Api Con Metodo Post Man

app.js

```
// Importaciones y configuración básica
const express = require('express');
const fs = require('fs');
const app = express();
const port = 3000;
app.use(express.json());
const ARCHIVO_ALUMNOS = 'alumnos.json';
```

Esta sección inicial configura el servidor web usando Express.js y establece las dependencias necesarias:

- express: Framework web para Node.js
- fs: Módulo para manejar archivos
- Se configura el puerto 3000 para el servidor

express.json(): Middleware para procesar datos JSON

```
// Funciones auxiliares para manejar datos
function leerAlumnos() {
    try {
        const datos = fs.readFileSync(ARCHIVO_ALUMNOS, 'utf8');
        return JSON.parse(datos);
    } catch (error) {
        return [];
    }
}

function guardarAlumnos(alumnos) {
    fs.writeFileSync(ARCHIVO_ALUMNOS, JSON.stringify(alumnos, null, 2));
}
```

Estas funciones manejan la lectura y escritura de datos:

- leerAlumnos(): Lee el archivo JSON y devuelve los datos de alumnos
- guardarAlumnos(): Guarda los datos de alumnos en el archivo JSON

```
// Endpoint POST para crear alumnos
app.post('/alumno', (req, res) => {
    const nuevoAlumno = req.body;
    const camposRequeridos = ['cuenta', 'nombre', 'promedio', 'grado', 'grupo'];

    // Validación de campos
    const camposFaltantes = camposRequeridos.filter(campo => !nuevoAlumno[campo]);
    if (camposFaltantes.length > 0) {
        return res.status(400).json({
            mensaje: `Faltan los siguientes campos: ${camposFaltantes.join(', ')}`
        });
    }

    // Verificación de duplicados y guardado
    const alumnos = leerAlumnos();
    if (alumnos.some(alumno => alumno.cuenta === nuevoAlumno.cuenta)) {
        return res.status(400).json({
            mensaje: 'Ya existe un alumno con esa cuenta'
        });
    }

    alumnos.push(nuevoAlumno);
    guardarAlumnos(alumnos);

    res.status(201).json({
        mensaje: 'Alumno creado exitosamente',
        alumno: nuevoAlumno
    });
});
```

Este es el endpoint principal que maneja la creación de nuevos alumnos:

- Recibe datos del alumno en formato JSON
- Valida que todos los campos requeridos estén presentes
- Verifica que no exista un alumno con la misma cuenta
- Guarda el nuevo alumno en el archivo JSON

alumnos.json - Este archivo almacena los datos de los alumnos:

```
[
  {
    "cuenta": "10132123",
    "nombre": "AntonioSolis",
    "promedio": 9.9,
    "grado": "2",
    "grupo": "1"
  }
]
```

Este archivo muestra la estructura de datos de los alumnos:

- Es un array de objetos JSON
- Cada alumno tiene los siguientes campos:
- cuenta: Número de cuenta del alumno
- nombre: Nombre del alumno
- promedio: Calificación promedio
- grado: Grado que cursa
- grupo: Grupo al que pertenece