

Nombre del estudiante: Santiago Bedoya Arias

Documento 8161893

Curso: Ingeniería Web II

Código del curso: PREPROF2201PC-TDS0236

Nombre del programa:
TECNOLOGÍA EN DESARROLLO DE SOFTWARE

Evidencia de aprendizaje: EA1, API REST NodeJs

Fecha: 15/05/2022

Docente: Julio Cesar Martínez Zarate

Ciudad Medellín

Objetivo

Desarrollar una API REST utilizando JavaScript del lado del servidor con NodeJs.

Instrucciones

Para el desarrollo de la actividad se debe realizar lo siguiente:

Leer el documento Caso_de_Estudio.pdf [Descargar Caso_de_Estudio.pdf](#).

Diseñar y construir una API REST que permita realizar lo solicitado en los módulos de: Tipo de equipo, Estado de equipo, Usuarios y marcas.

Aspectos a tener en cuenta para construir la API REST:

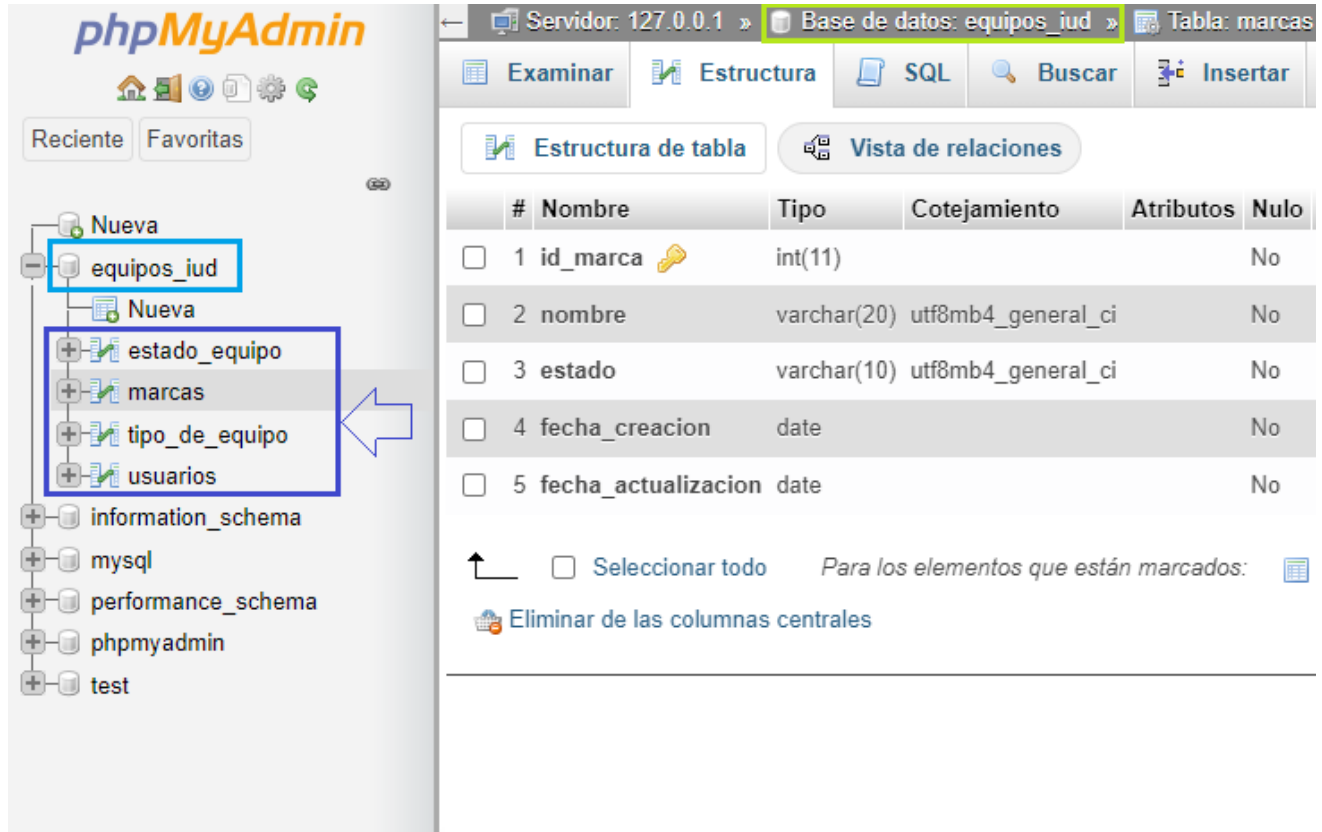
Diseñar la base de datos.

Construir los distintos métodos que realizarán las operaciones requeridas de la API (POST, PUT, DELETE, GET, etc.).

El funcionamiento de la API se realizará utilizando Postman o algún otro cliente que el estudiante considere.

Desarrollo

- Crear la base de datos y módulos solicitados en la actividad:



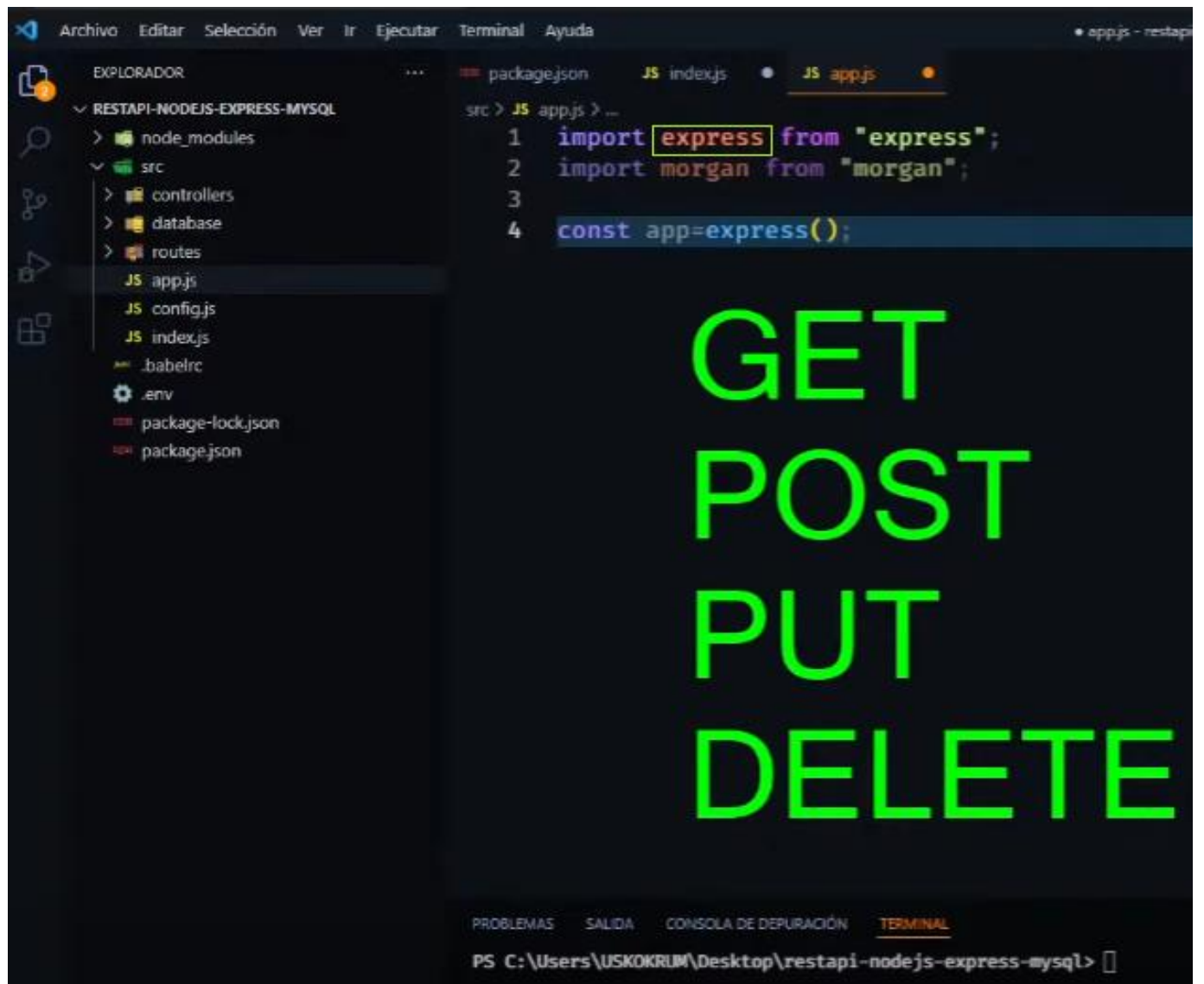
The screenshot shows the phpMyAdmin interface. On the left, the database 'equipos_iud' is selected, and its tables are listed: 'estado_equipo', 'marcas', 'tipo_de_equipo', and 'usuarios'. A blue box highlights the 'marcas' table. On the right, the 'Estructura de tabla' (Table Structure) view for the 'marcas' table is displayed. The table has 5 columns: 'id_marca' (int(11), primary key), 'nombre' (varchar(20), utf8mb4_general_ci), 'estado' (varchar(10), utf8mb4_general_ci), 'fecha_creacion' (date), and 'fecha_actualizacion' (date). The 'id_marca' column is marked as the primary key with a key icon. Below the table structure, there are options to 'Seleccionar todo' (Select all) and 'Eliminar de las columnas centrales' (Remove from central columns).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo
1	id_marca	int(11)			No
2	nombre	varchar(20)	utf8mb4_general_ci		No
3	estado	varchar(10)	utf8mb4_general_ci		No
4	fecha_creacion	date			No
5	fecha_actualizacion	date			No

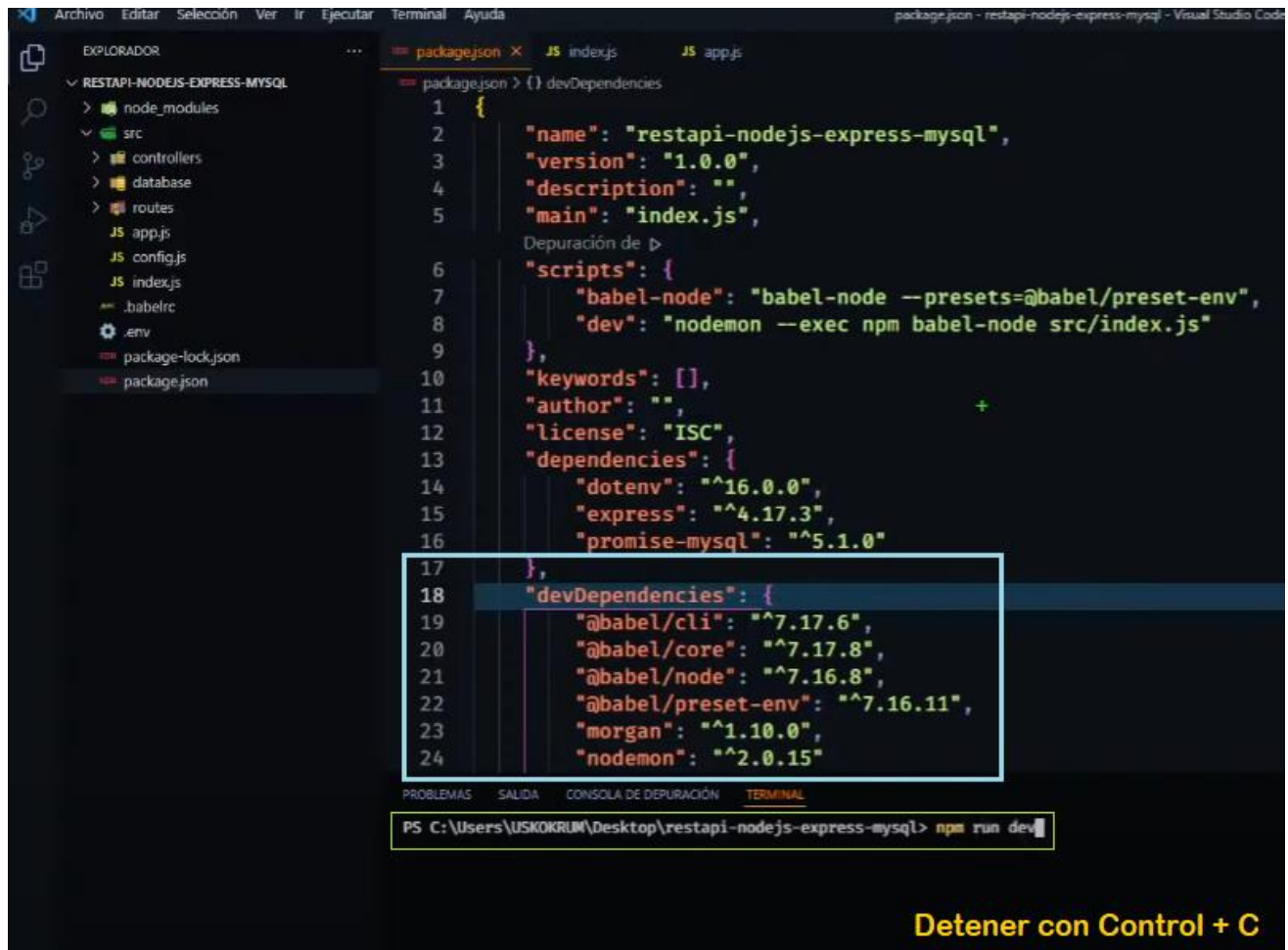
- Importar **Express**

Express es un framework web transigente, escrito en JavaScript y alojado dentro del entorno de ejecución NodeJS.

Proporciona los mecanismos para la escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).



- Dependencias babel, Morgan y nodemon



```
1 {
2   "name": "restapi-nodejs-express-mysql",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "babel-node": "babel-node --presets=@babel/preset-env",
8     "dev": "nodemon --exec npm babel-node src/index.js"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "dotenv": "^16.0.0",
15    "express": "^4.17.3",
16    "promise-mysql": "^5.1.0"
17  },
18  "devDependencies": {
19    "@babel/cli": "^7.17.6",
20    "@babel/core": "^7.17.8",
21    "@babel/node": "^7.16.8",
22    "@babel/preset-env": "^7.16.11",
23    "morgan": "^1.10.0",
24    "nodemon": "^2.0.15"
25  }
26}
```

PS C:\Users\USKOKRUM\Desktop\restapi-nodejs-express-mysql> npm run dev

Detener con Control + C

Babel es un "compilador" (o transpilador) para JavaScript. Básicamente permite transformar código escrito con las últimas y novedosas características de JavaScript y transformarlo en un código que sea entendido por navegadores más antiguos.

Morgan es un Middleware de nivel de solicitud HTTP. Es una gran herramienta que registra las requests junto con alguna otra información dependiendo de su configuración y el valor predeterminado utilizado. Demuestra ser muy útil durante la depuración y también si desea crear archivos de registro.

Nodemon es una utilidad de interfaz de línea de comandos (CLI) desarrollada por @rem que envuelve su aplicación Node, vigila el sistema de archivos y reinicia automáticamente el proceso.

- Servidor en puerto 4000 y Nodemon corriendo OK!

The screenshot shows the Visual Studio Code interface with the `package.json` file open in the editor. The file contains the following configuration:

```
1 {
2   "name": "rest_api_node",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "babel-node": "babel-node --presets=@babel/preset-env",
7     "dev": "nodemon --exec npm run babel-node src/index.js"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "dotenv": "^16.0.0",
14    "express": "^4.18.1",
15    "promise-mysql": "^5.2.0",
16    "psql": "^0.0.1"
17  },
18  "devDependencies": {
19    "@babel/cli": "^7.17.10",
20    "@babel/core": "^7.17.10",
21    "@babel/node": "^7.17.10",
22    "@babel/preset-env": "^7.17.10",
23    "morgan": "^1.10.0",
24    "nodemon": "^2.0.16"
25  }
26 }
```

The terminal output at the bottom shows the Nodemon process starting and running the application:

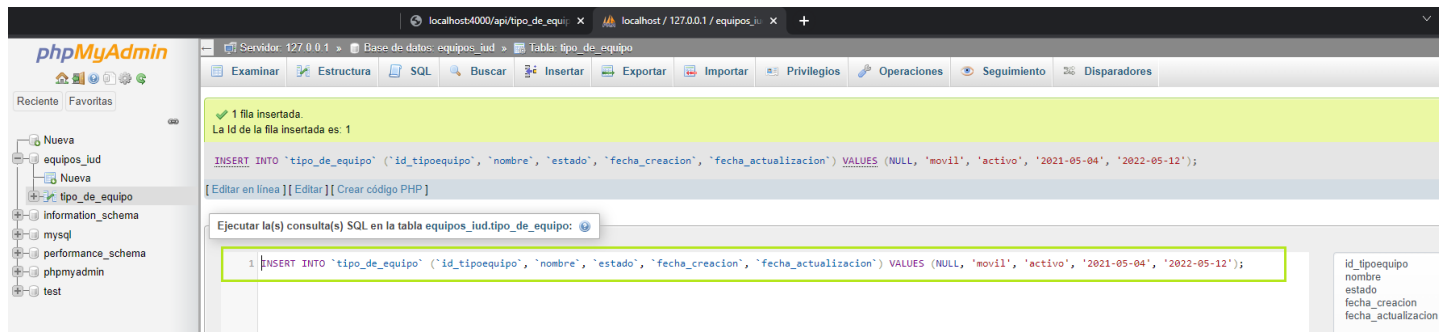
```
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `npm run babel-node src/index.js`

> rest_api_node@1.0.0 babel-node
> babel-node --presets=@babel/preset-env "src/index.js"

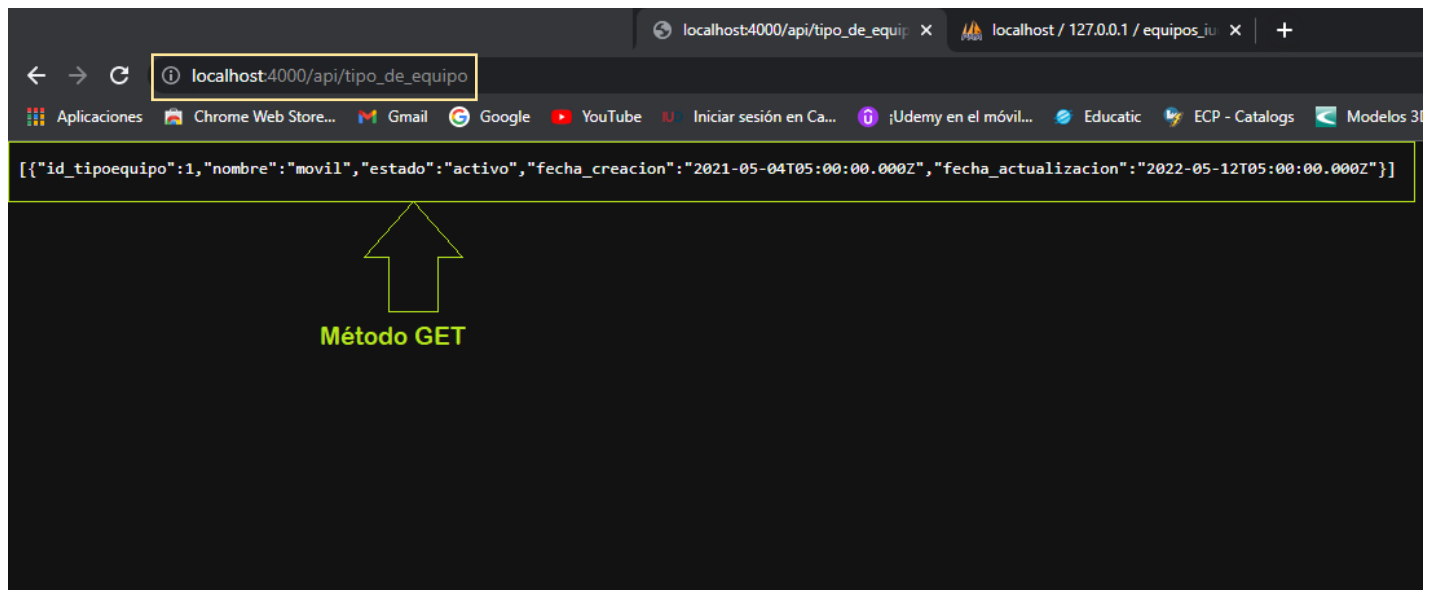
Server on port 4000
```

The terminal output is highlighted with a yellow box, and the text "OK" is written next to it.

- Ahora voy a insertar un registro en la base de datos para que exista información en ella y poder utilizar el método GET de prueba:



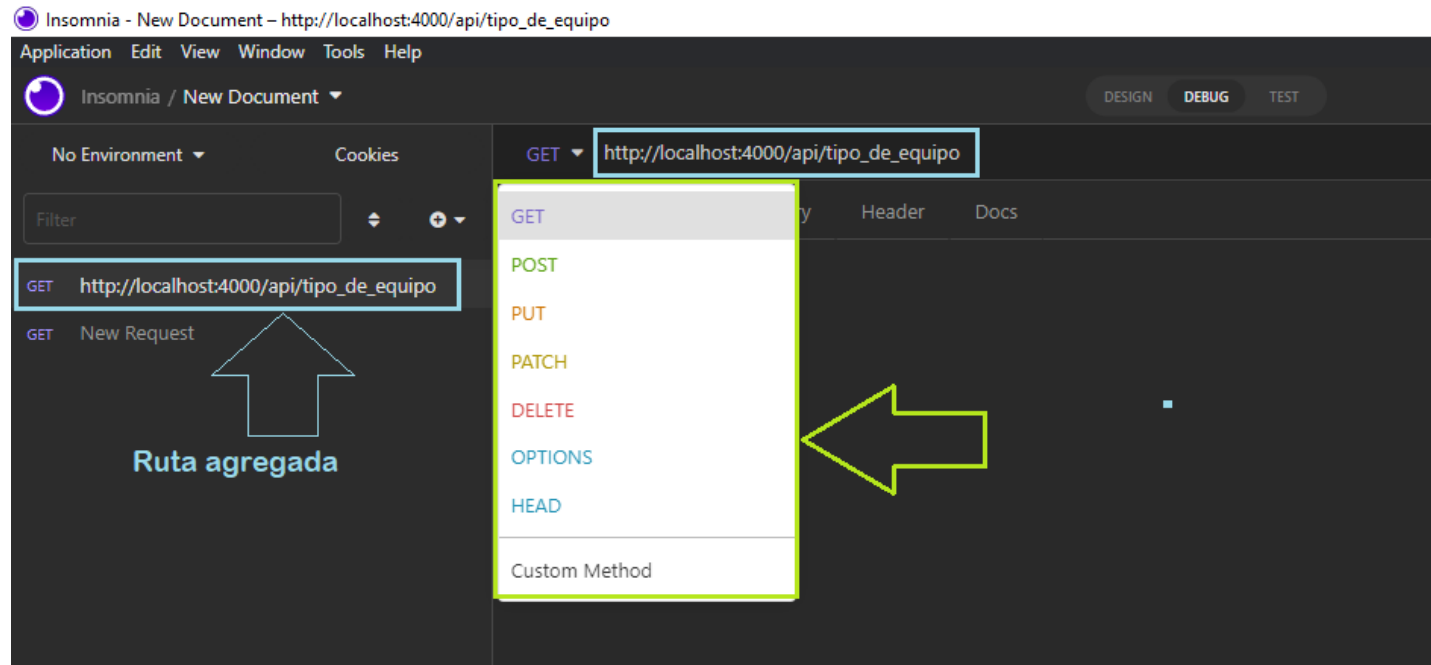
- Ya con un registro creado en la base de datos puedo hacer una petición GET de prueba desde el navegador:



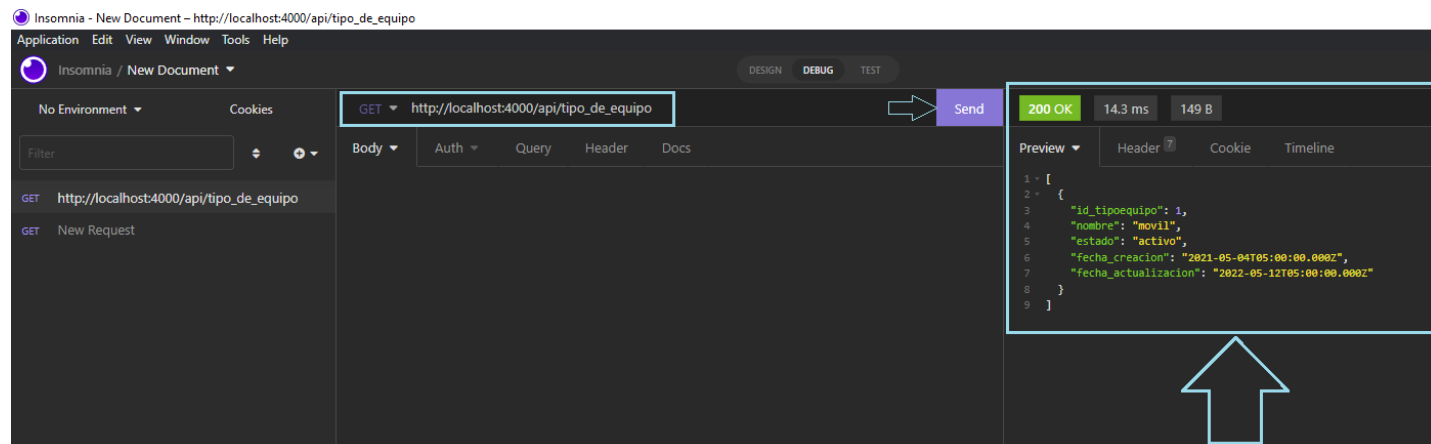
- Utilizaré **Insomnia** en lugar de Potsman.

Insomnia es una forma gráfica y sencilla de realizar consultas curl contra una API. La ventaja reside en que todas las consultas las guardas en la propia aplicación, para poderlas utilizar posteriormente.

Debo agregar la ruta para poder a empezar a realizar las **peticiones HTTP**:



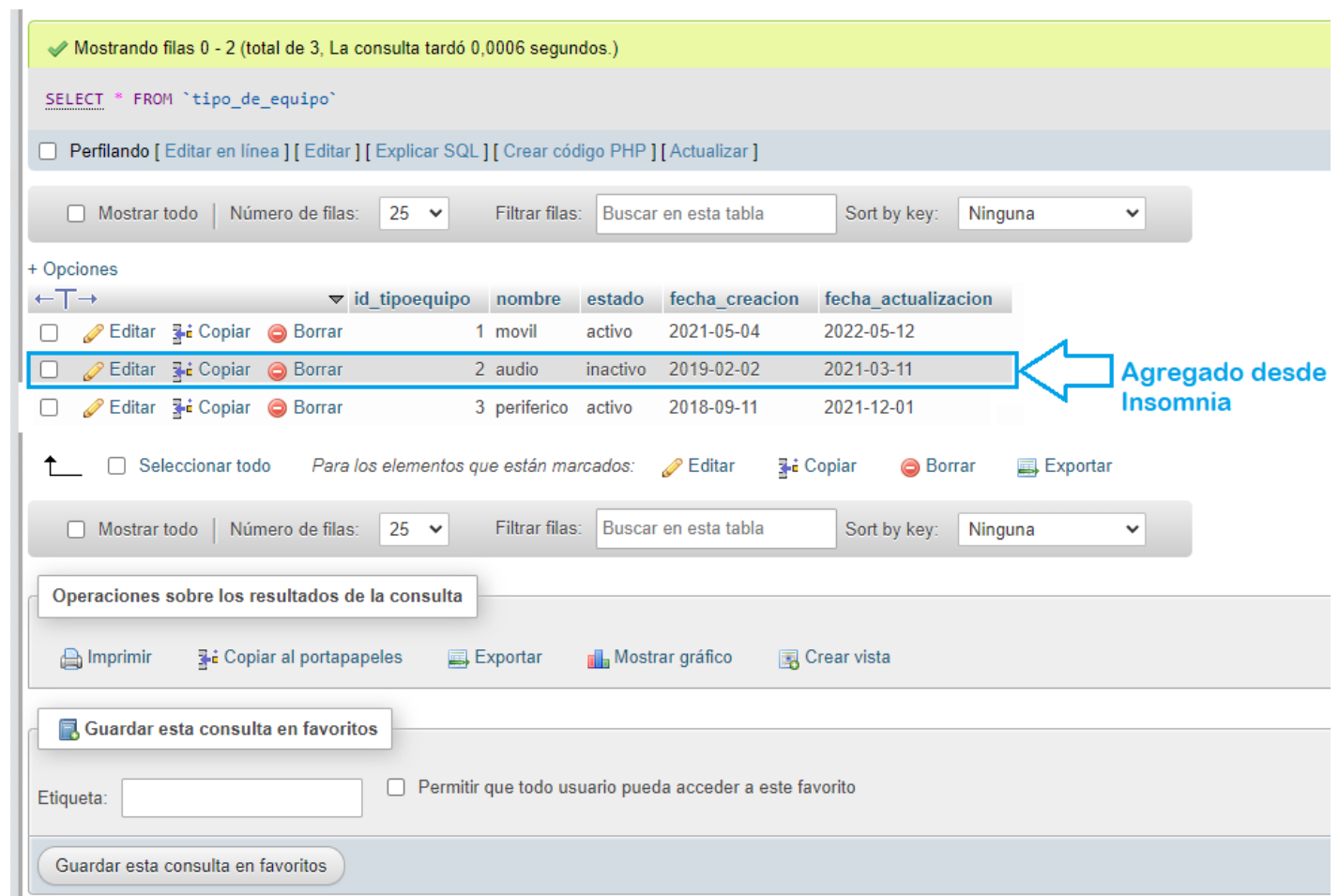
- Uso del método **GET** desde Insomnia, veremos el registro que tengo actualmente en la base de datos:



- Uso del método **POST** desde Insomnia para agregar un nuevo request:



- Comprobamos la inserción correcta en la base de datos:



- Voy a usar de nuevo el request **GET** para ver los registros actualizados:

GET `http://localhost:4000/api/tipo_de_equipo` Send 200 OK 5.76 ms 452 B

The request has succeeded.

JSON ▾ Auth ▾ Query Header 1 Docs

```
1 {
2   "nombre": "periferico",
3   "estado": "activo",
4   "fecha_creacion": "2018-09-11T05:00:00.000Z",
5   "fecha_actualizacion": "2021-12-01T05:00:00.000Z"
6 }
```

Metodo GET de lo que hay actualmente en la base de datos

```
1 [
2   {
3     "id_tipoequipo": 1,
4     "nombre": "movil",
5     "estado": "activo",
6     "fecha_creacion": "2021-05-04T05:00:00.000Z",
7     "fecha_actualizacion": "2022-05-12T05:00:00.000Z"
8   },
9   {
10    "id_tipoequipo": 2,
11    "nombre": "audio",
12    "estado": "inactivo",
13    "fecha_creacion": "2019-02-02T05:00:00.000Z",
14    "fecha_actualizacion": "2021-03-11T05:00:00.000Z"
15  },
16  {
17    "id_tipoequipo": 3,
18    "nombre": "periferico",
19    "estado": "activo",
20    "fecha_creacion": "2018-09-11T05:00:00.000Z",
21    "fecha_actualizacion": "2021-12-01T05:00:00.000Z"
22  }
23 ]
```

- Utilizar la petición **GET** enviándole un parámetro para que busque un registro por su ID:

DESIGN DEBUG TEST

GET `http://localhost:4000/api/tipo_de_equipo/3` Send 200 OK 40.1 ms 154 B

JSON ▾ Auth ▾ Query Header 1 Docs

```
1 {
2   "nombre": "periferico",
3   "estado": "activo",
4   "fecha_creacion": "2018-09-11T05:00:00.000Z",
5   "fecha_actualizacion": "2021-12-01T05:00:00.000Z"
6 }
```

Metodo GET utilizando solo el ID

Preview ▾ Header 7 Cookie Timeline

```
1 [
2   {
3     "id_tipoequipo": 3,
4     "nombre": "periferico",
5     "estado": "activo",
6     "fecha_creacion": "2018-09-11T05:00:00.000Z",
7     "fecha_actualizacion": "2021-12-01T05:00:00.000Z"
8   }
9 ]
```

- Método **DELETE** para eliminar un registro de la base de datos:

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** `http://localhost:4000/api/tipo_de_equipo/3`
- Status:** 200 OK
- Time:** 28.5 ms
- JSON Body:**

```
1 {
2   "nombre": "periferico",
3   "estado": "activo",
4   "fecha_creacion": "2018-09-11T05:00:00.000Z",
5   "fecha_actualizacion": "2021-12-01T05:00:00.000Z"
6 }
```
- Preview:**

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "serverStatus": 2,
6   "warningCount": 0,
7   "message": "",
8   "protocol41": true,
9   "changedRows": 0
10 }
```

A large arrow points from the JSON body to the Preview section.

Comprobar el DELETE en la terminal:

The screenshot shows a code editor and a terminal window. The code editor contains the following JavaScript code:

```
8 router.get("/", tipo_de_equipoController.getTipo_de_equipo);
9 router.get("/:id", tipo_de_equipoController.getTipo_de equip);
10 router.post("/", tipo_de_equipoController.addTipo_de_equipo);
11 router.put("/:id", tipo_de_equipoController.updateTipo_de_equipo);
12 router.delete("/:id", tipo_de_equipoController.deleteTipo_de_equipo);
13
14 export default router;
15
```

The terminal window shows the following output:

```
> rest_api_node@1.0.0 babel-node
> babel-node --presets=@babel/preset-env "src/index.js"

[nodemon] restarting due to changes...
[nodemon] starting `npm run babel-node src/index.js`

> rest_api_node@1.0.0 babel-node
> babel-node --presets=@babel/preset-env "src/index.js"

Server on port 4000
DELETE /api/tipo_de_equipo/3 200 22.098 ms - 127
```

A large arrow points from the terminal output to the REST client interface in the previous block.

Comprobar el **DELETE** en la base de datos:

✓ Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0006 segundos.)

```
SELECT * FROM `tipo_de_equipo`
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Sort by key: Ninguna

+ Opciones

		id_tipoequipo	nombre	estado	fecha_creacion	fecha_actualizacion		
<input type="checkbox"/>				1	movil	activo	2021-05-04	2022-05-12
<input type="checkbox"/>				2	audio	inactivo	2019-02-02	2021-03-11

← Selecc. todo Para los elementos que están marcados:

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Sort by key: Ninguna

Operaciones sobre los resultados de la consulta

Guardar esta consulta en favoritos

Etiqueta: ☐ Permitir que todo usuario pueda acceder a este favorito

DELETE OK, ya no está el registro con ID 3

- Método **PUT** para **editar** un registro existente.

En el lado izquierdo pongo los nuevos datos que quiero cambiar, estoy editando el id 1:

PUT 200 OK 5.18 ms 299 B

JSON Auth Query Header 1 Docs

```
1 {
2   "nombre": "escritorio",
3   "estado": "inactivo",
4   "fecha_creacion": "2021-05-04T05:00:00.000Z",
5   "fecha_actualizacion": "2022-05-14T05:00:00.000Z"
6 }
7
8
9
```

Registro con ID 1

Modificando los datos del registro con ID 1

Preview Header 7 Cookie Timeline

```
1 {
2   "id_tipoequipo": 1,
3   "nombre": "movil",
4   "estado": "activo",
5   "fecha_creacion": "2021-05-04T05:00:00.000Z",
6   "fecha_actualizacion": "2022-05-12T05:00:00.000Z"
7 },
8
9 {
10  "id_tipoequipo": 2,
11  "nombre": "audio",
12  "estado": "inactivo",
13  "fecha_creacion": "2019-02-02T05:00:00.000Z",
14  "fecha_actualizacion": "2021-03-11T05:00:00.000Z"
15 }
16 ]
```

Ahora voy a enviar el request:

The screenshot shows a REST client interface. The top bar indicates a PUT request to `http://localhost:4000/api/tipo_de_equipo/1` with a status of 200 OK, a response time of 8.94 ms, and a body size of 168 B. The JSON tab on the left shows the request body:

```
{  "nombre": "escritorio",  "estado": "inactivo",  "fecha_creacion": "2021-05-04T05:00:00.000Z",  "fecha_actualizacion": "2022-05-14T05:00:00.000Z"}
```

. The Preview tab on the right shows the response body:

```
{  "fieldCount": 0,  "affectedRows": 1,  "insertId": 0,  "serverStatus": 2,  "warningCount": 1,  "message": "(Rows matched: 1 Changed: 1 Warnings: 1",  "protocol41": true,  "changedRows": 1}
```

Verificar la operación **PUT** en la base de datos:

The screenshot shows a database management tool interface. At the top, it says "Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0006 segundos.)". Below this, the SQL query `SELECT * FROM `tipo_de_equipo`` is shown. The table view displays two rows. The first row is highlighted with a green box and an arrow pointing to it from the text "El UPDATE fue correcto en la base de datos". The first row has the following values: `id_tipoequipo`: 1, `nombre`: escritorio, `estado`: inactivo, `fecha_creacion`: 2021-05-04, `fecha_actualizacion`: 2022-05-14. The second row has the following values: `id_tipoequipo`: 2, `nombre`: audio, `estado`: inactivo, `fecha_creacion`: 2019-02-02, `fecha_actualizacion`: 2021-03-11.

Verificamos el **PUT** anterior en la terminal desde JavaScript:

```
79 res.status(500);  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
DELETE /api/tipo_de_equipo/3 200 22.098 ms - 127  
[nodemon] restarting due to changes...  
[nodemon] starting `npm run babel-node src/index.js`  
  
> rest_api_node@1.0.0 babel-node  
> babel-node --presets=@babel/preset-env "src/index.js"  
  
Server on port 4000  
GET /api/tipo_de_equipo/3 200 20.351 ms - 2  
GET /api/tipo_de_equipo/ 200 3.261 ms - 299  
PUT /api/tipo_de_equipo/1 200 7.233 ms - 168  
└─
```

Conclusión

Se realiza la Rest Api requerida con JavaScript y NodeJs, además de las dependencias mencionadas. Con **Insomnia** hago las pruebas API, funcionando como cliente HTTP que nos da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cual se obtienen los diferentes tipos de respuesta que posteriormente deberán ser validados.

Tras cada request enviado correctamente se verifican los cambios en la terminal con Node y la base de datos MySQL y se observa la interacción y conexión entre el front, el back y la base de datos.

Realicé las consultas GET, PUT, POST y DELETE con los módulos descritos en la actividad. Documenté solo los request con el **módulo de tipo de equipo** en este documento ya que hacerlo con todos extendería demasiado esta presentación.

Nota: se adjunta todo el código del proyecto en GITHUB y pantallazos de los pasos.

¡Gracias!

Bibliografía

Nodemon

<https://www.silversites.es/desarrollo-web/que-es-nodemon/>

Insomnia

<https://atareao.es/software/programacion/insomnia-un-productivo-cliente-rest/#:~:text=Insomnia%2C%20nos%20permite%20crear%20variables,producci%C3%B3n%20y%20otro%20de%20desarrollo.>

Babel

<https://www.freecodecamp.org/espanol/news/que-es-babel/>

Morgan

<https://es.acervolima.com/que-es-morgan-en-node-js/>

Express

https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction