Documentación del Backend - Aplicación de Cine

Índice

Descripción General del Proyecto

Servicio backend desarrollado en Node.js para una aplicación de cine que proporciona información sobre películas, autenticación de usuarios y gestión de contenido cinematográfico. El sistema se integra con la API de TMDB y soporta múltiples métodos de autenticación.

Stack Tecnológico

- **Entorno de Ejecución**: Node.js 20
- **Framework**: Express.js
- **Base de Datos**: MongoDB
- **Autenticación**: JWT, Passport.js (OAuth de Google)
- **Plataforma de Contenedores**: Docker
- **Gestor de Paquetes**: npm
- **Integración de API**: TMDB API

Estructura del Proyecto

```
backend/
├── config/
│   ├── cookieConfig.js    # Configuración de cookies
│   ├── cors.js           # Configuración de CORS
│   ├── db.js            # Conexión a base de datos
│   └── passport.js      # Estrategias de autenticación
├── controllers/
│   ├── authController.js  # Lógica de autenticación
│   ├── movieController.js # Gestión de películas
│   └── userController.js  # Gestión de usuarios
├── middlewares/
│   └── authMiddleware.js  # Verificación de JWT
├── models/
│   ├── Movie.js          # Esquema de películas
│   └── User.js           # Esquema de usuarios
├── routes/
│   ├── authRoutes.js     # Rutas de autenticación
│   ├── movieRoutes.js    # Rutas de películas
│   └── userRoutes.js     # Rutas de usuarios
│
├── server.js             # Punto de arranque del servidor
└── app.js                # Punto de entrada de la aplicación
```

Sistema de Autenticación

Métodos de Autenticación

1. **Autenticación Local**
    - Autenticación basada en JWT
    - Encriptación de contraseñas con bcrypt
    - Gestión segura de cookies
2. **Proveedores OAuth**
    - OAuth 2.0 de Google

Características de Seguridad

- Cookies HTTP-only
- Protección CORS
- Expiración de tokens JWT
- Encriptación de contraseñas
- Gestión segura de sesiones

Puntos destacables de este backend

Sistema de Autenticación Multicapa

Autenticación local

```
// authController.js
const login = async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email: email.toLowerCase()
}).select('+password');

  // Verificación segura de contraseña usando bcrypt
  const isValidPassword = await user.comparePassword(password);
```

Integración con Google OAuth

```
// passport.js
passport.use(
  new GoogleStrategy({
    clientID: process.env.GOOGLE_CLIENT_ID,
    clientSecret: process.env.GOOGLE_CLIENT_SECRET,
    // ...
  })
```

Gestión de Películas y API Externa

```
// movieController.js
export const fetchFromTMDB = async (endpoint, params = {}) => {
  const queryParams = new URLSearchParams({
    api_key: TMDB_API_KEY,
    language: "es-ES",
    ...params,
  });
```

Sistema de Reseñas y Favoritos

```
// userController.js
const toggleFavorite = async (req, res) => {
  const { movieId } = req.params;
  const user = await User.findById(req.userId);
  const movieIdNum = Number(movieId);
  const isFavorite = user.favoriteMovies.includes(movieIdNum);
```

Seguridad Robusta

CORS

```
// cors.js
const corsOptions = {
  origin: (origin, callback) => {
    if (allowedOrigins.includes(origin) || process.env.NODE_ENV === "production")
{
      return callback(null, origin);
    }
  },
  credentials: true,
```

Cookies

```
// cookieConfig.js
export const cookieConfig = {
  httpOnly: true,
  secure: process.env.NODE_ENV === "production",
  sameSite: process.env.NODE_ENV === "production" ? "strict" : "lax",
```

Guía de Pruebas con Thunder Client

Configuración del Entorno

1. Crear un nuevo entorno en Thunder Client
2. Configurar variables:

```
TOKEN: [se actualizará automáticamente tras login]
```

Rutas Auth

1. **Registro de Usuario**

- Método: POST
- URL: `http://localhost:4000/api/auth/register`
- Descripción: Registra un nuevo usuario recibiendo username, email y password, y devuelve un mensaje de confirmación al crear la cuenta exitosamente.
- Body:

```
{
  "username": "ejemplo",
  "email": "ejemplo@email.com",
  "password": "contraseña123"
}
```

- Response:

```
{
    "message": "Usuario registrado exitosamente"
}
```

1. **Inicio de Sesión**

- Método: POST
- URL: `http://localhost:4000/api/auth/login`
- Descripcion: Autentica a un usuario utilizando su correo electrónico y contraseña, devolviendo un token JWT y los datos del usuario al iniciar sesión exitosamente.
- Headers:

```
Content-Type: application/json
```

- Body:

```
{
  "email": "ejemplo@email.com",
  "password": "contraseña123"
}
```

- Response:

```
{
"message": "Inicio de sesión exitoso",
"token": "JWT",
"user": {
    "id": "67b836302d55b51fa48ee8a7",
    "username": "ejemplo",
    "email": "ejemplo@email.com",
    "authType": "local"
    }
}
```

2. **Cierre de Sesión**

- Método: POST
- URL: `http://localhost:4000/api/auth/logout`
- Descripcion: Autentica a un usuario utilizando su correo electrónico y contraseña, devolviendo un token JWT y los datos del usuario al iniciar sesión exitosamente.
- Headers:

```
Content-Type: application/json
```

- Body:

```
{
  "email": "ejemplo@email.com",
  "password": "contraseña123"
}
```

- Response:

```
  {
  "message": "Inicio de sesión exitoso",
  "token": "JWT",
  "user": {
      "id": "67b836302d55b51fa48ee8a7",
      "username": "ejemplo",
      "email": "ejemplo@email.com",
      "authType": "local"
      }
  }
```

2. **Inicio de Sesión OAuth**

- Método: GET
- URL: `http://localhost:4000/api/auth/google`
- Descripción: Inicia el flujo OAuth con Google, redirigiendo al usuario a la página de login de Google y, tras autenticarse, retorna un token JWT junto con los datos del usuario.
- No requiere headers ni body ya que redirige al flujo de OAuth de Google

El usuario es redirigido a la página de login de Google

Después de autenticarse en Google, el usuario es redirigido a:
`http://localhost:4000/api/auth/google/callback`

Finalmente, el usuario es redirigido a: `http://localhost:5173/login?token={JWT}`

- Response:

```
{
    "message": "Inicio de sesión exitoso",
    "token": "JWT",
    "user": {
        "id": "67b836302d55b51fa48ee8a7",
        "username": "Nombre de Google",
```

```
        "email": "usuario@gmail.com",
        "authType": "google",
        "googleId": "google_id_number"
    }
}
```

Error Response (401):

```
{
    "message": "Error en autenticación con Google",
    "status": 401,
    "timestamp": "2024-02-20T16:45:32Z"
}
```

> [!NOTE] Notas importantes:
>
> A diferencia del login tradicional, este es un flujo de redirección OAuth. El token JWT se envía como query parameter en la URL de redirección final. No requiere envío de credenciales directamente ya que las maneja Google. El campo authType en la respuesta será "google" en lugar de "local"

Rutas para User

3. **Obtener el perfil de usuario**

- Método: GET
- URL: `http://localhost:4000/api/users/me`
- Descripción: Recupera el perfil del usuario autenticado, devolviendo su ID y nombre de usuario mediante el token proporcionado
- Authorization: `Bearer {token}`
- Response:

```
{
    "id": "67b836302d55b51fa48ee8a7",
    "username": "ejemplo"
}
```

4. **Obtener reseñas de usuario**

- Método: GET
- URL: `http://localhost:4000/api/users/reviews`
- Authorization: `Bearer {token}`
- Response:

```
[
  {
    "movieId": 939243,
    "content": "hewhbewfhbfewhbjferwhjbferwbhewferbj",
    "rating": 5,
    "_id": "67b9a74015310860a7bebf07",
    "createdAt": "2025-02-22T10:30:24.432Z",
    "movie": {
      "title": "Sonic 3: La película",
      "poster_path": "/3aDWCRXLYOCuxjrjiPfLd79tcI6.jpg",
      "release_date": "2024-12-19",
      "vote_average": 7.8,
      ...
    }
  }
]
```


6. **Alternar añadir/eliminar peliculas favoritas**

- Método: POST
- URL: `http://localhost:4000/api/users/favorites/603`
- Authorization: `Bearer {token}`
- Respuestas:


```
    {
        "message": "Película añadida a favoritos",
        "isFavorite": true
    }

    ó

    {
        "message": "Película eliminada de favoritos",
        "isFavorite": false
    }
```


1. **Mostrar reseñas del usuario**

- Método: GET
- URL: `http://localhost:4000/api/users/reviews`
- Descripción:
- Respuesta:

```
[
  {
    "movieId": 939243,
    "content": "hewhbewfhbfewhbjferwhjbferwbhewferbj",
    "rating": 5,
    "_id": "67b9a74015310860a7bebf07",
    "createdAt": "2025-02-22T10:30:24.432Z",
    "movie": {
      "title": "Sonic 3: La película",
      "poster_path": "/3aDWCRXLYOCuxjrjiPfLd79tcI6.jpg",
      "release_date": "2024-12-19",
      "vote_average": 7.8
    }, ....
  }
]
```


Rutas para Movies

1. **Búsqueda de Películas Populares**

- Método: GET
- URL: `http://localhost:4000/api/movies/popular`
- Descripción
- Respuesta:


```
{
  "page": 1,
  "results": [
    {
      "id": 762509,
      "title": "Mufasa: El rey león",
      "poster_path": "/dmw74cWIEKaEgl5Dv3kUTcCob6D.jpg",
      "release_date": "2024-12-18",
      "vote_average": 7.4,
      ...
    },
    {
      "id": 950396,
      "title": "El abismo secreto",
      "poster_path": "/3s0jkMh0YUhIeIeioH3kt2X4st4.jpg",
      "release_date": "2025-02-13",
      "vote_average": 7.8,
      ...
    },...
  ],
  "total_pages": 48816,
  "total_results": 976312
}
```

2. **Búsqueda de Películas por nombe y/o página**

- Método: GET
- URL: `http://localhost:4000/api/movies/search?query=matrix`
- Descripción: Esta url obtiene un listado de peliculas que coinciden con el texto introducido en el input, si se introduce "matrix" aparecerán todas las pelúclas que conténgan esta palabra en el título.
- URL: `http://localhost:4000/api/movies/search?query=matrix&page=2`
- Descripción: Esta url obtiene un listado de peliculas que coinciden con el texto introducido en el input pero solo las contenidas en la página 2, si no se indica página, por defecto es la 1.

> [!NOTE] Ambas muestran el total de páginas que contienen peliculas con un titulo similar al introducido y la cantidad exacta de películas que contienen esa palabra.

- Respuesta:


```
{
  "page": 1,
  "results": [
    {
      "id": 603,
      "title": "The Matrix",
      "poster_path": "/2P14LNK2zDBf84tF016blkz4Q4C.jpg",
      "release_date": "1999-03-31",
      "vote_average": 8.221
    },
    {
      "id": 555879,
      "title": "Matrix",
      "poster_path": "/AfFD10ZqEx2vkxM2yvRZkybsGB7.jpg",
      "release_date": "1998-12-31",
      "vote_average": 7.083
    },...
  ],
  "total_pages": 5,
  "total_results": 87
}
```


1. **Mostrar detalles de una película**

- Método: GET
- URL: `http://localhost:4000/api/movies/603`
- Descripción: Permite obtener información detallada de una película a través de su identificador.
- Respuesta:

```
{
  "id": 603,
  "title": "Matrix",
  "poster_path": "/2P14LNK2zDBf84tF016blkz4Q4C.jpg",
  "release_date": "1999-03-31",
  "vote_average": 8.2,
  "overview": "Thomas Anderson lleva una doble vida: por el día es programador y
por la noche un hacker llamado Neo. Su vida cambia al descubrir la verdadera
naturaleza de Matrix.",
  "tagline": "Bienvenido al mundo real.",
  "runtime": 138
}
```


4. **Añadir una reseña a una película**

- Método: POST
- URL: `http://localhost:4000/api/movies/603/review`
- Descripción: Permite agregar una reseña a la película indicada por id. Requiere autenticación y espera un JSON con el contenido de la reseña y la calificación, devolviendo un mensaje de confirmación
- Authorization: `Bearer {token}`
- Body:


```
{
  "content": "Excelente película",
  "rating": 5
}
```


- Response:


```
    {
        "message": "Reseña añadida correctamente"
    }
```


5. **Mostrar reseñas de una película**

- Método: GET
- URL: `http://localhost:4000/api/movies/603/reviews`
- Descripción: Muestra las reseñas de una película indicada por id.
- Respuesta:

```
[
  {
    "username": "ana",
    "content": "Excelente película",
    "rating": 5,
    "createdAt": "2025-02-21T09:19:21.718Z"
  }
]
```

7. **Obtener géneros**

- Método: GET
- URL: `http://localhost:4000/api/movies/genres`
- Authorization: `Bearer {token}`
- Respuestas:

```
 {
"genres": [
  {
    "id": 28,
    "name": "Acción"
  },
  {
    "id": 12,
    "name": "Aventura"
  },
  {
    "id": 16,
    "name": "Animación"
  },
  ...
]
}
```