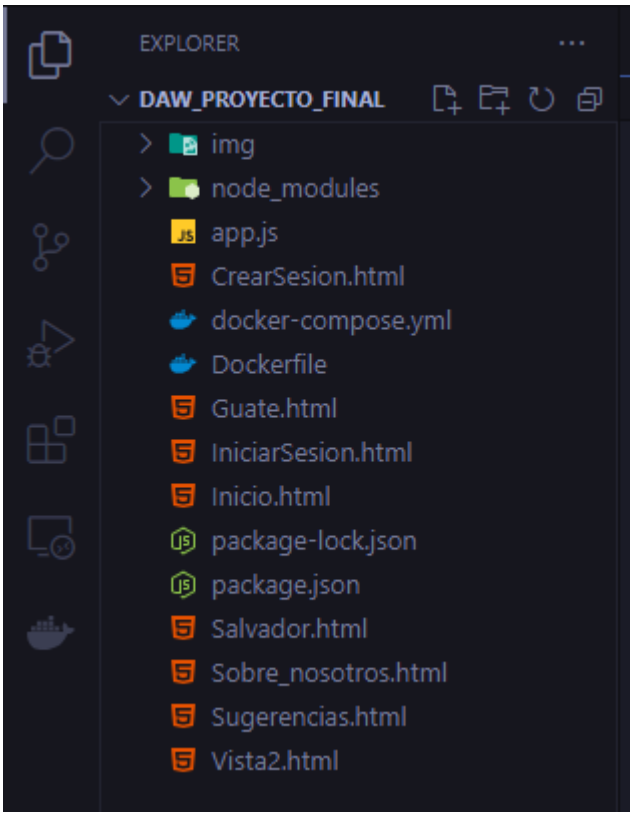
Manual Técnico: Implementación de una API REST en Docker

Nombre del Sitio: TURISMO SV

Objetivo del Sitio: Proporcionar a los usuarios una guía visual e informativa sobre destinos turísticos en Centroamérica, destacando especialmente El Salvador y Guatemala. El sitio ofrece fotos, nombres de lugares turísticos, precios y sugerencias de viaje para explorar lo mejor de la región.

Para ello, este documento ha sido creado con el fin para los futuros programadores o alguien diseñado con interés de averiguar sobre como funciona y se compone cada elemento

A la hora de abrir y ejecutar el programa dentro de visual estudio code, te encontraras con una serie de archivos que son los siguientes:



Para ello, es obligatorio conta con los siguientes criterios:

Requisitos Previos

Antes de comenzar, asegúrate de tener instalados los siguientes componentes en tu sistema:

- **Node.js**: Para desarrollar la API REST.
- **XAMPP**: Para el servidor local y la gestión de archivos HTML. Puedes obtenerlo en apachefriends.org.
- **Docker**: Para contenerizar la aplicación.
- **Docker Compose**: Para gestionar aplicaciones multi-contenedor. Docker Compose suele instalarse junto con Docker.

1. Creación de la API REST

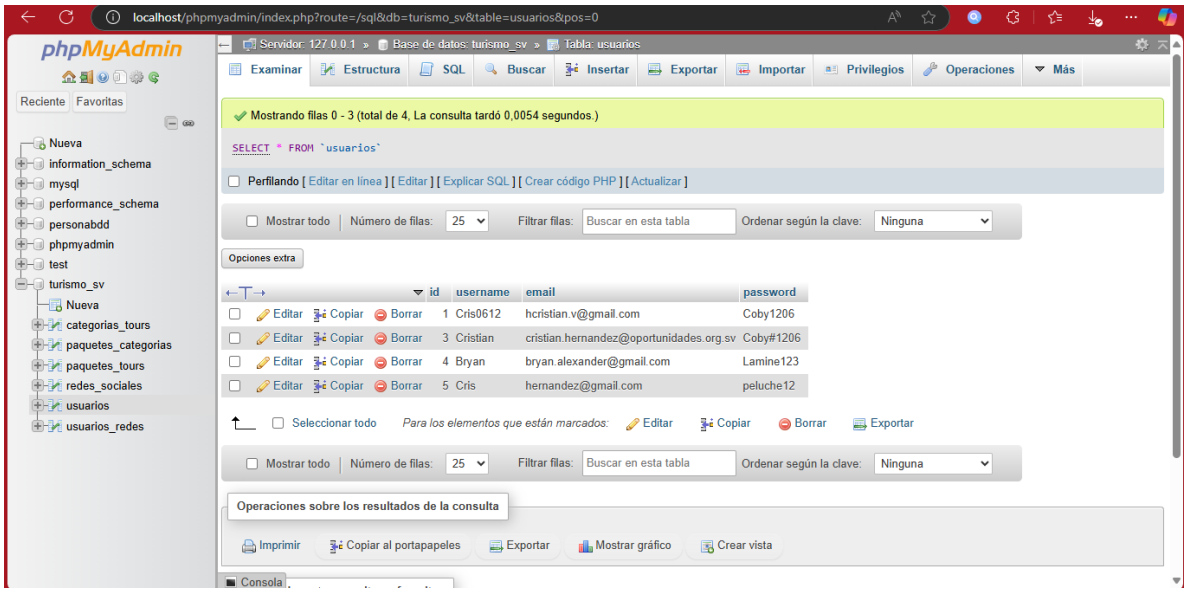
- Instalar Node.js
- Inicializa un nuevo proyecto Node.js con el siguiente comando, que creará un archivo package.json con “npm init -y”
- Después de la instalación, con el comando de “npm i express “ instalamos las dependencias de express.
- Luego de ello crear un archivo que se llama app.js. Este archivo contendrá la lógica de la API REST.

2. Creación de las Vistas con HTML

- Crear los diferentes archivos html con su respectiva información que desea mostrar

3. Creación de la base de datos

- Inicializar xampp con las opciones de Apache y MySQL
- Creamos la base de datos que se llama turismo_sv, en ella permitirá guardar los datos cuando el usuario crea una cuenta por primera vez, además de acceder e ingresar a la plataforma con dicha cuenta.



4. Contenerización con Docker

- Se creo un archivo llamado Dockerfile en el directorio raíz del proyecto. Este archivo define cómo se construirá la imagen Docker para la aplicación. contiene todas las instrucciones necesarias para construir una imagen Docker. Define el entorno en el que se ejecutará tu aplicación, incluyendo el sistema operativo, las dependencias y cómo iniciar la aplicación.
- **FROM**: Especifica la imagen base (en este caso, Node.js).
- **WORKDIR**: Define el directorio de trabajo en el contenedor.
- **COPY**: Copia archivos desde el sistema de archivos del host al contenedor.
- **RUN**: Ejecuta comandos (como la instalación de dependencias).
- **EXPOSE**: Informa a Docker que el contenedor escucha en el puerto especificado en tiempo de ejecución.
- **CMD**: Proporciona el comando por defecto que se ejecutará al iniciar el contenedor.

```
# Usa una imagen oficial de Node.js como base
FROM node:16

# Establece el directorio de trabajo en el contenedor
WORKDIR /app

# Copia el package.json y package-lock.json
COPY package*.json ./
```

```
# Instala las dependencias
RUN npm install

# Exponer el puerto que utiliza la aplicación
EXPOSE 3000

# Comando para ejecutar la aplicación
CMD ["node", "app.js"]
```

- Crea un archivo llamado docker-compose.yml en el mismo directorio. Este archivo te permite definir y ejecutar aplicaciones compuestas por múltiples contenedores. Facilita la configuración de la red entre contenedores, así como la gestión de sus dependencias.
- **services**: Define los distintos contenedores que componen la aplicación.
- **build**: Indica que se debe construir la imagen desde el Dockerfile.
- **image**: Especifica una imagen existente para el contenedor (como MySQL o phpMyAdmin).
- **environment**: Establece variables de entorno que configuran el comportamiento de los contenedores.
- **ports**: Mapea puertos del contenedor a puertos del host, permitiendo el acceso a la aplicación desde el exterior.