

METODOS DE PLAYLIST

Método: Crear Playlist

Ruteo de endpoint: `router.post('/create-playlist', verifyToken, (req,res) => addPlaylistToUser(req,res));`

Petición: POST con body conformado por {name, movie}, o simplemente {name}.

Respuesta de verifyToken:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** { auth: false, message: 'No token provided' } si no se encuentra un token.
- **Estado 400:** { message:'Invalido token' } si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** { message: 'Playlist creada exitosamente', id: playlist_id } si la respuesta es exitosa.
- **Estado 409:** { message: 'Ya existe una playlist con ese nombre', id: playlist_id} si la lista de reproducción ya existe y no fue posible crearla.
- **Estado 500:** { message: 'Error interno del servidor' } si hubiera ocurrido un error interno del servidor.

Método: Agregar Película a Playlist

Ruteo de endpoint: `router.post('/add-movies-playlist', verifyToken, (req,res) => addMoviesToPlaylist(req,res));`

Petición: POST con query `_id_playlist` y body conformado por `{movie}`.

Respuesta de `verifyToken`:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** { auth: false, message: 'No token provided' } si no se encuentra un token.
- **Estado 400:** { message:'Invalido token' } si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** { message: 'Contenido añadido exitosamente a la playlist.', id: playlist._id } si la respuesta es exitosa.
- **Estado 409:** { message: 'La película ya está en la playlist "{Playlist.name}"', id: movie } si la película ya está en la lista de reproducción.
- **Estado 422:** { error: 'Usuario no encontrado' } si el usuario no es encontrado.
- **Estado 404:** { error: 'Playlist no encontrada' } si la playlist no es encontrada.
- **Estado 500:** { error: 'Ocurrió un error en el servidor' } si hubiera ocurrido un error interno del servidor.

Método: Eliminar Playlist

Ruteo de endpoint: `router.delete('/delete-playlist', verifyToken, (req,res) => deletePlaylistFromUser(req,res));`

Petición: DELETE con query `_id_playlist`.

Respuesta de `verifyToken`:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** { auth: false, message: 'No token provided' } si no se encuentra un token.
- **Estado 400:** { message:'Invalido token' } si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** 'Playlist deleted successfully.' si la respuesta es exitosa.
- **Estado 404:** 'Playlist not found in user account.' si la playlist no es encontrada.
- **Estado 422:** 'User not found.' si el usuario no es encontrado.
- **Estado 500:** 'Server error occurred.' si hubiera ocurrido un error interno del servidor.

Método: Eliminar Película de Playlist

Ruteo de endpoint: `router.delete('/delete-movies-playlist', verifyToken, (req,res) => deleteMoviesFromPlaylist(req,res));`

Petición: DELETE con query `_id_playlist` y body conformado por `{movie}`.

Respuesta de `verifyToken`:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** `{ auth: false, message: 'No token provided' }` si no se encuentra un token.
- **Estado 400:** `{ message:'Invalido token' }` si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** `{ message: 'Películas eliminadas exitosamente de la playlist.' }` si la respuesta es exitosa.
- **Estado 400:** `{ error: 'No se encontraron películas para eliminar de la playlist.' }` si no se encuentran las películas a eliminar.
- **Estado 404:** `{ error: 'Playlist no encontrada' }` si la playlist no es encontrada.
- **Estado 422:** `{ error: 'Usuario no encontrado' }` si el usuario no es encontrado.
- **Estado 500:** `{ error: 'Ocurrió un error en el servidor' }` si hubiera ocurrido un error interno del servidor.

Método: Modificar Nombre de Playlist

Ruteo de endpoint: `router.post('/modify-name-playlist', verifyToken, (req,res) => modifyNamePlaylist(req,res));`

Petición: POST con query `_id_playlist` y body conformado por `{name}`.

Respuesta de `verifyToken`:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** `{ auth: false, message: 'No token provided' }` si no se encuentra un token.
- **Estado 400:** `{ message:'Invalido token' }` si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** `{ message: 'Nombre de la playlist actualizado correctamente' }` si la respuesta es exitosa.
- **Estado 404:** `{ error: 'Playlist no encontrada' }` si la playlist no es encontrada.
- **Estado 422:** `{ error: 'Usuario no encontrado' }` si el usuario no es encontrado.
- **Estado 500:** `{ error: 'Ocurrió un error en el servidor' }` si hubiera ocurrido un error interno del servidor.

METODOS DE USUARIO

Método: Crear Usuario

Ruteo de endpoint: `router.post('/create-user', (req,res) => createUser(req,res));`

Petición: POST con body conformado por {username, email, password}.

Respuesta del metodo:

- **Estado 200:** { data: User, message: 'Account created successfully' } si la respuesta es exitosa.
- **Estado 422:** `json(err)` si hubo un error en la creación del usuario.

Método: Validar Login

Ruteo de endpoint: `router.post('/validate-login', (req,res) => validateLogin(req,res));`

Petición: POST con body conformado por {email, password}.

Respuesta del metodo:

- **Estado 200:** { username: User.username, playlists: User.playlists } y setea una cookie de sesión si la respuesta es exitosa.
- **Estado 422:** 'password not valid' si la contraseña no es válida.
- **Estado 422:** 'User not found' si el usuario no es encontrado.

Método: Validar Email

Ruteo de endpoint: `router.post('/validate-email', (req,res) => validateEmail(req,res));`

Petición: POST con body conformado por { `_email` }.

Respuesta del metodo:

- **Estado 200:** { message: 'El usuario está registrado' } si el email está registrado.
- **Estado 404:** { message: 'El usuario no está registrado' } si el email no está registrado.
- **Estado 500:** { message: 'Error al validar el email' } si hubo un error al validar el email.

Método: Recuperar Contraseña

Ruteo de endpoint: `router.put('/password-recover', (req,res) => passwordRecover(req,res));`

Petición: PUT con body conformado por { _email, new_password }.

Respuesta del metodo:

- **Estado 200:** { message: 'Password updated successfully' } si la respuesta es exitosa.
- **Estado 404:** { message: 'User not found' } si el usuario no es encontrado.
- **Estado 500:** { message: 'Error updating password' } si hubo un error al actualizar la contraseña.

Método: Obtener Perfil de Usuario

Ruteo de endpoint: `router.get('/get-profile', verifyToken, (req,res) => getProfile(req,res));`

Petición: GET.

Respuesta de verifyToken:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** { auth: false, message: 'No token provided' } si no se encuentra un token.
- **Estado 400:** { message:'Invalido token' } si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** { username: User.username, playlists: User.playlists } si la respuesta es exitosa.
- **Estado 422:** si el usuario no es encontrado.

Método: Logout

Ruteo de endpoint: `router.delete('/logout', verifyToken, (req,res) => deleteToken(req,res));`

Petición: DELETE.

Respuesta de verifyToken:

- **next():** Si el token pudo ser verificado y se puede avanzar con la ejecución de la próxima función.
- **Estado 401:** { auth: false, message: 'No token provided' } si no se encuentra un token.
- **Estado 400:** { message:'Invalido token' } si se encuentra un token pero este es invalido.

Respuesta del metodo:

- **Estado 200:** true y setea una cookie de duración 1 para reemplazar el token actual si la respuesta es exitosa.

MÉTODOS DE API

Método: Fetch Movies

Ruteo de endpoint: `router.get('/fetch-movies', (req,res) => fetchMovies(req,res));`

Petición: GET con parámetros de query conformados por { genres, title }.

Respuesta del metodo:

- **Estado 200:** JSON con un array de objetos que contiene detalles de las películas, incluyendo trailers.
- **Estado 400:** { message: 'Error fetching movies', error } si hubo un error al obtener las películas.

Método: Fetch Movie By ID

Ruteo de endpoint: `router.get('/fetch-movies-by-id', (req,res) => fetchMovieById(req,res));`

Petición: GET con parámetros de query conformados por { id }.

Respuesta del metodo:

- **Estado 200:** JSON con un objeto que contiene detalles de la película, incluyendo trailer.
- **Estado 400:** { message: 'Error fetching movie details', error } si hubo un error al obtener los detalles de la película.