



**Universidad Autónoma de Sinaloa**

**Facultad de informática Culiacán**

**Licenciatura en informática**

**Tema:**

Actividad 3

API con método POST

**Materia:**

Desarrollo Web Del Lado Del Servidor

**Alumno:**

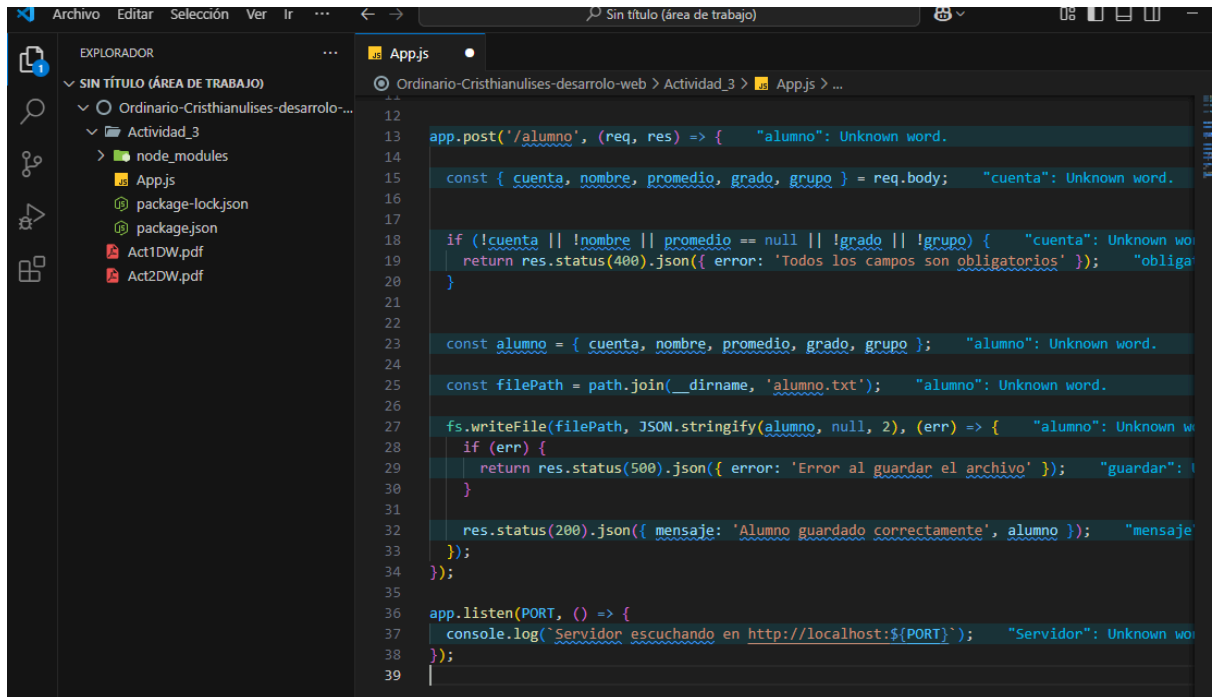
López López Cristhian Ulises

**Grupo:**

“2-3”

**Docente:**

José Manuel Cazarez Alderete



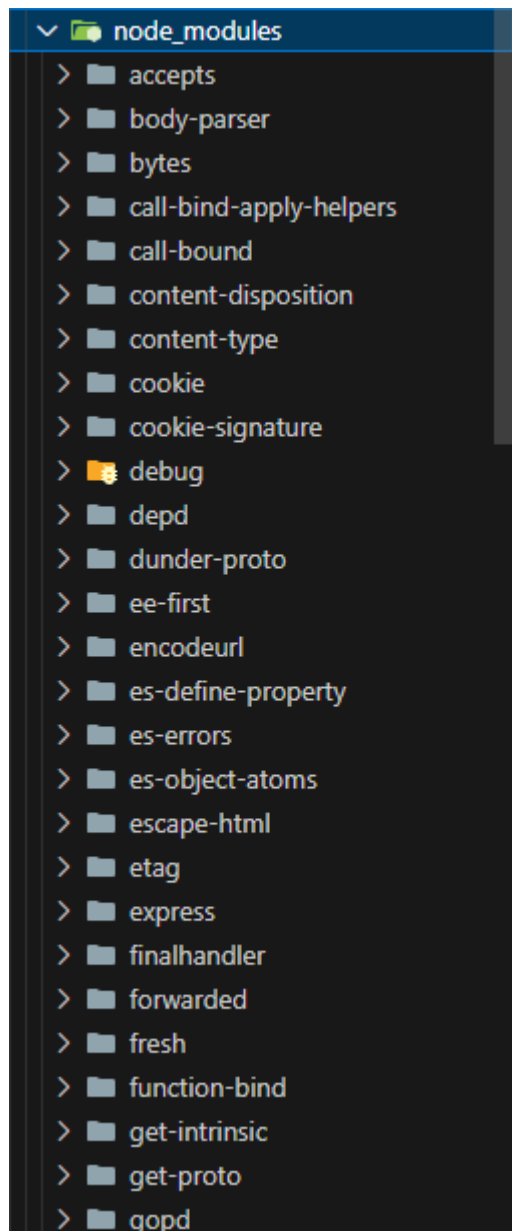
The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORADOR' (Explorer) sidebar shows a project structure with a folder named 'Actividad\_3' containing 'App.js', 'package-lock.json', and 'package.json'. The main editor area displays the code for 'App.js'. The code is a Node.js script using Express.js to create a web server. It defines a POST endpoint at '/alumno' that receives a request, extracts the body, and checks if all required fields (cuenta, nombre, promedio, grado, grupo) are present. If any field is missing, it returns a 400 status with an error message. If all fields are present, it constructs an 'alumno' object, creates a file path 'alumno.txt' in the current directory, and writes the object to the file using 'fs.writeFile'. If the write operation fails, it returns a 500 status with an error message. If successful, it returns a 200 status with a success message and the saved student data. Finally, it listens on a specified port (3000) and logs the server's listening status.

```
12 app.post('/alumno', (req, res) => { "alumno": Unknown word.
13
14   const { cuenta, nombre, promedio, grado, grupo } = req.body; "cuenta": Unknown word.
15
16   if (!cuenta || !nombre || promedio == null || !grado || !grupo) { "cuenta": Unknown wo
17     return res.status(400).json({ error: 'Todos los campos son obligatorios' }); "obliga
18   }
19
20   const alumno = { cuenta, nombre, promedio, grado, grupo }; "alumno": Unknown word.
21
22   const filePath = path.join(__dirname, 'alumno.txt'); "alumno": Unknown word.
23
24   fs.writeFile(filePath, JSON.stringify(alumno, null, 2), (err) => { "alumno": Unknown w
25     if (err) {
26       return res.status(500).json({ error: 'Error al guardar el archivo' }); "guardar": t
27     }
28
29     res.status(200).json({ mensaje: 'Alumno guardado correctamente', alumno }); "mensaje
30   });
31
32   app.listen(PORT, () => {
33     console.log(`Servidor escuchando en http://localhost:${PORT}`); "Servidor": Unknown wo
34   });
35
36
37
38
39
```

Este archivo crea un servidor web con Node.js y Express que escucha en el puerto 3000. Tiene una única función: recibir datos de un alumno mediante una petición POST, verificar que todos los campos estén completos y guardar esa información en un archivo llamado `alumno.txt`.

```
1 {
2   "name": "act3dw",
3   "version": "1.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "act3dw",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "express": "^5.1.0"
13      }
14    },
15    "node_modules/accepts": {
16      "version": "2.0.0",
17      "resolved": "https://registry.npmjs.org/accepts/-/accepts-2.0.0.tgz",
18      "integrity": "sha512-5cv798u1/7K/0uWV9/3n/7+2W/6ao+6Y7gu/RcRuAhGEzh5B4K1s",
19      "license": "MIT",
20      "dependencies": {
21        "mime-types": "^3.0.0",
22        "negotiator": "^1.0.0"
23      },
24      "engines": {
25        "node": ">= 0.6"
26      }
27    },
28    "node_modules/body-parser": {
29      "version": "2.2.0",
30      "resolved": "https://registry.npmjs.org/body-parser/-/body-parser-2.2.0.tgz",
31      "integrity": "sha512-02q00u207l6aMKGIZL9aP9X5I0Z8L7A0G+Vag0U+TgZc0eDn0kH4F0GKwDl0H0T0G",
32      "license": "MIT",
33      "dependencies": {
34        "bytes": "^3.1.2",
35        "content-type": "^1.0.5"
```

El archivo `package.json` es un documento que usan los proyectos de Node.js para guardar información importante. Es como una hoja de control que dice cómo se llama el proyecto, qué versión es, qué archivo debe arrancar primero y qué programas extra necesita para funcionar bien (como Express, por ejemplo).



La carpeta `node_modules` se usa para guardar archivos que explican cómo van a ser los datos en tu programa. Por ejemplo, si vas a trabajar con alumnos, ahí puedes poner qué datos tendrá cada alumno, como nombre, cuenta o promedio. Esta carpeta ayuda a que el programa sepa cómo organizar y guardar la información, sobre todo si usas una base de datos.