UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1
CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA
TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO


# MANUAL TÉCNICO

ELEAZAR NEFTALÍ COLOP COLOP
CARNÉ: 3198935960914
SECCIÓN: A

GUATEMALA, 28 DE JUNIO DEL 2,024

# ÍNDICE

# INTRODUCCIÓN

Este manual técnico tiene como finalidad proporcionar una guía detallada sobre la implementación, uso y mantenimiento del sistema de gestión de alquiler de películas. Aquí se detallarán los pasos necesarios para la configuración del entorno, la estructura del código, los endpoints disponibles y cómo interactuar con ellos. El objetivo es facilitar el desarrollo, la ampliación y la solución de problemas del sistema por parte de los programadores y administradores.

# OBJETIVOS

## 1. GENERAL

1.1. Proporcionar una guía comprensiva y detallada sobre la funcionalidad, estructura y mantenimiento del sistema de gestión de alquiler de películas.

## 2. ESPECÍFICOS

2.1. Objetivo 1: Describir la estructura del código y explicar cómo se implementan las principales funcionalidades del sistema.

2.2. Objetivo 2: Proveer instrucciones claras para la configuración del entorno de desarrollo y los requisitos necesarios para el correcto funcionamiento del sistema.

# DIRIGIDO

Este manual está dirigido a programadores, desarrolladores de software y administradores del sistema que se encargan de mantener y expandir la funcionalidad del sistema de gestión de alquiler de películas.

# ALCANCES DEL SISTEMA

El objetivo de este manual es ofrecer una guía técnica detallada que permita a los desarrolladores comprender la estructura y el funcionamiento del sistema de gestión de alquiler de películas, facilitando así su mantenimiento, solución de problemas y posibles ampliaciones.

# ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**

Para trabajar con la aplicación y continuar con el desarrollo futuro, se recomienda que el programador tenga acceso a los siguientes recursos de hardware:

- Procesador: Intel i5 o equivalente
- Memoria RAM: 8GB mínimo (16GB recomendados para un rendimiento óptimo)
- Almacenamiento: SSD con al menos 50GB de espacio libre
- Conexión a Internet estable para la descarga de dependencias y la interacción con servicios externos

- **REQUISITOS DE SOFTWARE**

Para trabajar con la aplicación y seguir con el desarrollo futuro, el programador necesitará los siguientes requisitos de software:

- **Sistema Operativo**: Windows 10 o superior, macOS, o distribuciones de Linux (Ubuntu, Fedora)
- **Node.js**: Versión 14.17.0 o superior
- **NPM**: Versión 6.14.13 o superior (incluido con Node.js)
- **Editor de Código**: Visual Studio Code o cualquier editor de texto con soporte para JavaScript/Node.js
- **Postman/Thunderclient**: Para probar los endpoints del API
- **Navegador Web**: Chrome, Firefox o cualquier navegador moderno
- **Git**: Para el control de versiones y la colaboración en el desarrollo del proyecto

# DESCRIPCIÓN DE LA SOLUCIÓN

El sistema de gestión de alquiler de películas es una aplicación web que permite a los usuarios ver, alquilar y devolver películas, así como a los administradores gestionar usuarios y posts de películas. La solución está desarrollada utilizando tecnologías modernas como Node.js para el backend y React para el frontend, con almacenamiento de datos en archivos JSON.

**Componentes Principales**

1. **Backend (Node.js y Express)**
    - **Funcionalidad**: El backend proporciona una API RESTful que permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre usuarios y posts de películas, así como gestionar el proceso de alquiler y devolución de películas.
    - **Almacenamiento**: Los datos se almacenan en archivos JSON (Usuarios.json, Posts.json, moviesAlquiladas.json).
    - **Endpoints**:
        - **/login**: Autenticar a los usuarios.
        - **/getPosts**: Obtener todos los posts de películas.
        - **/students**: Gestionar usuarios (crear, leer, actualizar, eliminar).
        - **/createPost**: Crear un nuevo post de película.
        - **/deletePost/**

: Eliminar un post de película.

        - **/alquilar**: Alquilar una película.
        - **/devolver**: Devolver una película.
        - **/getRentedMovies**: Obtener las películas alquiladas.
        - **/getPost/**

: Obtener un post específico.

        - **/updatePost/**

: Actualizar un post de película.

2. **Frontend (React)**
    - **Componentes**: El frontend incluye varios componentes para diferentes funcionalidades:
        - **UsuarioNormal**: Permite a los usuarios ver y alquilar películas.

        - **UsuarioAdmin**: Gestión de usuarios y posts.
        - **RentedMovies**: Muestra las películas alquiladas por el usuario.
        - **ReturnMovie**: Gestiona la devolución de películas.
    - **Bibliotecas Utilizadas**:
        - **React Router**: Para la navegación entre componentes.
        - **Bootstrap**: Para el diseño y estilo de la interfaz de usuario.
        - **SweetAlert2**: Para mostrar alertas y confirmaciones.
        - **React-Cookie**: Para manejar las cookies de autenticación.

**Flujo de Trabajo**

1. **Autenticación**
    - Los usuarios se autentican a través del endpoint /login utilizando su correo y contraseña. Si las credenciales son correctas, se guarda una cookie con la información del usuario.
2. **Gestión de Usuarios y Posts**
    - Los administradores pueden gestionar usuarios y posts de películas a través de los endpoints correspondientes (/students, /createPost, /deletePost/:postId, etc.).
3. **Alquiler y Devolución de Películas**
    - Los usuarios pueden alquilar películas mediante el endpoint /alquilar, que registra la fecha y hora del alquiler.
    - La devolución de películas se maneja a través del endpoint /devolver, donde se calcula el costo adicional en caso de retraso.
4. **Visualización de Películas Alquiladas**
    - Los usuarios pueden ver las películas que han alquilado utilizando el endpoint /getRentedMovies, que compara los IDs de las películas alquiladas con los posts disponibles y muestra la información correspondiente.

**Implementación Técnica**

- **Node.js y Express**: Utilizados para crear el servidor backend que maneja las solicitudes HTTP y gestiona la lógica del negocio.
- **React**: Utilizado para construir una interfaz de usuario dinámica y responsiva.
- **JSON**: Utilizado como sistema de almacenamiento de datos simple y fácil de manejar.

- **Middlewares**: cors y body-parser para manejar las solicitudes HTTP de manera eficiente.

Esta solución está diseñada para ser escalable y fácil de mantener, permitiendo futuras ampliaciones y mejoras en la funcionalidad del sistema.

# LÓGICA DEL PROGRAMA

| Método | Dirección | Descripción | Body | Respuesta |
|---|---|---|---|---|
| POST | /login | Autenticar a los usuarios. | { "correo": "string", "password": "string" } | { "success": boolean, "user": object } |
| GET | /getPosts | Obtener todos los posts de películas. | | Array de objetos post |
| GET | /students | Obtener todos los estudiantes. | | Array de objetos student |
| GET | /students/ | Obtener un estudiante por ID. | | Objeto student o mensaje de error |
| GET | /getLastPostId | Obtener el último ID de post. | | { "lastId": number } |
| POST | /students | Crear un nuevo estudiante. | Objeto student | { "response": "Elemento creado correctamente." } |
| POST | /createPost | Crear un nuevo post de película. | Objeto post | { "response": "Publicación guardada correctamente." } |
| PUT | /students/ | Actualizar un estudiante por ID. | Objeto student | { "response": "Usuario actualizado correctamente" } |
| DELETE | /deletePost/ | Eliminar un post de película por ID. | | { "response": "Publicación eliminada correctamente" } |
| DELETE | /students/ | Eliminar un estudiante por ID. | | { "mensaje": "Usuario eliminado correctamente" } |
| GET | /getPost/ | Obtener un post de película por ID. | | Objeto post o mensaje de error |
| PUT | /updatePost/ | Actualizar un post de película por ID. | Objeto post | { "response": "Publicación actualizada correctamente" } |
| POST | /alquilar | Alquilar una película. | { "idPost": "string", "idUsuario": "string" } | { "response": "Película alquilada correctamente." } |
| GET | /getRentedMovies | Obtener las películas alquiladas. | | Array de objetos post con id_alquiler |
| POST | /devolver | Devolver una película. | { "idPost": "string", "idUsuario": "string", "returnDate": "date" } | { "message": "Película devuelta correctamente" } |

| | | | | |
|---|---|---|---|---|
| GET | /movies/ | Obtener un post de película por ID. | | Objeto post o mensaje de error |
| PUT | /movies/ | Actualizar un post de película por ID. | Objeto post | { "response": "Publicación actualizada correctamente" } |

# FLUJO DE LA APLICACIÓN

### Comunicación entre Frontend y Backend


Donde el usuario al interactuar con el frontend manda las peticiones al backend(Python) y retorna las respuestas al frontend.

# CRÉDITOS

Elaborado por el alumno Eleazar Neftalí Colop Colop para el curso de IPC1, en el país de Guatemala con fecha desde el 20 de junio de 2024 al 27 de junio de 2024.