MANUAL DE LÓGICA DEL FRONTEND DEL PROGRAMA

Índice

Introducción

El frontend de este programa está diseñado para presentar una interfaz intuitiva al usuario, permitir la interacción dinámica con los datos y conectarse con un backend a través de llamadas a una API. Utiliza una estructura de archivos clara y organizada para separar estilos, lógica, recursos visuales y vistas.

1. Requisitos para Usar el Programa

- Tener el servidor de la base de datos en ejecución (ver manual de Backend).
- Navegador web moderno (como Google Chrome o Firefox).
- Tener **Visual Studio Code** instalado.
- Acceso a Internet (para que el programa pueda hacer el llamado a la API).
- Live Server configurado y en ejecución.

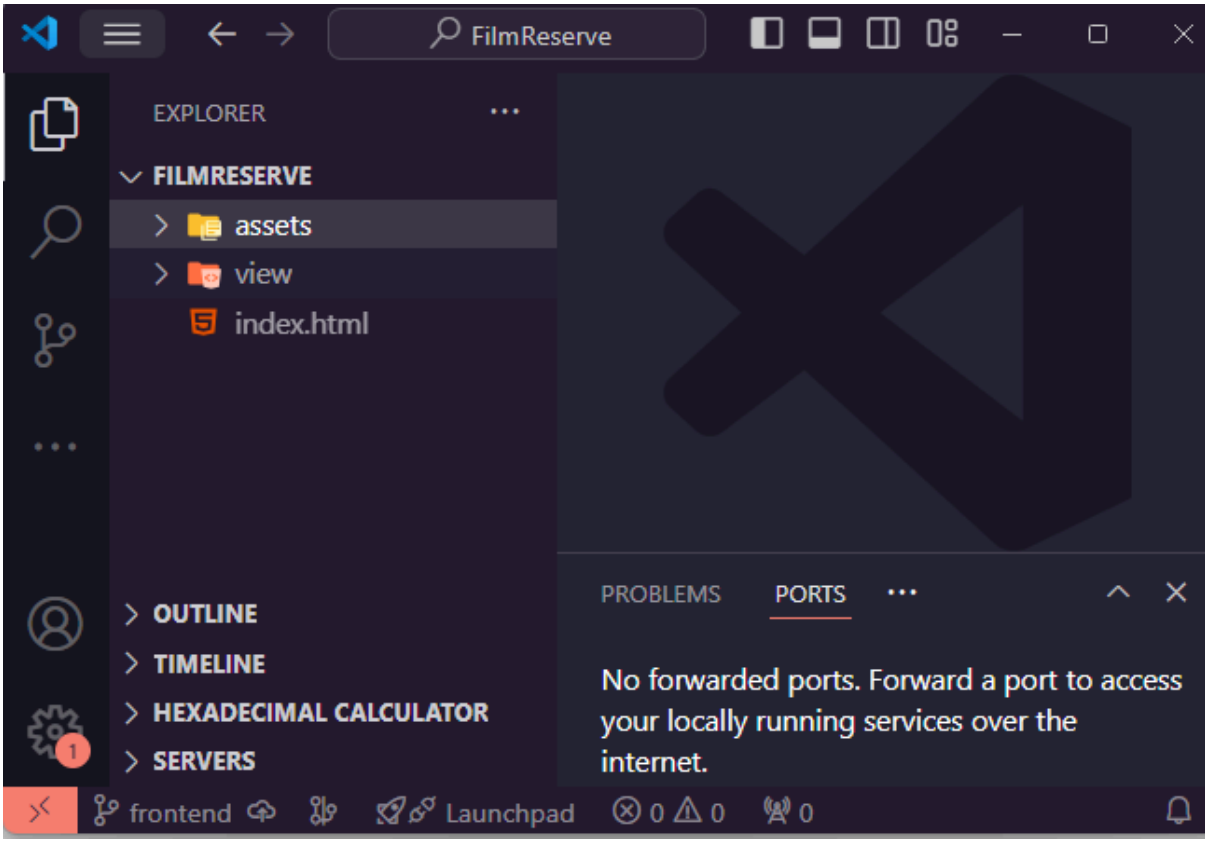
2. Flujo de Funcionamiento

El funcionamiento del frontend comienza cuando el usuario abre el archivo `index.html` utilizando Live Server. Esto permite que el navegador cargue la estructura base definida en el archivo HTML, aplique los estilos desde la carpeta css y ejecute los scripts alojados en la carpeta js. La interacción con la API es gestionada por `ClientAPI.js`, que realiza solicitudes HTTP para comunicarse con el backend, utilizando autenticación mediante encabezados con credenciales codificadas. Entre las principales funcionalidades disponibles, se encuentran: `GET` para obtener información (por ejemplo, datos de un cliente), `POST` para enviar datos al backend (como crear un nuevo administrador), `PUT` para actualizar información existente (como una contraseña), y `DELETE` para eliminar registros (como un cliente). Finalmente, la información obtenida de la API es procesada por `app.js` y se presenta dinámicamente en la página mediante el DOM. Además, se manejan errores y respuestas para proporcionar retroalimentación al usuario de manera efectiva.

3. Pasos para Ejecutar el Programa

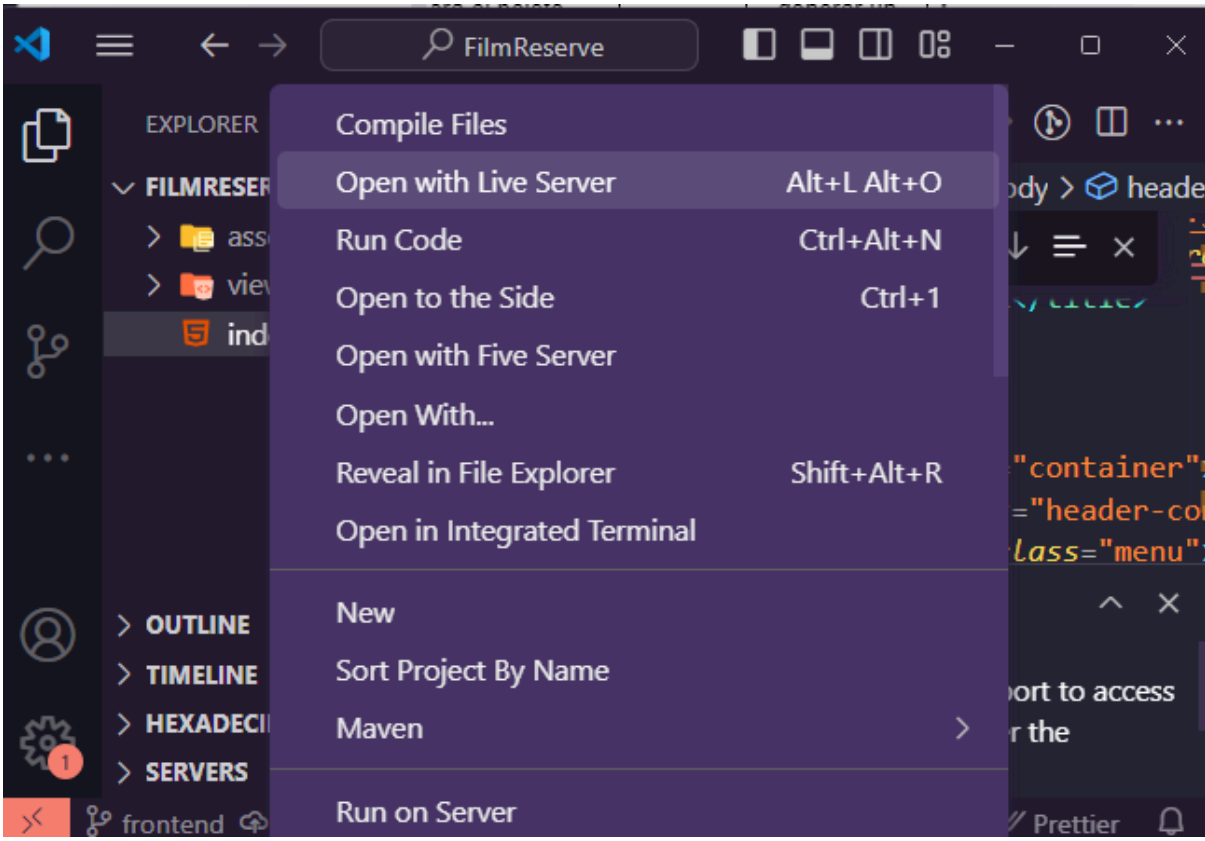
3.1. Abrir el Proyecto

- Asegúrate de tener los archivos del programa disponibles en tu equipo.
- Abre la carpeta que contiene el programa en Visual Studio Code.



3.2 Iniciar Live Server

- Haz clic derecho sobre el archivo principal (`index.html`).
- Selecciona **"Open with Live Server"**.



- Se abrirá una nueva pestaña en tu navegador con la URL del servidor local, por ejemplo, http://127.0.0.1:5500/.

Interacción con el Programa

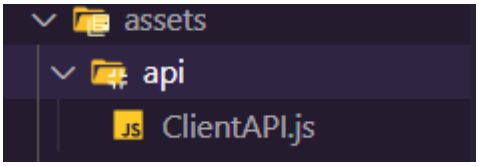
- Al abrir la página, el programa automáticamente hará los llamados a la API

4. Archivos Incluidos en el Programa

Assets

- **api**
    - ClientAPI: Este código define una clase llamada `ClientAPI` que sirve como un cliente genérico para interactuar con APIs mediante los métodos HTTP (`GET`, `POST`, `PUT`, y `DELETE`). También incluye un objeto preconfigurado. A continuación, se desglosan los elementos principales del código:

1. Clase ClientAPI



La clase encapsula toda la lógica para realizar peticiones HTTP y autenticación básica.

Constructor

- **Parámetros**:
    - `prmUsername`: Nombre de usuario para la autenticación.
    - `prmPassword`: Contraseña para la autenticación.
    - `prmAddres`: Dirección base de la API (e.g., `"https://www.api.com"`).
- **Función**:
    - Inicializa atributos como `atrUsername`, `atrPassword`, `atrHeaders` (encabezados con autenticación básica en formato `Authorization`), y `atrAddres` (URL base de la API).

2. Métodos HTTP

Todos los métodos utilizan `fetch` para realizar las peticiones y manejan la respuesta con una función callback (`prmAction`) pasada como parámetro.

2.1. `get(prmEndpoint, prmPathVariable = "", prmAction)`

- Realiza una petición HTTP `GET`.
- **Parámetros**:

1. `prmEndpoint`: Endpoint de la API (por ejemplo, `"/api/v1/customer"`).
2. `prmPathVariable`: Variable opcional para el path (por ejemplo, `"/123"`).
3. `prmAction`: Función lambda para manejar la respuesta.

- **Flujo**:
  1. Construye la URL combinando `atrAddres`, `prmEndpoint` y `prmPathVariable`.
  2. Realiza la petición con encabezados preconfigurados.
  3. Ejecuta `prmAction` con la respuesta en formato JSON.

2.2. `post(prmEndpoint, prmBody, prmAction)`

- Realiza una petición HTTP `POST`.
- **Parámetros**:
  1. `prmEndpoint`: Endpoint de la API.
  2. `prmBody`: Datos a enviar (en formato `FormData`).
  3. `prmAction`: Función lambda para manejar la respuesta.
- **Flujo**:
  1. Construye la URL con `atrAddres` y `prmEndpoint`.
  2. Envía la petición con el método `POST`, encabezados, y `prmBody`.
  3. Ejecuta `prmAction` con la respuesta en JSON.

2.3. `put(prmEndpoint, prmBody, prmAction)`

- Similar a `post`, pero utiliza el método HTTP `PUT`.
- Usado para actualizar datos en la API.

2.4. `delete(prmEndpoint, prmPathVariable = "", prmAction)`

- Realiza una petición HTTP `DELETE`.
- **Parámetros**:
  1. `prmEndpoint`: Endpoint de la API.
  2. `prmPathVariable`: Path variable opcional.
  3. `prmAction`: Función lambda para manejar la respuesta.
- **Flujo**:
  1. Construye la URL con `atrAddres`, `prmEndpoint`, y `prmPathVariable`.
  2. Envía la petición con el método `DELETE` y encabezados.
  3. Ejecuta `prmAction` con la respuesta en JSON.

3. Objeto `objClient`

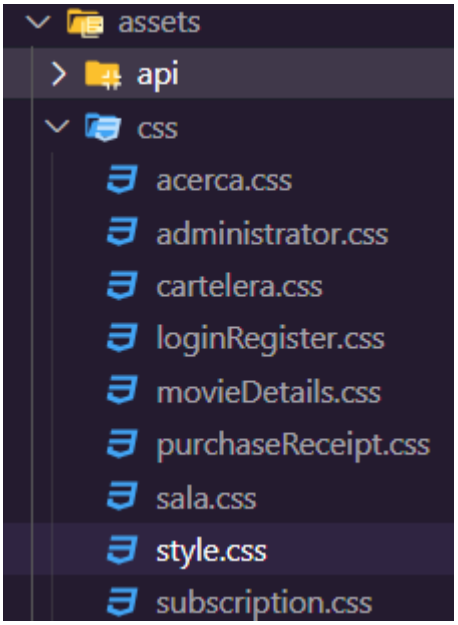
- Crear una instancia de la clase `ClientAPI`:
  - Usuario: `"filmreserve"`.
  - Contraseña: `"123"`.
  - Dirección base: `"http://localhost:8001"`.

```
const objClientAPI = new ClientAPI(
  "filmreserve",
  "123",
  "http://localhost:8001"
);
```

Usado para interactuar con un servidor local.

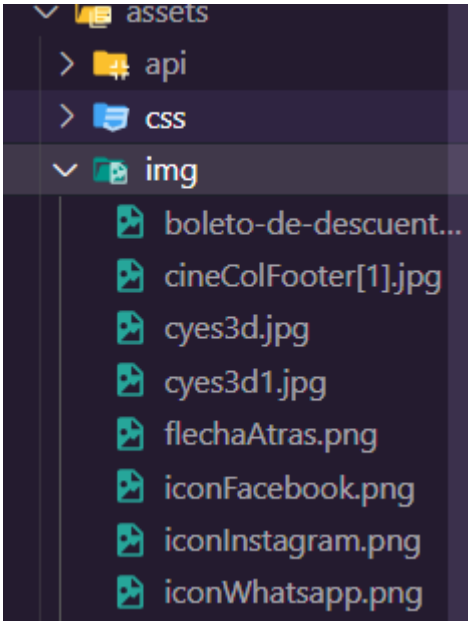
- **css**

  Contiene todos los archivos relacionados con los estilos del programa. Define la apariencia visual de las páginas web (colores, fuentes, tamaños, márgenes, etc.).

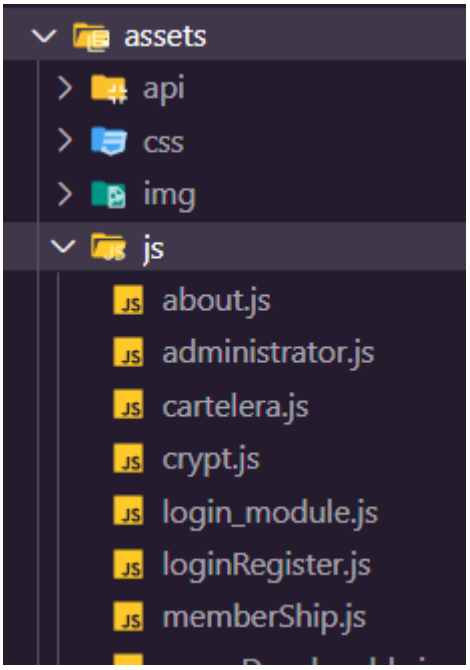


- **img**

  almacena las diferentes imágenes utilizadas en el programa

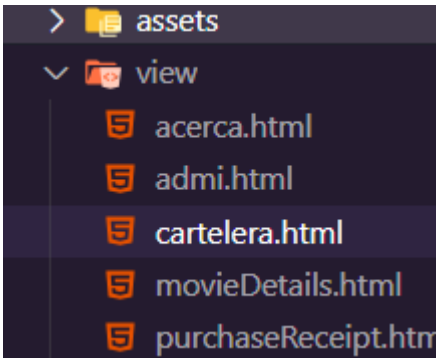


- **js**
  - Contiene los archivos de JavaScript responsables de la lógica del programa.



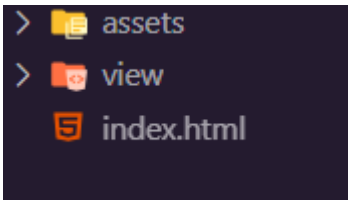
  - Define el comportamiento dinámico e interactivo de la aplicación.
  - Realiza llamados a APIs, validación de formularios, manejo de eventos, etc.

view

- La carpeta `view` contiene todos los archivos HTML del programa, que definen la estructura y el contenido de las páginas web que se muestran al usuario.
- 

HTML (index.html)

- Este archivo contiene la estructura de la página web:



- **Descripción**

  El archivo `index.html` es el punto de entrada principal de la aplicación web. Es el archivo que el navegador carga inicialmente cuando se accede al sitio.

- **Propósito**

  Actuar como la **página principal** de la aplicación web.

5. Errores Comunes y Soluciones

- La API no responde
  - Verifica que el servidor backend esté en ejecución.
  - Asegúrate de tener conexión a Internet.
- El programa no carga correctamente
  - Confirma que Live Server esté configurado y en ejecución.