UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1

CATEDRÁTICO: ING. NEFTALI DE JESUS CALDERON MENDEZ

TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO

MANUAL TÉCNICO

MANUEL ALEJANDRO LÓPEZ CANEL
DAVID NORBERTO FABRO GUZMAN
CARNÉ: 202302035
202307499
SECCIÓN: A

GUATEMALA, 27 DE JUNIO DEL 2,024

ÍNDICE

INTRODUCCIÓN

El presente manual técnico tiene como propósito brindar una guía detallada sobre la creación y desarrollo de la plataforma web de streaming para el alquiler de películas, PopCornflix. Este documento está diseñado para orientar a los desarrolladores en la implementación y el mantenimiento del proyecto, asegurando una comprensión clara de su arquitectura y funcionalidad.

OBJETIVOS

1. GENERAL

1.1. Proporcionar una guía detallada para la implementación y mantenimiento de la plataforma web PopCornflix, destacando la arquitectura API REST utilizada y la integración entre el frontend y el backend.

2. ESPECÍFICOS

2.1. Objetivo 1: Describir paso a paso el desarrollo del backend utilizando el framework Express.

2.2. Objetivo 2: Explicar la construcción del frontend con React y la forma en que se conecta con el backend mediante operaciones CRUD.

ALCANCES DEL SISTEMA

Este manual tiene como objetivo proporcionar una guía completa y detallada para el desarrollo y mantenimiento de la plataforma PopCornflix. Se enfoca en la implementación técnica de la arquitectura, el desarrollo de funcionalidades tanto para administradores como para usuarios, y la integración entre el frontend y el backend.

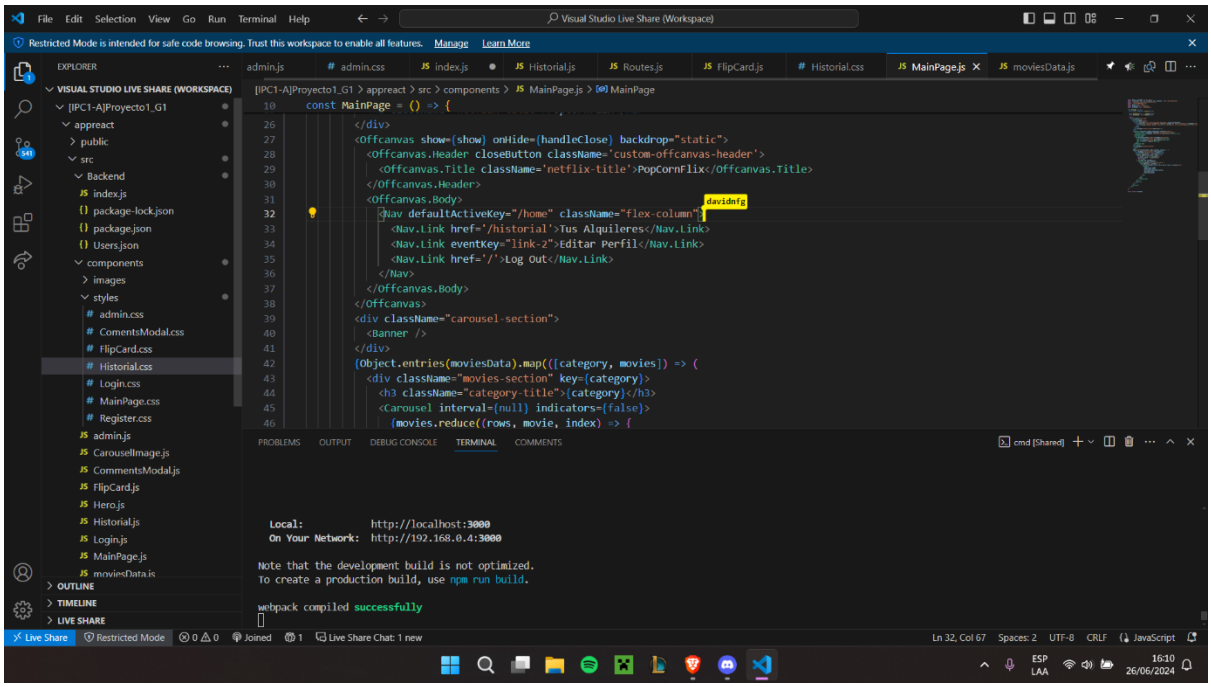
ESPECIFICACIÓN TÉCNICA

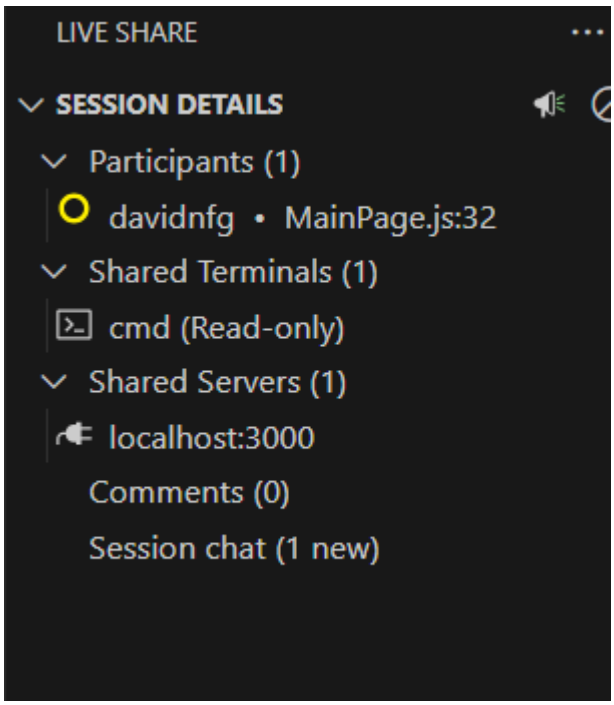
- **REQUISITOS DE HARDWARE**
  - Computadora con procesador de al menos 2.0 GHz
  - Memoria RAM de al menos 4 GB
  - Almacenamiento de al menos 1 GB disponible
- **REQUISITOS DE SOFTWARE**
  - Conexión a internet estable
  - Node.js y npm instalados
  - Visual Studio Code u otro editor de código
  - Navegador web moderno (Google Chrome, Firefox, etc.)

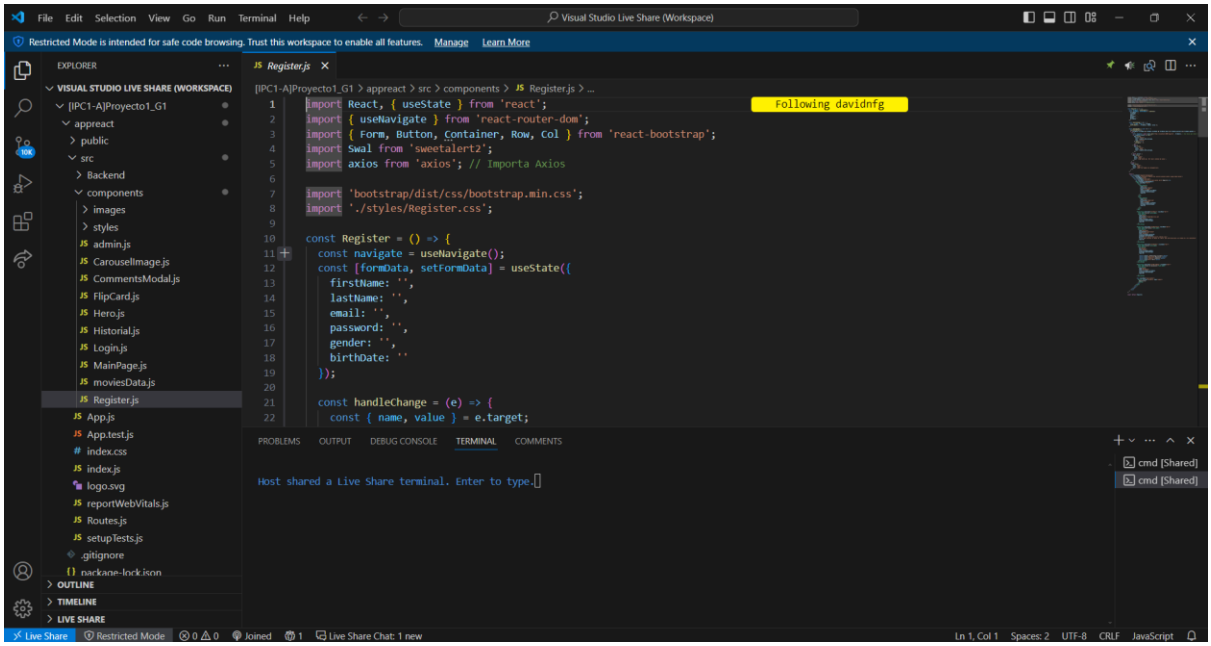
DESCRIPCIÓN DE LA SOLUCIÓN

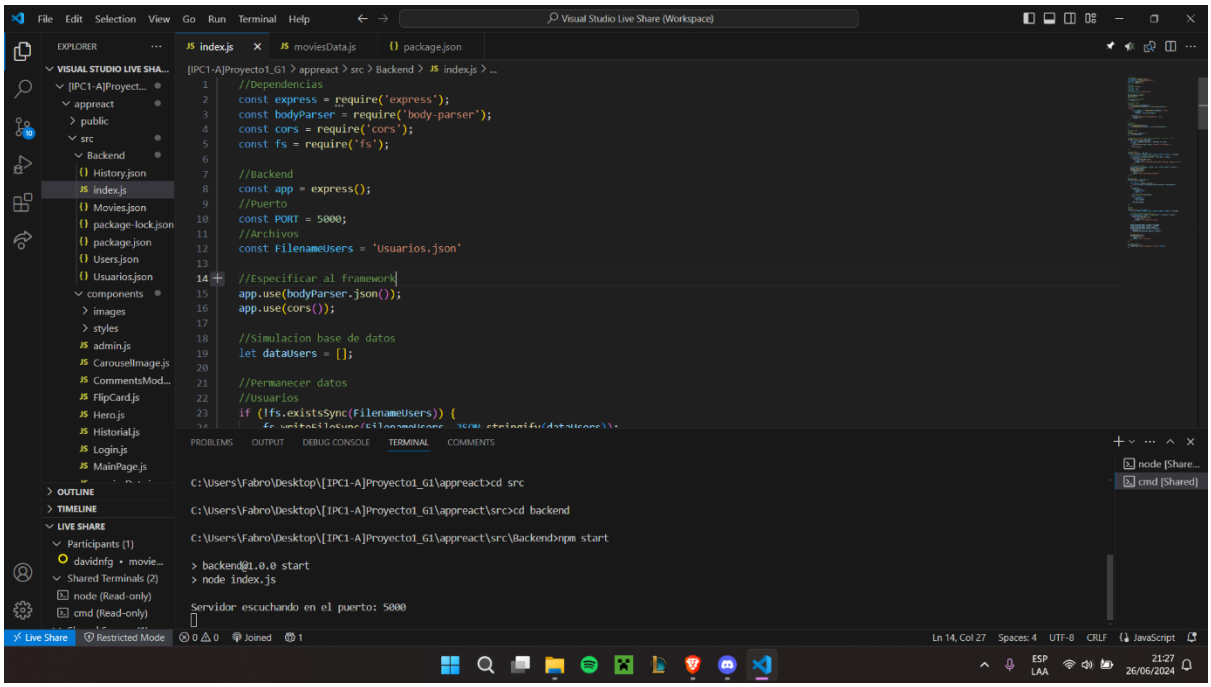
- La solución fue diseñada y analizada teniendo en cuenta los requisitos del enunciado, centrando el desarrollo en una arquitectura API REST. Se utilizaron tecnologías modernas como Express para el backend y React para el frontend, permitiendo una integración fluida entre ambos. La persistencia de datos se maneja mediante archivos JSON, facilitando la gestión de usuarios y películas.

VERIFICACION DEL TRABAJO






Register.js
Following davidnfg
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Form, Button, Container, Row, Col } from 'react-bootstrap';
import Swal from 'sweetalert2';
import axios from 'axios'; // Importa Axios
import 'bootstrap/dist/css/bootstrap.min.css';
import './styles/Register.css';
const Register = () => {
const navigate = useNavigate();
const [formData, setFormData] = useState({
firstName: '',
lastName: '',
email: '',
password: '',
gender: '',
birthDate: ''
});
const handleChange = (e) => {
const { name, value } = e.target;
Host shared a Live Share terminal. Enter to type.


index.js
moviesData.js
package.json
//Dependencias
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const fs = require('fs');
//Backend
const app = express();
//Puerto
const PORT = 5000;
//Archivos
const FilenameUsers = 'Usuarios.json'
//Especificar al framework
app.use(bodyParser.json());
app.use(cors());
//Simulacion base de datos
let dataUsers = [];
//Permanecer datos
//Usuarios
if (!fs.existsSync(FilenameUsers)) {
C:\Users\Fabro\Desktop\[IPC1-A]Proyecto1_G1\appreact>cd src
C:\Users\Fabro\Desktop\[IPC1-A]Proyecto1_G1\appreact\src>cd backend
C:\Users\Fabro\Desktop\[IPC1-A]Proyecto1_G1\appreact\src\Backend>npm start
> backend@1.0.0 start
> node index.js
Servidor escuchando en el puerto: 5000

LÓGICA DEL PROGRAMA

- **Index.js (Backend)**

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const fs = require("fs");
```

- **CarouselImage.js (Frontend)**

```
import React from 'react';
```

- **CommentsModal.js (Frontend)**

```
import React, { useState } from 'react';
import { Modal, Button } from 'react-bootstrap';
import './styles/ComentsModal.css';
```

- **FlipCard.js (Frontend)**

```
import React, { useState } from 'react';
import Swal from 'sweetalert2';
import CommentsModal from './CommentsModal';
import './styles/FlipCard.css';
```

- **Hero.js (Frontend)**

```
import React from 'react'
import banner_image from "./images/banner_img.jpg"
import dolar_icon from "./images/dolar.png"
import info_icon from "./images/info.png"
import { Button } from 'react-bootstrap'
```

- **Historial.js (Frontend)**

```
import React, { useState, useEffect } from 'react';
import { Container, Navbar, Table, Button } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import './styles/Historial.css';
import moviesData from './moviesData';
```

❖ **Login.js (Frontend)**

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import Swal from "sweetalert2";
import "bootstrap/dist/css/bootstrap.min.css";
import "./styles/Login.css";
```

❖ **MainPage.js (Frontend)**

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col, Offcanvas, Nav, Carousel } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import './styles/MainPage.css';
import Banner from './Hero';
import FlipCard from './FlipCard';
import moviesData from './moviesData';
```

❖ **Register.js (Frontend)**

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Form, Button, Container, Row, Col } from 'react-bootstrap';
import Swal from 'sweetalert2';
import axios from 'axios';
```

❖ **admin.js (Frontend)**

```
import React, { useState, useEffect } from "react";
import {
  Container,
  Navbar,
  Nav,
  Table,
  Button,
  Modal,
  Form,
} from "react-bootstrap";
import "bootstrap/dist/css/bootstrap.min.css";
import "./styles/admin.css";
import moviesData from "./moviesData";
import axios from "axios";
```

❖ **editprofile.js (Frontend)**

```
import React, { useState, useEffect } from 'react';
import { Form, Button, Container, Navbar } from 'react-bootstrap';
import Swal from 'sweetalert2';
import 'bootstrap/dist/css/bootstrap.min.css';
import './styles/editprofile.css';
import axios from 'axios';
```

➢ **Librerías**

- express: Framework utilizado para crear el servidor y manejar las rutas.
- fs: Módulo de Node.js utilizado para manejar el sistema de archivos, especialmente para leer y escribir archivos JSON.
- body-parser: Middleware para manejar los datos del cuerpo de las solicitudes HTTP.

➢ **Variables Globales de la clase CommentsModal.js**

```
const CommentsModal = ({ show, handleClose, title }) => {
  const [comments, setComments] = useState([]);
  const [newComment, setNewComment] = useState('');
  const [username, setUsername] = useState('');
```

➢ **Variables Globales de la clase FlipCard.js**

```
const FlipCard = ({ image, title, description, btn1, btn2, onRent, visible }) => {
  const [showComments, setShowComments] = useState(false);

  const handleShowComments = () => setShowComments(true);
  const handleCloseComments = () => setShowComments(false);
```

➢ **Variables Globales de la clase Historial.js**

```
const Historial = () => {
  const [rentals, setRentals] = useState([]);
```

➢ **Variables Globales de la clase Login.js**

```
const Login = () => {
  const navigate = useNavigate();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
```

➢ **Variables Globales de la clase MainPage.js**

```
const MainPage = () => {
  const [show, setShow] = useState(false);
  const [visibleMovies, setVisibleMovies] = useState({});
  const [rentedMovies, setRentedMovies] = useState([]);
```

➢ **Variables Globales de la clase Register.js**

```
const Register = () => {
  const navigate = useNavigate();
  const [formData, setFormData] = useState({
    firstName: '',
    lastName: '',
    email: '',
    password: '',
    gender: '',
    birthDate: ''
  });
```

➢ **Variables Globales de la clase admin.js**

```
const Admin = () => {
  const [activeTab, setActiveTab] = useState("usuarios");
  const [users, setUsers] = useState([]);
  const [showAddMovieModal, setShowAddMovieModal] = useState(false);
  const [showUpdateMovieModal, setShowUpdateMovieModal] = useState(false);
  const [newMovie, setNewMovie] = useState({
    category: "NUEVAS",
    title: "",
    description: "",
    image: null,
  });
  const [selectedMovie, setSelectedMovie] = useState(null);
```

➢ **Variables Globales de la clase editprofile.js**

```
const EditProfile = () => {
  const [userProfile, setUserProfile] = useState({
    firstName: '',
    lastName: '',
    email: '',
    birthDate: '',
    password: ''
  });
```

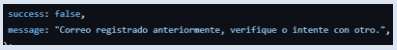
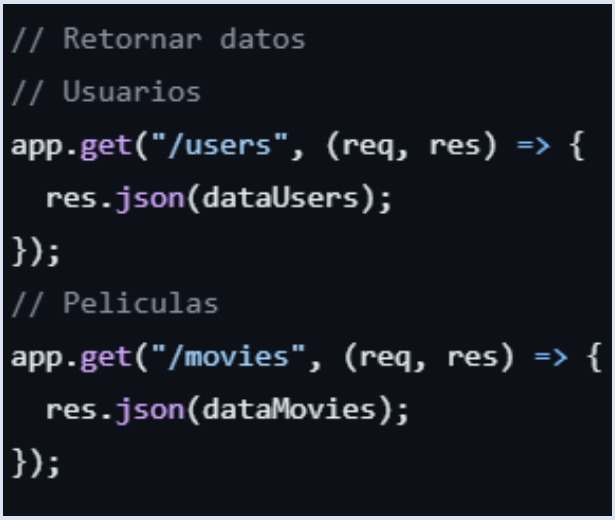
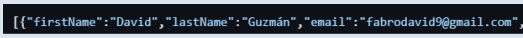
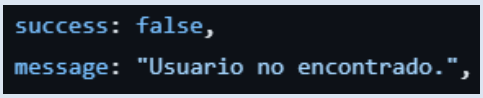
➢ **Variables Globales de la clase index.js (Backend)**

```
// Backend
const app = express();
// Puerto
const PORT = 5000;
// Archivos
const FilenameUsers = "Usuarios.json";
const FilenameMovies = "Movies.json";
```

➢ **Función Main (index.js del front)**

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import App from './App';
import 'bootstrap/dist/css/bootstrap.min.css'; // Importa Bootstrap CSS

const container = document.getElementById('root');
const root = createRoot(container);
root.render(<App />);
```

Método	Dirección y tipo de Método	Body	Respuesta
Registrar Usuario Recibe un JSON con los datos que el usuario ingresa desde el frontend	/register		
	POST		
Mostrar Usuarios y Películas Muestra a los usuarios y a las películas registradas en el archivo json	/users /movies		
	GET		
Eliminar Usuario Elimina al usuario seleccionado incluso del archivo json	/users/:id		
	DELETE		

<table>
<tr>
<td rowspan="2">Eliminar Película

Elimina la película seleccionada del archivo json</td>
<td>/movies/:id</td>
<td rowspan="2">

```
// Película por ID
app.delete("/movies/:id", (req, res) => {
  const movieId = parseInt(req.params.id);
  const movieIndex = dataMovies.findIndex((movie) => movie.id === movieId);

  if (movieIndex === -1) {
    return res.status(404).json({
      success: false,
      message: "Película no encontrada.",
    });
  }

  dataMovies.splice(movieIndex, 1);
  updateFileMovies();
  res.status(200).json({
    success: true,
    message: "Película eliminada exitosamente.",
  });
});
```

</td>
<td>

```
success: false,
message: "Película no encontrada."
```

</td>
</tr>
<tr>
<td>DELETE</td>
<td>

```
success: true,
message: "Película eliminada exitosamente.",
```

</td>
</tr>
<tr>
<td rowspan="2">Agregar Película

guarda la categoría, el titulo y la descripción de la película, la imagen es guardada con base 64</td>
<td>/admin/:id</td>
<td rowspan="2">

```
// Endpoint para agregar una nueva película
app.post("/movies", (req, res) => {
  const { category, title, description } = req.body;

  const newMovie = {
    id: new Date().getTime(), // Usar timestamp como ID único
    category,
    title,
    description,
  };

  dataMovies.push(newMovie);
  updateFileMovies();

  res.status(201).json({
    success: true,
    message: "Película agregada exitosamente.",
    movie: newMovie,
  });
});
```

</td>
<td>

```
message: "Película agregada exitosamente.",
movie: newMovie,
```

</td>
</tr>
<tr>
<td>PUT</td>
<td></td>
</tr>
<tr>
<td rowspan="2">Actualizar película

Toma la película seleccionada y actualiza los datos que se hayan cambiado</td>
<td>/admin/:id</td>
<td rowspan="2">

```
// Endpoint para actualizar una película por ID
app.put("/admin/:id", (req, res) => {
  const movieId = parseInt(req.params.id);
  const { category, title, description } = req.body;
  const movieIndex = dataMovies.findIndex((movie) => movie.id === movieId);

  if (movieIndex === -1) {
    return res.status(404).json({
      success: false,
      message: "Película no encontrada.",
    });
  }

  dataMovies[movieIndex] = {
    ...dataMovies[movieIndex],
    category,
    title,
    description,
  };
  updateFileMovies();
  res.status(200).json({
    success: true,
    message: "Película actualizada exitosamente.",
    movie: dataMovies[movieIndex],
  });
});
```

</td>
<td>

```
success: false,
message: "Película no encontrada.",
```

</td>
</tr>
<tr>
<td>PUT</td>
<td>

```
success: true,
message: "Película actualizada exitosamente.",
```

</td>
</tr>
</table>

<table>
<tr>
<td rowspan="2">Login

Esta función se encarga de verificar si al entrar existe el usuario en la lista de usuarios</td>
<td>/login</td>
<td rowspan="2"> 
</td>
<td>success: false,
user: null,</td>
</tr>
<tr>
<td>POST</td>
<td>success: true,
user: tempuser,</td>
</tr>
<tr>
<td rowspan="2">Editar Perfil

Actualiza los datos del usuario loggeado</td>
<td>/editarperfil</td>
<td rowspan="2"> 
</td>
<td>success: true,
message: "Perfil actualizado.",</td>
</tr>
<tr>
<td>PUT</td>
<td>success: false,
message: "No se encontró al usuario.",</td>
</tr>
</table>

DOCUMENTO EXTRA

ESTUDIANTE 1	ESTUDIANTE 2
Manuel Alejandro López Canel 202302035	David Norberto Fabro Guzmán 202307499
BACKEND	FRONTEND
MANUAL TECNICO	MANUAL DE USUARIO