\$pTH0qSniP1p8J6HX\$32ddc6fb9995e4a...
6	6	fernando	scrypt:32768:8:1\$OkPQDlIxM4lmbfAT\$7d57ae7b1750be...
7	7	fernando1	scrypt:32768:8:1\$qObvmftE7z1tcA75\$2963ddc1b7a5df9...
+	8		

## Funcionalidades del cliente

- Iniciar sesión con usuario y contraseña ( POST /login )
- Registrarse si el login falla ( POST /registro )
- Ver tareas si el login fue exitoso ( GET /tareas )
- Tiene un límite de 3 intentos fallidos de ingreso de contraseñas incorrectas, luego cierra el cliente.

# Repositorio Público

[Ver en GitHub](#)

---

## Demo / Documentación:

[Ver en GitHub Pages](#)

Este proyecto implementa una aplicación de consola en Python que se comunica con un servidor Flask (API REST). Por su naturaleza, no puede desplegarse en GitHub Pages, ya que esta plataforma solo permite publicar contenido estático (HTML, CSS, JS), y no ejecuta código del lado del servidor ni scripts de consola como `cliente.py`.

Por eso, la sección de GitHub Pages del repositorio fue utilizada exclusivamente para alojar una documentación visual del proyecto, incluyendo capturas de pantalla, flujos de uso y explicaciones de endpoints.

Como alternativa real de despliegue para probar el servidor Flask, se podría utilizar una plataforma como **Render** (<https://render.com>), que sí permite publicar aplicaciones Python con backend web. Allí podría alojarse `servidor.py` y permitir que `cliente.py` interactúe remotamente.

---

## Backlog de funcionalidades pendientes

- [X] Reorganizar la estructura del proyecto con carpetas (ej. `/app`)
  - [X] Agregar confirmación de contraseña en el registro (validación mínima)
  - [X] Mostrar una página HTML más personalizada en `/tarear` (usuario logueado)
  - [X] Documentar el uso del cliente en consola
  - [X] Revisar flujo en Error en el login: Credenciales inválidas
  - [X] Limitar la cantidad a 3 la cantidad de intentos fallidos antes de cerrar el cliente
- 

## Instalación y ejecución (Windows)

### 1. Clonar el repositorio

```
bash git clone https://github.com/paezzanini/PF02-ProgRedes cd PF02-ProgRedes
```

### 2. Crear entorno virtual (opcional pero recomendado)

```
bash python -m venv env env\Scripts\activate
```

### 3. Instalar dependencias

```
bash pip install -r requirements.txt
```

### 4. Ejecutar el servidor Flask

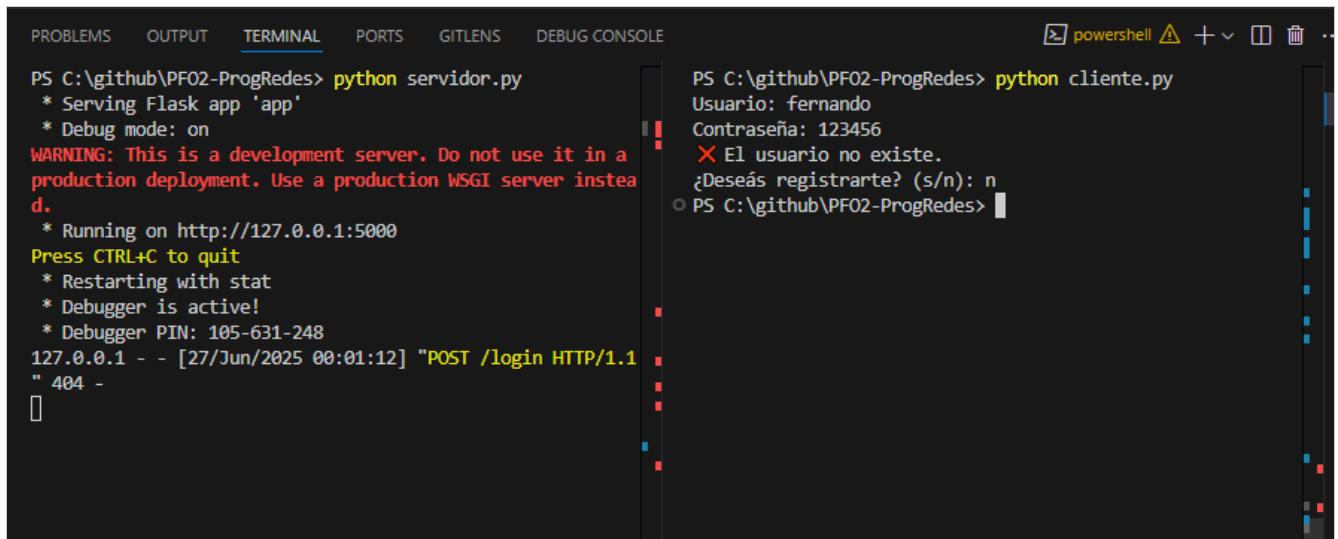
```
bash python servidor.py
```

### 5. Ejecutar el cliente en consola en otra consola

```
bash python cliente.py
```

## Ejemplos de Uso

#### Usuario no registrado y no acepto registrarme



```
PROBLEMS OUTPUT TERMINAL PORTS GITLENS DEBUG CONSOLE
PS C:\github\PF02-ProgRedes> python servidor.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://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 105-631-248
127.0.0.1 - - [27/Jun/2025 00:01:12] "POST /login HTTP/1.1" 404 -

PS C:\github\PF02-ProgRedes> python cliente.py
Usuario: fernando
Contraseña: 123456
❌ El usuario no existe.
¿Deseás registrarte? (s/n): n
PS C:\github\PF02-ProgRedes>
```

Usuario: fernando Contraseña: 123456

❌ El usuario no existe. ¿Deseás registrarte? (s/n): n

#### Usuario no registrado y acepto registrarme

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  GITLENS  DEBUG CONSOLE

PS C:\github\PF02-ProgRedes> python servidor.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://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 105-631-248
127.0.0.1 - - [27/Jun/2025 00:07:21] "POST /login HTTP/1.1" 404 -
127.0.0.1 - - [27/Jun/2025 00:07:29] "POST /registro HTTP/1.1" 200 -

PS C:\github\PF02-ProgRedes> python cliente.py
Usuario: fernando
Contraseña: 123456
X El usuario no existe.
¿Deseás registrarte? (s/n): s
Confirmá la contraseña: 123456
✓ Usuario registrado con éxito. Ahora podés iniciar sesión
n
PS C:\github\PF02-ProgRedes>
```

Usuario: fernando Contraseña: 123456

✖ El usuario no existe. ¿Deseás

registrarte? (s/n): s

Confirma la contraseña: 123456

✅ Usuario registrado con éxito. Ahora podés iniciar sesión

Se ejecuta nuevamente el cliente y ya permite loguearse con los datos de registro ``

**Usuario registrado - Credenciales invalidas - Limita a 3 intentos fallidos antes de cerrar el cliente**

```
PROBLEMS OUTPUT TERMINAL PORTS GITLENS DEBUG CONSOLE
```


```
PS C:\github\PF02-ProgramRes> python servidor.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://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 105-631-248
127.0.0.1 - - [27/Jun/2025 00:07:21] "POST /login HTTP/1.1" 404 -
127.0.0.1 - - [27/Jun/2025 00:07:29] "POST /registro HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:09:06] "POST /login HTTP/1.1" 401 -
```

```
PS C:\github\PF02-ProgramRes> python cliente.py
Usuario: fernando
Contraseña: 123456
X El usuario no existe.
¿Deseás registrarte? (s/n): s
Confirmá la contraseña: 123456
✓ Usuario registrado con éxito. Ahora podés iniciar sesión
PS C:\github\PF02-ProgramRes> python cliente.py
Usuario: fernando
Contraseña: 123465
X Contraseña incorrecta.
Intento 1/3. Intente nuevamente.

Contraseña: 
```

Usuario: fernando

Contraseña: 123465

 Contraseña incorrecta.

Intento 1/3. Intente nuevamente.

Contraseña: 1234

❌ Contraseña incorrecta.

Intento 2/3. Intente nuevamente.

Contraseña: 143

❌ Contraseña incorrecta.

❌ Demasiados intentos fallidos. Cerrando.

## Usuario registrado - login exitoso

```
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 105-631-248
127.0.0.1 - - [27/Jun/2025 00:07:21] "POST /login HTTP/1.1" 404 -
127.0.0.1 - - [27/Jun/2025 00:07:29] "POST /registro HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:09:06] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:10:39] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:10:44] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:15:05] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:15:07] "GET /tareas?usuario=fernando HTTP/1.1" 200 -

PS C:\github\PF02-ProgRedes> python cliente.py
Usuario: fernando
Contraseña: 123456

✅ ¡Login exitoso!

TAREAS:
<!DOCTYPE html>
<html>
<head><title>Tareas</title></head>
<body>
  <h1>Bienvenido, fernando</h1>
  <p>Acá vas a ver tus tareas (¡próximamente!).</p>
</body>
</html>

PS C:\github\PF02-ProgRedes>
```

Usuario: fernando

Contraseña: 123456

✅ ¡Login exitoso!

TAREAS:

<!DOCTYPE html>

<html>

<head>

<title>Tareas</title>

</head>

<body>

<h1>Bienvenido, fernando</h1>

<p>Acá vas a ver tus tareas (¡próximamente!).</p>

</body>

</html>

## Usuario no registrado confirmacion de pass erronea

```
PROBLEMS OUTPUT TERMINAL PORTS GITLENS DEBUG CONSOLE
* Debugger is active!
* Debugger PIN: 105-631-248
127.0.0.1 - - [27/Jun/2025 00:07:21] "POST /login HTTP/1.1" 404 -
127.0.0.1 - - [27/Jun/2025 00:07:29] "POST /registro HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:09:06] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:10:39] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:10:44] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:15:05] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:15:07] "GET /tareass?usuario=fernando HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:18:06] "POST /login HTTP/1.1" 404 -

PS C:\github\PF02-ProgRedes> python cliente.py
Usuario: fernando1
Contraseña: 12345
❌ El usuario no existe.
¿Deseás registrarte? (s/n): s
Confirmá la contraseña: 1234
❌ Las contraseñas no coinciden.
PS C:\github\PF02-ProgRedes>
```

Usuario: fernando1  
Contraseña: 12345  
❌ El usuario no existe.  
¿Deseás registrarte? (s/n): s  
Confirmá la contraseña: 1234  
❌ Las contraseñas no coinciden.

### Usuario no registrado confirmacion de pass correcta

```
127.0.0.1 - - [27/Jun/2025 00:07:29] "POST /registro HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:09:06] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:10:39] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:10:44] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [27/Jun/2025 00:15:05] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:15:07] "GET /tareass?usuario=fernando HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2025 00:18:06] "POST /login HTTP/1.1" 404 -
127.0.0.1 - - [27/Jun/2025 00:20:31] "POST /login HTTP/1.1" 404 -
127.0.0.1 - - [27/Jun/2025 00:20:38] "POST /registro HTTP/1.1" 200 -

PS C:\github\PF02-ProgRedes> python cliente.py
Usuario: fernando1
Contraseña: 12345
❌ El usuario no existe.
¿Deseás registrarte? (s/n): s
Confirmá la contraseña: 1234
❌ Las contraseñas no coinciden.
PS C:\github\PF02-ProgRedes> python cliente.py
Usuario: fernando1
Contraseña: 12345
❌ El usuario no existe.
¿Deseás registrarte? (s/n): s
Confirmá la contraseña: 12345
✅ Usuario registrado con éxito. Ahora podés iniciar sesión
PS C:\github\PF02-ProgRedes>
```

Usuario: fernando1  
Contraseña: 12345  
❌ El usuario no existe.  
¿Deseás registrarte? (s/n): s  
Confirmá la contraseña: 12345  
✅ Usuario registrado con éxito. Ahora podés iniciar sesión

---

## Requisitos técnicos

- Python 3.8 o superior
  - Flask
  - SQLite3
  - Werkzeug (para hashing de contraseñas)
- 

## Respuestas conceptuales

### ¿Por qué hashear contraseñas?

Hashear contraseñas significa aplicar una función matemática que transforma la contraseña original en una cadena irreconocible. Es un proceso que no se puede revertir fácilmente, por eso se dice que es unidireccional.

Esto se hace para no guardar contraseñas en texto plano (es decir, tal como las escribe el usuario). Si alguien llegara a robar la base de datos, no podría ver directamente las contraseñas reales, porque solo tendría los valores ya hasheados.

Además, muchas veces se le agrega una “sal” (un valor aleatorio) antes de aplicar el hash, para hacer más difícil que alguien use ataques automáticos para adivinar las contraseñas.

En resumen, hashear sirve para proteger los datos de los usuarios y evitar que, si alguien accede a la base, pueda ver las contraseñas reales.

### Ventajas de usar SQLite

SQLite es una base de datos liviana que no necesita instalar un servidor aparte ni configuraciones complicadas. Todo se guarda en un solo archivo .db, lo que la hace muy práctica para proyectos chicos o medianos como este.

Una de las ventajas más grandes es que es fácil de usar y muy rápida para desarrollos locales o aplicaciones que no van a tener miles de usuarios al mismo tiempo.

También es ideal para practicar o hacer trabajos prácticos, porque no depende de conexiones externas ni servicios adicionales. Basta con importar la librería en Python y ya se puede empezar a guardar datos.

En resumen, SQLite es simple, funciona bien para este tipo de proyectos educativos o personales, y permite centrarse en la lógica del programa sin preocuparse por configurar un sistema de base de datos más complejo.

---