

# PROYECTO SILKUNG

REALIZADO POR Fº JAVIER FIESTAS BOTELLA  
TOKIO SCHOOL

MAYO DE 2023

Silvia García Cuesta



Qigong & Pilates

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

## Manual de instalación.

El proyecto 'Silkung' está creado con Python 3.9.0 utilizando :

- PyCharm 2022.3.1 (Edición Comunitaria)  
Versión # PC-223.8214.51, compilado el 20 de diciembre de 2022  
Versión de tiempo de ejecución: 17.0.5+1-b653.23 amd64  
Máquina virtual: OpenJDK 64-Bit Server VM de JetBrains s.r.o.  
Windows 11 10.0  
GC: Generación Joven G1, Generación Antigua G1  
Memoria: 2038M  
Núcleos: 8
- Sistema Operativo Windows 11 Home  
Versión 22H2  
Versión del sistema operativo 22621.1555

### 1-Entorno Virtual.

El proyecto cuenta con su propio entorno virtual en pycharm, claro que éste viene desinstalado, habiéndose creado el archivo requirement.txt mediante el comando; `pip freeze > requirement`

*Para la instalación del del archivo requeriment.txt una vez descargado al archivo zip con el código del proyecto en Pycharm*

- Abre el proyecto
- Ve al menú 'File' y selecciona 'Settings'
- Selecciona el nombre del proyecto 'Silkung' y 'Python interpreter'
- *Añadimos el entorno virtual eligiendo 'Virtualenv Environment' y la ubicación*
- *Ahora creamos en 'Create'*
- *Luego abrimos la terminal y navegamos hasta el directorio del proyecto*
- *Ejecutamos el comando: `pip install -r requirements.txt` instalando todas las dependencias del proyecto*

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

## 2-Dependencias del proyecto.

A continuación un listado de las librerías con sus versiones del proyecto que se encuentran en el archivo requirements.txt.

```
aiohttp==3.8.4  
aiosignal==1.3.1  
async-timeout==4.0.2  
attrs==23.1.0  
blinker==1.6.2  
blis==0.7.9  
catalogue==2.0.8  
certifi==2022.12.7  
charset-normalizer==3.1.0  
click==8.1.3  
colorama==0.4.6  
confection==0.0.4  
contourpy==1.0.7  
cycler==0.11.0  
cymem==2.0.7  
es-core-news-md @ https://github.com/explosion/spacy-models/releases/download/es_core_news_md-3.5.0/es_core_news_md-3.5.0-py3-none-any.whl#sha256=b65a00a896debcfcaec84245bc94a21b397c6b151e70d098d84b4d44d38ac14c  
Flask==2.2.3  
Flask-Mail==0.9.1  
fonttools==4.39.3  
frozenset==1.3.3  
greenlet==2.0.2  
idna==3.4  
importlib-metadata==6.0.0  
importlib-resources==5.12.0  
itsdangerous==2.1.2  
Jinja2==3.1.2  
kiwisolver==1.4.4  
langcodes==3.3.0  
MarkupSafe==2.1.2  
matplotlib==3.7.1  
multidict==6.0.4  
murmurhash==1.0.9  
numpy==1.24.2  
openai==0.27.6  
packaging==23.1  
pathy==0.10.1  
Pillow==9.5.0  
playsound==1.3.0  
preshed==3.0.8
```

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

```
pydantic==1.10.7  
pyparsing==3.0.9  
python-dateutil==2.8.2  
python-dotenv==1.0.0  
pywin32==306  
requests==2.29.0  
six==1.16.0  
smart-open==6.3.0  
spacy==3.5.2  
spacy-legacy==3.0.12  
spacy-loggers==1.0.4  
SQLAlchemy==2.0.6  
srsly==2.4.6  
thinc==8.1.9  
tqdm==4.65.0  
typer==0.7.0  
typing_extensions==4.5.0  
urllib3==1.26.15  
wasabi==1.1.1  
Werkzeug==2.2.3  
yarl==1.9.2  
zipp==3.15.0
```

### 3-Base de datos.

- Verifica que tengas SQLite, que generalmente ya viene instalado con Python.
- Crea una nueva base de datos. Para hacer esto, puedes utilizar la línea de código siguiente:  
`engine = create_engine('sqlite:///database/usuarios.db')`, que crea una nueva base de datos en un archivo llamado `usuarios.db` en una carpeta llamada `database` en tu proyecto.
- Crea las tablas necesarias para la base de datos. En tu caso, las tablas ya están definidas en `models.py` y están vinculadas a la base de datos con la siguiente línea de código: `Base = declarative_base()`
- Configura la sesión. Para hacer esto, puedes utilizar la línea de código siguiente: `Session = sessionmaker(bind=engine)`, que crea una sesión para conectarse a la base de datos a través del motor que has creado.

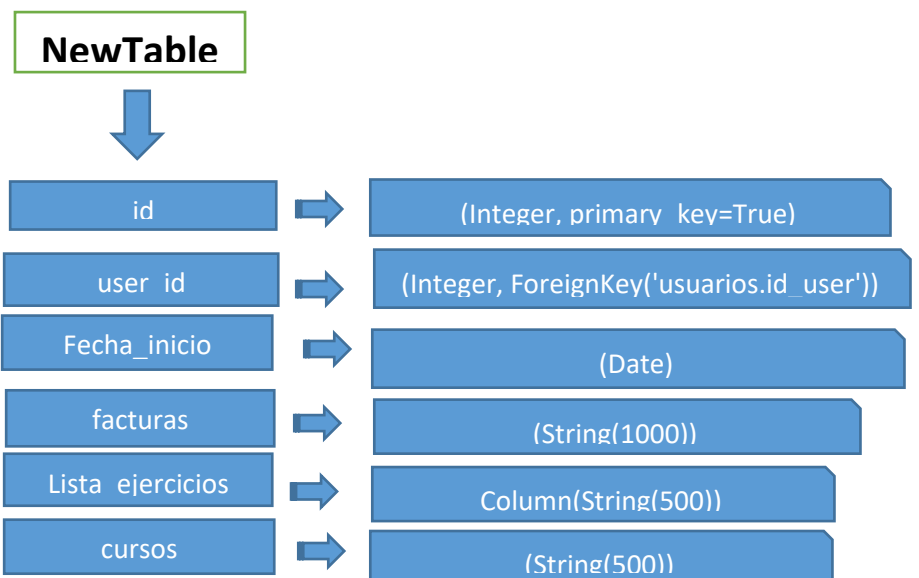
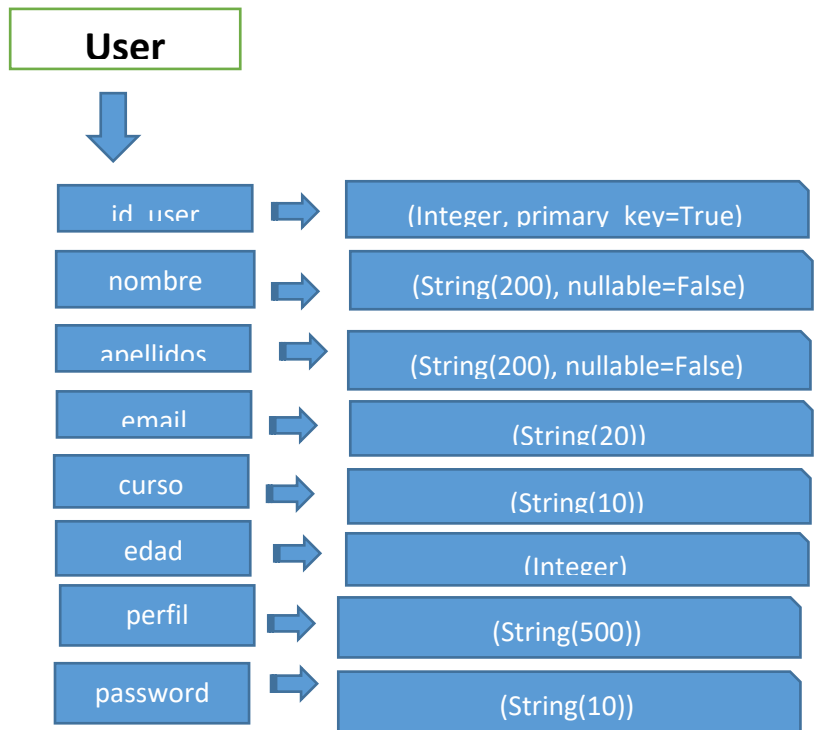
Mostramos ahora diagrama de las dos clases creadas para SQLite:

**usuarios** y **datos** teniendo una conexión entre ambas con **user\_id**.

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com



En el proyecto y siguiendo con lo aprendido durante el curso he utilizado de SQLAlchemy que permite interactuar con bases de datos SQLite utilizando ORM

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

- ORM (Object-Relational Mapping) es una técnica de programación que permite mapear las relaciones entre objetos de una aplicación y las tablas de una base de datos relacional.
- SQLAlchemy es una biblioteca de Python que proporciona una ORM y un conjunto de herramientas de bajo nivel para trabajar con bases de datos relacionales, incluyendo SQLite. SQLAlchemy es muy flexible y se puede utilizar para trabajar con diferentes bases de datos, no solo con SQLite.

## 4-GitHub.

Este proyecto lo podemos subir a *GitHub* que es una plataforma creada para que los programadores y desarrolladores de software puedan publicar sus proyectos en código abierto.

Primero debes registrarte y luego crear un repositorio.

- Haz clic en el botón "+" en la esquina superior derecha de la pantalla y selecciona "Nuevo repositorio".
- Ingresa un nombre para tu repositorio. El nombre debe ser único y puede incluir letras, números y guiones.
- Agrega una descripción opcional para tu repositorio.
- Decide si quieres que el repositorio sea público o privado. Los repositorios públicos son visibles para cualquier persona en internet, mientras que los repositorios privados solo son visibles para los miembros del equipo.
- Selecciona la opción "Agregar un archivo README" para crear un archivo README.md en tu repositorio. Este archivo se usará para proporcionar información sobre tu proyecto.
- Haz clic en "Crear repositorio".

Luego para ir actualizando incluyendo archivos y creando commit tenemos varios comandos que explico a continuación. También importante es la creación de un archivo que se llama **.gitignore** donde en el cual pondremos que archivos no queremos subir al repositorio para evitar enseñar o exponer contraseñas de la aplicación en un repositorio público.

- **git init**: Inicializa un repositorio Git en un directorio local.
- **git clone 'url'**: Clona un repositorio existente desde una URL remota a tu directorio local.
- **git add 'nombre\_archivo']**: Agrega un archivo al área de preparación de Git para incluirlo en el siguiente commit. También podemos poner `git add .` lo que incluirá todo lo que esté en esa carpeta

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

- **git commit -m "nombre de la actualización"**: Crea un nuevo commit con los cambios agregados al área de preparación. El mensaje describe los cambios realizados en el commit.
- **git push**: Sube los cambios realizados en el repositorio local al repositorio remoto.
- **git pull**: Descarga los cambios realizados en el repositorio remoto y los fusiona con el repositorio local.
- **git status**: Muestra el estado actual del repositorio, incluyendo los archivos modificados, agregados y eliminados.
- **git log**: Muestra una lista de todos los commits realizados en el repositorio.

## 5-Flask.

Para configurar la variable de entorno **FLASK\_APP** en tu sistema operativo, sigue estos pasos:

- Abre una terminal y navega hasta el directorio raíz de tu proyecto.
- Exporta la variable de entorno con el siguiente comando: **export FLASK\_APP=main.py** (Nota: asegúrate de reemplazar **main.py** con el nombre del archivo que contiene tu aplicación Flask).
- Verifica que la variable de entorno se ha configurado correctamente ejecutando el comando **echo \$FLASK\_APP**.

Para iniciar el servidor web de Flask, asegúrate de estar en el directorio raíz del proyecto y ejecuta el siguiente comando en la terminal: **flask run**. Esto iniciará el servidor en el puerto 5000 de forma predeterminada.

Para acceder a la aplicación desde un navegador web, abre tu navegador y navega a la dirección **http://localhost:5000**. Si has configurado el servidor en un puerto diferente, reemplaza **5000** con el número de puerto que has especificado. Esto debería mostrar la página de inicio de tu aplicación Flask.

Para trabajar con Flask lo primero es su importación:

```
from flask import Flask
```

A continuación debemos crear una instancia de la clase Flask

```
app = Flask(__name__)
```

Al final para ejecutar tu aplicación Flask, debes llamar al método **run()**

```
if __name__ == '__main__':
```

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

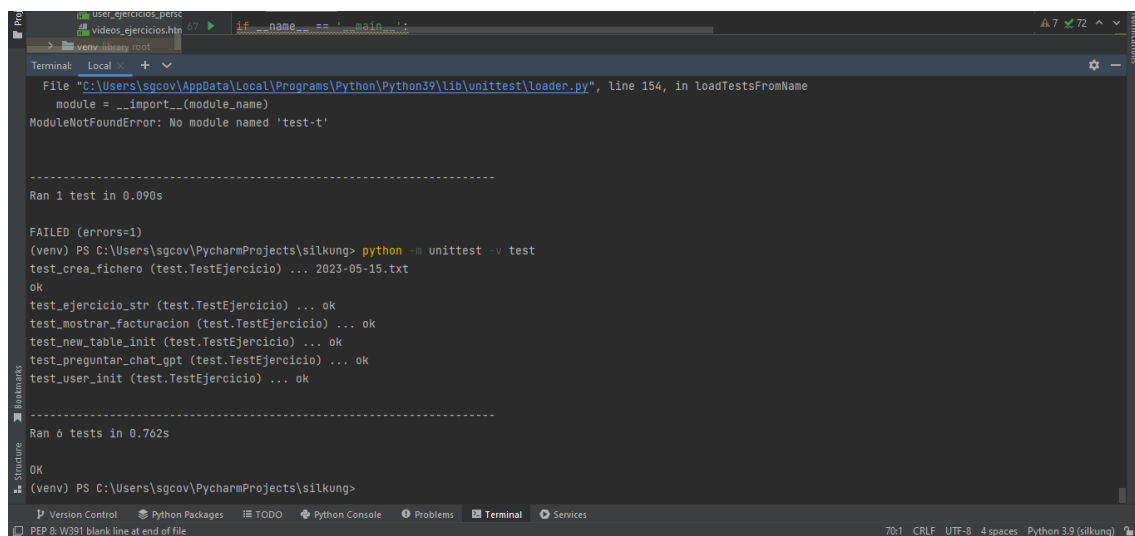
Email: javierfiestasbotella@gmail.com

```
app.run(debug=True)
```

## 6-Testing.

Para testear utilizamos unittest una de las librerías para poder realizar pruebas unitarias. La importancia de **unittest** es que permite crear pruebas unitarias para asegurarse de que el código funciona correctamente. Es un módulo integrado de Python que proporciona una estructura para escribir pruebas y ejecutarlas. Permite verificar que una determinada función o método produce el resultado esperado y asegurarse de que cualquier cambio realizado en el código no afecte su funcionamiento.

En el archivo se realizan pruebas unitarias en diferentes aspectos del código, como la creación de objetos, la lectura de archivos, la respuesta de una API, etc. Estas pruebas son importantes para detectar errores y asegurarse de que el código se comporta como se espera. Además, las pruebas unitarias son útiles para garantizar la calidad del código y para facilitar su mantenimiento y evolución en el futuro.



```
File "C:\Users\sgcov\AppData\Local\Programs\Python\Python39\lib\unittest\loader.py", line 154, in loadTestsFromName
  module = __import__(module_name)
ModuleNotFoundError: No module named 'test-t'

-----
Ran 1 test in 0.090s

FAILED (errors=1)
(venv) PS C:\Users\sgcov\PycharmProjects\silkung> python -m unittest -v test
test_crea_fichero (test.TestEjercicio) ... 2023-05-15.txt
OK
test_ejercicio_str (test.TestEjercicio) ... OK
test_mostrar_facturacion (test.TestEjercicio) ... OK
test_new_table_init (test.TestEjercicio) ... OK
test_preguntar_chat_gpt (test.TestEjercicio) ... OK
test_user_init (test.TestEjercicio) ... OK
-----
Ran 6 tests in 0.762s

OK
(venv) PS C:\Users\sgcov\PycharmProjects\silkung>
```

## 7-Desarrollo del Proyecto Silkung.

Silkung es una página web orientada al Qigong (Chikung) y Pilates.

Se ofrecen servicios de clases de Qigong para diferentes edades y dificultad al igual que de Pilates, ofreciendo especializaciones en embarazadas y niños.

La web dispone también de una BBDD para guardar información sobre los alumnos al igual que un login donde cada alumno puede logearse para



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

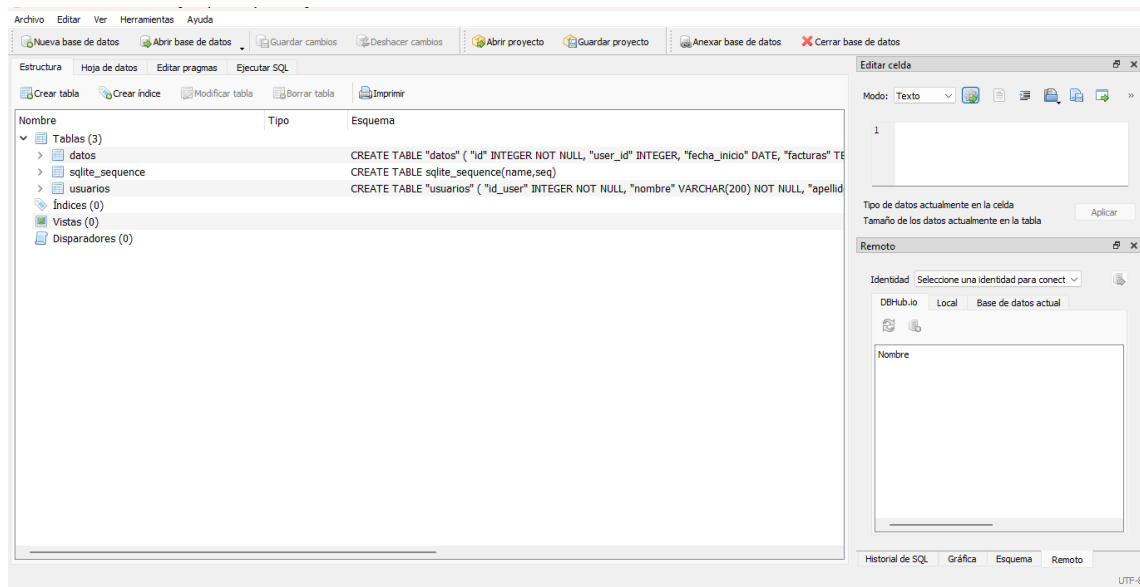
Email: javierfiestasbotella@gmail.com

poder ver ejercicios recomendados de la monitora y poder consultarlos al igual que hacer consultas a la instructora.

La Instructora también tiene su propio sitio del administrador, donde podrá adjudicar ejercicios a los alumnos, poder ver los alumnos matriculados de los dos cursos o filtrar sólo los alumnos del qigong o los alumnos de pilates. También puede dar de alta a nuevos alumnos y dar los de baja. Puede crear comunicados vía email.

## BBDD

La base datos creada en Sqlite, dispone de dos tablas enlazadas por el id del usuario.

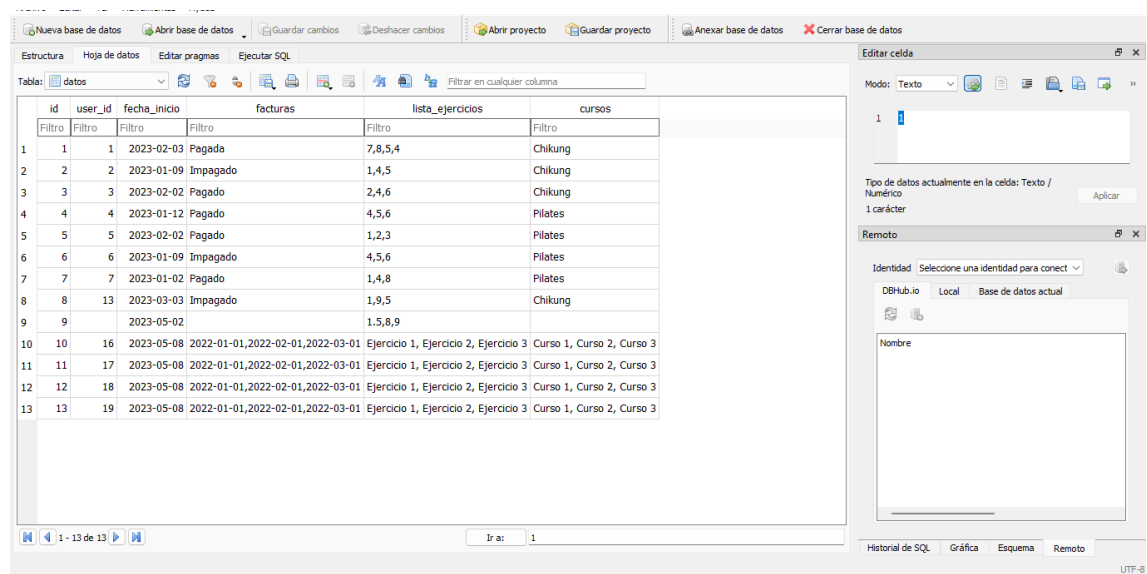
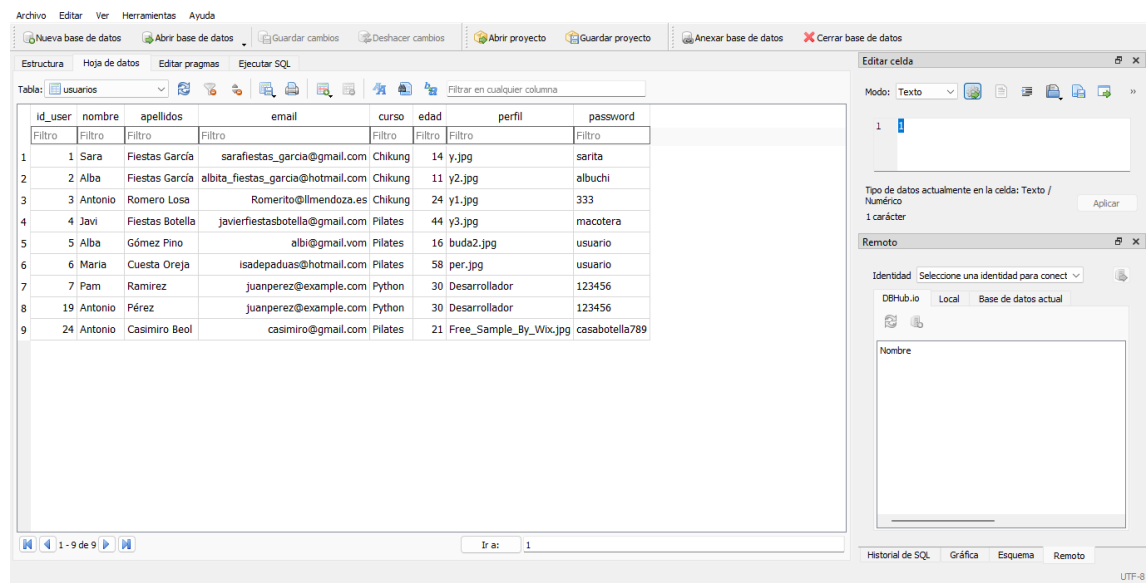


Las tablas son ***usuarios*** y ***datos***.

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com



## Galería de imágenes

Para la galería de imágenes que están repartidas en 3 html dedicados a Qigong, Pilates y Para niños, junto a un html de Videos.

Para las imágenes tras búsqueda en internet de distintas galerías de imágenes quise optar por una con algún efecto pero utilizando solo css y html y encontré un buen tutorial que utilizaba una técnica con target.

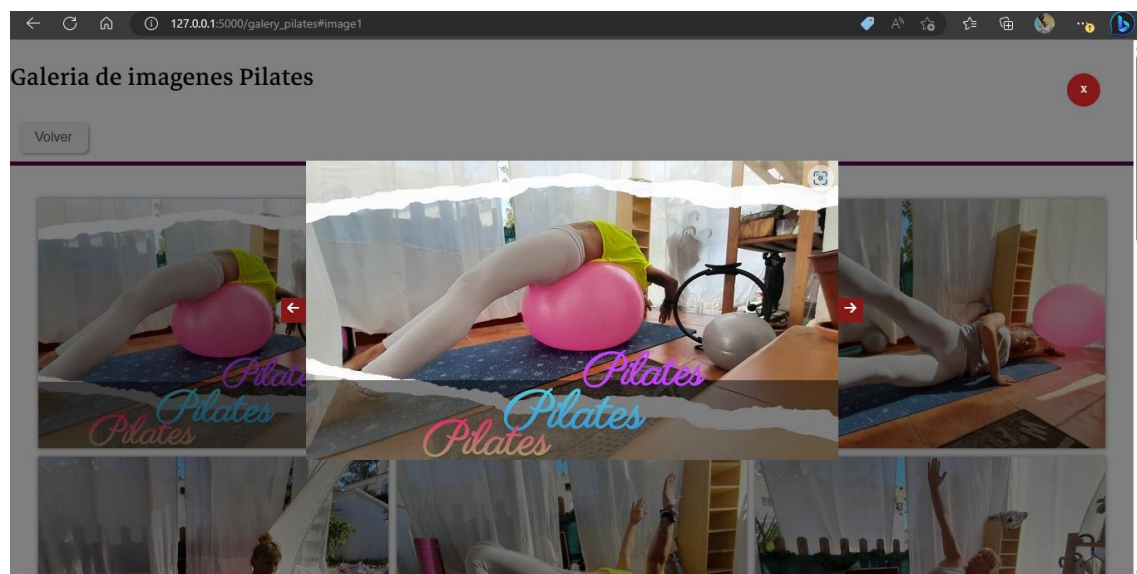
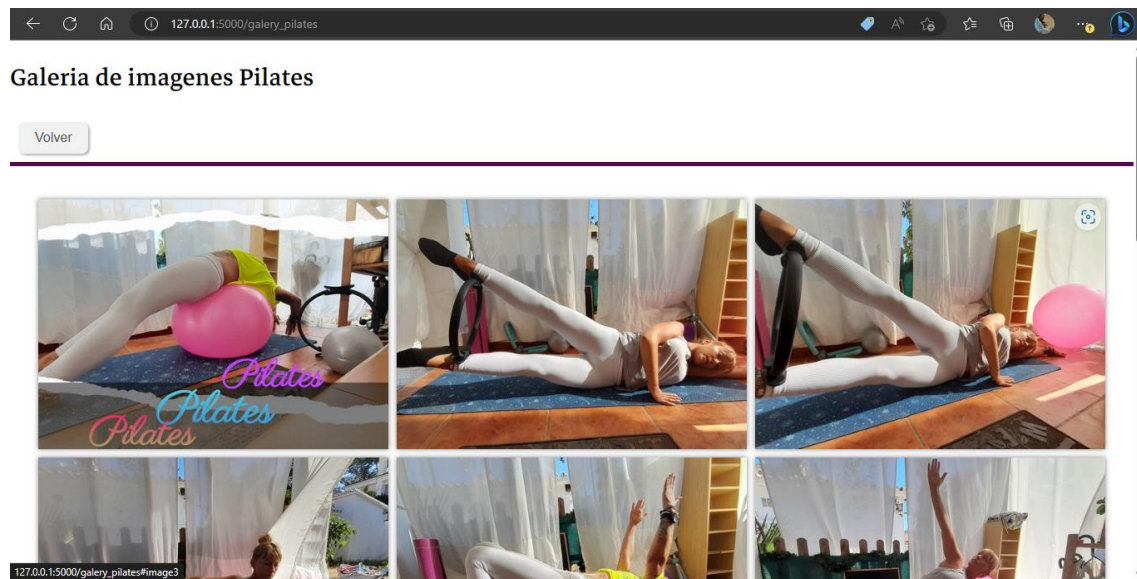
Los efectos con **:target** y **lightbox** son utilizados para crear una ventana emergente que se superpone al contenido principal del sitio web, para mostrar una imagen de manera más destacada.

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

La técnica de **:target** se utiliza para crear un enlace que, al hacer clic en él, abre una ventana emergente en la página web. El **:target** se aplica a un elemento de la página que tiene un ID específico, y se utiliza para aplicar estilos CSS a ese elemento sólo cuando es el objetivo del enlace. Cuando la conseguí implementarla al ver la metodología casi me echo atrás simplemente por la incomodidad del ir metiendo más imágenes, por la arquitectura que lleva.



Los videos lo he organizado como una galería con enlaces a videos organizados con su ancho y altura que cada video utiliza la etiqueta **<source>** para proporcionar la URL del archivo de video en formato MP4. Si el navegador no es compatible con la etiqueta **<video>**, se muestra un mensaje alternativo.

Proyecto: Silkung

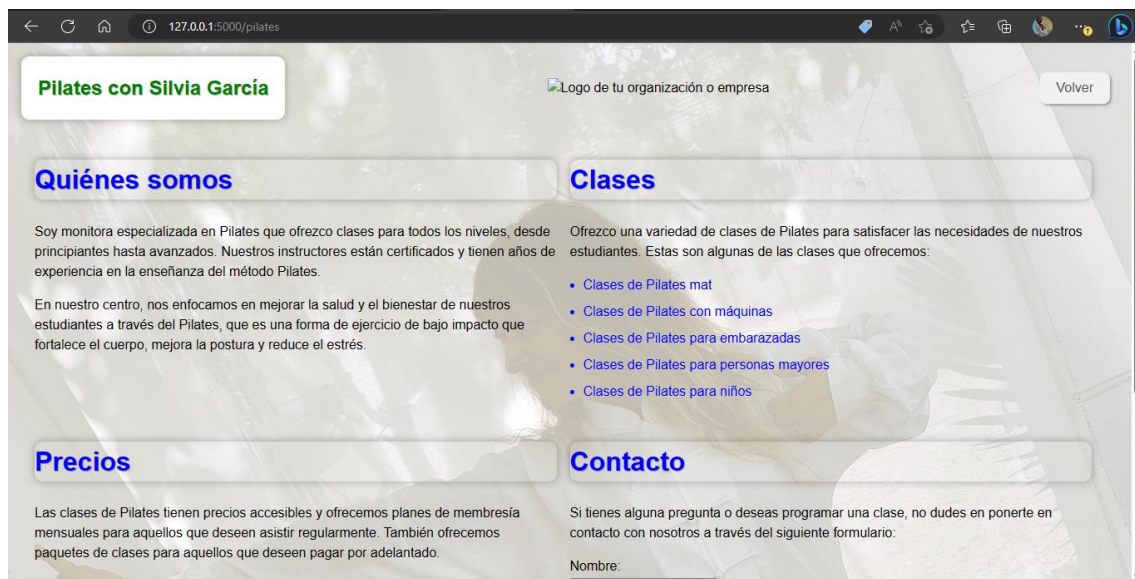
Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com



## Información

En el menú desplegable en información podemos ver que nos podemos ir a información sobre qigong, pilates, precios y horarios, llevándonos a otro html cada uno con su información y su hoja de estilos propia.



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

4. Concentración: Concéntrate en la respiración y en la sensación de energía que fluye a través de tu cuerpo.

2. Qigong para la salud

5. Relajación: Termina la práctica con unos minutos de relajación, respirando profundamente y liberando cualquier tensión en el cuerpo y la mente.

Contacto

Si tienes alguna pregunta o deseas programar una clase, no dudes en ponerte en contacto con nosotros a través del siguiente formulario:

Nombre:

Email:

Asunto:

Mensaje:

Enviar

Borrar

Horarios de Qigong y Pilates

Volver

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
7:00 - 8:00	Pilates	Qigong	Pilates	Qigong	Pilates	Qigong	Descanso
8:00 - 9:00	Descanso	Descanso	Descanso	Descanso	Descanso	Descanso	Descanso
9:00 - 10:00	Pilates	Qigong	Pilates	Qigong	Pilates	Qigong	Descanso
10:00 - 11:00	Descanso	Descanso	Descanso	Descanso	Descanso	Descanso	Descanso
11:00 - 12:00	Pilates	Qigong	Pilates	Qigong	Pilates	Qigong	Pilates
12:00 - 13:00	Pilates	Qigong	Pilates	Qigong	Pilates	Qigong	Qigong
13:00 - 14:00	Qigong	Pilates	Qigong	Pilates	Qigong	Pilates	Pilates
14:00 - 15:00	Pilates	Qigong	Pilates	Qigong	Pilates	Qigong	Qigong
15:00 - 16:00	Qigong	Pilates	Qigong	Pilates	Qigong	Pilates	Pilates
16:00 - 17:00	Pilates	Qigong	Pilates	Qigong	Pilates	Qigong	Qigong

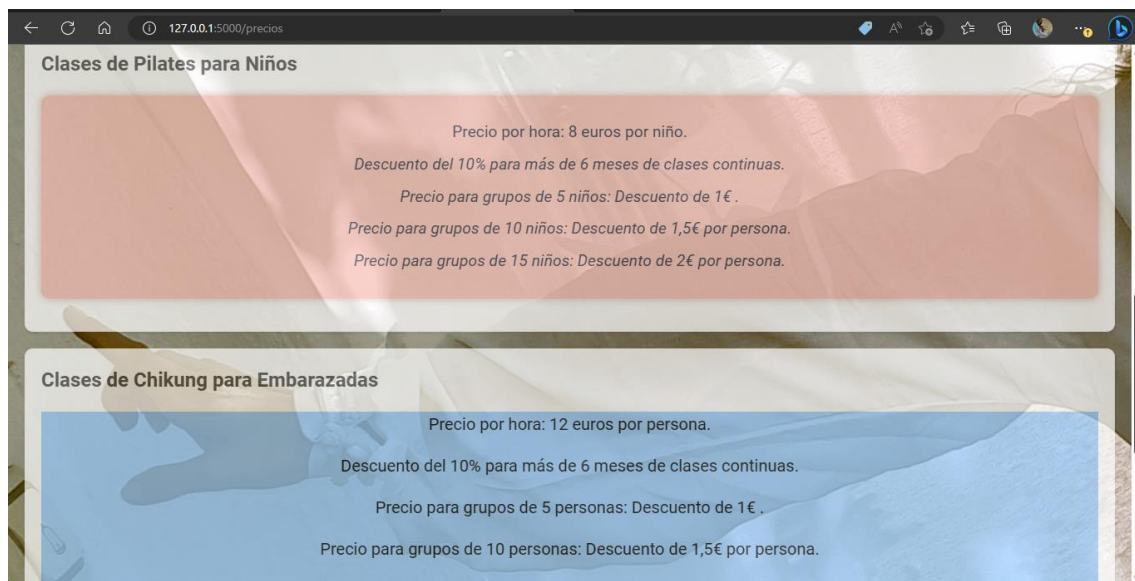
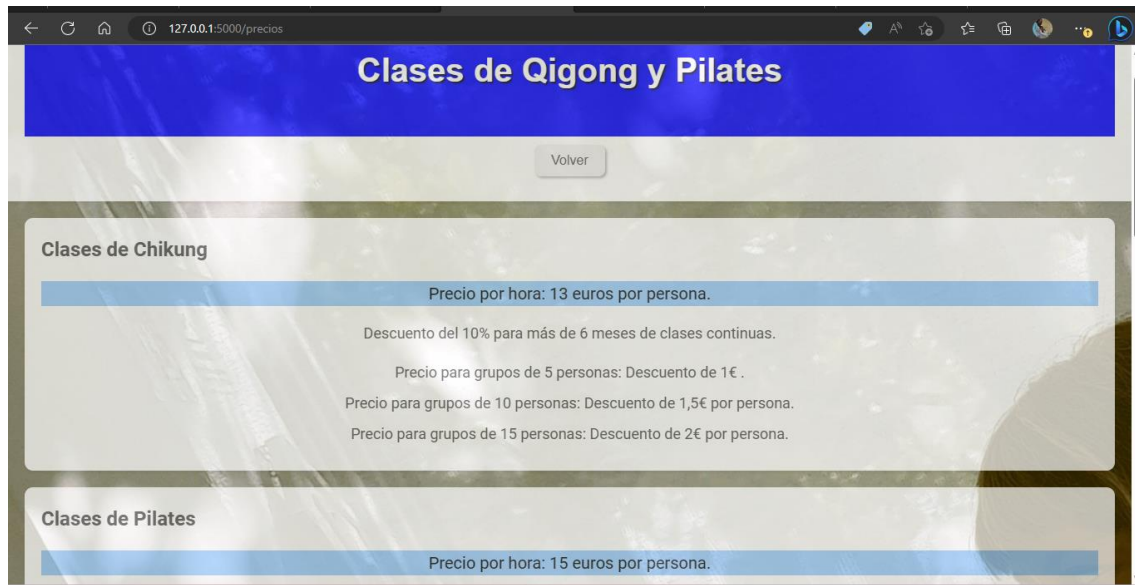
Los viernes a primera hora de la mañana, disfruta de una clase de meditación con el amanecer.



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com



### Menú desplegable

En éste menú que es uno de los muchos que enseñan por internet y gracias a mis primeros pasos en la programación cuando empecé con Html5 pude practicar un poco con javascript y opte por un código fácil para el menú:

```
$(document).ready(function(){
  $('.menu-toggle').click(function(){
    $('nav').toggleClass('active')
  })
})
```

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

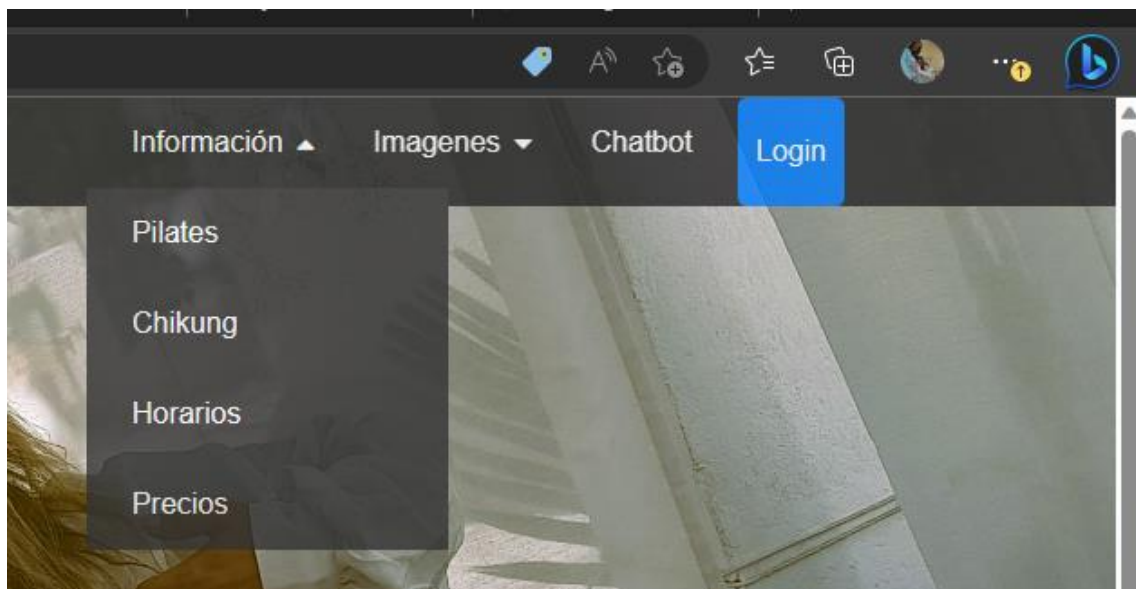
Email: javierfiestasbotella@gmail.com

```
$('.ul li').click(function(){  
    $(this).siblings().removeClass('active');  
    $(this).toggleClass('active');  
})  
})
```

El código se envuelve en `$(document).ready(function() { ... })`, lo que significa que se ejecutará una vez que se haya cargado completamente el documento HTML.

En la primera línea del código, `$('.menu-toggle').click(function() { ... })` selecciona el elemento del DOM con la clase "menu-toggle" y agrega un controlador de eventos clic que se ejecutará cada vez que se hace clic en él. Dentro de la función, `$('.nav').toggleClass('active')` selecciona el elemento de navegación (una etiqueta `<nav>` en este caso) y agrega o quita la clase "active", dependiendo de si ya la tiene o no.

La siguiente sección, `$('.ul li').click(function() { ... })`, agrega un controlador de eventos clic a cada elemento `<li>` dentro de una lista no ordenada (`<ul>`). Dentro de la función, `$(this).siblings().removeClass('active')` selecciona todos los elementos hermanos del elemento clickeado y les quita la clase "active". Después, `$(this).toggleClass('active')` agrega o quita la clase "active" al elemento clickeado.



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

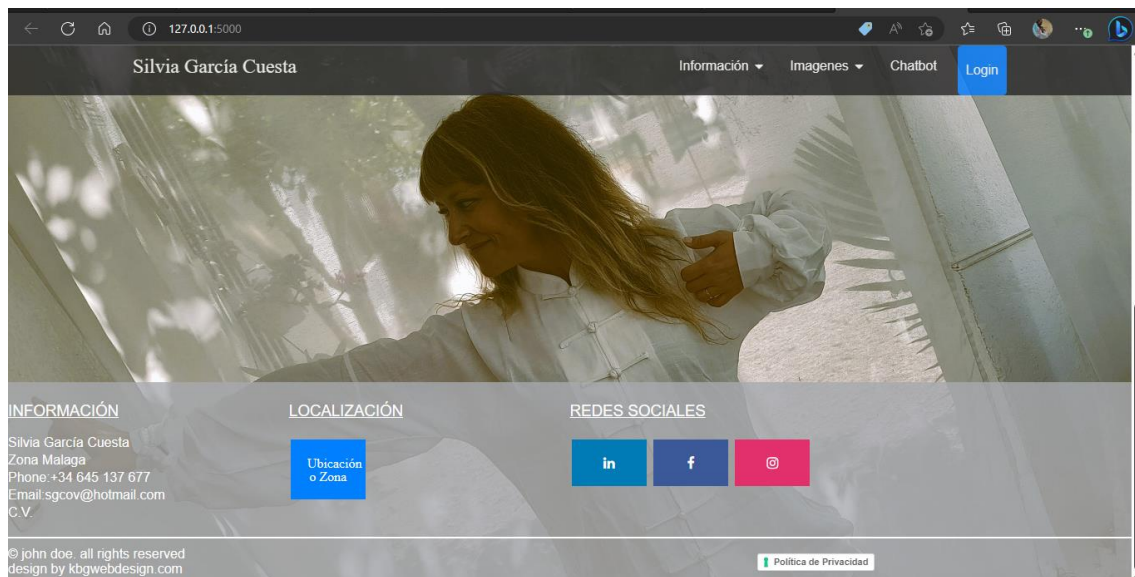
Email: javierfiestasbotella@gmail.com

## Footer

La etiqueta **<footer>** indica que se trata del pie de página. Dentro del pie de página, hay tres divisiones diferentes (**<div>**), cada una con su propia información:

1. **footer-information-container**: contiene información de contacto para la empresa o persona que posee el sitio web, incluyendo el nombre, la ubicación, el teléfono y el correo electrónico.
2. **our-location-container**: incluye un enlace para mostrar la ubicación o zona de la empresa o persona que posee el sitio web en Google Maps.
3. **social-media-container**: contiene enlaces a las cuentas de redes sociales de la empresa o persona que posee el sitio web, incluyendo LinkedIn, Facebook e Instagram.

En la sección inferior del pie de página, hay una división adicional (**footer-developer-container**) que incluye información sobre la empresa que desarrolló el sitio web (**footer-developed-by**) y un enlace a su sitio web. También hay un enlace para la política de privacidad del sitio web, que se carga desde el servicio de terceros de iubenda.



## ChatBot (IA)

Debido al fenómeno GPT y la IA estuve leyendo sobre la implementación de GPT con python para aplicaciones o webs, estuve viendo videos y leyendo documentación y quise implementar un chatbot a la web tras un pequeño curso realizado con Udemy para la implementación de openai.



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

```
import os
from dotenv import load_dotenv
import openai
import spacy

load_dotenv()

#llamamos a la clave que está en el archivo .env
api_key = os.getenv('OPENAI_API_KEY')

#ahora le decimos que es nuestra clave la que tiene que utilizar
openai.api_key = api_key

preguntas_anteriores = []
respuestas_anteriores = []

def preguntar_chat_gpt(prompt, modelo="text-davinci-002"):
    respuesta = openai.Completion.create(
        engine=modelo,
        prompt=prompt,
        n=1,
        temperature=0.7,
        max_tokens=200
    )
    return respuesta.choices[0].text.strip()

def iniciar_chatbot():
    print('Bienvenido a nuestro chatbot basico; escribe salir si quieres terminar')
    while True:
        conversacion_historica= ""
        ingreso_usuario = input('\nTu:')
        if ingreso_usuario.lower() == 'salir':
            break

        for pregunta, respuesta in zip(preguntas_anteriores, respuestas_anteriores):
            conversacion_historica += f'El usuario pregunta: {pregunta}/n'
            conversacion_historica += f'Buda Silkung responde: {respuesta}/n'

        prompt = f'El usuario pregunta: {ingreso_usuario}\n'
        conversacion_historica += prompt
        respuesta_gpt = preguntar_chat_gpt(conversacion_historica)
        print(f'{respuesta_gpt}')

        preguntas_anteriores.append(ingreso_usuario)
        respuestas_anteriores.append(respuesta_gpt)

if __name__ == "__main__":
    iniciar_chatbot()
```

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

Explicación paso a paso:

- **import os** - importa el módulo **os** para interactuar con el sistema operativo.
- **from dotenv import load\_dotenv** - importa la función **load\_dotenv** del módulo **dotenv**, que carga variables de entorno desde un archivo **.env**.
- **import openai** - importa el módulo **openai**, que proporciona acceso a la API de OpenAI.
- **import spacy** - importa el módulo **spacy**, que se utiliza para el procesamiento del lenguaje natural.
- **load\_dotenv()** - carga las variables de entorno desde el archivo **.env**.
- **api\_key = os.getenv('OPENAI\_API\_KEY')** - asigna a **api\_key** el valor de la variable de entorno **OPENAI\_API\_KEY**.
- **openai.api\_key = api\_key** - establece la clave de API de OpenAI para el valor de **api\_key**.
- **preguntas\_anteriores = []** - inicializa una lista vacía llamada **preguntas\_anteriores**.
- **respuestas\_anteriores = []** - inicializa una lista vacía llamada **respuestas\_anteriores**.
- **def preguntar\_chat\_gpt(prompt, modelo="text-davinci-002"):**  
- define una función llamada **preguntar\_chat\_gpt** que toma dos argumentos, **prompt** y **modelo**, y devuelve una respuesta generada por el modelo de OpenAI.
- **respuesta = openai.Completion.create(** - crea una solicitud a la API de OpenAI para generar una respuesta en base al modelo, la temperatura y el número de tokens especificados.
- **return respuesta.choices[0].text.strip()** - devuelve la respuesta generada por el modelo.
- **def iniciar\_chatbot():** - define una función llamada **iniciar\_chatbot**.
- **print('Bienvenido a nuestro chatbot basico; escribe salir si quieres terminar')** - muestra un mensaje de bienvenida al usuario.
- **while True:** - crea un bucle infinito que se ejecutará hasta que el usuario ingrese la palabra "salir".
- **conversacion\_historica= ""** - inicializa una cadena de texto vacía llamada **conversacion\_historica**.
- **ingreso\_usuario = input('\nTu:')** - pide al usuario que ingrese una pregunta o comentario y asigna su respuesta a la variable **ingreso\_usuario**.
- **if ingreso\_usuario.lower() == 'salir':** - verifica si el usuario ha ingresado la palabra "salir" y, si es así, sale del bucle while.
- **for pregunta, respuesta in zip(preguntas\_anteriores, respuestas\_anteriores):** - itera sobre las listas **preguntas\_anteriores** y **respuestas\_anteriores** al mismo tiempo.

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

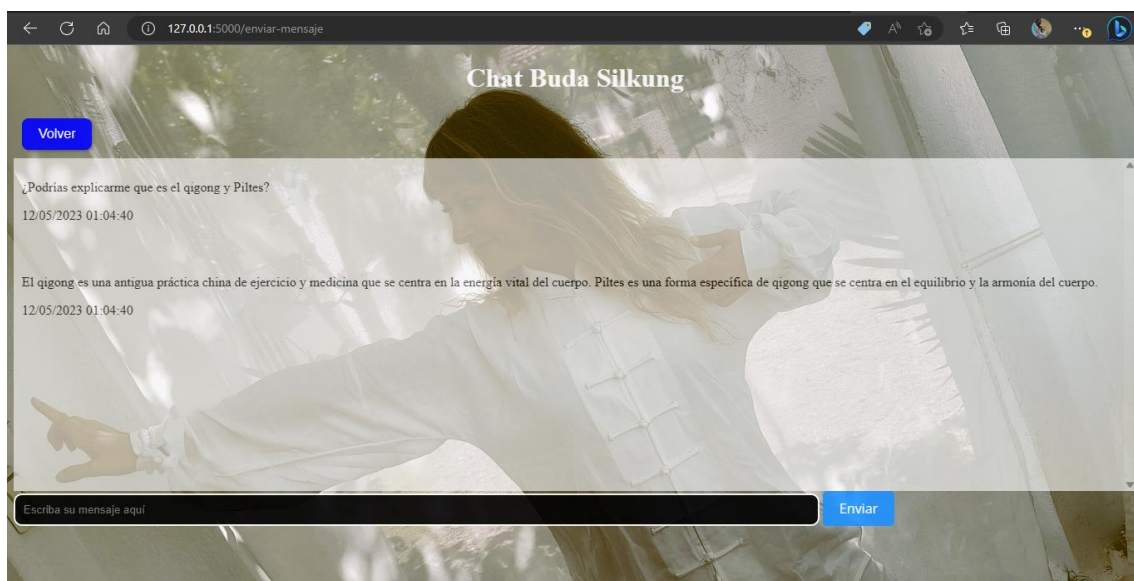
- **conversacion\_historica += f'El usuario pregunta: {pregunta}/n'** - agrega la pregunta anterior del usuario a la cadena de **conversacion\_historica**.
- **conversacion\_historica += f'Buda Silkung responde: {respuesta}/n'** - agrega la respuesta anterior del chatbot a la cadena de **conversacion\_historica**.
- Luego, definimos una función llamada **preguntar\_chat\_gpt(prompt, modelo="text-davinci-002")**. Esta función utiliza la API de OpenAI para generar una respuesta a partir de una pregunta, utilizando un modelo de lenguaje pre-entrenado. La función recibe dos parámetros: **prompt**, que es la pregunta que se desea hacer al modelo, y **modelo**, que es el modelo de lenguaje que se utilizará para generar la respuesta (por defecto, utilizamos el modelo "text-davinci-002").
- Dentro de la función **preguntar\_chat\_gpt**, utilizamos el método **openai.Completion.create** para generar una respuesta a partir de la pregunta. Este método recibe varios parámetros, como el nombre del modelo a utilizar (**engine**), la pregunta en sí (**prompt**), la cantidad de respuestas a generar (**n**), la "temperatura" de generación de respuestas (**temperature**) y el número máximo de tokens que puede tener la respuesta (**max\_tokens**).
- La respuesta generada por la API de OpenAI es una lista de objetos, pero como solo estamos pidiendo una respuesta (**n=1**), tomamos el primer elemento de la lista utilizando **respuesta.choices[0]**. Luego, utilizamos el atributo **text** de este objeto para obtener la respuesta en sí, y eliminamos los espacios en blanco al principio y al final de la respuesta utilizando el método **strip()**. Finalmente, la función devuelve la respuesta generada.
- Después, definimos otra función llamada **iniciar\_chatbot()**. Esta función es la que se encarga de interactuar con el usuario y de hacer uso de la función **preguntar\_chat\_gpt** para generar respuestas a partir de las preguntas del usuario.
- Dentro de la función **iniciar\_chatbot**, primero imprimimos un mensaje de bienvenida y un mensaje indicando cómo salir del chatbot.
- Luego, iniciamos un bucle **while** que se ejecutará hasta que el usuario ingrese la palabra "salir".
- Dentro del bucle, creamos una variable llamada **conversacion\_historica** que contendrá todas las preguntas y respuestas previas, separadas por el carácter "/n".
- Luego, le pedimos al usuario que ingrese una pregunta. Si el usuario ingresa la palabra "salir", salimos del bucle con la instrucción **break**.
- A continuación, recorremos las preguntas y respuestas previas utilizando la función **zip**, y vamos agregando cada pregunta y respuesta a la variable **conversacion\_historica** en el formato correspondiente.

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

- Luego, creamos una variable llamada **prompt** que contiene la pregunta actual del usuario en el formato correspondiente. Agregamos esta pregunta a la variable **conversacion\_historica**.
- Utilizamos la función **preguntar\_chat\_gpt** para generar una respuesta a partir de la conversación histórica.
- Imprimimos la respuesta generada por el modelo.
- Finalmente, agregamos la pregunta actual del usuario y la respuesta generada por el modelo a las listas **preguntas\_anteriores** y **respuestas\_anteriores**, respectivamente.



### Archivo .env

El archivo **.env** es utilizado para almacenar variables de entorno en la aplicación, de forma que puedan ser utilizadas por el código sin necesidad de ser expuestas públicamente. En este caso, las variables almacenadas son:

- **OPENAI\_API\_KEY**: es una clave de autenticación utilizada para acceder a la API de OpenAI. El valor de esta clave está siendo cargado en la variable **api\_key** en el código.
- **SECRET\_KEY**: es una cadena de texto que es utilizada como una clave secreta en la aplicación.
- **SMTP\_PASSWORD**: Contraseña generada por Gmail para el envío de mensajes

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

## Login

En el login de la web puedes acceder con los datos de un alumno que esté matriculado o también puedes acceder como administrador. Si accedes como administrador:

En la sección principal del administrador se muestra una lista de los alumnos matriculados con información como el nombre, apellidos, curso, número de alumno, correo electrónico y los ejercicios que han realizado. Además, se incluye una imagen de perfil del alumno y un enlace para editar sus datos. También hay botones que permiten filtrar la lista de alumnos por curso y un botón para volver a la página de inicio.

Si entramos como un alumno:

Contiene un encabezado con una barra de navegación que incluye opciones para editar el perfil, facturación, ejercicios y contacto.

La página también muestra una imagen de perfil del usuario, así como una lista de datos del perfil que incluyen su nombre, apellidos, correo electrónico, curso y edad.

Además, la página incluye un formulario de contacto que se puede mostrar y ocultar haciendo clic en un botón. El formulario recopila el nombre, correo electrónico, asunto y mensaje del usuario para que puedan enviar un correo electrónico al propietario de la página.

El funcionamiento de éste botón se hace mediante javascript en la siguiente función:

```
var botonMostrarFormulario = document.getElementById("mostrar-formulario");
var formularioContacto = document.getElementById("contacto");

botonMostrarFormulario.addEventListener("click", function(evento) {
    evento.preventDefault();
    formularioContacto.classList.toggle("oculto");
});
```

Este código de JavaScript utiliza el método **addEventListener()** para asignar un controlador de eventos al botón con el id "mostrar-formulario". El controlador de eventos es una función anónima que se ejecutará cuando se haga clic en el botón. Dentro de la función, se utiliza el método **preventDefault()** para evitar que el formulario se envíe automáticamente

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

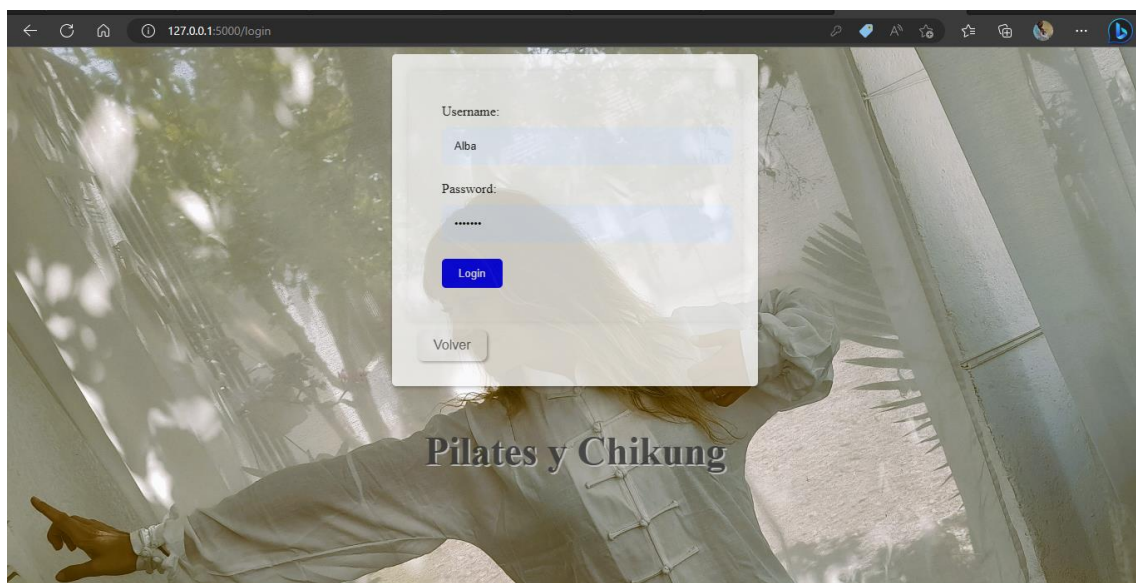
Email: javierfiestasbotella@gmail.com

al hacer clic en el botón. Luego se utiliza el método **classList.toggle()** para cambiar la clase CSS del formulario con el id "contacto" de "visible" a "oculto" (o viceversa) cada vez que se hace clic en el botón.

Paso a paso, esto es lo que sucede cuando se hace clic en el botón:

- Se obtiene el elemento del botón "mostrar-formulario" y se asigna a la variable **botonMostrarFormulario**.
- Se obtiene el elemento del formulario "contacto" y se asigna a la variable **formularioContacto**.
- Se llama al método **addEventListener()** del botón "mostrar-formulario" y se le pasa como primer argumento el evento "click" y como segundo argumento una función anónima.
- Cuando se hace clic en el botón, se ejecuta la función anónima, que recibe como parámetro el objeto evento **evento**.
- Se llama al método **preventDefault()** en el objeto evento **evento** para evitar que el formulario se envíe automáticamente al hacer clic en el botón.
- Se llama al método **classList.toggle()** en el formulario "contacto" con el id "contacto" para cambiar su clase CSS de "visible" a "oculto" (o viceversa), lo que muestra u oculta el formulario según su estado anterior.

En la parte inferior de la página, hay un mensaje de bienvenida y una breve descripción de la monitora que se especializa en Qigong y Pilates.



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

← ↻ 🏠 ⓘ 127.0.0.1:5000/alumnos

Lista Alumnos Matriculados

Edición ▾ Envío

Registrar Alumno


Editar ejercicios

Dar de baja Alumno

Página Inicio

Alumnos de Qigong lt

Alumnos de Pilates lt



**Sara ----> N° Alumno:1**

Fiestas García

Curso: Chikung

[sarafiestas\\_garcia@gmail.com](mailto:sarafiestas_garcia@gmail.com)

Presione para actualizar datos

Fecha Inicio: 2023-02-03

Facturas: Pagada

[7.8.5.4](#)

Alba ----> N° Alumno:2

← ↻ 🏠 ⓘ 127.0.0.1:5000/user

☐ Editar Perfil


☐ Facturación

☐ Ejercicios

☐ Contacto

Salir sesión

**Perfil de Alba Fiestas García**



☐ Nombre: Alba

☐ Apellidos: Fiestas García

☐ Email: albita\_fiestas\_garcia@hotmail.com

☐ Curso: Chikung

☐ Edad: 11

**¡Bienvenido a Qigong y Pilates!**

Soy monitora especializada en la práctica de Qigong y Pilates. Mi misión es ayudar a los alumnos a alcanzar un estado óptimo de salud y bienestar.



Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

Registro datos de alumno

Edite sus nuevos datos y presione Registrar

**Id del usuario:**

Id del alumno

**Fecha de inicio:**

15/05/2023

**Última Factura:**

Pagada/Impagada

**Lista Ejercicios:**

Ejercicios separados por ','

**Curso:**

Cursos en trámite

Registrar

Registro datos de alumno Nuevo

Edite sus nuevos datos y presione Registrar

**Nombre:**

**Apellidos:**

**Email:**

administradora\_garcia\_cuesta

**Curso:**

**Edad:**

**IMG Perfil:**

Perfil de usuario

Registrar

errores\_archivos.py

Crea archivo en caso de que no exista, o agregar contenido al archivo si ya existe, para registrar errores en una carpeta llamada "static/errores" con el nombre del archivo que corresponde a la fecha actual.



Proyecto: Silkung

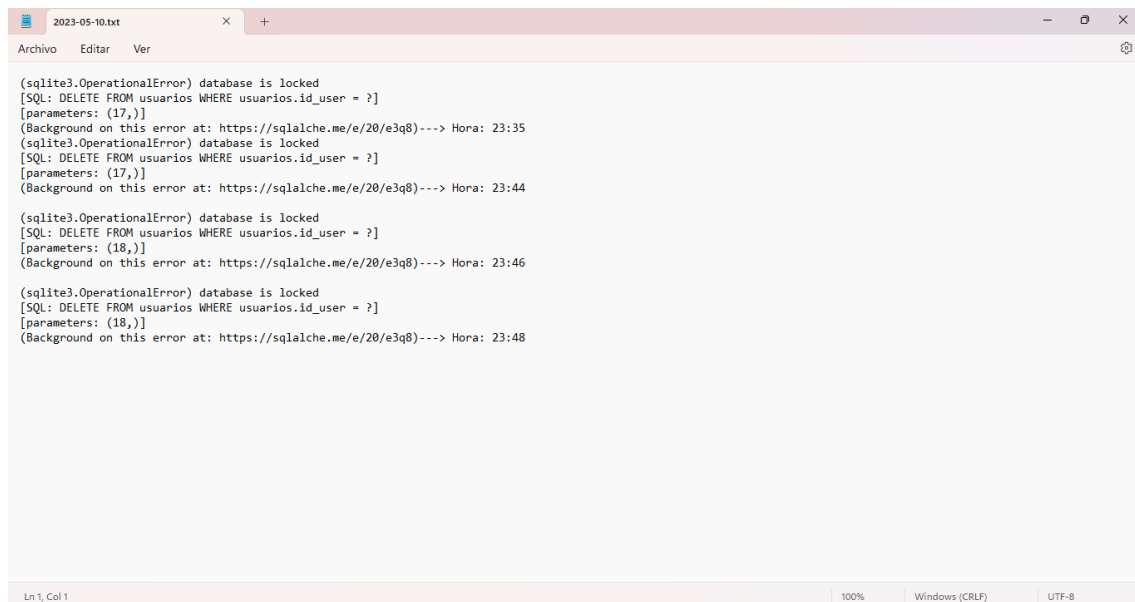
Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

Se importa el módulo `os` para verificar si el archivo existe y la clase `datetime` para obtener la fecha y hora actual. También se utiliza `pathlib` para obtener la ruta del archivo en caso que se cambie la ruta.

La función `crea_fichero` toma un argumento `name` que se utilizará para registrar el error en el archivo. Primero, se obtiene la fecha actual y la hora actual, y luego se verifica si el archivo ya existe en la carpeta "static/errores" utilizando el método `os.path.exists`. Si existe, se abre el archivo en modo de "agregar" y se escribe el error. Si no existe, se crea un nuevo archivo con el nombre de la fecha actual y se escribe el error.

Todo esto se hace no el motivo de ver cada periodo de tiempo que errores son más comunes para su corrección u optimización del código.



The screenshot shows a text editor window titled "2023-05-10.txt" with a menu bar (Archivo, Editar, Ver) and a toolbar. The text content consists of three identical log entries, each representing a database error. Each entry starts with "(sqlite3.OperationalError) database is locked", followed by the SQL query "[SQL: DELETE FROM usuarios WHERE usuarios.id\_user = ?]", the parameters "[parameters: (17,)]", and a background message "(Background on this error at: https://sqlalche.me/e/20/e3q8)---> Hora: 23:35" (with times 23:35, 23:44, and 23:46 respectively). The status bar at the bottom indicates "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

```
(sqlite3.OperationalError) database is locked
[SQL: DELETE FROM usuarios WHERE usuarios.id_user = ?]
[parameters: (17,)]
(Background on this error at: https://sqlalche.me/e/20/e3q8)---> Hora: 23:35

(sqlite3.OperationalError) database is locked
[SQL: DELETE FROM usuarios WHERE usuarios.id_user = ?]
[parameters: (17,)]
(Background on this error at: https://sqlalche.me/e/20/e3q8)---> Hora: 23:44

(sqlite3.OperationalError) database is locked
[SQL: DELETE FROM usuarios WHERE usuarios.id_user = ?]
[parameters: (18,)]
(Background on this error at: https://sqlalche.me/e/20/e3q8)---> Hora: 23:46

(sqlite3.OperationalError) database is locked
[SQL: DELETE FROM usuarios WHERE usuarios.id_user = ?]
[parameters: (18,)]
(Background on this error at: https://sqlalche.me/e/20/e3q8)---> Hora: 23:48
```

## 8-Opinion Personal.

Bueno éste proyecto decidí hacerlo porque mi mujer ha concluido unos cursos como monitora de Pilates y de Qigong y está empezando con clases.

Hoy en día las redes sociales y tener una web propia es imprescindible para potenciar cualquier proyecto y negocio.

He tenido varias dificultades a la hora de compaginar las rutas cuando enviamos con POST o GET luego al final se va comprendiendo mucho mejor.

Proyecto: Silkung

Alumno: Fº Javier Fiestas Botella

Email: javierfiestasbotella@gmail.com

Con respecto a los CSS he optado por darle una hoja de estilos por html compartiendo el nombre para hacerlo más legible a la hora de leerlo y comprenderlo. Quise introducir un chat-bot por la novedad de gpt, aprendí como implementarlo en Python y quise hacer un guiño, pero reconozco que un chat-bot es mejor para otro tipo de web o aplicaciones.

Me gusta mucho ver códigos de compañeros y funcionalidades diferentes para el tema de blog, funcionalidades de menú, menús desplegables, galerías de imágenes, y realmente se aprende mogollón.

Nada más empezar con la programación empecé aprendiendo Html5, aprendiendo también JavaScript, hice una web para mi restaurante con el que me di cuenta de muchísimas cosas a la hora de crearla, luego contratar un hostinguer que me ayudó muchísimo a entender cómo funcionan y la importancia de las BBDD. Ahora al hacerlo con Python y Flask, ha sido una manera diferente pero por la cual te das cuenta de la iteración de Python y el potencial que tiene pero si he notado que mis pequeños conocimientos de html y css me han venido genial para poder llegar a comprender mucho mejor.

Ha sido una muy buena experiencia e interesante proyecto que me ha abierto curiosidades a otros posibles proyectos. Muy importante y le he dado gran valor a la arquitectura en los proyectos, ya que tan importante es el naming como la organización de carpetas como la arquitectura en general no solo para el mejor funcionamiento si no también la una buena legibilidad y comprensión del código.

Seguir adelante con mi pasión por la programación y continuar aprendiendo y creciendo es lo que me planteo como meta.