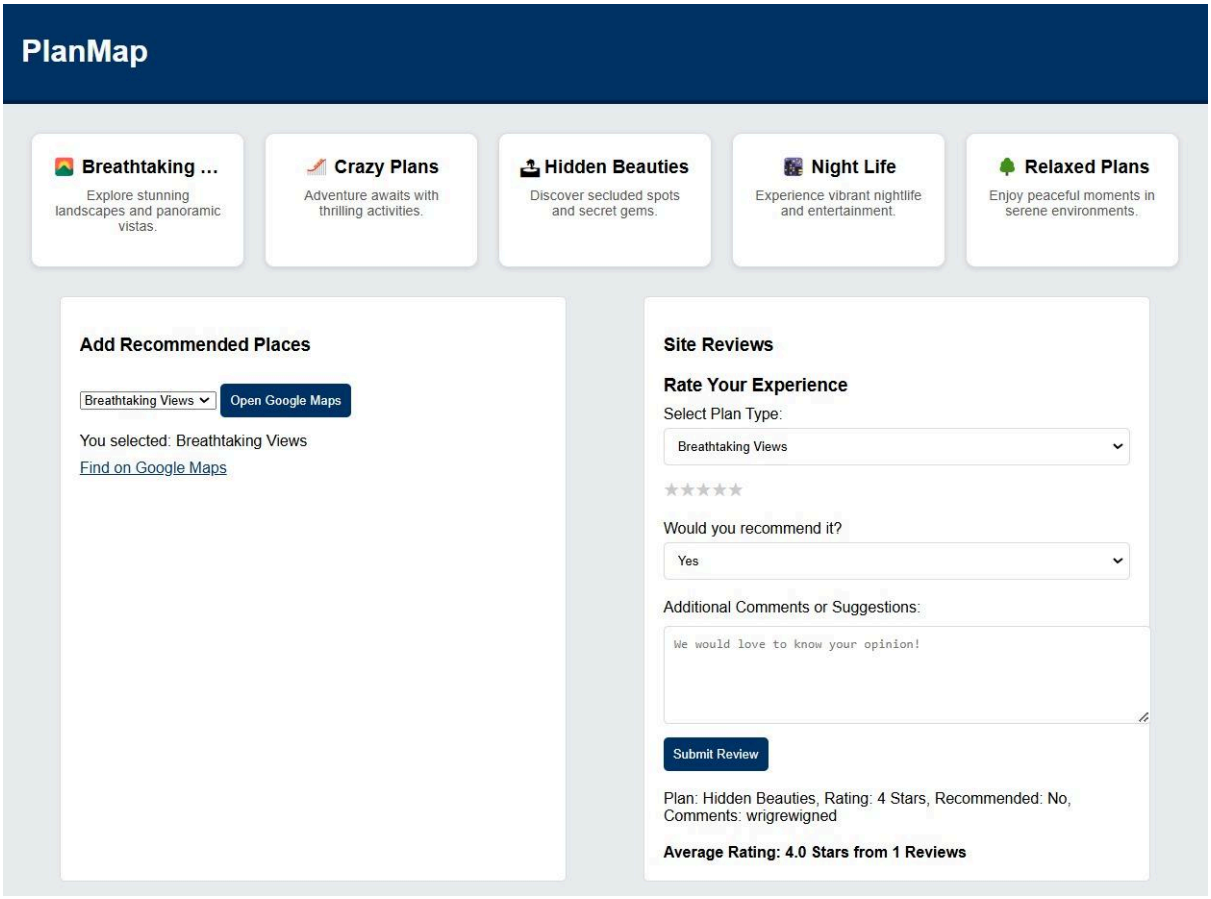
Plan Map



Trabajo realizado por: Pablo Barreiro Corral, Sabela Fiaño García, Lucas Lijó Triñanes, Bruno Vázquez Jorge

Nota: Crear en el escritorio una carpeta llamada “data” 1

AGRADECIMIENTOS:

Queremos agradecer a la organización del HACK UDC por habernos dado la oportunidad de participar en este proyecto.

Agradecemos el interés y la colaboración de nuestros mentores que aún con su propio proyecto nos guiaron para que aprendieramos a realizar nuestra propia web.

"Ley de Alzheimer de la programación: si lees un código que escribiste hace más de dos semanas es como si lo vieras por primera vez"

-- Via Dan Hurvitz

ÍNDICE:

1. INTRODUCCIÓN
    - 1.1. Motivos por los que se escogió este tema
    - 1.2. Objetivos del trabajo
    - 1.3. Competencias a adquirir
2. MATERIALES, MÉTODOS Y FASES DE TRABAJO:
    - 2.1. FASE 1: Bases para crear la web (python)
        - 2.1.1. buscar los import flask
        - 2.1.2. creación del servidor web
    - 2.2. FASE 2: creación web (html)
        - 2.2.1. introducción a visual studio code
        - 2.2.2. creación de la web
    - 2.3. FASE 3: unión.
3. CONCLUSIONES
4. PROPUESTA DE CONTINUIDAD
5. WEBGRAFÍA

1 INTRODUCCIÓN

1.1 Motivos por los que se escogió este tema

Hoy en día, en el mundo de la navegación, existe una gran sobrecarga de información, sobre todo información relacionada con el “postureo”, tanta que suele abrumar a la mayoría de los internautas, por ejemplo, en redes sociales como Instagram, Pinterest, Tiktok, … donde la gran mayoría de publicaciones tratan de exponer sus vidas o presumir ante la cámara. ¿Quién no querría vivir experiencias de acuerdo a las últimas tendencias?

Escogimos este tema porque nos atrajo la idea de poder crear nuestra web, algo que ninguno de nosotros habíamos hecho y la temática vino dada por la constante indecisión de los jóvenes al no saber qué hacer en un día de relax.

1.2 Objetivos del trabajo

El objetivo de este trabajo era crear un prototipo de una web dada nuestra temática usando html y un servidor web en python el cual permite servir archivos estáticos desde “data/” permitiendo redirigir desde el directorio raiz (/) hasta index.html, responder solicitudes GET y HEAD. Nuestro objetivo era recomendar dichas actividades a otros usuarios, atendiendo a las especificaciones y rango de precio de los diferentes planes.

1.3 Competencias a adquirir

- Navegar por internet para buscar la solución a un problema. Durante la realización de este trabajo de investigación fue necesario recurrir a múltiples búsquedas en internet para solucionar algunas dudas puntuales.
- Trabajar con un equipo utilizando algunos de sus recursos principales, como python, comandos para ejecutar en el terminal, programas de edición de texto y algún editor libre de html.
- Trabajar en equipo y utilizando el método científico, registrando de manera exacta y objetiva el trabajo realizado. De este modo aprendimos a realizar la documentación de un proyecto.

2 MATERIALES, MÉTODOS Y FASES DE TRABAJO:

Para poder llevar a cabo este proyecto, partimos desde dos extremos el primero la creación de la web y el servidor. El objetivo es crear una web funcional y para ello debemos combinar estos puntos.

Dividimos la estructura del trabajo en las 3 siguientes fases:

2.1 FASE 1: Bases para crear la web (python)

Nuestro código implementa un servidor web básico con Flask, que sirve archivos estáticos a clientes desde un directorio llamado data.

Entre algunas de las funciones principales que lleva a cabo están:

- **Rutas principales**:
    - /: Redirige automáticamente a index.html.
    - /<path:filename>: Si existen archivos de la carpeta data los sirve.
- **Métodos soportados**:
    - GET: Devuelve el archivo solicitado al cliente.
    - HEAD: Devuelve solo los encabezados (sin cuerpo) con información del tipo y tamaño del archivo.
- **Contenido**: Define los tipos MIME para archivos comunes como HTML, texto, imágenes JPEG o GIF. Si la extensión no coincide, usa application/octet-stream.
- **Errores**:
    - 404 Not Found: Devuelve una página de error si el archivo no existe.
    - 400 Bad Request: Maneja solicitudes incorrectas.
- **Ejecución del servidor**:
    - Se ejecuta en localhost (127.0.0.1) en el puerto 8000.

Por lo tanto, este servidor actúa como un servidor web estático y pequeño, que puede entregar archivos y manejar solicitudes básicas.

2.1.1 buscar los import flask:

Para trabajar con Flask en Python, primero es necesario instalar la biblioteca utilizando una herramienta de gestión de paquetes como "pip". Una vez instalada, se puede importar Flask en el proyecto para crear aplicaciones web.

La biblioteca proporciona las herramientas necesarias para manejar solicitudes HTTP, definir rutas que determinen las páginas o recursos que verá el usuario y devolver respuestas personalizadas.

Al importarlo, se crea una instancia de la aplicación que actúa como núcleo del servidor web y se configura para escuchar y responder las solicitudes del navegador. Flask también permite gestionar errores y definir cómo debe comportarse la aplicación en situaciones inesperadas (como páginas no encontradas o solicitudes inválidas).

2.1.2 creación del servidor web

Importación de bibliotecas:
Se importan las funciones necesarias de Flask para crear la aplicación, gestionar rutas y manejar archivos estáticos. También se usan módulos de Python, como ‘os’ para trabajar con archivos y ‘datetime’ para manejar fechas.

Configuración básica:
-Definimos un conjunto de constantes, como los métodos HTTP (GET y HEAD), los tipos de contenido (HTML, imágenes, texto), y los códigos de estado HTTP (200, 404, 400, etc.).
-Configuramos la base de la aplicación web mediante app = Flask(__name__).

Definir rutas y manejar solicitudes:

Ruta principal /: Redirige automáticamente a index.html.

Ruta dinámica /<path:filename>: Devuelve archivos del directorio data verificando si existen.

Dado un GET, devuelve el archivo solicitado.

Si solicita una HEAD, solo envía los encabezados HTTP sin contenido.

Ejecutar el servidor:
El servidor se inicia con app.run(), configurado para escuchar en localhost (127.0.0.1) en el puerto 8000.

1. IMPORTACIONES
    - Flask: Crea la aplicación web.
    - send_from_directori: Servir archivos desde una carpeta específica.
    - sbort: Manejar errores HTTP.
    - redirect, url_for: Redirigir a otras rutas.
    - request: Manejar solicitudes HTTP
    - os: Interactuar con el sistema de archivos

2. Configuraciones iniciales
    - Métodos: GET y HEAD.
    - Tipo de contenidos (MIME types): Extensiones de archivo con sus tipos MME correspondientes (html, txt, jpg, gif, etc.).
    - HTTP: Definimos un diccionario con algunos códigos HTTP y sus descripciones.

3. Ruta Principal (/)

"""Redirige a index.html."""

- Cuando un usuario accede a /, el servidor lo redirige automáticamente a index.html, suponiendo que este archivo existe en el directorio data/.

4. Ruta para Servir Archivos (/path:filename>)

Accedemos a cualquier archivo dentro del directorio data/.

- Verificación: si el archivo no existe, devuelve un error 404.
- Obtención del tipo MIME: Se extrae y busca su tipo en el diccionario de tipos MIME.
- Respuesta según el método HTTP:
    - Si HEAD, solo devuelve los encabezados HTTP sin enviar el contenido.
    - Si GET, el archivo es enviado al cliente.

5. Manejo de Errores

- Error 404 (No encontrado):

Si un usuario solicita un archivo inexistente, se devuelve un mensaje HTML con el error 404.

- Error 400 (Solicitudes incorrectas)

Se maneja el error 400 en caso de solicitudes malformadas.

6. Ejecución del Servidor

```
if __name__ == '_main_':
   app.run(host='127.0.0.1', port=8000)
```

- Inicia el servidor en http://127.0.0.1:8000.

2.2 FASE 2: creación web (html)

2.2.1 introducción a visual studio code

Visual Studio Code (VS Code) es un editor de código fuente ligero, potente y personalizable desarrollado por Microsoft. Es compatible con múltiples lenguajes de programación como con python. Su soporte constante asegura una evolución continua para adaptarse a las necesidades del desarrollador.

2.2.2 creación de la web

- Interactividad con JavaScript:
    - Selección de planes: Los usuarios pueden hacer clic en diferentes tarjetas de planes (ej. "Breathtaking Views", "Crazy Plans", etc.).
    - Ventana emergente: Al seleccionar un plan, aparece un popup donde se puede ajustar un rango de precios.
    - Búsqueda en Google Maps: Permite encontrar lugares relacionados con el plan elegido dentro de un presupuesto definido.
- Sección de Opiniones y Calificaciones:
    - Los usuarios pueden calificar planes con estrellas.
    - También pueden dejar comentarios y especificar si recomiendan el plan.
    - Las reseñas se guardan en localStorage para que persistan entre sesiones.
    - Se calcula y muestra la calificación promedio basada en todas las reseñas registradas.
- En resumen, es una plataforma interactiva que permite descubrir, compartir y evaluar diferentes experiencias de ocio, facilitando la búsqueda de planes adecuados a sus preferencias.

2.3 FASE 3: unión.

Colocar el html en una carpeta con el python después una vez ejecutado el python crea el servidor web, por lo que nos devuelve “running in http:// 127.0.0.1.8000” .
Entramos a un buscador y buscamos localhost:8000/Plan_Map, de este modo nos redirige a nuestra web

3. CONCLUSIONES

Creamos el prototipo de una web funcional:

- Para realizar este trabajo fueron necesarios conocimientos previos sobre el manejo de html y python.
- Conseguimos crear una web y implementar un servidor HTTP con dominio propio en Python usando socket.
- Pudimos comprobar y mejorar nuestras habilidades informáticas.

4. PROPUESTA DE CONTINUIDAD

- Ampliar los datos y valoración para tener más variedad de planes y actividades.
- Usar datos reales de los clientes de nuestra web para implementar un sistema de recomendación basandonos en los perfiles de nuestros usuarios.
- mplementar un mapa interactivo propio donde visualizar todos los planes.
- Con un asistente virtual ayudar a los usuarios de la web a crear planes más ersonalizados y nuevas esperienzas.

5. WEBGRAFÍA

- Flask

  Welcome to Flask — Flask Documentation (3.1.x), consultado el 22/02/2025.

- GitHub

  GitHub - dirk-makerhafen/pyHtmlGui: PyHtmlGui - A Python library for building user interfaces, consultado el 22/02/2025.

- GeeksforGeeks

  Creación y visualización de archivos HTML con Python - GeeksforGeeks, consultado el 22/02/2025.

- HTML YA

  HTML Ya, consultado el 22/02/2025.

- Python

  html.parser — Simple HTML and XHTML parser — Python 3.13.2 documentation, consultado el 22/02/2025.

- W3 Schools

  HTML Tutorial , consultado el 22/02/2025.

- Web Debs

  El mejor Tutorial de HTML y HTML5 más completo del mundo - Web Devs, consultado el 22/02/2025.