Taller de Investigación y Desarrollo Web con Flask, Python, HTML, CSS y Bases de Datos

Proyecto Pagina web Heladería Annia

Integrantes:

Ana Sofia Rodriguez Alvarez

Nicolk Anelca Diaz Hernandez

Maria Isabela Figueroa Imbacuán

Annjely Velez Solis

Instructor Mairon Antonio Salazar

Instructora Katherine Giraldo

Ficha 3192191

Módulo 1 – Introducción e investigación inicial

1. Qué es Flask y por qué se usa en el desarrollo web?

Flask es un framework para el desarrollo de proyectos web en Python. Se califica dentro del segmento de los «microframework», ya que tiene un enfoque minimalista y está orientado para un tipo de trabajos más específicos como servicios web y microservicios, en vez del renderizado de HTML como Django.

De todos modos, que sea un «microframework» no significa que Flask esté limitado a proyectos pequeños o que no tenga suficientes herramientas. En realidad, significa que no impone ciertas capas y se centra solamente en resolver las solicitudes HTTP de una manera ágil y sencilla**.**

Utilización en el desarrollo web.
Con Flask el desarrollador tiene una gran libertad a la hora de elegir las herramientas que mejor le convengan para cada proyecto, dentro de lo que ofrece el ecosistema de Python, que es enorme.
Es un framework minimalista, esto quiere decir que es sencillo de aprender y de aplicar a los proyectos. A la vez, lo hace muy ligero y especialmente indicado para aplicaciones pequeñas o medianas, donde estas características son más apreciadas.

Flask está basado en WSGI, que es un estándar de Python para el desarrollo de aplicaciones web (siglas de Web Server Gateway Interface). Eso hace que la comunidad de Python se sienta muy cómoda al usarlo, ya que su forma de trabajar le resultará familiar si viene de otros proyectos.

Además, usa el motor de plantillas Jinja2, que permite generar HTML dinámico renderizado en el lado del servidor. Aunque a decir verdad este no es el uso más frecuente de Flask, ya que la mayoría de los proyectos desarrollados con este framework son servicios web que devuelven JSON como respuesta.

2. Plantillas HTML y CSS (ejemplo: Bootstrap, Tailwind, Bulma).

Bootstrap es un framework front-end gratuito y de código abierto que proporciona a los desarrolladores web herramientas y componentes (HTML, CSS y JavaScript) para crear sitios web y aplicaciones web de forma rápida y sencilla. Su principal ventaja es el sistema de diseño responsivo, que permite que los sitios se adapten automáticamente a cualquier tamaño de pantalla, desde móviles hasta ordenadores.

Tailwind CSS es un framework CSS de bajo nivel y alta personalización que funciona con un enfoque "utility-first". En lugar de proporcionar estilos predefinidos para componentes (como

botones o tarjetas), Tailwind te da un conjunto de clases de utilidad (como text-center, bg-blue 500, flex) que se aplican directamente en el marcado HTML para construir interfaces de usuario personalizadas. Esto permite diseñar rápidamente sitios web utilizando tus propios estilos, optimizando al mismo tiempo el peso del código CSS y facilitando la toma de decisiones de diseño.

Bulma es un framework CSS moderno, gratuito y de código abierto, desarrollado con Flexbox y Sass, que permite crear interfaces web responsivas de forma eficiente y rápida. Se basa en un sistema de clases para aplicar estilos, ofreciendo componentes prediseñados y modularidad para personalizar el diseño sin escribir CSS desde cero.

3. Entorno a jinja2, ¿cómo funcionan las plantillas dinámicas?

En un proyecto real, necesitas mostrar información que cambia constantemente: usuarios diferentes, productos nuevos, pedidos en tiempo real. Sin plantillas dinámicas, tendrías que crear páginas HTML separadas para cada caso, lo que sería imposible de mantener.

Jinja es un motor de plantillas moderno y fácil de usar para Python. Permite incrustar código Python directamente en archivos HTML, haciéndolos dinámicos. Eso significa que se puede lograr mostrar datos variables en tu página web.

¿Cómo Funciona?

Jinja2 actúa como tu asistente personal de HTML. Tú creas una plantilla base con "espacios vacíos" donde irá la información variable. Cuando un usuario visita tu sitio, tu aplicación toma los datos específicos (el perfil del usuario, sus pedidos, etc.) y Jinja2 automáticamente llena esos espacios, generando una página única para cada persona.

Cuando Jinja2 procesa una plantilla, lo hace mediante un análisis en dos etapas. Primero, escanea todo el texto identificando las etiquetas especiales para variables y {% %} para lógica. Luego, ejecuta las instrucciones encontradas: sustituye las variables por sus valores reales y procesa las estructuras de control como bucles y condicionales. Este proceso no modifica tu plantilla original, sino que genera un nuevo archivo HTML completamente renderizado.

4. Informe inicial: “Arquitectura básica de una aplicación web con Flask”.

```python
from flask import Flask, render_template, request
import sqlite3
app = Flask(__name__)
# funcion para iniciar nuestras bases de datos
Tabnine | Edit | Test | Explain | Document
def init_db():
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS contactos (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 nombre TEXT NOT NULL,
                 email TEXT NOT NULL,
                 telefono TEXT NOT NULL,
                 edad INTEGER NOT NULL,
                 mensaje TEXT NOT NULL)''')
    conn.commit()
    conn.close()

# funcion para insertar datos en la base de datos
Tabnine | Edit | Test | Explain | Document
def insert_data(nombre, email, telefono, edad, mensaje):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute('INSERT INTO contactos (nombre, email, telefono, edad, mensaje) VALUES (?, ?, ?, ?, ?)', (nombre, email,
    telefono, edad, mensaje))
    conn.commit()
    conn.close()
Tabnine | Edit | Test | Explain | Document
@app.route('/', methods=['GET', 'POST'])
```

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        nombre = request.form['nombre']
        email = request.form['email']
        telefono = request.form['telefono']
        edad = request.form['edad']
        mensaje = request.form['mensaje']
        insert_data(nombre, email, telefono, edad, mensaje)
        return 'Formulario enviado con exito!'
    return render_template('index.html')

if __name__ == '__main__':
    init_db()
    app.run(debug=True)
```

5. Investigación de entornos virtuales (venv, pipenv) y por qué son importantes.

¿Qué es un Entorno virtual?

Es un espacio aislado donde instalas librerías y dependencias solo para tu proyecto, sin afectar el Python global de tu sistema. Esto evita conflictos entre proyectos, además, genera políticas de seguridad.

El problema real que resuelven

Cuando trabajas en múltiples proyectos de Python, cada uno puede necesitar versiones diferentes de las mismas bibliotecas. Sin entornos virtuales, instalarías todo en el mismo lugar, creando conflictos.

Ejemplo: Si el Proyecto A necesita Flask 1.0 y el Proyecto B necesita Flask 2.0, sin entornos virtuales instalarías una versión y el otro proyecto se rompería**.**

Cómo funcionan los entornos virtuales:

1.Creación del espacio aislado
Al crear un entorno virtual, generas una carpeta independiente que contiene su propia versión de Python y sus propias bibliotecas instaladas.

2.Activacion del entorno
Cuando activas el entorno virtual, le dices a tu sistema: "Ahora trabaja solo con las herramientas de este proyecto específico".

3. Instalación de bibliotecas
Las bibliotecas que instalas quedan confinadas al entorno virtual actual, sin afectar otros proyectos.

4.Desativación
Al terminar, desactivas el entorno y vuelves al sistema general.

Venv - El Método Básico (Incluido en Python)

Paso 1: Crear el Entorno

```
python -m venv mi_entorno
```

Paso 2: Activar el Entorno

Windows

```
mi_entorno\Scripts\activate
```

Mac/Linux

```
source mi_entorno/bin/activate
```

Paso 3: Instalar Bibliotecas

```
pip install flask pandas
```

Paso 4: Guardar Dependencias

```
pip freeze > requirements.txt
```

Paso 5: Desactivar

```
Deactivate
```

Pipenv - El Método Automatizado

Paso 1: Instalar pipenv

```
pip install pipenv
```

Paso 2: Crear y Activar Automáticamente

```
pipenv install flask pandas
```

Paso 3: Trabajar en el Entorno

```
pipenv shell
```

Diferencias Claras Entre Ambos

venv (Como hacerlo manual)

- Tú controlas cada paso.
- Más trabajo, pero más control.
- Ideal para aprender cómo funciona.
- Viene incluido con Python.
- Usas pip para instalar bibliotecas.
- Manualmente debes crear archivos de requisitos.

pipenv (Como tener un asistente)

- Combina entorno virtual + gestión de bibliotecas.
- Automatiza los procesos.
- Menos probabilidad de errores.
- Crea automáticamente archivos de configuración.
- Mejor control de versiones.
- Más recomendado para proyectos serios.

Por Qué Son Esenciales para tu Proyecto

- ✓ Evitan Conflictos.
- ✓ Facilitan el Trabajo en Equipo.

- ✓ Permiten Reproducir el Entorno.
- ✓ Mantienen tu Sistema Limpio.
- ✓ Previenen Desastres.

Ejemplo de la Vida Real

Proyecto A: Una tienda online que requiere Flask versión 2.0 y pandas 1.5.0

Proyecto B: Un blog personal que funciona con Flask versión 1.0 y pandas 1.2.0

Sin entornos virtuales: Instalas Flask 2.0 y el blog se rompe, o instalas pandas 1.2.0 y la tienda online deja de funcionar.

Con entornos virtuales: Cada proyecto tiene su versión correcta de cada biblioteca y ambos funcionan perfectamente en el mismo computador.

Conclusión Práctica

Los entornos virtuales son como habitaciones separadas para cada proyecto: mantienen el orden, evitan problemas y hacen tu trabajo más profesional. Son esenciales para cualquier desarrollo serio en Python.

Módulo 2 – Estructura y plantillas del proyecto

1. ¿Cómo organizar un proyecto Flask?

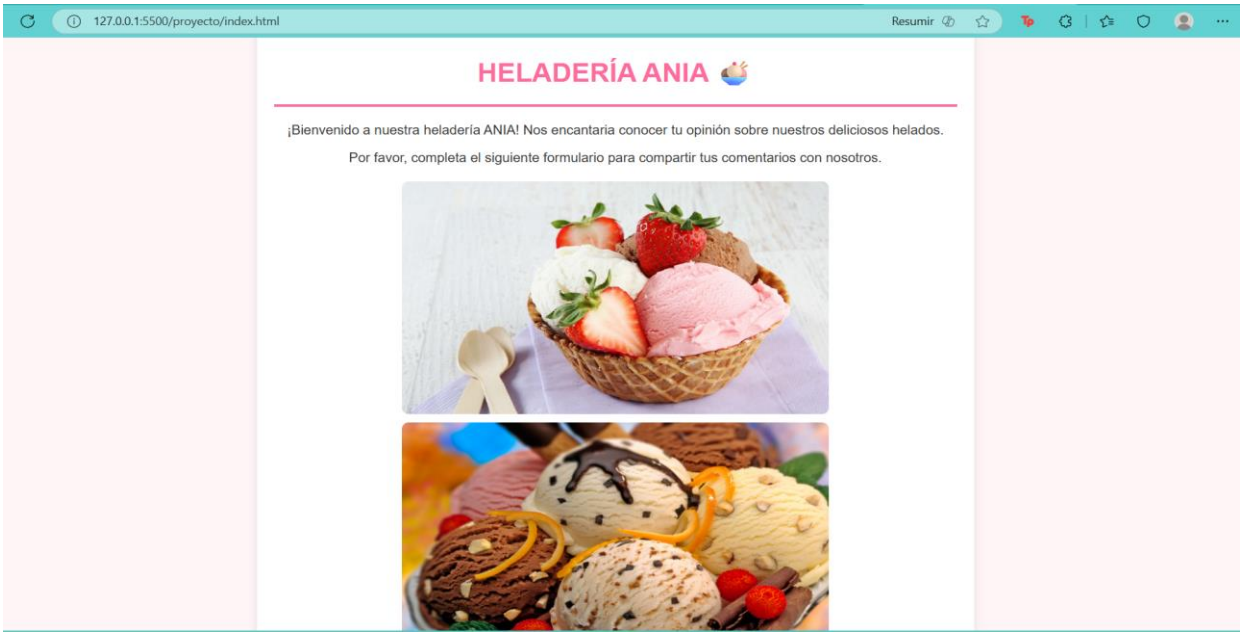
+ mi_proyecto
|_ app.py
|_ forms.py
|_ models.py
|_ static/
|_ templates/

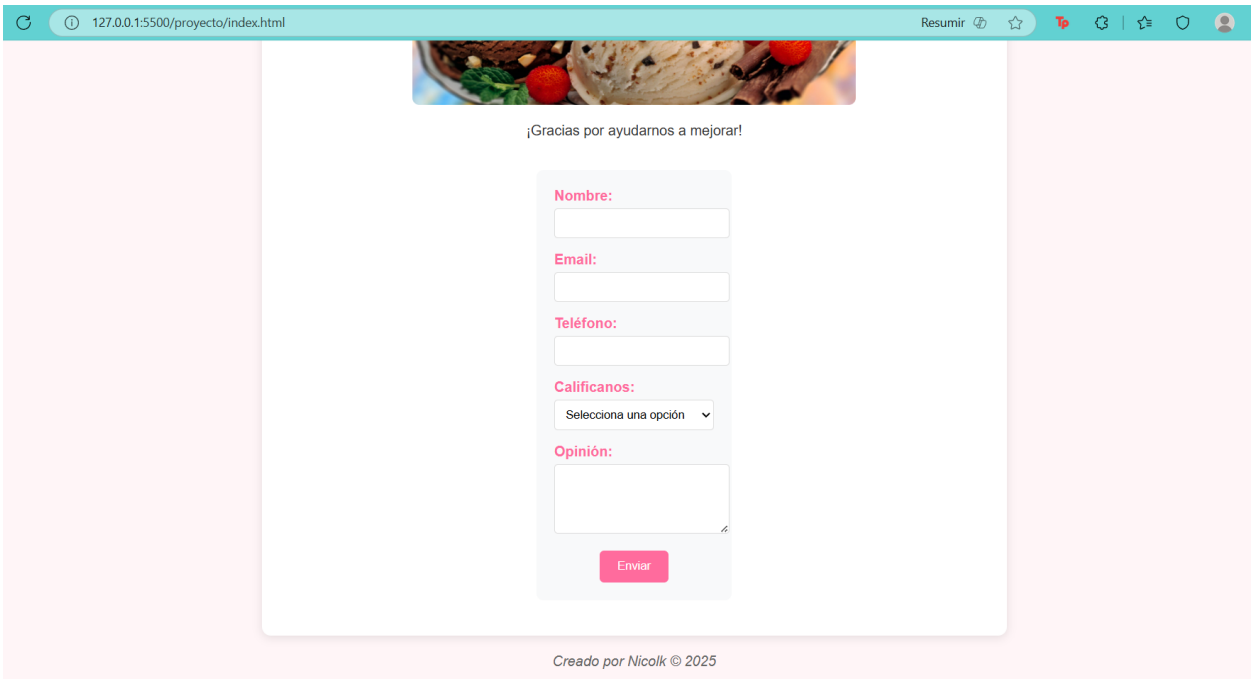
2. Selección de plantillas HTML y CSS (cuadro comparativo Bootstrap vs Tailwind)

	Bootstrap	**Tailwind**
Eficiencia	Ensamble rápidamente interfaces responsivas sin escribir CSS extenso.	Reduce la necesidad de escribir CSS personalizado, mejorando la velocidad de desarrollo.
Flexibilidad	Ideal para desarrolladores que prefieren elementos listos para usar.	Ideal para crear diseños únicos y personalizados.
Enfoque de Estilo	Proporciona componentes predefinidos y listos para	Proporciona clases de utilidad para construir componentes desde cero.

	usar (botones, cards, navbars).	
Personalización	Se basa en variables SASS para personalizar temas, requiere compilación.	Altamente personalizable mediante el archivo de configuración
Diseño Único	Puede ser un desafío salirse del "look" estándar de Bootstrap sin esfuerzo adicional.	Fomenta y facilita la creación de diseños completamente únicos y a medida.
Curva de Aprendizaje	Baja; fácil de empezar con componentes conocidos.	Moderada; requiere aprender las clases de utilidad y el enfoque "utility-first".
Integración con Frameworks JS	Se integra bien, pero los componentes pueden requerir wrappers (ej: React-Bootstrap).	Se integra de forma nativa y excelente con frameworks como React, Vue, Svelte, etc.

3. Ejemplo de un diseño simple con HTML y CSS elegido.





4. Investigación sobre buenas prácticas de diseño web (usabilidad, accesibilidad).

Diseño Claro y Sencillo: Mantén la interfaz simple y sin desorden para facilitar la interacción. Te dejo enlace al artículo en el cual encontrarás una reflexión sobre interfaces web sencillas.

Navegación Intuitiva: Utiliza una estructura de navegación clara y coherente en todo el sitio web. Te dejo enlace sobre menús de navegación.

Legibilidad: Usa fuentes legibles y contraste suficiente entre el texto y el fondo. Te dejo enlace al artículo sobre elegir tipografías web.

Responsive Design: Asegúrate de que tu sitio web sea fácilmente accesible y usable en diferentes dispositivos y tamaños de pantalla. Enlace al artículo Diseño web responsive; ejemplos y buenas prácticas.

Botones y Enlaces Claros: Haz que los botones y enlaces sean reconocibles y descriptivos, que digan qué hacen. Te dejo un enlace para optimizar los botones y enlaces de tu web.

Prevención y Corrección de Errores: Haz todas las pruebas necesarias para detectar errores y anticipar posibles problemas al usuario. Añade mensajes de error claros y opciones e instrucciones para corregirlos fácilmente.

Proporcionar Ayuda y Documentación: Habilita la posibilidad al usuario de acceder a soporte en línea o guías. Lo puedes hacer con un simple formulario web para que el usuario tenga fácil contactar con el responsable de la web. Te dejo enlace al artículo sobre hacer formularios web sencillos.

Privacidad y Seguridad: Informa claramente a los usuarios sobre las políticas de privacidad y asegura que los datos sean manejados de manera segura.

Módulo 3: Conexión con bases de datos

1.Investigación sobre SQLite vs MySQL: ventajas y desventajas

MySQL	
Ventajas	**Desventajas**

SQLite	
Ventajas	**Desventajas**
Beneficios de usar SQLite Es una biblioteca liviana que se puede integrar en aplicaciones sin problemas, lo que la hace ideal para aplicaciones móviles o de IoT con recursos limitados. SQLite admite transacciones, lo que garantiza la integridad y la consistencia de los datos. Permite a los desarrolladores trabajar sin un servidor de base de datos separado y elimina la necesidad de configuraciones complejas. SQLite es de código abierto y de uso gratuito, lo que lo hace accesible para desarrolladores de todos los niveles.	Las principales desventajas de SQLite son su incapacidad para manejar un alto volumen de usuarios concurrentes, su rendimiento decreciente con grandes volúmenes de datos, la falta de características de seguridad y administración de usuarios, y su funcionalidad limitada en operaciones de escritura y consultas complejas. Además, no está diseñado para ser accedido a través de una red, limitando su uso a aplicaciones locales o de un solo usuario.

Fácil de usar y configurar. Gran comunidad de usuarios y desarrolladores. Excelente rendimiento en operaciones de lectura. Soporte para transacciones y claves foráneas. Gestión eficiente de tablas múltiples Compatibilidad con diversos sistemas de hosting Capacidad para manejar diferentes cargas de trabajo Soporte para múltiples consultas simultáneas Sistema de funciones personalizable Escalabilidad para diferentes tipos de clientes	Limitaciones de seguridad: MySQL puede tener algunas limitaciones de seguridad, especialmente si no se configura adecuadamente. Deberás asegurarte de que la base de datos esté bien protegida para evitar vulnerabilidades. Varias de las utilidades de MySQL no están documentadas. No es del todo intuitivo, en comparación con otros programas. No maneja de manera tan eficiente una base de datos con un tamaño muy grande.

2.Investigación sobre librerías en Python para conexión con BD: sqlite3, SQLAlchemy, PyMySQL.

sqlite3 es una librería estándar incluida con Python que permite trabajar con bases de datos SQLite. Este tipo de base de datos es liviana, no requiere servidor y guarda la información en un archivo local. Es ideal para proyectos pequeños, pruebas rápidas o aplicaciones que no necesitan múltiples usuarios accediendo al mismo tiempo.

La principal ventaja de sqlite3 es su simplicidad: no requiere instalación adicional y se puede comenzar a usar con pocas líneas de código. Además, ofrece soporte para transacciones y es compatible con la API estándar para bases de datos en Python (DB-API 2.0). Sin embargo, no es recomendable para aplicaciones que necesiten escalar o manejar mucha concurrencia, ya que SQLite no está diseñado para eso.

En resumen, sqlite3 es una excelente opción para desarrollos simples, prototipos o almacenamiento local de datos. No requiere configuración de servidor y permite operar con bases de datos directamente desde el código Python.

SQLAlchemy es una librería muy poderosa para trabajar con bases de datos en Python. A diferencia de sqlite3, SQLAlchemy es más que un simple conector: ofrece una capa de abstracción que permite

trabajar tanto con SQL directo (modo “Core”) como con programación orientada a objetos a través de su ORM (Object Relational Mapper).

Una de sus principales ventajas es la flexibilidad y compatibilidad con múltiples motores de bases de datos como SQLite, MySQL, PostgreSQL, entre otros. Esto permite cambiar de base de datos sin reescribir todo el código. También maneja de forma avanzada las conexiones, transacciones y relaciones entre tablas.

Aunque SQLAlchemy es muy completo, su curva de aprendizaje puede ser más alta, especialmente si se usa el ORM. Sin embargo, para proyectos medianos o grandes, es una herramienta muy recomendable por su robustez, escalabilidad y buenas prácticas que promueve.

PyMySQL es una librería que permite conectar Python con bases de datos MySQL o MariaDB. Es un conector 100% en Python, lo que facilita su instalación y portabilidad. Es compatible con la especificación DB-API 2.0, y permite ejecutar comandos SQL, manejar transacciones y obtener resultados con cursores.

Su uso es directo: se establecen los parámetros de conexión (como host, usuario y contraseña) y se pueden realizar consultas o inserciones fácilmente. También se puede usar junto con otras herramientas como SQLAlchemy, actuando como el driver de conexión a bases de datos MySQL.

La principal desventaja de PyMySQL es que solo funciona con MySQL/MariaDB, por lo que no es una solución general. Tampoco ofrece abstracciones como un ORM por sí solo. Aun así, es una excelente opción cuando se trabaja exclusivamente con estos motores y se necesita una conexión sencilla.

3.Ejemplo de cómo crear tablas y conectarlas con Flask.

```python
from flask import Flask, render_template, request
import sqlite3
app = Flask(__name__)
# funcion para iniciar nuestras bases de datos
```

Primero, se importan las librerías necesarias **(Flask, render_template, request y sqlite3**). Luego, se crea la aplicación con **app = Flask(__name__**), lo cual permite definir las rutas y el funcionamiento del servidor.

```python
Tabnine | Edit | Test | Explain | Document
def init_db():
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS contactos (
                 id INTEGER PRIMARY KEY AUTOINCREMENT,
                 nombre TEXT NOT NULL,
                 email TEXT NOT NULL,
                 telefono TEXT NOT NULL,
                 edad INTEGER NOT NULL,
                 mensaje TEXT NOT NULL)''')
    conn.commit()
    conn.close()
```

A continuación, la función **init_db()** establece la conexión con **database.db** y crea la tabla **contactos** si aún no existe. En ella se definen los campos id, nombre, email, teléfono, edad y mensaje. Esta función garantiza que la base de datos esté lista para almacenar los registros antes de iniciar la aplicación.

```python
def insert_data(nombre, email, telefono, edad, mensaje):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute('INSERT INTO contactos (nombre, email, telefono, edad, mensaje) VALUES (?, ?, ?, ?, ?)', (nombre, email,
    telefono, edad, mensaje))
    conn.commit()
    conn.close()
```

La función **insert_data** (nombre, email, teléfono, edad, mensaje) realiza la conexión con la base de datos y ejecuta un comando **INSERT INTO** para agregar los datos enviados por el usuario.
Los valores se pasan como parámetros (?, ?, ?) para prevenir inyecciones SQL.
Finalmente, se confirma la transacción con **conn.commit()** y se cierra la conexión.

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        nombre = request.form['nombre']
        email = request.form['email']
        telefono = request.form['telefono']
        edad = request.form['edad']
        mensaje = request.form['mensaje']
        insert_data(nombre, email, telefono, edad, mensaje)
        return 'Formulario enviado con exito!'
    return render_template('index.html')
```

La ruta **@app.route**('/', **methods=['GET', 'POST'])** define la página principal del sitio y permite recibir solicitudes **GET** (mostrar la página) y POST (enviar datos del formulario).

Dentro de la función **index(),** se verifica si el método de solicitud es POST. Si lo es, se obtienen los datos del formulario mediante **request.form[]** y se envían a la función **insert_data()** para ser almacenados en la base de datos.
Finalmente, se muestra un mensaje de confirmación o se renderiza el archivo index.html.

```python
if __name__ == '__main__':
    init_db()
    app.run(debug=True)
```

El bloque final **if __name__ == '__main__'**: indica que el archivo se ejecuta directamente (no como módulo importado).

Dentro de él se llama a **init_db()** para asegurar que la base de datos esté lista y luego se inicia el servidor local con **app.run(debug=True**), lo que permite visualizar la aplicación en el navegador y detectar errores en tiempo real.

4. Investigación sobre conceptos básicos de seguridad en BD (inyección SQL, contraseñas).

Inyección SQL

La inyección SQL es una vulnerabilidad de seguridad que ocurre cuando una aplicación permite que datos ingresados por el usuario se integren directamente en una consulta SQL sin validación adecuada. Esto puede permitir a un atacante modificar la consulta original para acceder, manipular o eliminar

información de la base de datos. Es una de las técnicas de ataque más conocidas y puede comprometer gravemente la integridad y confidencialidad de los datos.

Para prevenir este tipo de ataques, se recomienda el uso de consultas preparadas o parametrizadas, que separan el código SQL de los datos del usuario, evitando que estos sean interpretados como parte del comando. Además, es importante validar y sanear todas las entradas, aplicar el principio de mínimos privilegios en el acceso a la base de datos y mantener buenas prácticas de desarrollo seguro.

Gestión Segura de Contraseñas

Una práctica fundamental en la seguridad de bases de datos es el correcto almacenamiento de contraseñas. Las contraseñas nunca deben guardarse en texto plano, ya que, en caso de una filtración quedarían expuestas fácilmente. En su lugar, deben almacenarse utilizando algoritmos de hash seguro como bcrypt, Argon2 o PBKDF2, que hacen que la contraseña no se pueda revertir ni adivinar fácilmente.

También es recomendable aplicar técnicas como el salting, que consiste en añadir un valor aleatorio único antes de aplicar el hash, para evitar ataques con bases de datos de contraseñas predecibles. Adicionalmente, es buena práctica implementar políticas de contraseñas fuertes, autenticación en dos pasos (MFA) y mecanismos de bloqueo ante intentos fallidos. Todo esto ayuda a proteger tanto al usuario como a la integridad del sistema.