CINE UTP

Nicolas Serna Gonzalez

Álvaro José Cotes Carmona

Universidad Tecnológica de Pereira

2024

**INTRODUCCION**

**Nombre del Proyecto**: Cine UTP

**Descripción General**: El proyecto Cine UTP consiste en una página web donde, al registrarse, los usuarios podrán acceder a un catálogo de películas. La plataforma está diseñada para mejorar la experiencia de exploración de películas, ofreciendo información detallada sobre cada título y funcionalidades adicionales para mejorar la interacción del usuario con el sistema.

**Objetivo Principal**: Proveer una herramienta intuitiva y moderna que permita a los usuarios de la comunidad UTP consultar, explorar y gestionar sus intereses relacionados con películas.

**Alcance**: A los **usuarios**: Registrarse, consultar un catálogo de películas, comprar entradas, verificar la disponibilidad de asientos en tiempo real y acceder a su historial de compras.

A los **administradores**: Gestionar inventarios de películas, horarios y la disponibilidad de asientos.

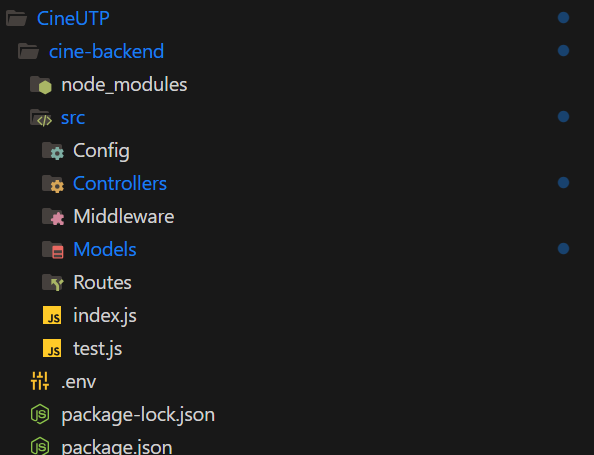
**Diseño del Sistema**

* **Arquitectura**: El sistema se basa en una arquitectura cliente-servidor:
* **Frontend**: Los frameworks utilizados para el desarrollo de las IU/UX fueron React.js y Boostrap.
* **Backend**: Para la parte de la lógica del sistema se utilizaron las siguientes tecnologías:
  + **Node.js**: Para ejecutar JavaScript fuera del navegador
  + **Express.js**: Framework para Node.js permite hacer conexiones entre paginas Web y APIs (Manejo de Rutas/Solicitudes)
  + **Mongoose:** Biblioteca de NODE.js para hacer la conexión entre la lógica del proyecto y la BD MongoDB.
  + **JWT:** Utilizado para la autenticación y autorización de la plataforma, para dar manejo a los usuarios administradores de la plataforma
  + **Bcrypt.js:** Para encriptar las contraseñas antes de almacenarlas en la BD.
  + **Axios:** Biblioteca para darle manejo a las solicitudes que se realizan desde el lado del cliente hacia la BD.
  + **Nodemon:** Herramienta que ejecuta la parte de la lógica cada vez que se guarda cambios en esta parte del proyecto.
* **Base de Datos**: MongoDB se utilizará para almacenar documentos JSON que representen usuarios, películas y transacciones.

**CINE UTP CONTEXTO DEL SISTEMA**

**CINE-BACKEND**

**Src (Componente Recursos) Componente de la lógica del**

**proyecto:**

* **Config/db.js/connectBD:**Conexión a la BD donde se apoya con  
  el archivo .env donde se encuentra la   
  conexión a la BD MongoDB y el puerto  
  que se define para que sea donde se   
  ejecuta esta parte del proyecto  
  En esta conexión se establecen una serie de   
  logs para poder identificar los siguientes datos de conexión:

**Servidor corriendo en puerto 5000:**

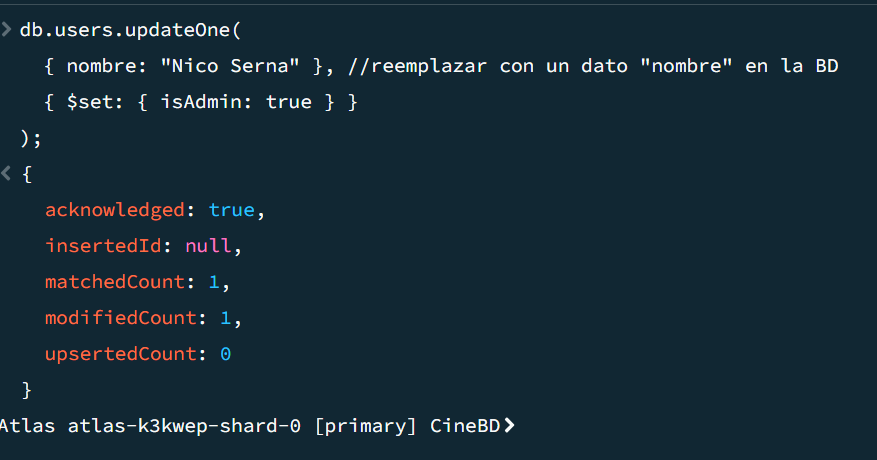
* + Mongoose conectado a MongoDB
  + MongoDB Conectado
  + Host: bd2cluster-shard-00-00.5d9jg.mongodb.net
  + Base de datos: CineBD
  + Estado: Conectado

Esto para poder identificar en que parte de la ejecución del back está fallando, si es un tema de la configuración del back o un tema de caída del servidor de mongoDB (Con esto se pudo identificar que cada vez que se accede a MongoDB Atlas con una dirección IP no configurada, se debe ir hasta MongoDB Atlas y configurar la dirección IP actual con la que se está accediendo para que la BD está disponible nuevamente).

* La parte de la lógica se ejecuta bajo la siguiente herramienta: Nodemon, con el comando nodemon src/index.js para esto se debe estar en la ruta cine-backend:

Ruta completa: /cine-backend> nodemon src/index.js

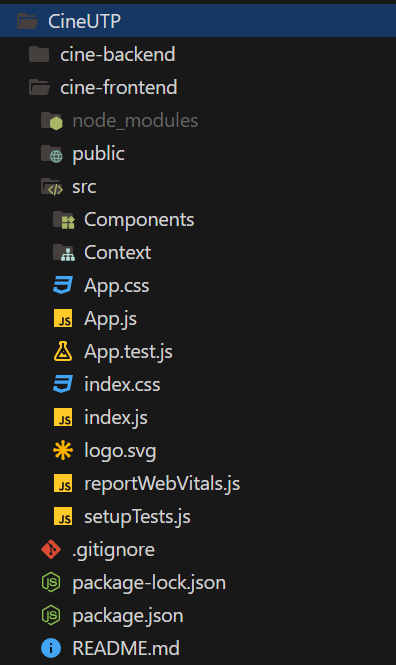
* **Controladores:**
  + logController.js: Controlador que maneja todas las validaciones y acceso que se tiene en el inicio de sesión de la plataforma, en este se incluyen validaciones como credenciales invalidas, error en el servidor (500) y validaciones con el token jwt.
  + movieController.js: Controlador para darle manejo a la parte de gestión de películas en la plataforma, donde se incluye el crud que se hace para las películas almacenadas en la BD, al igual, se añade un centralizador de mensajes de error para poder llevar un control de que es lo que está fallando en la operación, si es por parte del cliente o por parte del servidor (Interno).
  + userController.js: Controlador para darle manejo al elemento seleccionable “Mi perfil” donde se incluyen las funcionalidades de listar información del usuario, editar información de usuario, validaciones si se edita el correo por un existente.
* **Middleware:**
  + admin.js: Componente que verifica si el usuario que ingresa es admin, esto se logra con el parámetro definido desde base de datos isAdmin, el cual por defecto está definido como false, desde base de datos con el comando:

* + auth.js: Autentica a los usuarios mediante un token JWT. Extrae el token del encabezado de la solicitud, verifica su existencia y validez, y si es correcto, permite el acceso a rutas protegidas.
  + log.js: Se encarga de obtener y verificar el token (JWT) de la solicitud, comprueba si el token está presente y si es válido. Si el token es correcto, permite su uso en etapas posteriores de la solicitud.
* **Models**: Se definen las “entidades” u objetos que se van a utilizar para manejar la información, se define incialmente “Movies.js” y “Users.js” a lo cual se podría definir en una FASE 2 modelos como “Tickets.js” o “Rooms.js”, en estos modelos se definen las información que se va a manejar por cada objeto; Con los usuarios, datos como “nombre”, “email”, “preferencias”, etc. En películas datos como “titulo”, géneros [], imagen, fechaEstreno, etc. y se definen su tipo de dato: string, date, array [], Number, booleano, etc.
* **Routes:**
  + adminRoute.js: Valida si el usuario que está ingresando es administrador, en este archivo se implementan logs para facilitar la validación de la información que entra y como la toma el sistema, esto se desarrolla para tener un registro de la información que está siendo procesada, aunq tmbn se define que si el usuario no es admin, se dirija al home (Comportamiento normal), esto para evitar implementaciones adicionales, si no que particionar esta redireccion aprovechando que ya se está haciendo la validación de estos usuarios.
  + logRoute.js: Es el componente que nos proporciona las funciones de registrar e iniciar sesion desde el lado del cliente, por ende se establece las rutas de las funciones POST, se establece POST(login) por que en el login se define si el usuario que ingresó (sing) es admin (Cuenta con JWT) o es usuario normal.
  + MovieRoute.js: Es el componente que utiliza la pantalla del elemento “Gestionar películas” la cual establece las rutas que se utilizan desde el front para hacer el crud a la colección de documentos movies, en este se establece las funciones que se ejecutan a nivel publico, que sería la de obtener películas, las funciones POST, PUT y DELETE se definen únicamente para los usuarios que son administradores (isAdmin = true).
  + userRoute: Es el componente que define las rutas para obtener y editar la información de usuario en el elemento “Mi Perfil” desde el lado del cliente.
* Index.js: Es el corazón del lado back del proyecto, ya que con este se establece el enviroment que se tomará (Con esto se establece el .env que tomará la conexión a la BD “db.js”), se hace la conexión a la BD y se establece esta como almacenamiento de la información, se define el cors para que el acceso a esta lógica se pueda hacer desde otros dominios (Como el del front del proyecto) se establece express para la conexión cliente y procesamiento back, se establecen las rutas por las cuales se procesarán los componentes, las cuales se definen así:
  + <http://localhost:5000/api/Auth>/[Metodo]
  + <http://localhost:5000/api/Users>/[Metodo]
  + [http://localhost:5000/api/Movies/[Metodo](http://localhost:5000/api/Movies/%5bMetodo)]

Al igual, se define un manejo de errores para cuando una de estas llamdas al servidor que se hacen desde el lado del cliente, falla, por ejemplo cuando se está ingresando un parámetro que no corresponde al tipo o que se ingresa un dato no registrado, se podría decir que se define el error 500 cuando el servidor (back del proyecto) falla y el front no aplica ninguna restricción, en otras palabras este error se presenta cuando se presenta un error no catalogado. Por otro lado, también se incluyen logs para verificar el puerto en el que se está ejecutando la plataforma.

* Test.js: Se creó este archivo al principio del desarrollo del proyecto para realizar una validación de que la conexión a la BD estaba bien configurada, en esta están incluidos unos logs que se implementarían posteriormente en el archivo db.js, esto para que cada vez que se haga un cambio en la BD se mostraran estos logs, puesto que test.js se ejecuta de manera individual con npm start y estar sobre la ruta /test.js: /test.js> npm start.
* .env: Se implementa como una herramienta para seguir buenas prácticas en el desarrollo de software. Al estar ubicado fuera del directorio de código fuente (src), este archivo ayuda a proteger información sensible, como la URL de la base de datos MongoDB y la clave JWT, que solo deben ser accesibles para los usuarios administradores. De esta manera, se minimiza el riesgo de exposición de credenciales y se facilita la gestión de configuraciones específicas para diferentes entornos (desarrollo, pruebas, producción).

**CINE-FRONTEND**

En esta parte del proyecto lo que se implementó fueron las   
vistas para la UI, utilizando únicamente JavaScript mediante   
React.js para el diseño, para los estilos se utilizó boostrap   
para tener un estilado estético en la pagina y algo corto de   
implementar, esto para acortar tiempo estilando la pagina   
desde 0 como sería con css, scss o Tailwind.

* **Components:**
  + AdminMovies/moviesManagement.js: Esta es la   
    pantalla que se muestra al seleccionar la opción  
     “Gestionar películas” siendo administrador, en esta pantalla se puede visualizar tarjetas con la información mas relevante de la película como el titulo, la sinopsis, sus géneros, duración, etc. se tenía pensado en una posible FASE 2, agregar un elemento que al seleccionarlo el sistema despliegue un modal con toda la información que se pide en el formulario para agregar una película.Además, por cada tarjeta se habilitan un botón **eliminar** y uno **editar**, los cuales eliminan o editan la información de la película en la BD.  
    El botón editar, sigue la lógica que se implementa tanto en las pestañas de la pantalla mi perfil como el formulario que se utiliza para agregar películas (Reutilización de código), Para aclarar, al momento de seleccionar la opción editar el sistema despliega un formulario (Agregar película) con la información pre cargada de la película, con esto se asegura de que el formulario se muestre en vez de las películas en inventario, y evitando que el usuario ingrese toda la información nuevamente optimizando que este no consulte la información que ya se tenía.
  + AdminMovies/moviesFormPost.js: Es un componente que permite a los usuarios crear o editar películas, gestionando el estado del formulario y proporcionando una interfaz para ingresar y enviar datos de películas, para aclarar este es un componente que sirve tanto para añadir como para editar información de películas, puesto que este no se ejecuta bajo otra dirección url, tanto editar como agregar se ejecutan bajo <http://localhost:3000/admin/movies>, que es la dirección en la que se ejecuta “Gestionar películas”.
  + Auth/login.js: Pantalla que enseña al usuario el formulario de inicio de sesión, el cual utiliza para el reponse body el correo y la contraseña ingresadas, y el servicio al que apuntan es <http://localhost:5000/api/Auth/login>, el cual fue especificado anteriormente, además habilita el hipervínculo para pasar a la pantalla de registro “Regístrate”
  + Auth/register.js: Pantalla que enseña al usuario el formulario de registro de usuario, el cual utiliza para el reponse body el nombre, correo y la contraseña ingresadas, y el servicio al que apuntan es http://localhost:5000/auth/api/registro, el cual fue especificado anteriormente.
  + Profile/profile.js: Es un componente que actúa como la vista principal del perfil del usuario en la aplicación. Gestiona la carga de la información del usuario desde la API y permite la navegación entre diferentes secciones, como "Mi Perfil", "Preferencias" y "Historial de Compras". Dependiendo de la pestaña activa, muestra la información del usuario, que incluye la opción de editarla a través del componente userInfo, así como otras funcionalidades relacionadas con las preferencias y el historial de compras. Además, maneja el estado de carga y errores durante la obtención de datos.
  + Profile/userInfo.js: Es un componente que trae un formulario con algunos datos del usuario registrado, además de habilitar un botón para editar esta información que se visualiza, esto esta configurado para que al guardar automáticamente guarde los cambios en BD y al instante se vea esa información cambiada en este componente inicialmente.
  + Profile/ history.js y preference.js: Son controladores que traen la información relacionada a las preferencias configuradas al usuario y el historial de compras (Reserva de asientos) que tenga el usuario registrado en BD.  
    Al seleccionar preferences.js este trae una serie de acciones seleccionables para configurar las preferencias del usuario, como: Los géneros, idioma preferido y una posible acción de enviar notificaciones al correo registrado por el usuario.
  + Home.js: Es la pagina de inicio que se visualiza al proseguir del formulario de inicio de sesión o registro de usuario, en este se listan las películas registradas en el inventario de películas, en este componente se visualizan por primera vez los botones de acción “Mi perfil”, “Cerrar sesión”, “Inicio”, “Gestionar películas”(Solo admin).  
    El botón inicio hace que el sistema asegura dirigir al usuario a la ruta especificada para este componente (home.js).  
    El botón Cerrar sesión, dirige al usuario al formulario de inicio de sesión y asegura que por ningún método el usuario pueda volver a la pantalla anterior sin volver a iniciar sesión, esto se logra con los métodos de autenticación y método de contexto.
  + Context/logContext.js: Este componente (Método de contexto) gestiona la autenticación del usuario en la aplicación, permitiendo verificar el estado de autenticación, iniciar sesión y cerrar sesión, y proporcionando esta información a otros componentes a través del contexto de React.
* **App.js**: Archivo en el que se define la ejecución del front del programa para el lado del cliente, en este se definen los elementos que perduren a lo largo de la plataforma, como en este caso el navbar, el cual se visualiza en todo momento de la plataforma.  
  Además se define cuando se ejecuta el front por medio de “npm start” cuando se esté sobre la ruta de cine-frontend:

/cine-frontend>npm start

El cual se actualiza constantemente al estar ejecutándose en base al back que se re ejecuta cada vez que se realiza un cambio gracias a la herramienta nodemon

Este proyecto es una aplicación de gestión de películas que permite a los usuarios explorar, registrar y administrar información sobre diversas películas. La aplicación está dividida en dos partes principales: el frontend, desarrollado en React, y el backend, construido con Node.js y MongoDB.

Las funcionalidades implementadas en la plataforma cubren los siguientes objetivos propuestos:

* **Autenticación de Usuarios**: Los usuarios pueden registrarse e iniciar sesión, lo que les permite acceder a funciones personalizadas.
* **Catálogo de Películas:** Los usuarios pueden ver un catálogo de películas disponibles, con detalles como título, sinopsis, géneros, duración, director y estado.
* **Perfil de Usuario:** Los usuarios pueden ver y editar su información personal, incluyendo nombre, email y teléfono.
* **Gestión de Películas para Administradores:** Los administradores tienen acceso a funcionalidades para agregar, editar y eliminar películas de la base de datos.

A pesar de que se implementaron varias funcionalidades clave, no se alcanzó a implementar todos los objetivos propuestos. Entre las características que no se completaron se incluyen:

* **Sistema de Reserva de Asientos por Película:** Esta funcionalidad, que permitiría a los usuarios reservar asientos para las proyecciones de películas, no fue implementada.
* **Filtrado por Género en la Página Principal:** La opción de filtrar las películas por género en la página principal no se logró, lo que limita la capacidad de los usuarios para encontrar películas específicas según sus preferencias.

El proyecto proporciona una base sólida para la gestión de películas y la interacción del usuario, aunque hay áreas de mejora y funcionalidades adicionales que podrían ser implementadas en futuras iteraciones. La aplicación demuestra un enfoque en la usabilidad y la organización del contenido, y sienta las bases para un sistema más completo en el futuro.

Pd: Junto a la documentación técnica, se adjunta un video demostrativo de la página, y un documento con los requerimientos.