Universidad Tecmilenio

Campus Las Torres

Materia: Desarrollo FullStack

# Actividad 3 – API RESTful utilizando Node.js y Express.js

Alumno: Patricio Calvo

Matricula: 07097795

Febrero / 2026

Link de Github:

https://github.com/Diablo650cc/Desarrollo-FullStack/tree/main/ACT%203

**REPORTE**

**Configuración Inicial**
Se creó la carpeta "api-tareas" y se inicializó con npm init -y. Se instalaron las dependencias: express, body-parser, jsonwebtoken y bcryptjs. Esta base permitió construir el servidor y las funcionalidades de seguridad.

**Servidor con Express.js**
Se configuró el archivo server.js con un servidor Express escuchando en puerto 3000. Se incluyó middleware para parsear JSON y un sistema de logging que registra cada petición (método, URL, headers y body), facilitando el debugging.

**Almacenamiento con Módulo fs**
Se utilizó fs.promises para operaciones asincrónicas no bloqueantes. Se crearon funciones obtenerTareas() y guardarTareas() para leer/escribir en tareas.json. Similarmente, obtenerUsuarios() y guardarUsuarios() para manejar usuarios.json. Esto garantiza persistencia sin afectar el Event Loop.

**Rutas CRUD**

- GET /tareas: Devuelve todas las tareas del archivo JSON.
- POST /tareas: Crea nueva tarea con ID único (timestamp), fecha de creación y la guarda.
- PUT /tareas/:id: Actualiza tarea existente, agregando fecha de actualización. Error 404 si no existe.
- DELETE /tareas/:id: Elimina tarea por ID. Error 404 si no se encuentra.

**Sistema de Autenticación**

- POST /register: Recibe email y contraseña. Verifica que el email no exista, encripta contraseña con bcryptjs (10 rondas de sal) y guarda el usuario.
- POST /login: Verifica credenciales. Si son válidas, genera token JWT con ID y email del usuario, válido por 2 horas. El token debe incluirse en peticiones posteriores.

**Middleware de Autenticación**
La función autenticarToken extrae el token del header Authorization, verifica su validez con jwt.verify() y adjunta los datos del usuario a req.user. Si no hay token, responde 401; si es inválido, 403.

Todas las rutas de tareas incorporan este middleware, quedando protegidas. Al crear tareas, se asocia el userId del token para futuras personalizaciones.

**Manejo de Errores y Debugging**
Se implementó un sistema de errores de tres capas:

- Middleware de logging: Registra cada petición con detalles completos.
- Try-catch en rutas: Captura errores y los pasa al manejador con next(error).
- Middleware 404: Captura rutas no definidas.
- Manejador central de errores: Con cuatro parámetros, registra el error en consola, determina código HTTP (400, 401, 404, 500) y envía respuesta JSON estructurada. En desarrollo incluye stack trace.

Se agregó ruta /error-test para probar el sistema.

**Validación de Datos**

- Registro: Valida email y contraseña no vacíos, y email no duplicado.
- Login: Valida que ambos campos estén presentes.
- Las rutas CRUD pueden extenderse fácilmente para requerir título y descripción.

**Herramientas de Debugging**

- console.log() estratégicos
- Middleware de logging automático
- Node.js Inspector (node --inspect server.js)
- Postman para pruebas de endpoints

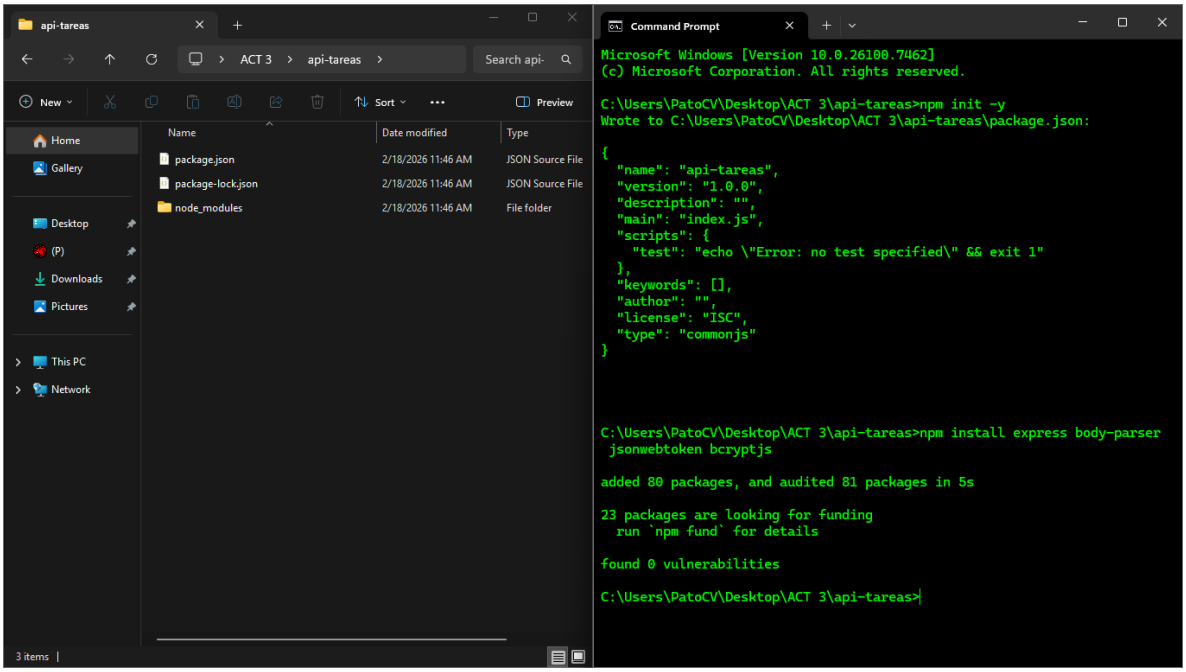
**Resultado Final**

- La API cumple todos los requisitos:
- Servidor Express en puerto 3000
- CRUD completo con almacenamiento JSON
- Autenticación segura con bcrypt y JWT
- Rutas protegidas por middleware
- Manejo centralizado de errores
- Validación básica de datos
- Herramientas de debugging integradas

**Conclusión**

El proyecto demostró la implementación práctica de conceptos clave de Node.js: Event Loop, asincronía con async/await, manejo de archivos, middleware en Express, y autenticación segura. La estructura modular y el manejo de errores facilitan el mantenimiento y escalabilidad futura.

# A. CONFIGURAR EL PROYECTO NODE.JS




# B. CREACIÓN DE UN SERVIDOR BÁSICO CON EXPRESS.JS.

## C. CREAR LAS RUTAS PARA LA API RESTFUL

```
    // RUTAS PARA LA API RESTFUL
    // LAS RUTAS A CONTINUACION EXPONEN LOS ENDPOINTS DE LA API (PELÍCULAS)


    peliculas.push(nuevaPelicula);
    await guardarPeliculas(peliculas);
    res.status(201).json(nuevaPelicula);
  } catch (error) {
    next(error);
  }
});
```

## D. MANEJO DE DATOS CON EL MÓDULO FS

```
// MANEJO DE DATOS CON EL MODULO FS
// LAS FUNCIONES ABAJO REALIZAN LECTURA/ESCRITURA EN ARCHIVOS JSON

async function obtenerPeliculas() {
  try {
    const data = await fs.readFile(PELICULAS_FILE, 'utf8');
    return JSON.parse(data);
  } catch {
    return [];
  }
}
```

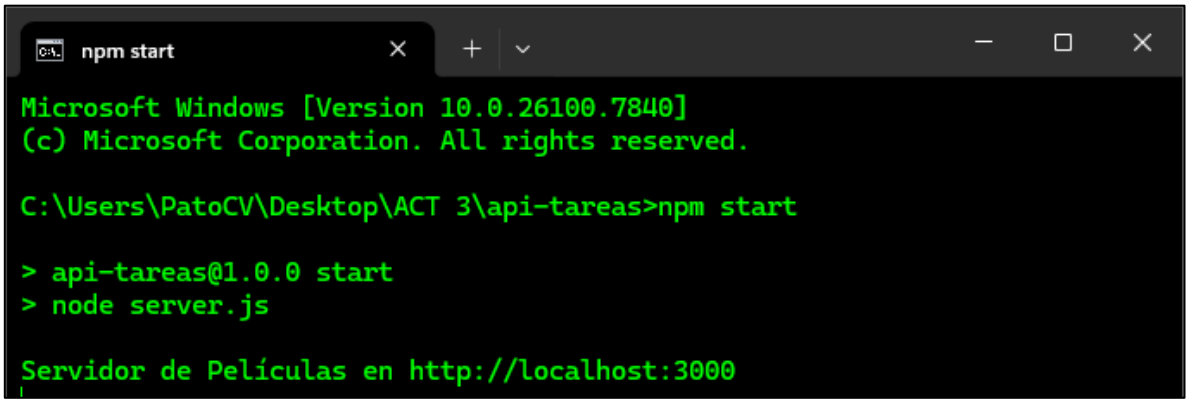
## E. IMPLEMENTACIÓN DE AUTENTICACIÓN Y SESIONES

```
// Usuarios (auth)
async function obtenerUsuarios() {
  try {
    const data = await fs.readFile(USERS_FILE, 'utf8');
    const parsed = JSON.parse(data);
    return Array.isArray(parsed) ? parsed : [];
  } catch {
    return [];
  }
}

async function guardarUsuarios(usuarios) {
  await fs.writeFile(USERS_FILE, JSON.stringify(usuarios, null, 2));
}

function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.status(401).json({ error: 'Token requerido' });
  try {
    const payload = jwt.verify(token, JWT_SECRET);
    req.user = payload;
    next();
  } catch (err) {
    return res.status(403).json({ error: 'Token inválido' });
```

## F. MANEJO DE ERRORES Y DEBUGGING

```
211  // MANEJO DE ERRORES Y DEBUGGING
212  // EL MIDDLEWARE SIGUIENTE CAPTURA ERRORES Y DEVUELVE CÓDIGO HTTP
213  app.use((err, req, res, next) => {
214    const statusCode = err.status || 500;
215    res.status(statusCode).json({ error: err.message });
216  });
217
218  app.listen(PORT, () => {
219    console.log(`Servidor de Películas en http://localhost:${PORT}`);
220  });
```

# PRUEBAS DE FUNCIONAMIENTO

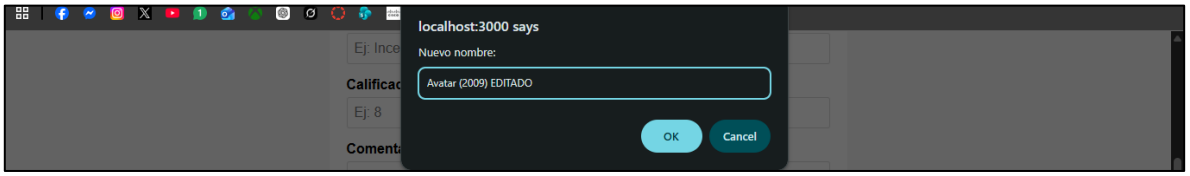


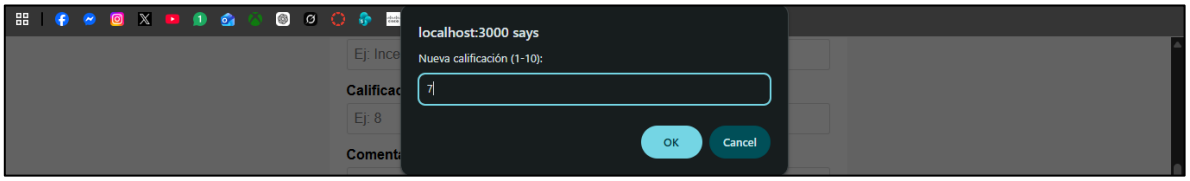


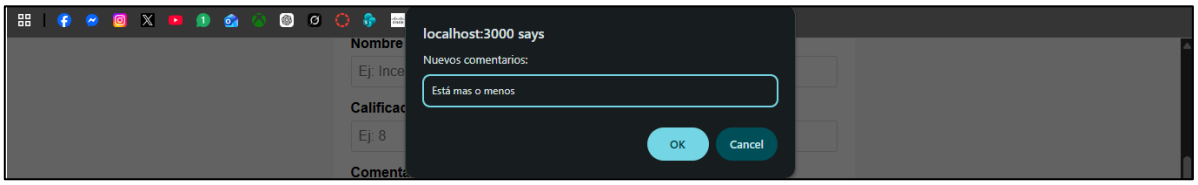


Películas Vistas
Película agregada
Logout
Agregar Película
Nombre de la película:
Ej: Inception
Calificación (1-10):
Ej: 8
Comentarios:
Ej: Excelente película, muy recomendada
Agregar Película

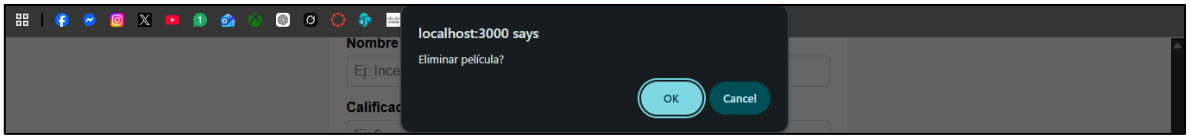

Mis Películas
Avatar (2009)
Calificación: 9/10
Revolucionó el cine en 3D y efectos visuales. Visualmente impresionante.
Agregado: 2/18/2026, 4:45:08 PM
Editar
Eliminar

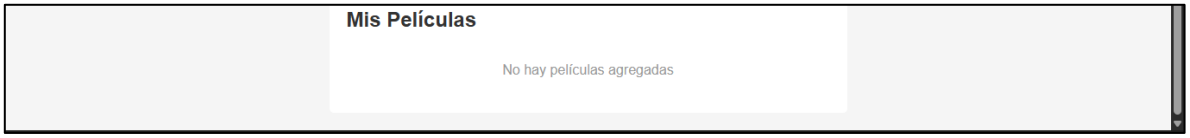

localhost:3000 says
Nuevo nombre:
Avatar (2009) EDITADO
OK
Cancel

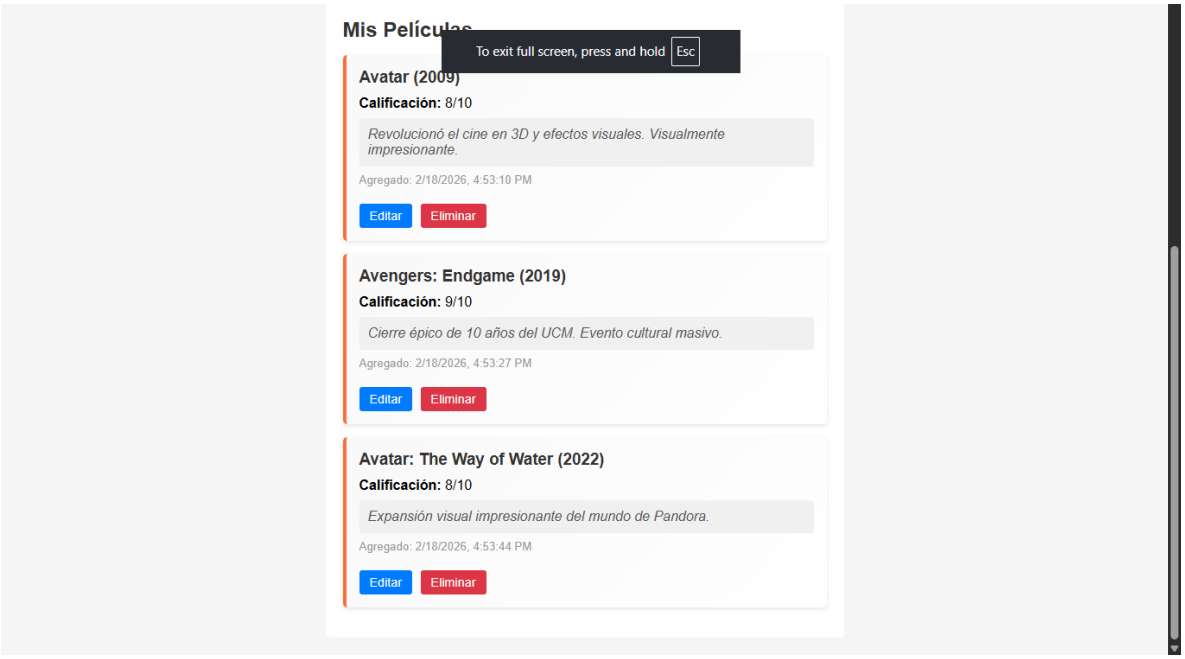

localhost:3000 says
Nueva calificación (1-10):
7
OK
Cancel


localhost:3000 says
Nuevos comentarios:
Está mas o menos
OK
Cancel


Mis Películas
Avatar (2009) EDITADO
Calificación: 7/10
Está mas o menos
Agregado: 2/18/2026, 4:45:08 PM
Editar
Eliminar


Nombre
Ej: Ince
Calificac
localhost:3000 says
Eliminar película?
OK
Cancel


Mis Películas
No hay películas agregadas


Mis Películas
To exit full screen, press and hold Esc
Avatar (2009)
Calificación: 8/10
Revolucionó el cine en 3D y efectos visuales. Visualmente impresionante.
Agregado: 2/18/2026, 4:53:10 PM
Editar
Eliminar
Avengers: Endgame (2019)
Calificación: 9/10
Cierre épico de 10 años del UCM. Evento cultural masivo.
Agregado: 2/18/2026, 4:53:27 PM
Editar
Eliminar
Avatar: The Way of Water (2022)
Calificación: 8/10
Expansión visual impresionante del mundo de Pandora.
Agregado: 2/18/2026, 4:53:44 PM
Editar
Eliminar