

## Trabajo Práctico Evaluatorio

### Servidor ExpressJS

Práctica Profesional

Para este trabajo debes crear un servidor web utilizando Node.js y Express.js.

Este servidor debe implementar rutas para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de libros, autores, usuarios y préstamos mediante los metodos http get, put, post, delete.

Utiliza middlewares para el manejo de errores y el procesamiento de solicitudes HTTP.

Para los que no estén realizando la materia Gestión de bases de datos deberán utilizar herramientas mock para emular la información de la base de datos, te dejo algunas que pueden servirte, pero puedes buscar otras.

- <https://app.mockfly.dev/>
- <https://mockapi.io/>
- <https://mockoon.com/>

También puedes trabajarlo localmente (esta última opción sería la menos recomendada), para ello crea la carpeta assets y dentro la carpeta mocks que contendrá los archivos con la información.

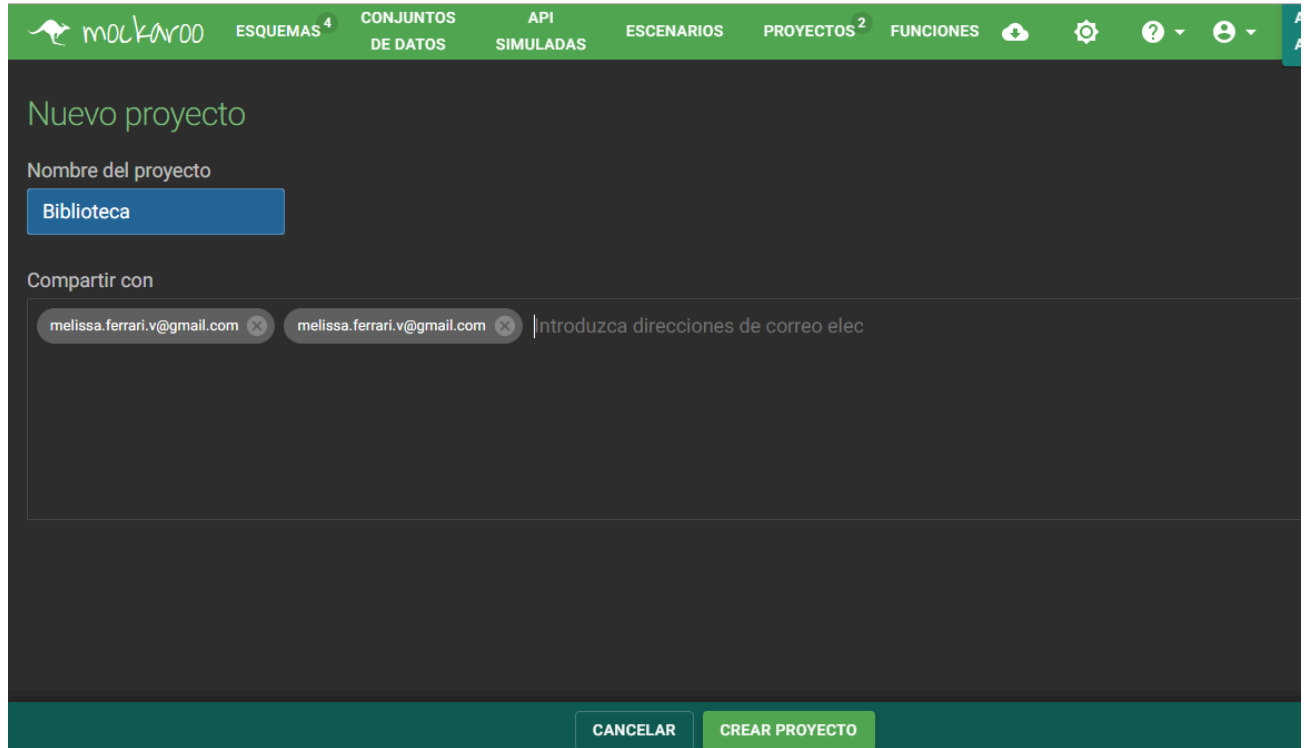
Debes entregar el repositorio de github donde se encuentra el trabajo realizado.

En clase deberás defender lo realizado, analizando el funcionamiento correcto y el fallido. En caso de no llegar a terminar algún punto podrás justificarlo en el momento de la defensa.

El trabajo puede realizarse en grupos de 2 integrantes.

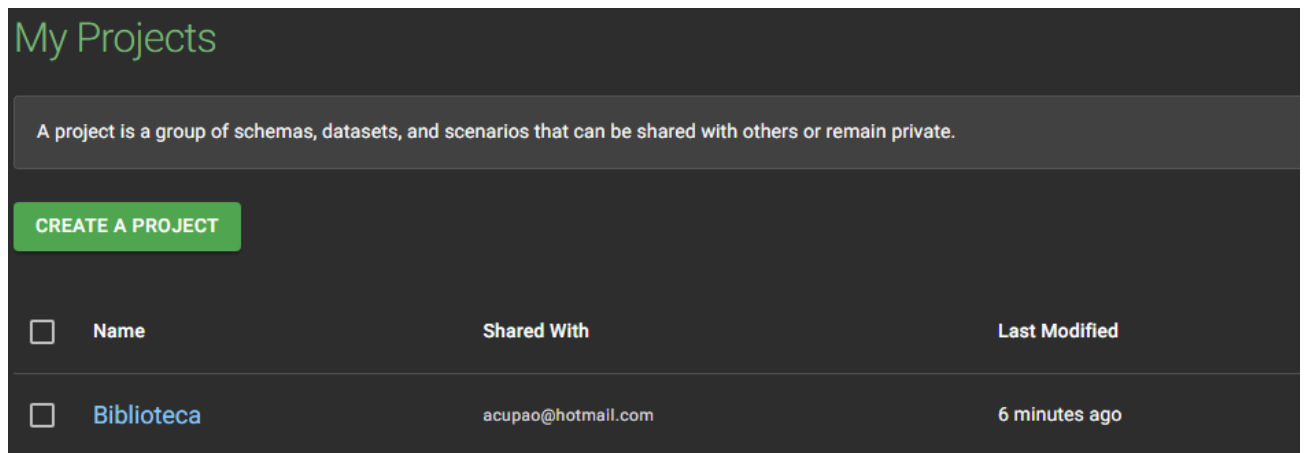
## Creamos un Proyecto en Mockaroo y esquemas

Ingresamos a la plataforma para crear el proyecto Biblioteca



The screenshot shows the 'Nuevo proyecto' (New project) form in the Mockaroo application. The top navigation bar is green and contains the Mockaroo logo and several menu items: ESQUEMAS<sup>4</sup>, CONJUNTOS DE DATOS, API SIMULADAS, ESCENARIOS, PROYECTOS<sup>2</sup>, and FUNCIONES. The form itself has a dark gray background. At the top, it says 'Nuevo proyecto' in green. Below that, there's a section 'Nombre del proyecto' (Project name) with a blue button labeled 'Biblioteca'. Underneath is a 'Compartir con' (Share with) section, which includes two email addresses in gray buttons: 'melissa.ferrari.v@gmail.com' and 'melissa.ferrari.v@gmail.com', followed by a text input field with the placeholder 'Introduzca direcciones de correo elec'. At the bottom of the form, there are two buttons: 'CANCELAR' (Cancel) and 'CREAR PROYECTO' (Create project).

## Creamos el proyecto

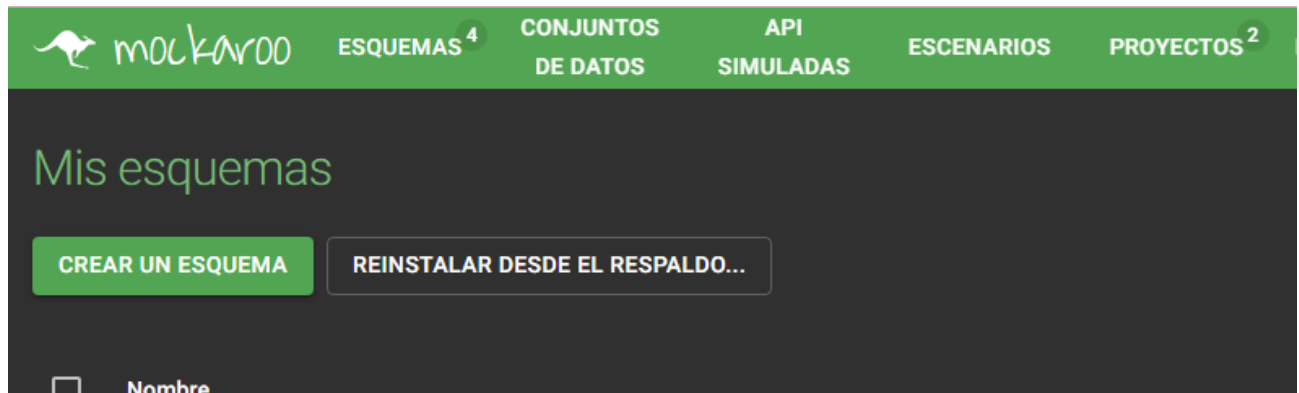


The screenshot shows the 'My Projects' page in the Mockaroo application. The page has a dark gray background. At the top, it says 'My Projects' in green. Below that, there's a gray box with the text 'A project is a group of schemas, datasets, and scenarios that can be shared with others or remain private.' Underneath is a green button labeled 'CREATE A PROJECT'. Below that is a table with three columns: 'Name', 'Shared With', and 'Last Modified'. The table has one row with the project 'Biblioteca'.

<input type="checkbox"/>	Name	Shared With	Last Modified
<input type="checkbox"/>	Biblioteca	acupao@hotmail.com	6 minutes ago

## Crear los esquemas

Un esquema en Mockaroo define la estructura de los datos. Vamos a crear esquemas como libros, autores, usuarios, y préstamos.



Añadir Campos: vemos una tabla donde podemos añadir campos, hacemos clic en el botón + añadir otro campo a añadir un campo.

## Generar datos y guardar el Esquema

Generar datos: una vez configurados todos los campos, seleccionamos cuántos registros deseamos generar (por ejemplo, 10).

Guardar el Esquema: después de configurar todos los campos, hacemos clic en guardar y asignamos un nombre al esquema y guardamos. Como se observa en la siguiente imagen se crearon los esquemas.

## autores

	Nombre del campo	Tipo	Opciones
⋮	id_autor	Numero de fila	blanco: 0 % $\Sigma$ ✕
⋮	nombre	Nombre de pila	blanco: 0 % $\Sigma$ ✕
⋮	apellido	Apellido	blanco: 0 % $\Sigma$ ✕

+ AGREGAR OTRO CAMPO


GENERAR CAMPOS USANDO IA...

# Filas: 20 Formato: JSON ☒ formación ☒ incluir valores nulos

Sugerencia: utilice "." en nombres de columnas para generar objetos json anidados, corchetes para generar matrices. [Más información...](#)

Agregar conjunto de datos: elige un conjunto de datos

```
[{
  "id_autor": 1,
  "nombre": "Yehudit",
  "apellido": "Pressnell"
}, {
  "id_autor": 2,
  "nombre": "Marybeth",
  "apellido": "Brauner"
}, {
  "id_autor": 3,
  "nombre": "Zebadiah",
  "apellido": "Bloore"
}, {
  "id_autor": 4,
  "nombre": "Auberta",
  "apellido": "Airds"
}, {
  "id_autor": 5,
  "nombre": "Madelle",
  "apellido": "Ilyas"
}]
```



ESQUEMAS 4

CONJUNTOS DE DATOS

API SIMULADAS

ESCENARIOS


PROYECTOS 2

FUNCIONES


Biblioteca / libros

MOVER

libros

Nombre del campo	Tipo	Opciones
id_libro	Numero de fila	blanco: 0 % $\Sigma$ $\times$
titulo	Palabras	al menos 1 pero no más que 5 blanco: 0 % $\Sigma$ $\times$
estado	Lista personalizada	disponible, prestado  aleatorio $\nabla$ blanco: $\Sigma$ $\times$
id_autor	Numero de fila	blanco: 0 % $\Sigma$ $\times$

+ AGREGAR OTRO CAMPO

 GENERAR CAMPOS USANDO IA...

# Filas: 30

Formato: JSON  $\nabla$

☒ formación

☒ incluir valores nulos

Sugerencia: utilice "." en nombres de columnas para generar objetos json anidados, corchetes para generar matrices. [Más información...](#)

Actualizar conjunto de datos

GENERAR DATOS  $\nabla$


AVANCE

CAMBIOS GUARDADOS

CREAR API

MÁS  $\nabla$

```
[{"id_libro": 1,
"titulo": "volutpat eleifend donec ut dolor",
"estado": "disponible",
"id_autor": 1
}, {
"id_libro": 2,
"titulo": "elementum in",
"estado": "disponible",
"id_autor": 2
}, {
"id_libro": 3,
"titulo": "ultrices posuere cubilia curae",
"estado": "disponible",
"id_autor": 3
}, {
"id_libro": 4,
"titulo": "praesent id massa id",
"estado": "disponible",
"id_autor": 4
}, {
"id_libro": 5,
"titulo": "velit nec nisi vulputate nonummy",
"estado": "disponible",
"id_autor": 5
}, {
"id_libro": 6,
"titulo": "nisi l nunc",
"estado": "disponible",
"id_autor": 6
}, {
"id_libro": 7,
"titulo": "nulla integer pede justo lacinia",
"estado": "prestado",
"id_autor": 7
}, {
"id_libro": 8,
"titulo": "nascetur ridiculus",
"estado": "disponible",
"id_autor": 8
}, {
}
```



ESQUEMAS<sup>4</sup>

CONJUNTOS DE DATOS

API SIMULADAS

ESCENARIOS

PROYECTOS<sup>2</sup>

FUNCIONES

Biblioteca / Usuarios

M

## Usuarios

Nombre del campo	Tipo	Opciones
id_usuario	Numero de fila	blanco: 0 % $\Sigma$ X
nombre	Nombre de pila	blanco: 0 % $\Sigma$ X
apellido	Apellido	blanco: 0 % $\Sigma$ X
email	Dirección de correo ...	blanco: 0 % $\Sigma$ X

+ AGREGAR OTRO CAMPO

GENERAR CAMPOS USANDO IA...

# Filas: 15

Formato: JSON

☒ formación

☒ Incluir valores nulos

Sugerencia: utilice "." en nombres de columnas para generar objetos json anidados, corchetes para generar matrices. [Más información...](#)

Agregar conjunto de datos

GENERAR DATOS


AVANCE

CAMBIOS GUARDADOS

CREAR API

MÁS

```
[{
  "id_usuario": 1,
  "nombre": "Sheilakathryn",
  "apellido": "Goodee",
  "email": "sgoodee0@flickr.com"
}, {
  "id_usuario": 2,
  "nombre": "Darin",
  "apellido": "Vousden",
  "email": "dvousden1@netvibes.com"
}, {
  "id_usuario": 3,
  "nombre": "Karina",
  "apellido": "Manis",
  "email": "kmanis2@biblegateway.com"
}, {
  "id_usuario": 4,
  "nombre": "Myrtia",
  "apellido": "Jory",
  "email": "mjory3@drupal.org"
}, {
  "id_usuario": 5,
  "nombre": "Dody",
  "apellido": "Gothrup",
  "email": "dgothrup4@mail.ru"
}]
```



ESQUEMAS<sup>4</sup>

CONJUNTOS DE DATOS

API SIMULADAS

ESCENARIOS

PROYECTOS<sup>2</sup>

FUNCIONES

Biblioteca / prestamos

prestamos

Nombre del campo	Tipo	Opciones
id_prestamo	Numero de fila	blanco: 0 %
id_usuario	Numero de fila	blanco: 0 %
id_libro	Numero de fila	blanco: 0 %
fecha_inicio	Fecha y hora	07/01/2024 a 07/01/2024 formato: m/d/aaaa blanco: 0 %
fecha_devolucion	Fecha y hora	08/01/2024 a 08/01/2024 formato: m/d/aaaa blanco: 0 %

+ AGREGAR OTRO CAMPO

GENERAR CAMPOS USANDO IA...

# Filas: 5

Formato: JSON

☒ formación

☒ incluir valores nulos

Sugerencia: utilice " en nombres de campo

GENERAR DATOS

AVANCE

CAMBIOS GUARDADOS

CREAR API

MÁS

```
[{
  "id_prestamo": 1,
  "id_libro": 1,
  "id_usuario": 1,
  "fecha_inicio": "7/2/2024",
  "fecha_devolucion": "7/15/2024"
}, {
  "id_prestamo": 2,
  "id_libro": 2,
  "id_usuario": 2,
  "fecha_inicio": "7/2/2024",
  "fecha_devolucion": "7/15/2024"
}, {
  "id_prestamo": 3,
  "id_libro": 3,
  "id_usuario": 3,
  "fecha_inicio": "7/2/2024",
  "fecha_devolucion": "7/15/2024"
}, {
  "id_prestamo": 4,
  "id_libro": 4,
  "id_usuario": 4,
  "fecha_inicio": "7/2/2024",
  "fecha_devolucion": "7/15/2024"
}, {
  "id_prestamo": 5,
  "id_libro": 5,
  "id_usuario": 5,
  "fecha_inicio": "7/2/2024",
  "fecha_devolucion": "7/15/2024"
}]
```

## Usar los Datos

Obtener la URL: una vez guardados los esquemas, vamos a obtener la URL que vamos usar para obtener datos dinámicamente, por ejemplo observemos autores:

The screenshot shows the Mockaroo web interface for generating data. At the top, the schema name 'autores' is displayed in green. Below it, there is a table with three columns: 'Nombre del campo', 'Tipo', and 'Opciones'. The table contains three rows of fields: 'id\_autor' (Numero de fila), 'nombre' (Nombre de pila), and 'apellido' (Apellido). Each row has a 'blanco' option set to '0 %' and a 'Σ' icon. Below the table, there are buttons for '+ AGREGAR OTRO CAMPO' and 'GENERAR CAMPOS USANDO IA...'. Further down, there are settings for '# Filas' (20), 'Formato' (JSON), and checkboxes for 'formación' and 'Incluir valores nulos'. A suggestion text is present: 'Sugerencia: utilice "" en nombres de columnas para generar objetos json anidados, corchetes para generar matrices. Más Información...'. Below this, there is a section for 'Agregar conjunto de datos:' with a dropdown menu. The main section is titled 'Generando datos a través de cURL' and contains a text box with a cURL command: 'curl "https://api.mockaroo.com/api/6ded64c0?count=5&key=a0bb0e60" > "autores.json"'. Below the text box, there is a note: 'También puedes generar datos a través de la API.'. At the bottom, there is a section titled 'Intercambio' with a note: 'Este esquema es privado. Sólo tú y los colaboradores del proyecto podéis acceder a él.' and a button 'COMPARTE ESTE ESQUEMA PÚBLICAMENTE'.

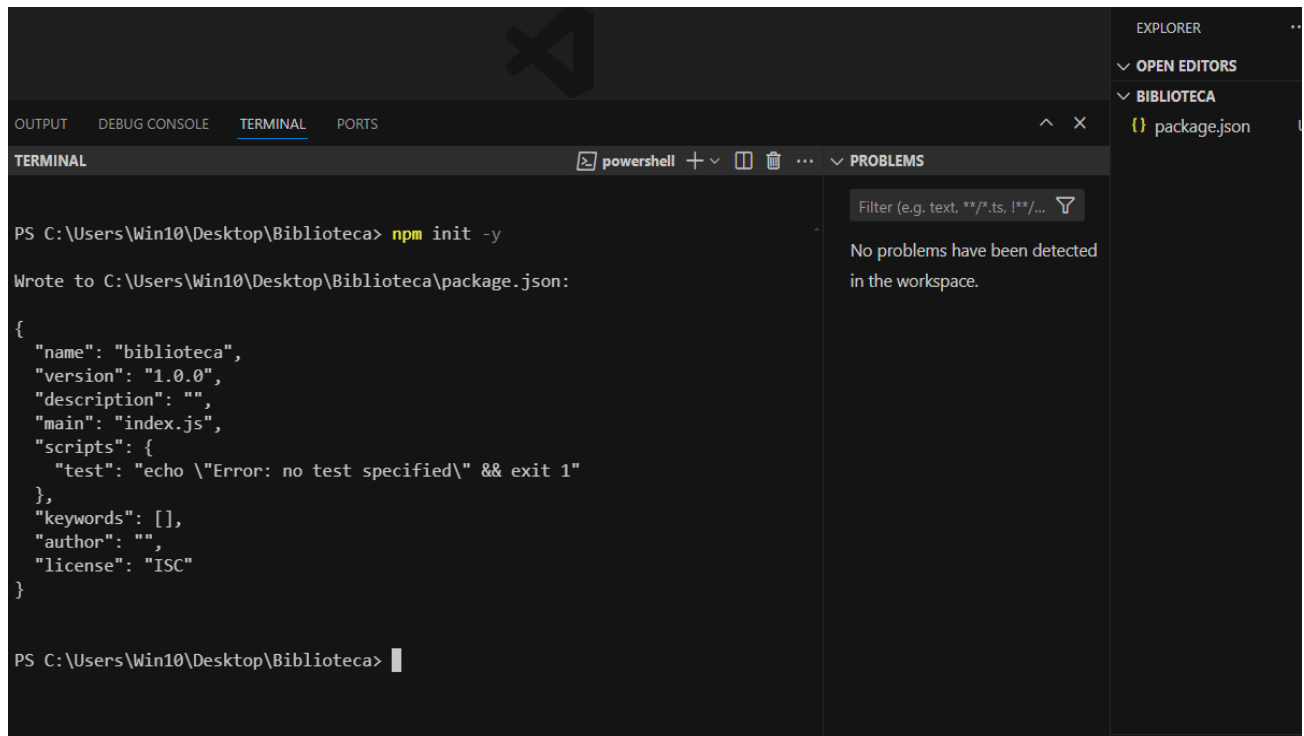
**"https://api.mockaroo.com/api/6ded64c0?count=5&key=a0bb0e60"**

Para implementar el servidor web con Node.js y Express.js y utilizar Mockaroo para generar datos iniciamos un nuevo proyecto Node.js utilizando npm

```
PS C:\Users\Win10\Desktop\Biblioteca> npm init -y
```



Instalamos las dependencias necesarias: express para el servidor y axios para hacer solicitudes HTTP a Mockaroo

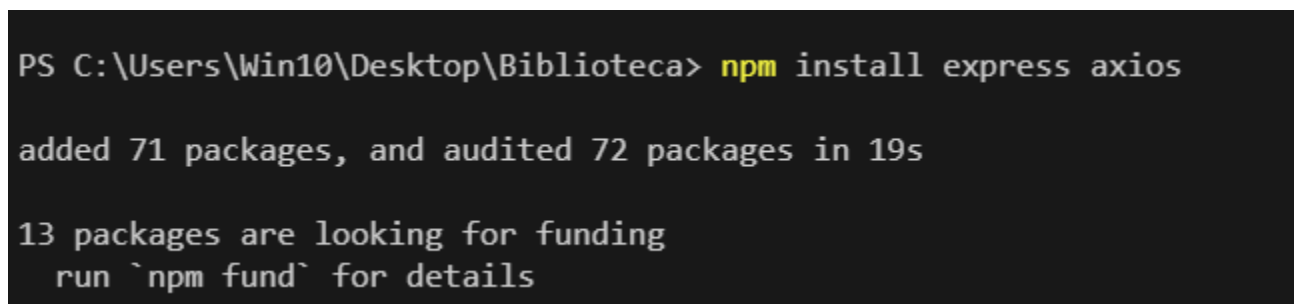


The screenshot shows the Visual Studio Code interface with the terminal panel active. The terminal is running a PowerShell session in the directory C:\Users\Win10\Desktop\Biblioteca. The command `npm init -y` has been executed, and the output shows the creation of a `package.json` file with the following content:

```
Wrote to C:\Users\Win10\Desktop\Biblioteca\package.json:

{
  "name": "biblioteca",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

The terminal prompt is now `PS C:\Users\Win10\Desktop\Biblioteca>`. The Explorer panel on the right shows the `package.json` file.



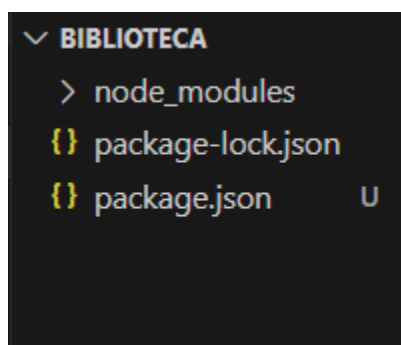
The screenshot shows the Visual Studio Code terminal with the output of the `npm install express axios` command. The output indicates that 71 packages were added and 72 packages were audited in 19 seconds. It also mentions that 13 packages are looking for funding and suggests running `npm fund` for details.

```
PS C:\Users\Win10\Desktop\Biblioteca> npm install express axios

added 71 packages, and audited 72 packages in 19s

13 packages are looking for funding
  run `npm fund` for details
```

Se generan estos archivos en nuestro proyecto:



The screenshot shows the Explorer panel in VS Code, displaying the contents of the `BIBLIOTECA` folder. The files listed are `node_modules`, `package-lock.json`, and `package.json`.

```
▼ BIBLIOTECA
  > node_modules
  {} package-lock.json
  {} package.json
```

```

{} package-lock.json X
{} package-lock.json > {} packages > {} "" > {} dependencies
1  {
2    "name": "biblioteca",
3    "version": "1.0.0",
4    "lockfileVersion": 2,
5    "requires": true,
6    "packages": {
7      "": {
8        "name": "biblioteca",
9        "version": "1.0.0",
10       "license": "ISC",
11       "dependencies": {
12         "axios": "^1.7.2",
13         "express": "^4.19.2"
14       },
15     },
16     "node_modules/accepts": {
17       "version": "1.3.8",
18       "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.8.tgz",
19       "integrity": "sha512-Py+U0089tXU39F4742z39C0j568Bt2w388930X0261d1299741379fde70f74686",
20       "dependencies": {
21         "mime-types": "~2.1.34",
22         "negotiator": "0.6.3"
23       },
24       "engines": {
25         "node": ">= 0.6"
26       }
27     },
28     "node_modules/array-flatten": {

```

```

{} package.json > ...
1  {
2    "name": "biblioteca",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "axios": "^1.7.2",
14     "express": "^4.19.2"
15   }
16 }
17

```

**Crear Archivos para el Servidor:** creamos el archivo server.js

## Importación de Módulos y Configuración Inicial

```
// Importamos módulos necesarios
const express = require('express');
const axios = require('axios');
const app = express();
const PORT = process.env.PORT || 3000;
```

- express: framework web para Node.js.
- axios: librería para realizar peticiones HTTP.
- app: instancia de la aplicación Express.
- PORT: puerto en el cual el servidor escuchará, definido por una variable de entorno o por defecto el puerto 3000.

## Middleware para Parsear JSON y Manejo de Errores

```
// Middleware para parsear JSON y URLencoded
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Middleware para manejar errores
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Error interno del servidor');
});
```

- express.json(): Middleware para parsear cuerpos de solicitud JSON.
- express.urlencoded({ extended: true }): Middleware para parsear cuerpos de solicitud URL-encoded.
- Middleware de manejo de errores que captura cualquier error en la pila de middleware y envía una respuesta con un mensaje de error.

## URLs de Mockaroo

```
const URL_LIBROS = "https://api.mockaroo.com/api/63b8e580?count=10&key=a0bb0e60";
const URL_USUARIOS = "https://api.mockaroo.com/api/79ba2a10?count=5&key=a0bb0e60";
const URL_AUTORES = "https://api.mockaroo.com/api/6ded64c0?count=5&key=a0bb0e60";
const URL_PRESTAMOS = "https://api.mockaroo.com/api/57740b90?count=5&key=a0bb0e60";
```

## Función para Cargar Datos desde una URL

```
async function cargarDatosDesdeURL(url) {
  try {
    const response = await axios.get(url);
    // Asigna un id único a cada libro
    const datos = response.data.map((libro, index) => ({
      ...libro,
      id: index + 1 // Asigna un id único basado en el índice + 1
    }));
    return datos;
  } catch (error) {
    console.error('Error al cargar datos desde Mockaroo:', error.message);
    return [];
  }
}
```

- `cargarDatosDesdeURL(url)`: función asíncrona que realiza una petición GET a la URL proporcionada y asigna un ID único a cada elemento basado en su índice en el array.

```
// Variables para almacenar los datos
let libros = [];
let usuarios = [];
let autores = [];
let prestamos = [];

// Cargar datos al iniciar el servidor
async function cargarDatos() {
  libros = await cargarDatosDesdeURL(URL_LIBROS);
  usuarios = await cargarDatosDesdeURL(URL_USUARIOS);
  autores = await cargarDatosDesdeURL(URL_AUTORES);
  prestamos = await cargarDatosDesdeURL(URL_PRESTAMOS);

  console.log('Datos cargados:');
  console.log('Libros:', libros);
  console.log('Usuarios:', usuarios);
  console.log('Autores:', autores);
  console.log('Prestamos:', prestamos);
}

cargarDatos();
```

## Implementar Rutas CRUD para Libros, Autores, Usuarios y Préstamos

Añadimos funciones para manejar las operaciones CRUD utilizando Axios para las URLs proporcionadas por Mockaroo:

### Rutas CRUD para Libros

```
// Variables para almacenar los datos (simulando una "base de datos")
let libros = [];
let usuarios = [];
let autores = [];
let prestamos = [];

// Cargar datos al iniciar el servidor
async function cargarDatos() {
  libros = await cargarDatosDesdeURL(URL_LIBROS);
  usuarios = await cargarDatosDesdeURL(URL_USUARIOS);
  autores = await cargarDatosDesdeURL(URL_AUTORES);
  prestamos = await cargarDatosDesdeURL(URL_PRESTAMOS);

  console.log('Datos cargados:');
  console.log('Libros:', libros);
  console.log('Usuarios:', usuarios);
  console.log('Autores:', autores);
  console.log('Préstamos:', prestamos);
}

cargarDatos();

// Rutas CRUD para libros
app.get('/libros', (req, res) => {
  res.json(libros);
});

app.post('/libros', (req, res) => {
  const nuevolibro = req.body;
  // Asigna un nuevo id al nuevo libro
  nuevolibro.id = libros.length + 1;
  libros.push(nuevolibro);
  res.status(201).json(nuevolibro);
});

app.put('/libros/:id', (req, res) => {
  const id = req.params.id;
  const index = libros.findIndex(libro => libro.id === parseInt(id));
  if (index !== -1) {
    libros[index] = req.body;
    res.json(libros[index]);
  } else {
    res.status(404).send('Libro no encontrado');
  }
});

app.delete('/libros/:id', (req, res) => {
  const id = req.params.id;
  libros = libros.filter(libro => libro.id !== parseInt(id));
  res.status(204).send();
});
```

- **GET** /libros: retorna todos los libros.
- **POST** /libros: añade un nuevo libro.
- **PUT** /libros/:id: actualiza un libro existente por su ID.
- **DELETE** /libros/:id: elimina un libro por su ID.

### Rutas CRUD para Autores:

```
// Rutas CRUD para autores
app.get('/autores', (req, res) => {
  res.json(autores);
});

app.post('/autores', (req, res) => {
  const nuevoAutor = req.body;
  nuevoAutor.id = autores.length + 1;
  autores.push(nuevoAutor);
  res.status(201).json(nuevoAutor);
});

app.get('/libros', (req, res) => {
  res.json(libros);
});

app.put('/autores/:id', (req, res) => {
  const id = req.params.id;
  const index = autores.findIndex(autor => autor.id === parseInt(id));
  if (index !== -1) {
    autores[index] = req.body;
    res.json(autores[index]);
  } else {
    res.status(404).send('Autor no encontrado');
  }
});

app.delete('/autores/:id', (req, res) => {
  const id = req.params.id;
  autores = autores.filter(autor => autor.id !== parseInt(id));
  res.status(204).send();
});
```

- **GET** /autores: retorna todos los autores.
- **POST** /autores: añade un nuevo autor.
- **PUT** /autores/:id: actualiza un autor existente por su ID.
- **DELETE** /autores/:id: elimina un autor por su ID.

## Rutas CRUD para Usuarios

```
// Rutas CRUD para usuarios
app.get('/usuarios', (req, res) => {
  res.json(usuarios);
});

app.post('/usuarios', (req, res) => {
  const nuevoUsuario = req.body;
  nuevoUsuario.id = usuarios.length + 1;
  usuarios.push(nuevoUsuario);
  res.status(201).json(nuevoUsuario);
});

app.put('/usuarios/:id', (req, res) => {
  const id = req.params.id;
  const index = usuarios.findIndex(usuario => usuario.id === parseInt(id));
  if (index !== -1) {
    usuarios[index] = req.body;
    res.json(usuarios[index]);
  } else {
    res.status(404).send('Usuario no encontrado');
  }
});

app.delete('/usuarios/:id', (req, res) => {
  const id = req.params.id;
  usuarios = usuarios.filter(usuario => usuario.id !== parseInt(id));
  res.status(204).send();
});
```

- **GET** /usuarios: retorna todos los usuarios.
- **POST** /usuarios: añade un nuevo usuario.
- **PUT** /usuarios/:id: actualiza un usuario existente por su ID.
- **DELETE** /usuarios/:id: elimina un usuario por su ID.

## Rutas CRUD para Préstamos

```
// Rutas CRUD para préstamos
app.get('/prestamos', (req, res) => {
  res.json(prestamos);
});

app.post('/prestamos', (req, res) => {
  const nuevoPrestamo = req.body;
  nuevoPrestamo.id = prestamos.length + 1;
  prestamos.push(nuevoPrestamo);
  res.status(201).json(nuevoPrestamo);
});

app.put('/prestamos/:id', (req, res) => {
  const id = req.params.id;
  const index = prestamos.findIndex(prestamo => prestamo.id === parseInt(id));
  if (index !== -1) {
    prestamos[index] = req.body;
    res.json(prestamos[index]);
  } else {
    res.status(404).send('Préstamo no encontrado');
  }
});

app.delete('/prestamos/:id', (req, res) => {
  const id = req.params.id;
  prestamos = prestamos.filter(prestamo => prestamo.id !== parseInt(id));
  res.status(204).send();
});
```

- **GET** /prestamos: retorna todos los préstamos.
- **POST** /prestamos: añade un nuevo préstamo.
- **PUT** /prestamos/:id: actualiza un préstamo existente por su ID.
- **DELETE** /prestamos/:id: elimina un préstamo por su ID

## Iniciar el servidor

```
app.listen(PORT, () => {
  console.log(`Servidor escuchando en el puerto ${PORT}`);
});
```



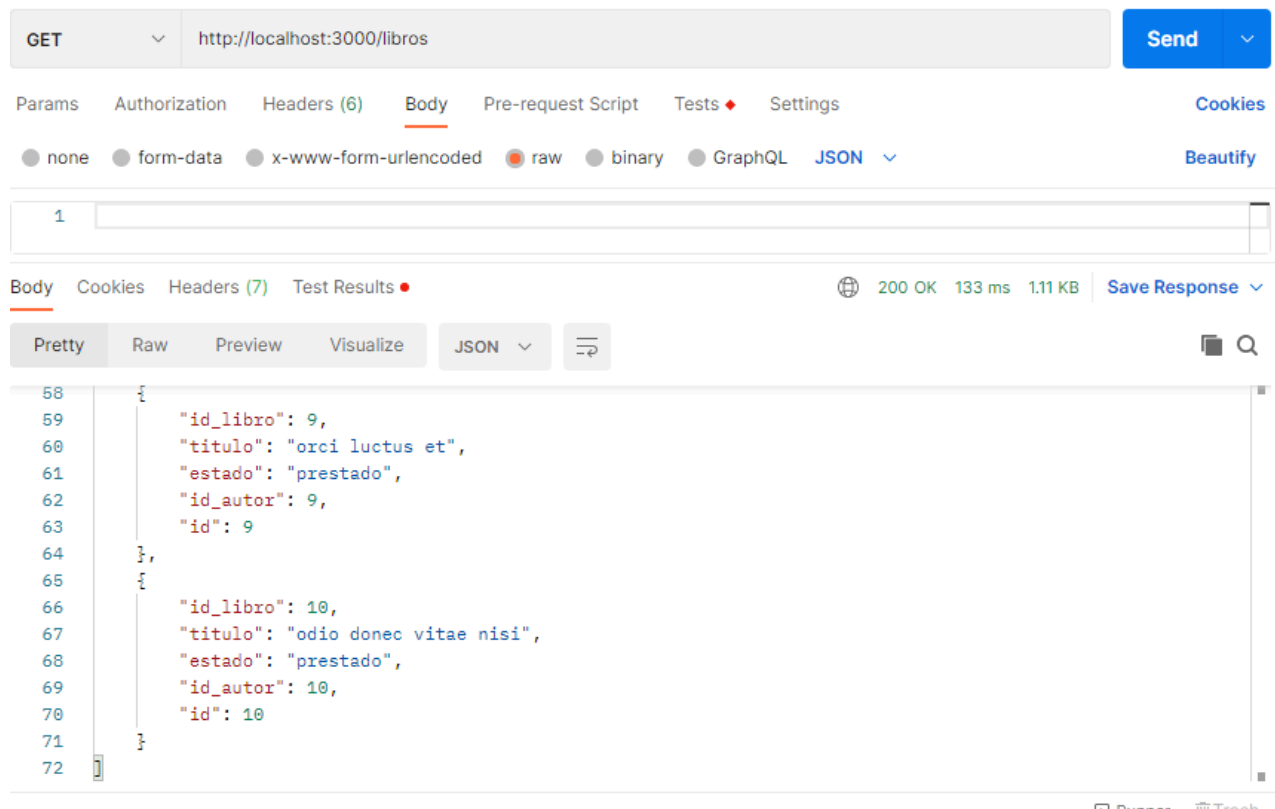
Inicia el servidor y lo pone a escuchar en el puerto definido, mostrando un mensaje en la consola para confirmar que está en funcionamiento.

```
PS C:\Users\Win10\Desktop\Biblioteca> node server.js
Servidor escuchando en el puerto 3000
```

## Probamos CRUD libros

Vamos a realizar los get, post, put y delete en Postman y poder visualizarlos.

**GET** en libros en postman, retorna 10 libros, observando 200 ok(es una respuesta estándar para una solicitud HTTP exitosa)



**POST** en libros con `id_libro :11`

POST http://localhost:3000/libros

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON Beautify

```
1 {"id_libro":11,"titulo":"Mountain","estado":"disponible","id_autor":1,"id":11}
```

Body Cookies Headers (7) Test Results 201 Created 42 ms 318 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_libro": 11,
```

Teníamos 10 libros y se agregó el id\_libro:11. Realizamos otro **POST** id\_libro:12

POST http://localhost:3000/libros/

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookie

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON Beautify

```
1 {"id_libro":12,"titulo":"Brigherton","estado":"disponible","id_autor":2,"id":12}
```

Body Cookies Headers (7) Test Results 201 Created 17 ms 320 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_libro": 12,
3   "titulo": "Brigherton",
4   "estado": "disponible",
5   "id_autor": 2,
6   "id": 12
7 }
```

## Verificamos con GET

The screenshot shows a REST client interface with a GET request to `http://localhost:3000/libros/`. The response is a JSON array with one object, displayed in a pretty-printed format.

```
1 [{"id_libro":12,"titulo":"Brigherton","estado":"disponible","id_autor":2,"id":12}]
```

The response status is 200 OK, with a response time of 15 ms and a size of 1.27 KB. The response is saved.

```
76      "id_autor": 1,  
77      "id": 11  
78    },  
79    {  
80      "id_libro": 12,  
81      "titulo": "Brigherton",  
82      "estado": "disponible",  
83      "id_autor": 2,  
84      "id": 12  
85    }  
86  ]
```

## PUT id\_libro:2, se cambió el estado a prestado

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/libros/2`. The response is a JSON object, displayed in a pretty-printed format.

```
1 [{"id_libro":2,"titulo":"habitasse platea dictumst maecenas","estado":"prestado","id_autor":2,"id":2}]
```

The response status is 200 OK, with a response time of 23 ms and a size of 336 B. The response is saved.

```
1  [{"id_libro": 2,  
2    "titulo": "habitasse platea dictumst maecenas",  
3    "estado": "prestado",  
4    "id_autor": 2,  
5    "id": 2  
6  }  
7  ]
```

Below the REST client, a browser screenshot shows the URL `localhost:3000/libros` and a button for "impresión con formato estilístico". The page content displays a list of books in a JSON array format.

```
{  
  "id_libro":1,"titulo":"mauris vulputate","estado":"prestado","id_autor":1,"id":1},  
  {"id_libro":2,"titulo":"habitasse platea dictumst maecenas","estado":"prestado","id_autor":2,"id":2},  
  {"id_libro":3,"titulo":"feugiat non","estado":"prestado","id_autor":3,"id":3},  
  {"id_libro":4,"titulo":"acinia","estado":"disponible","id_autor":4,"id":4},  
  {"id_libro":5,"titulo":"nulla sed","estado":"disponible","id_autor":5,"id":5},  
  {"id_libro":6,"titulo":"hasellus id","estado":"prestado","id_autor":6,"id":6},  
  {"id_libro":7,"titulo":"sagittis sapien cum sociis","estado":"prestado","id_autor":7,"id":7},  
  {"id_libro":8,"titulo":"lectus","estado":"prestado","id_autor":8,"id":8},  
  {"id_libro":9,"titulo":"orci luctus et","estado":"prestado","id_autor":9,"id":9},  
  {"id_libro":10,"titulo":"isi","estado":"prestado","id_autor":10,"id":10},  
  {"id_libro":11,"titulo":"Mountain","estado":"disponible","id_autor":11,"id":11}  
}
```

## DELETE id\_libro:12

The screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/libros/12`. The response status is `204 No Content`, with a response time of `16 ms` and a size of `134 B`. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, and the response is displayed in a text area.

Luego en localhost podemos observar que el libro se borró y en postman realizamos un GET.

The screenshot shows a REST client interface with a GET request to `http://localhost:3000/libros`. The response status is `200 OK`, with a response time of `15 ms` and a size of `1.19 KB`. The response is displayed in a text area, showing a JSON array of 11 books. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, and the response is displayed in a text area.

```
":1,"titulo":"mauris vulputate","estado":"prestado","id_autor":1,"id":1},{ "id_libro":2,"titulo":"habitasse platea dictumst", "estado":"prestado","id_autor":2,"id":2},{ "id_libro":3,"titulo":"feugiat non","estado":"prestado","id_autor":3,"id":3},{ "id_libro":4,"t", "estado":"disponible","id_autor":4,"id":4},{ "id_libro":5,"titulo":"nulla sed","estado":"disponible","id_autor":5,"id":5},{ "id_libro":6,"", "estado":"prestado","id_autor":6,"id":6},{ "id_libro":7,"titulo":"sagittis sapien cum sociis","estado":"prestado","id_autor":7,"id":7}, {"id_libro":8,"id":8},{ "id_libro":9,"titulo":"orci luctus et","estado":"prestado","id_autor":9,"id":9},{ "id_libro":10,"do":"prestado","id_autor":10,"id":10},{ "id_libro":11,"titulo":"Mountain","estado":"disponible","id_autor":1,"id":11}]
```

## CURL AUTORES

**GET:** retorna 5 autores.

GET ⌵ http://localhost:3000/autores Send

Params Authorization Headers (6) **Body** Pre-request Script Tests ◆ Settings Cooki

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ⌵ Beautif

1

Body Cookies Headers (7) Test Results ● 🌐 200 OK 15 ms 538 B Save Response

Pretty Raw Preview Visualize **JSON** ⌵ ⌵

```
22     "nombre": "Tessi",
23     "apellido": "Billson",
24     "id": 4
25   },
26   {
27     "id_autor": 5,
28     "nombre": "Price",
29     "apellido": "Dickons",
30     "id": 5
31   }
32 ]
```

En localhost:3000/autores

🌐 localhost:3000/autores

Formato estilístico ☐

```
"nombre":"Danit","apellido":"Buxey","id":1},{ "id_autor":2,"nombre":"Nadiya","apellido":"Autin","id":2},{ "id_autor":3,"nombre":"Tessi","apellido":"Billson","id":4},{ "id_autor":5,"nombre":"Price","apellido":"Dickons","id":5}]
```

**POST** id\_autor: 6

POST http://localhost:3000/autores

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {"id_autor":6,"nombre":"Lucas","apellido":"Stanley","id":6}
```

Body Cookies Headers (7) Test Results 201 Created 20 ms 299 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_autor": 6,
3   "nombre": "Lucas",
4   "apellido": "Stanley",
5   "id": 6
6 }
```

POST id\_autor:7

POST http://localhost:3000/autores

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"id_autor":7,"nombre":"Nadia","apellido":"Christensen","id":7}
```

Body Cookies Headers (7) Test Results 201 Created 29 ms 303 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_autor": 7,
3   "nombre": "Nadia",
4   "apellido": "Christensen",
5   "id": 7
6 }
```

PUT id\_autor: 6, se cambia nombre y apellido

PUT http://localhost:3000/autores/5

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"id_autor":5,"nombre":"Carlos","apellido":"Gonzalez","id":5}
```

Body Cookies Headers (7) Test Results 200 OK 14 ms 296 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_autor": 5,
3   "nombre": "Carlos",
4   "apellido": "Gonzalez",
5   "id": 5
6 }
```

**DELETE** id\_autor: 7

DELETE http://localhost:3000/autores/7

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"id_autor":5,"nombre":"Carlos","apellido":"Gonzalez","id":5}
```

Body Cookies Headers (4) Test Results 204 No Content 17 ms

Implementamos un GET de autores nuevamente para observar autores

GET http://localhost:3000/autores

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 [{"id_autor":5,"nombre":"Carlos","apellido":"Gonzalez","id":5}]
```

Body Cookies Headers (7) Test Results 200 OK 31 ms 600 B

Pretty Raw Preview Visualize JSON

```
28     "nombre": "Carlos",
29     "apellido": "Gonzalez",
30     "id": 5
31   },
32   {
33     "id_autor": 6,
34     "nombre": "Lucas",
35     "apellido": "Stanley",
36     "id": 6
37   }
38 ]
```

## CURL USUARIOS

### GET usuarios retornando 5 usuarios

GET http://localhost:3000/usuarios

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 |
```

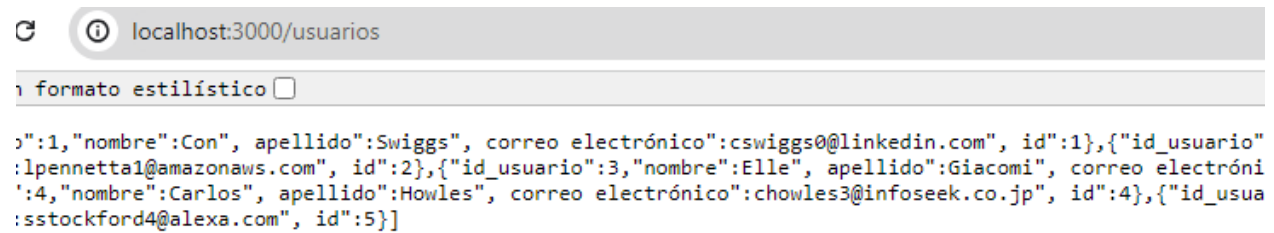
Body Cookies Headers (7) Test Results 200 OK 16 ms 718 B Save Response

Pretty Raw Preview Visualize JSON

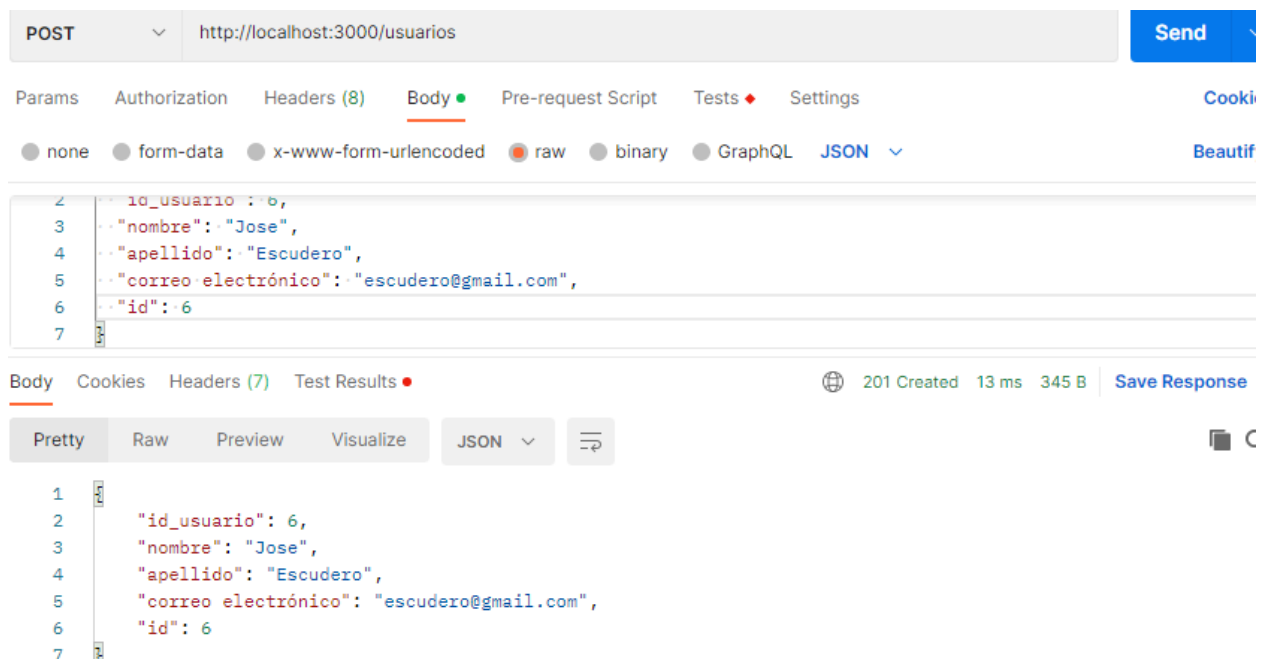
```
27     "email": "chowles3@infoseek.co.jp",
28     "id": 4
29   },
30   {
31     "id_usuario": 5,
32     "nombre": "Simeon",
33     "apellido": "Stockford",
34     "email": "sstockford4@alexa.com",
35     "id": 5
36   }
37 ]
```



En localhost:3000/usuarios



POST id\_usuario:6



Creamos el usuario con id:7

POST http://localhost:3000/usuarios

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
2 {
3   "nombre": "Marcelo",
4   "apellido": "Fernandez",
5   "correo electrónico": "fernandez@gmail.com",
6   "id": 7
7 }
```

Body Cookies Headers (7) Test Results 201 Created 13 ms 350 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_usuario": 7,
3   "nombre": "Marcelo",
4   "apellido": "Fernandez",
5   "correo electrónico": "fernandez@gmail.com",
6   "id": 7
7 }
```

**GET** para verificar que tenemos 7 usuarios

GET http://localhost:3000/usuarios Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cooki

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautif

```
2 {
3   "nombre": "Marcelo",
4   "apellido": "Fernandez",
5   "correo electrónico": "fernandez@gmail.com",
6   "id": 7
7 }
```

Body Cookies Headers (7) Test Results 200 OK 14 ms 933 B Save Response

Pretty Raw Preview Visualize JSON

```
41 {
42   "correo electrónico": "escudero@gmail.com",
43   "id": 6
44 },
45 {
46   "id_usuario": 7,
47   "nombre": "Marcelo",
48   "apellido": "Fernandez",
49   "correo electrónico": "fernandez@gmail.com",
50   "id": 7
51 }
```

**PUT** con el usuario id:7, cambiamos el nombre, apellido y email.Observamos el cambio realizado.

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/usuarios/7`. The 'Body' tab is selected, showing a JSON payload. The status bar indicates a 200 OK response with a 31 ms duration and 337 B of data.

```
1 {
2   "id_usuario": 7,
3   "nombre": "Maria",
4   "apellido": "Garcia",
5   "correo electrónico": "garcia@gmail.com",
6   "id": 7
7 }
```

Body Cookies Headers (7) Test Results • 200 OK 31 ms 337 B Save

Pretty Raw Preview Visualize JSON { }

```
1 {
2   "id_usuario": 7,
3   "nombre": "Maria",
4   "apellido": "Garcia",
5   "correo electrónico": "garcia@gmail.com",
6   "id": 7
7 }
```

Realizamos un **DELETE** id\_usuario: 7

The screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/usuarios/7`. The 'Body' tab is selected, showing the same JSON payload as the previous request. The status bar indicates a 204 No Content response with a 14 ms duration and 134 B of data.

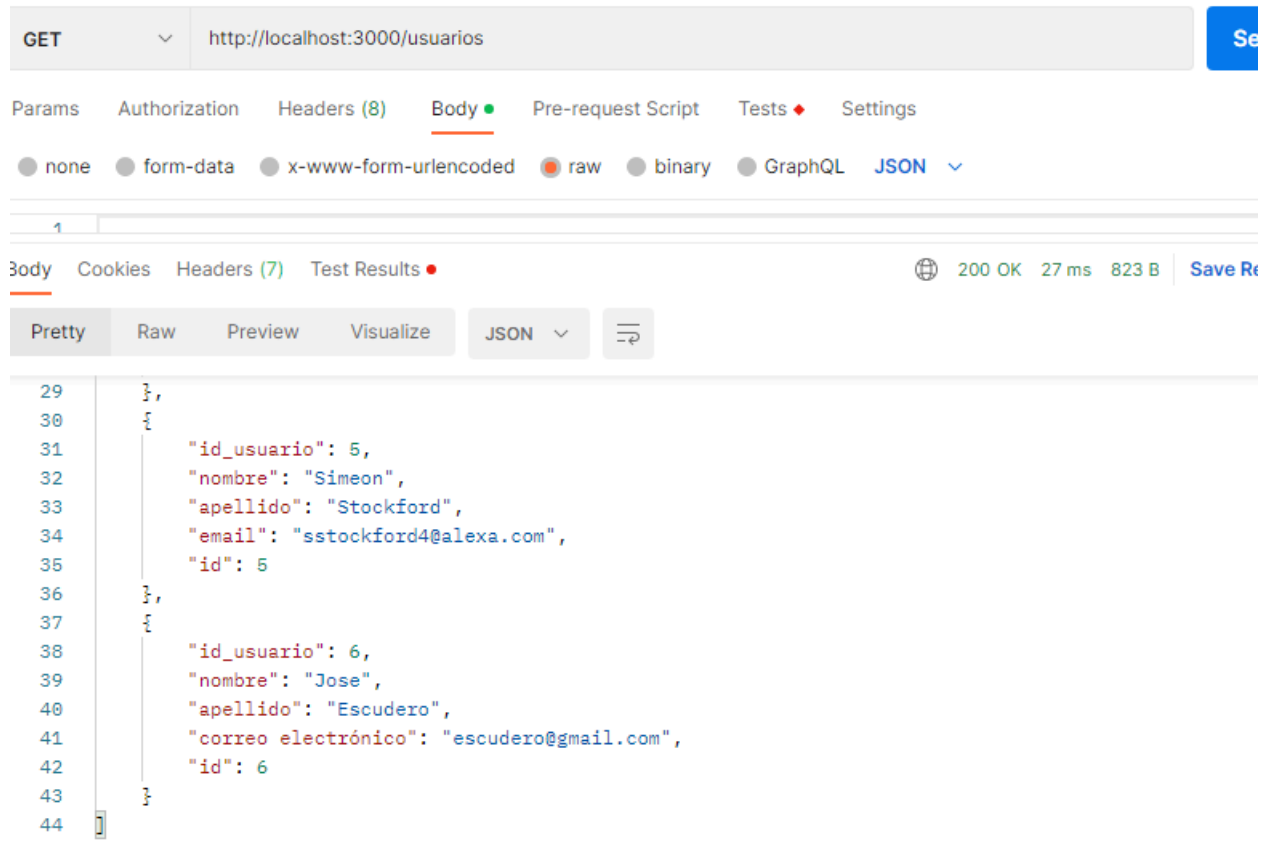
```
1 {
2   "id_usuario": 7,
3   "nombre": "Maria",
4   "apellido": "Garcia",
5   "correo electrónico": "garcia@gmail.com",
6   "id": 7
7 }
```

Body Cookies Headers (4) Test Results • 204 No Content 14 ms 134 B Save Res

Pretty Raw Preview Visualize Text { }

```
1
```

Verificamos que se borro



GET http://localhost:3000/usuarios

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

Body Cookies Headers (7) Test Results 200 OK 27 ms 823 B Save Response

Pretty Raw Preview Visualize JSON

```
29   },
30   {
31     "id_usuario": 5,
32     "nombre": "Simeon",
33     "apellido": "Stockford",
34     "email": "sstockford4@alexa.com",
35     "id": 5
36   },
37   {
38     "id_usuario": 6,
39     "nombre": "Jose",
40     "apellido": "Escudero",
41     "correo electrónico": "escudero@gmail.com",
42     "id": 6
43   }
44 ]
```

## CURL Préstamos

Observamos con **GET** que tenemos 5 préstamos

```
Impresión con formato estético
{"id_prestamo":1,"id_libro":1,"id_usuario":1,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":1},
{"id_prestamo":2,"id_libro":2,"id_usuario":2,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":2},
{"id_prestamo":3,"id_libro":3,"id_usuario":3,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":3},
{"id_prestamo":4,"id_libro":4,"id_usuario":4,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":4},
{"id_prestamo":5,"id_libro":5,"id_usuario":5,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":5}]
```

**POST** id\_prestamo: 6 y 7

POST http://localhost:3000/prestamos

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "id_prestamo": 6,
```

Body Cookies Headers (7) Test Results 201 Created 43 ms 350 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_prestamo": 6,
3   "id_libro": 6,
4   "id_usuario": 6,
5   "fecha_inicio": "7/2/2024",
6   "fecha_devolucion": "7/15/2024",
7   "id": 6
8 }
```

POST http://localhost:3000/prestamos

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
7   ... "id": 7
8 }
```

Body Cookies Headers (7) Test Results 201 Created 13 ms 350 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_prestamo": 7,
3   "id_libro": 7,
4   "id_usuario": 7,
5   "fecha_inicio": "7/2/2024",
6   "fecha_devolucion": "7/15/2024",
7   "id": 7
8 }
```

**GET** retorna 7 préstamos:

GET http://localhost:3000/prestamos Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Co

none form-data x-www-form-urlencoded raw binary GraphQL JSON Be

7 ... "id": 7  
8 ...

Body Cookies Headers (7) Test Results 200 OK 13 ms 1008 B Save Respon

Pretty Raw Preview Visualize JSON

```
45      "id_usuario": 6,  
46      "fecha_inicio": "7/2/2024",  
47      "fecha_devolucion": "7/15/2024",  
48      "id": 6  
49    },  
50    {  
51      "id_prestamo": 7,  
52      "id_libro": 7,  
53      "id_usuario": 7,  
54      "fecha_inicio": "7/2/2024",  
55      "fecha_devolucion": "7/15/2024",  
56      "id": 7  
57    }  
58  ]
```

Runner

Realizamos un **PUT** id\_prestamo: 7, modificando la fecha de devolución del préstamos.

PUT http://localhost:3000/prestamos/7 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings C

none form-data x-www-form-urlencoded raw binary GraphQL JSON Be

```
1  {  
2    "id_prestamo": 7,  
3    "id_libro": 7,  
4    "id_usuario": 7,  
5    "fecha_inicio": "7/2/2024",  
6    "fecha_devolucion": "7/31/2024",  
7    "id": 7  
8  }
```

Body Cookies Headers (7) Test Results 200 OK 18 ms 345 B Save Respo

Pretty Raw Preview Visualize JSON

```
1  {  
2    "id_prestamo": 7,  
3    "id_libro": 7,  
4    "id_usuario": 7,  
5    "fecha_inicio": "7/2/2024",  
6    "fecha devolucion": "7/31/2024".  
7  }
```

Runner

## DELETE id\_prestamo: 7

DELETE

http://localhost:3000/prestamos/7

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

Body

Cookies

Headers (4)

Test Results

204 No Content

15 ms

134 B

Save

Pretty

Raw

Preview

Visualize

Text

1

GET

http://localhost:3000/prestamos/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

Body

Cookies

Headers (7)

Test Results

200 OK

12 ms

81 B

Save

Pretty

Raw

Preview

Visualize

JSON

```
35     "id_prestamo": 5,
36     "id_libro": 5,
37     "id_usuario": 5,
38     "fecha_inicio": "7/2/2024",
39     "fecha_devolucion": "7/15/2024",
40     "id": 5
41   },
42   {
43     "id_prestamo": 6,
44     "id_libro": 6,
45     "id_usuario": 6,
46     "fecha_inicio": "7/2/2024",
47     "fecha_devolucion": "7/15/2024",
48     "id": 6
49   }
50 ]
```

localhost:3000/prestamos

Impresión con formato estilístico

```
[{"id_prestamo":1,"id_libro":1,"id_usuario":1,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":1}, {"id_prestamo":2,"id_libro":2,"id_usuario":2,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":2}, {"id_prestamo":3,"id_libro":3,"id_usuario":3,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":3}, {"id_prestamo":4,"id_libro":4,"id_usuario":4,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":4}, {"id_prestamo":5,"id_libro":5,"id_usuario":5,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":5}, {"id_prestamo":6,"id_libro":6,"id_usuario":6,"fecha_inicio":"7/2/2024","fecha_devolucion":"7/15/2024","id":6}]
```