



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/api-tareas>

## **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 asíncronas 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

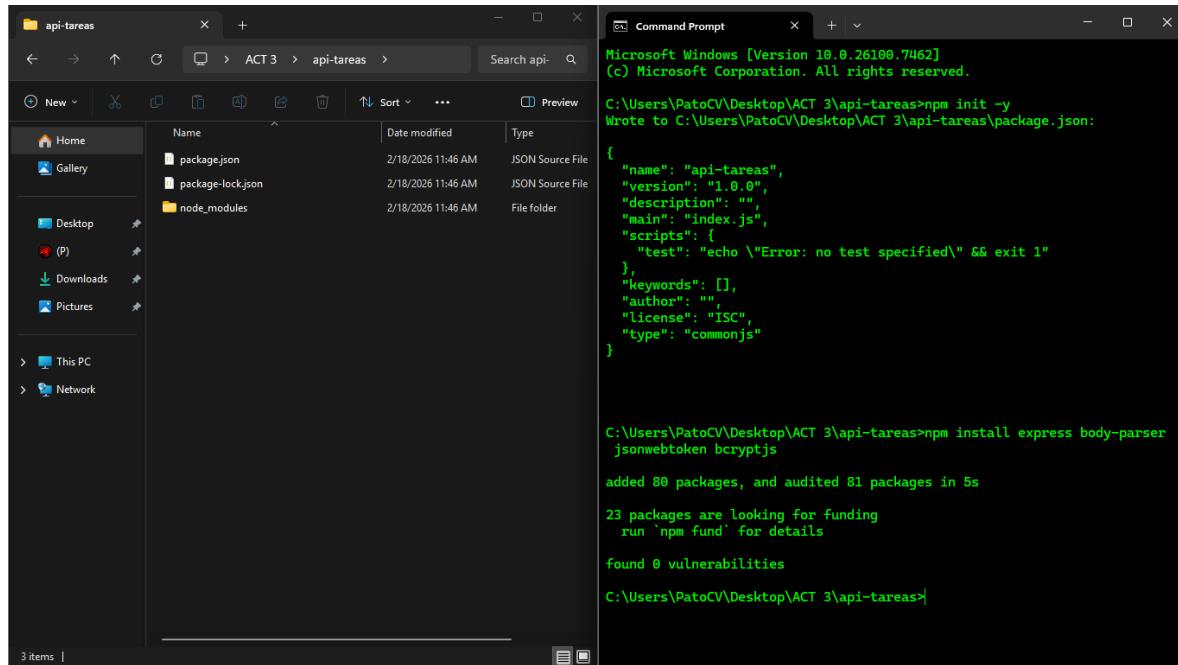
## **Resultado Final**

- 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, asíncronía 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



The screenshot shows a Windows File Explorer window on the left and a Command Prompt window on the right.

**File Explorer Content:**

Name	Date modified	Type
package.json	2/18/2026 11:46 AM	JSON Source File
package-lock.json	2/18/2026 11:46 AM	JSON Source File
node_modules	2/18/2026 11:46 AM	File folder

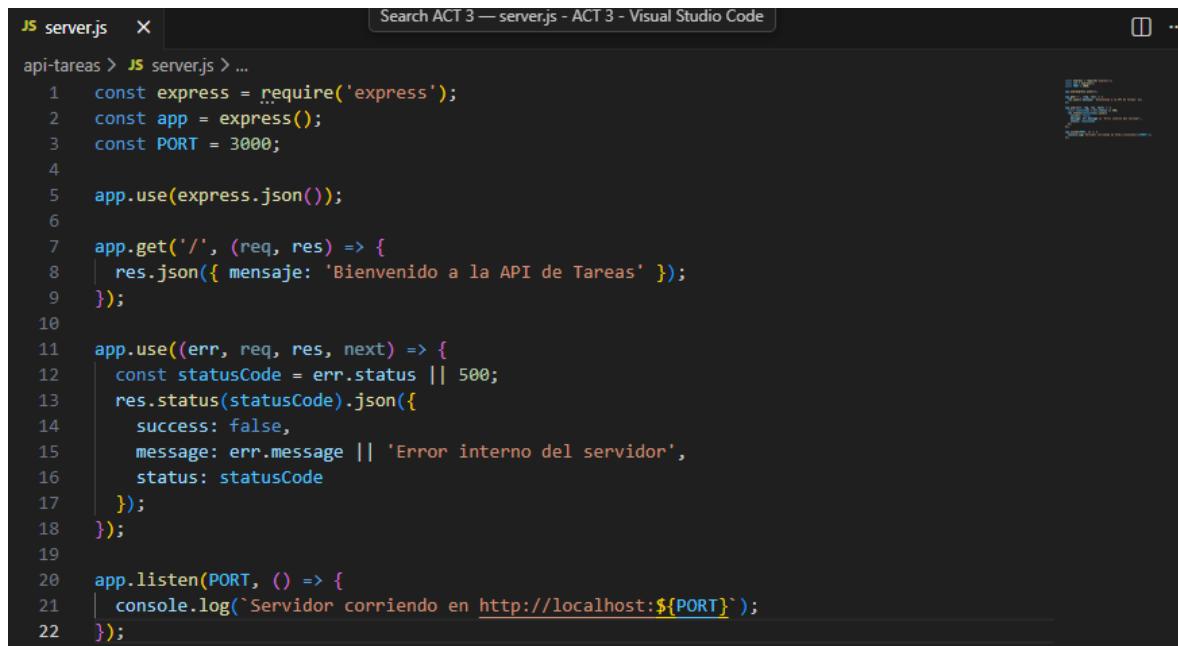
**Command Prompt History:**

```
C:\Users\PatoCV\Desktop\ACT 3\api-tareas>npm init -y
Write to C:\Users\PatoCV\Desktop\ACT 3\api-tareas\package.json:
{
  "name": "api-tareas",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}

C:\Users\PatoCV\Desktop\ACT 3\api-tareas>npm install express body-parser jsonwebtoken bcryptjs
added 80 packages, and audited 81 packages in 5s
23 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities

C:\Users\PatoCV\Desktop\ACT 3\api-tareas>
```

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



The screenshot shows a Visual Studio Code editor window with the file 'server.js' open.

```
JS server.js X Search ACT 3 — server.js - ACT 3 - Visual Studio Code ...
api-tareas > JS server.js > ...
1 const express = require('express');
2 const app = express();
3 const PORT = 3000;
4
5 app.use(express.json());
6
7 app.get('/', (req, res) => {
8   res.json({ mensaje: 'Bienvenido a la API de Tareas' });
9 });
10
11 app.use((err, req, res, next) => {
12   const statusCode = err.status || 500;
13   res.status(statusCode).json({
14     success: false,
15     message: err.message || 'Error interno del servidor',
16     status: statusCode
17   });
18 });
19
20 app.listen(PORT, () => {
21   console.log(`Servidor corriendo en http://localhost:${PORT}`);
22 });
```

## C. CREAR LAS RUTAS PARA LA API RESTFUL

```
app.get('/tareas', async (req, res, next) => {
  try {
    const datos = await fs.readFile(TAREAS_FILE, 'utf8');
    const tareas = JSON.parse(datos);
    res.json(tareas);
  } catch (error) {
    next(error);
  }
});

app.post('/tareas', async (req, res, next) => {
  try {
    const nuevaTarea = {
      id: Date.now().toString(),
      ...req.body
    };

    const datos = await fs.readFile(TAREAS_FILE, 'utf8');
    const tareas = JSON.parse(datos);

    tareas.push(nuevaTarea);

    await fs.writeFile(TAREAS_FILE, JSON.stringify(tareas, null, 2));

    res.status(201).json({ id: nuevaTarea.id });
  } catch (error) {
    next(error);
  }
});

app.put('/tareas/:id', async (req, res, next) => {
  try {
    const { id } = req.params;

    const datos = await fs.readFile(TAREAS_FILE, 'utf8');
    const tareas = JSON.parse(datos);

    const indice = tareas.findIndex(t => t.id === id);

    if (indice === -1) {
      const error = new Error('Tarea no encontrada');
      error.status = 404;
      throw error;
    }

    tareas[indice] = { ...tareas[indice], ...req.body };

    await fs.writeFile(TAREAS_FILE, JSON.stringify(tareas, null, 2));

    res.json({ id });
  } catch (error) {
    next(error);
  }
});

app.delete('/tareas/:id', async (req, res, next) => {
  try {
    const { id } = req.params;

    const datos = await fs.readFile(TAREAS_FILE, 'utf8');
    const tareas = JSON.parse(datos);

    const tareasFiltradas = tareas.filter(t => t.id !== id);

    if (tareasFiltradas.length === tareas.length) {
      const error = new Error('Tarea no encontrada');
      error.status = 404;
      throw error;
    }
  }
});
```

## D. MANEJO DE DATOS CON EL MÓDULO FS

```
1 // Importar módulos necesarios
2 const express = require('express');
3 const fs = require('fs').promises; // Módulo de archivos asincrónico
4 const path = require('path');
```

## E. IMPLEMENTACIÓN DE AUTENTICACIÓN Y SESIONES

```
1 // Importar módulos necesarios
2 const express = require('express');
3 const fs = require('fs').promises; // Módulo de archivos asincrónico
4 const path = require('path');
5 const bcrypt = require('bcryptjs'); // Para encriptar contraseñas
6 const jwt = require('jsonwebtoken'); // Para generar tokens
7
```

## F. MANEJO DE ERRORES Y DEBUGGING

```
// ===== MIDDLEWARE PARA RUTAS NO ENCONTRADAS (404) =====
app.use((req, res, next) => {
  const error = new Error('Ruta no encontrada');
  error.status = 404;
  next(error);
});

// ===== MIDDLEWARE PERSONALIZADO PARA MANEJO DE ERRORES =====
app.use((err, req, res, next) => {
  // Registrar el error en consola (para debugging)
  console.error('✖ ERROR DETECTADO:');
  console.error('Timestamp:', new Date().toISOString());
  console.error('Ruta:', req.method, req.url);
  console.error('Mensaje:', err.message);
  console.error('Stack:', err.stack);
```

## PRUEBAS DE FUNCIONAMIENTO

```
npm start  
Microsoft Windows [Version 10.0.26100.7462]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\PatoCV\Desktop\ACT 3\api-tareas>npm start  
  
> api-tareas@1.0.0 start  
> node server.js  
  
Servidor en http://localhost:3000  
|
```

# Gestor de Tareas

## Crear Tarea

**Título:**

**Descripción:**

API RESTful con Node.js/Express.js que gestiona tareas en archivo JSON con autenticación JWT y bcryptjs. Implementa CRUD, middleware de validación y manejo de errores. Rutas protegidas mediante token.

## Mis Tareas

**Gestor de Tareas**

API RESTful con Node.js/Express.js que gestiona tareas en archivo JSON con autenticación JWT y bcryptjs. Implementa CRUD, middleware de validación y manejo de errores. Rutas protegidas mediante token.

Creado: 2/18/2026, 12:26:57 PM

[Editar](#) [Eliminar](#)

## Mis Tareas

### Gestor de Tareas

API RESTful con Node.js/Express.js que gestiona tareas en archivo JSON con autenticación JWT y bcryptjs. Implementa CRUD, middleware de validación y manejo de errores. Rutas protegidas mediante token.

Creado: 2/18/2026, 12:26:57 PM

[Editar](#)

[Eliminar](#)

**localhost:3000 says**

Nuevo titulo:

Gestor de Tareas (EDITADO)

OK

Cancel

**localhost:3000 says**

Nueva descripcion:

Nueva descripcion (EDITADO)

OK

Cancel

## Mis Tareas

**Gestor de Tareas (EDITADO)**

Nueva descripcion (EDITADO)

Creado: 2/18/2026, 12:26:57 PM

[Editar](#)

[Eliminar](#)

localhost:3000 says

Eliminar tarea?

OK

Cancel

# Gestor de Tareas

Tarea eliminada

## Crear Tarea

**Titulo:**

Ej: Comprar leche

**Descripcion:**

Ej: Ir al supermercado

Agregar Tarea

## Mis Tareas

No hay tareas