Documentación `Api-Restaurante`

Introducción

Este documento detalla la estructura y configuración de la API Restaurante, diseñada para gestionar información relacionada con usuarios y restaurantes. La API utiliza una base de datos MySQL y está implementada con Docker para facilitar la portabilidad y escalabilidad del sistema.

Creación de las tablas de base de datos

Creacion de la tabla `usuarios:`

```
CREATE TABLE `usuarios` (
  `id` int(4) PRIMARY KEY AUTO_INCREMENT,
  `email` varchar(150) NOT NULL,
  `password` varchar(240) NOT NULL,
  `nombre` varchar(200) NOT NULL,
  `imagen` varchar(200) DEFAULT NULL,
  `disponible` tinyint(1) NOT NULL,
  `token` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='tabla de usuarios';
```

Creacion de la tabla `restaurantes:`

```
CREATE TABLE `restaurantes` (
  `id` int(7) AUTO_INCREMENT PRIMARY KEY,
  `id_usuario` int(11) NOT NULL,
  `nombre` varchar(100) NOT NULL,
  `ciudad` varchar(100) NOT NULL,
  `provincia` varchar(100) NOT NULL,
  `telefono` varchar(250) NOT NULL,
  `imagen` varchar(100) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Creamos la relación entre el usuario y restaurante

```
ALTER TABLE `restaurantes`
ADD CONSTRAINT FK_id_usuario FOREIGN KEY (id_usuario)
REFERENCES `usuarios` (id);
```

Creamos el archivo .env:

Este archivo de configuración debe estar en el gitinore para no subirlo cuando se hace el commit, por que contiene las credenciales de acceso.

```
MYSQL_DATABASE= aqui va el nombre de la base de datos
MYSQL_PASSWORD=  aqui va la contraseña del usuario
MYSQL_ROOT_PASSWORD= aqui  va la contraseña del root
MYSQL_USER= aqui va el username
MYSQL_PORT= aqui va el puerto al que se va a mapear
PHPMYADMIN_PORT= puerto de  phpmyadmin
PORT= puerto http
```

Creamos el docker-compose

```
version: '3'version: "3.1"
services:
    db:
        image: mysql    -Utiliza la imagen oficial de MySQL desde Docke
        ports:
            - "${MYSQL_PORT}:3306"   -Mapea el puerto del host al puert
        command: --default-authentication-plugin=mysql_native_password
        environment:
            MYSQL_DATABASE: ${MYSQL_DATABASE}   -Nombre de la base de da
            MYSQL_PASSWORD: ${MYSQL_PASSWORD}   -Contraseña de la base d
            MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}  -Contraseña de
        volumes:
            - ./dump:/docker-entrypoint-initdb.d  - Monta el directorio
            - ./conf:/etc/mysql/conf.d  -Monta el directorio 'conf' para
            - persistent:/var/lib/mysql  -Monta el volumen 'persistent'
        networks:
            - default  -Usa la red predeterminada
    www:
        build: .
        ports:
            - "{PORT}:80"
        volumes:
            - ./www:/var/www/html
        links:
            - db
        networks:
            - default
    phpmyadmin:
        image: phpmyadmin/phpmyadmin
        links:
            - db:db
        ports:
            - ${PHPMYADMIN_PORT}:80
        environment:
            MYSQL_USER: ${MYSQL_USER}
            MYSQL_PASSWORD: ${MYSQL_PASSWORD}
            MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
volumes:
    persistent:
```

RestauranteClass

Se realizaron modificaciones en ciertas partes del codigo

Se cambiaron los parametros permitidos para hacer las consultas con los que necesitaba para mi modelo de datos.

```
class restaurante extends Database
{
    private $table = 'restaurantes';

    //parámetros permitidos para hacer consultas selección.
    private $allowedConditions_get = array(
        'id',
        'id_usuario',
        'nombre',
        'ciudad',
        'provincia',
        'telefono',
        'imagen',
        'page'
    );


    //parámetros permitidos para la inserción.
    private $allowedConditions_insert = array(
        'id_usuario',
        'nombre',
        'ciudad',
        'provincia',
        'telefono',
        'imagen'
    );

    //parámetros permitidos para la actualización.
    private $allowedConditions_update = array(
        'id_usuario',
        'nombre',
        'ciudad',
        'provincia',
```

```
            'telefono',
            'imagen'

        );
    }
```

Se validan los nuevos parametros ingresados.

```
    private function validate($data){

                if(!isset($data['id_usuario']) || empty($data['id_usuar
                        $response = array(
                                'result' => 'error',
                                'details' => 'El campo id del usuario es
                        );
                        Response::result(400, $response);
                        exit;
                }

                if(!isset($data['nombre']) || empty($data['nombre'])){
                        $response = array(
                                'result' => 'error',
                                'details' => 'El campo nombre es obliga
                        );

                        Response::result(400, $response);
                        exit;
                }
                if(!isset($data['ciudad']) || empty($data['ciudad'])){
                        $response = array(
                                'result' => 'error',
                                'details' => 'El campo ciudad es obliga
                        );

                        Response::result(400, $response);
                        exit;
                }
                if(!isset($data['provincia']) || empty($data['provincia
                        $response = array(
                                'result' => 'error',
                                'details' => 'El campo provincia es obl
```

```
                        );

                        Response::result(400, $response);
                        exit;
                }

                if(!isset($data['telefono']) || empty($data['telefono']
                        $response = array(
                                'result' => 'error',
                                'details' => 'El campo telefono es obli
                        );

                        Response::result(400, $response);
                        exit;
                }
    }
```

Auth Model

Modificamos para que las variables sean constantes, mas adelante utilizare mi archivo de configuracion .env para estos dato, para que queden ocultos.

```
/**
 * Modelo para la authenticación.
 */
    class AuthModel{
        private $connection;
        const DB_HOST = 'db';
        const DB_USER = 'root';
        const DB_PASSWORD = 'john';
        const DB_NAME = 'restaurantDb';
        const DB_PORT = '3306';

        public function __construct(){
            $this->connection = new mysqli(self::DB_HOST, self::DB_USER

            if($this->connection->connect_errno){
                echo 'Error de conexión a la base de datos';
                exit;
            }
```

```
        }
    }
```

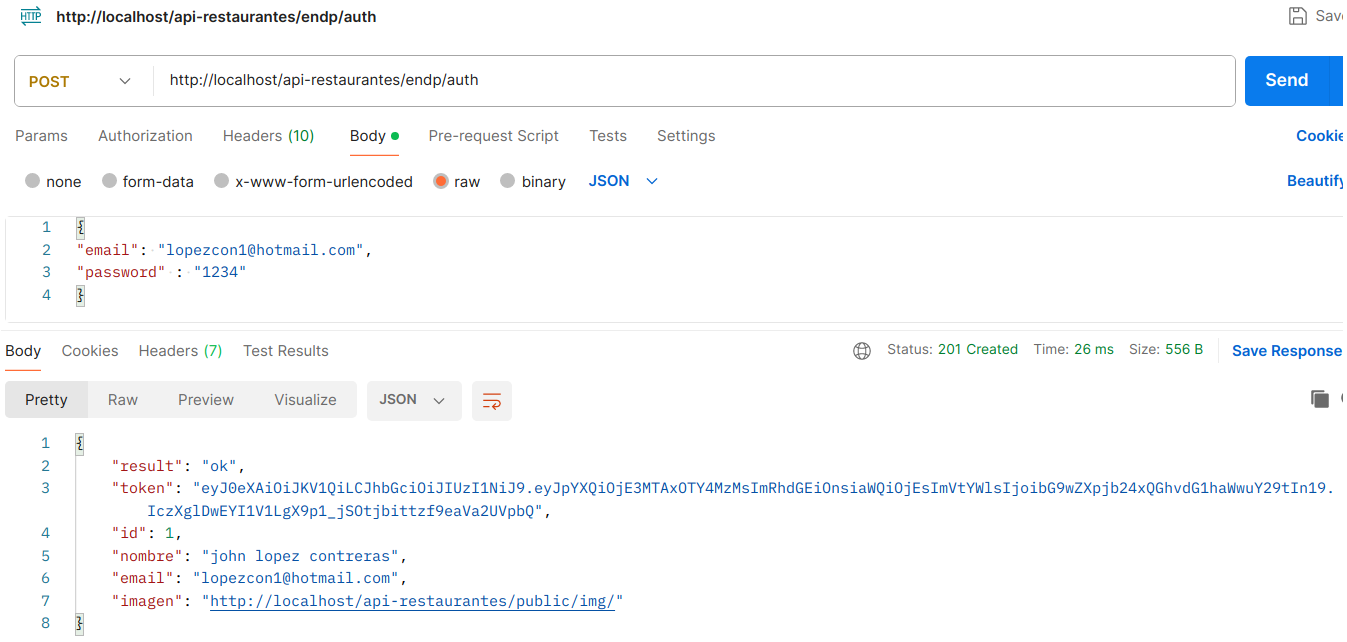
Se modifical las consultas a la base de datos para prevenir inyección sql

```
    class AuthModel{
        public function login($email, $password)
        {
            $query = "SELECT id, nombre, email FROM usuarios WHERE emai
            $statement = $this->connection->prepare($query);
            $statement->bind_param('ss', $email, $password);
            $statement->execute();
            $result = $statement->get_result();
            return $result->fetch_all(MYSQLI_ASSOC);
        }


        public function update($id, $token)
        {
            $query = "UPDATE usuarios SET token = ? WHERE id = ?";
            $statement = $this->connection->prepare($query);
            $statement->bind_param('si', $token, $id);
            $statement->execute();
            return $statement->affected_rows;
        }


        public function getById($id)
        {
            $query = "SELECT token FROM usuarios WHERE id = ?";
            $statement = $this->connection->prepare($query);
            $statement->bind_param('i', $id);
            $statement->execute();
            $result = $statement->get_result();
            $resultArray = $result->fetch_all(MYSQLI_ASSOC);
            return $resultArray;
        }

        public function insertarLog($milog){
            $query = "INSERT INTO log (log) VALUES(?)";
            $statement = $this->connection->prepare($query);
            $statement->bind_param('s', $milog);
```

```
            $statement->execute();
        }

        public function devUserModel($id)
        {
            $query = "SELECT id, nombre, email, imagen FROM usuarios WHI
            $statement = $this->connection->prepare($query);
            $statement->bind_param('i', $id);
            $statement->execute();
            $result = $statement->get_result();
            $resultArray = $result->fetch_all(MYSQLI_ASSOC);
            return $resultArray;
        }
    }
```

Imagenes de consultas a la api con posmant

Hacemos login



Listamos todos los restaurantes

http://localhost/api-restaurantes/endp/restaurante

GET http://localhost/api-restaurantes/endp/restaurante Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookie

☑	Content-Type	application/json
☑	api-key	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE3MTAxOTY4MzMsImRhdG...
	Key	Value

Body Cookies Headers (7) Test Results Status: 200 OK Time: 16 ms Size: 1.69 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "result": "ok",
    "restaurantes": [
        {
            "id": "8",
            "id_usuario": "1",
            "nombre": "Casa Marcial",
            "ciudad": "Arriondas",
            "provincia": "Asturias",
            "telefono": "985 84 09 91",
            "imagen": "http://localhost/api-restaurantes/public/img/65edc2afbd06f.JPEG"
        },
        {
            "id": "9"
```


Buscamos un resturante por su id

http://localhost/api-restaurantes/endp/restaurante

GET http://localhost/api-restaurantes/endp/restaurante?id=11 Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookie

☑	Content-Type	application/json
☑	api-key	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE3MTAxOTY4MzMsImRhdG...
	Key	Value

Body Cookies Headers (7) Test Results Status: 200 OK Time: 8 ms Size: 475 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "result": "ok",
    "restaurantes": [
        {
            "id": "11",
            "id_usuario": "1",
            "nombre": "magoga",
            "ciudad": "Abadía Retuerta LeDomaine",
            "provincia": "Valladolid",
            "telefono": "983 68 76 00",
            "imagen": "http://localhost/api-restaurantes/public/img/65edd4ba630db.JPEG"
        }
    ]
```

Editamos un restaurante por su id

http://localhost/api-restaurantes/endp/restaurante

PUT http://localhost/api-restaurantes/endp/restaurante?id=13 Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookie

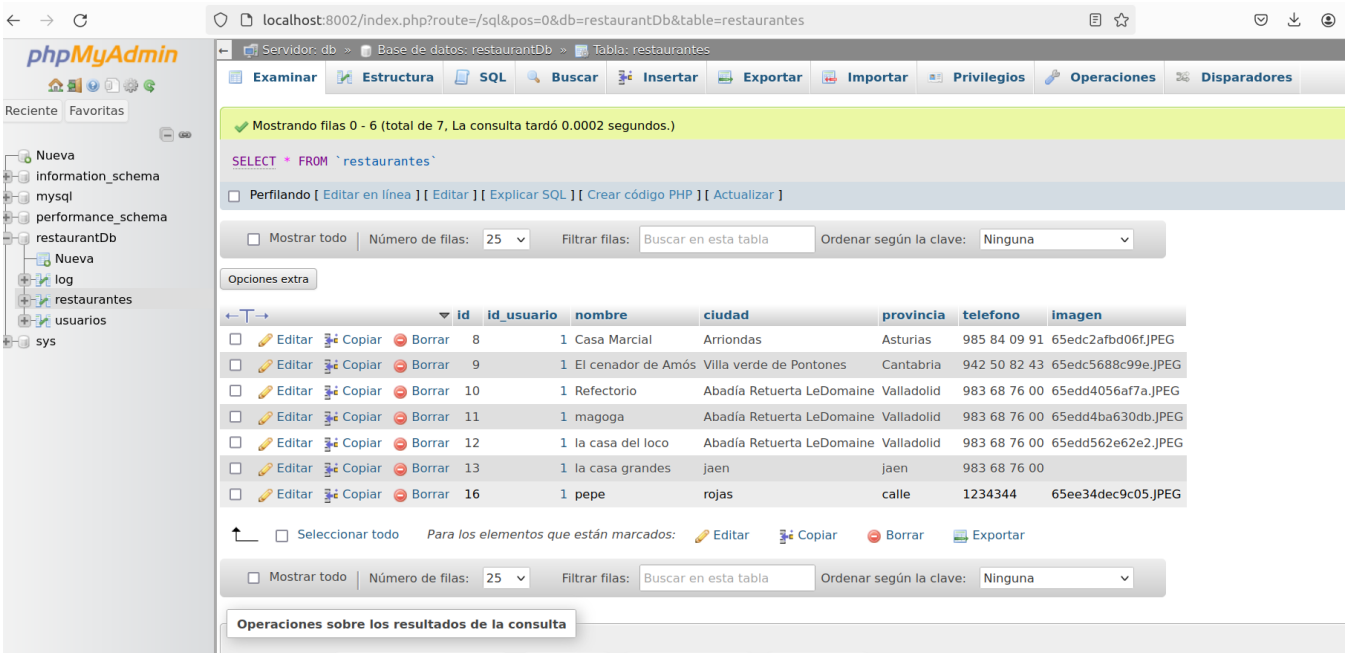
none form-data x-www-form-urlencoded raw binary JSON Beautify

```
{
    "nombre": "la casa grandes",
    "ciudad": "jaen",
    "provincia": "jaen",
    "telefono": "983 68 76 00"
}
```

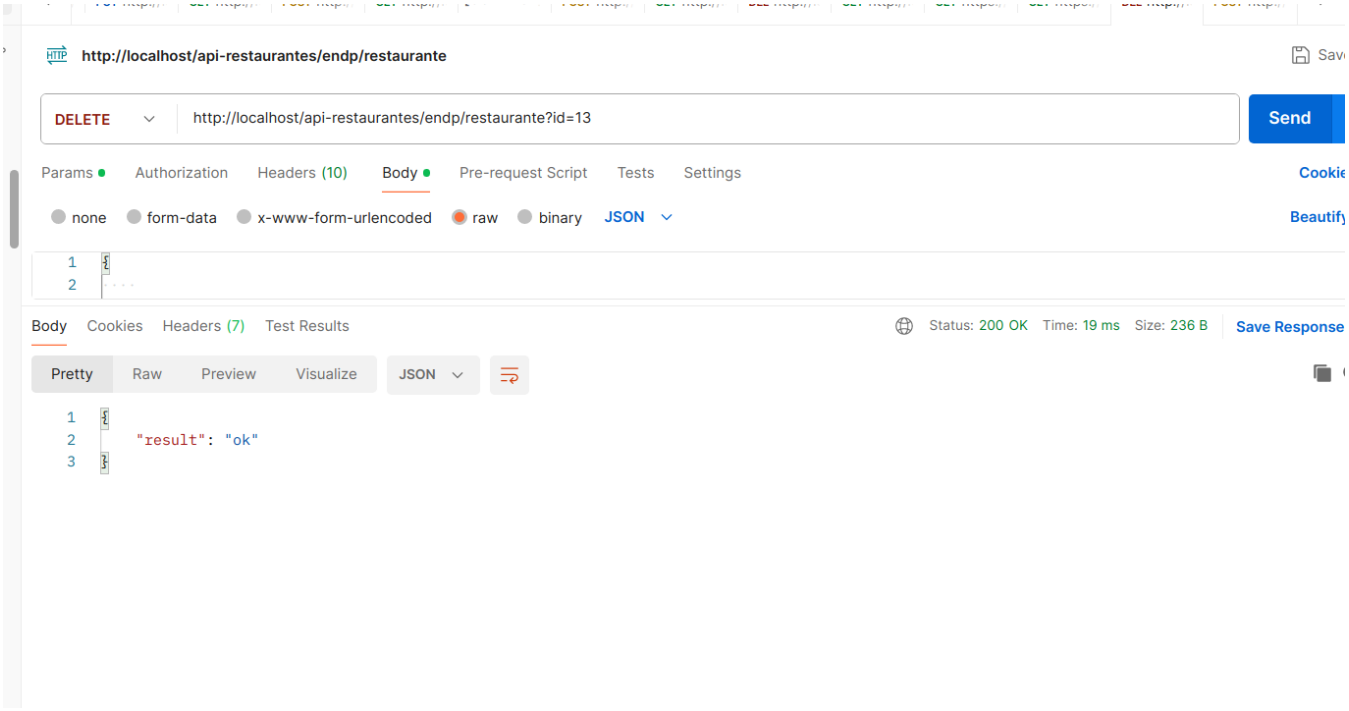
Body Cookies Headers (7) Test Results Status: 200 OK Time: 20 ms Size: 264 B Save Response

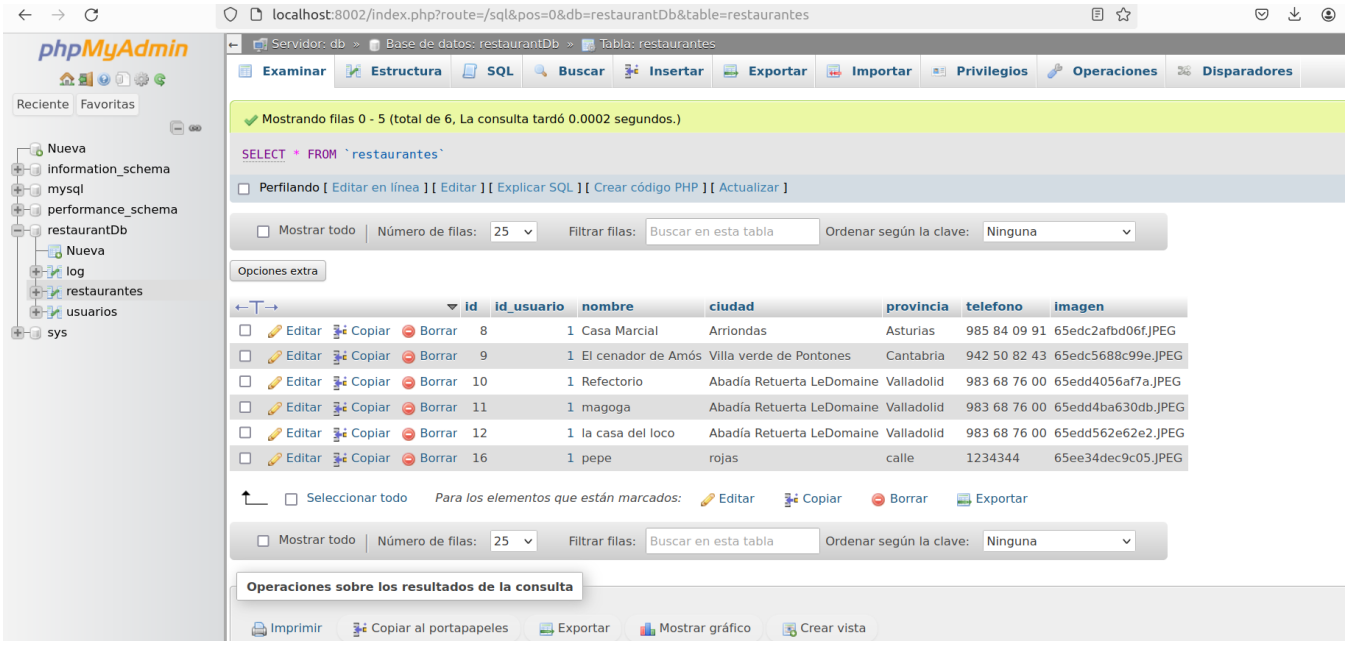
Pretty Raw Preview Visualize JSON

```
{
    "result": "ok actualizacion",
    "file_img": ""
}
```

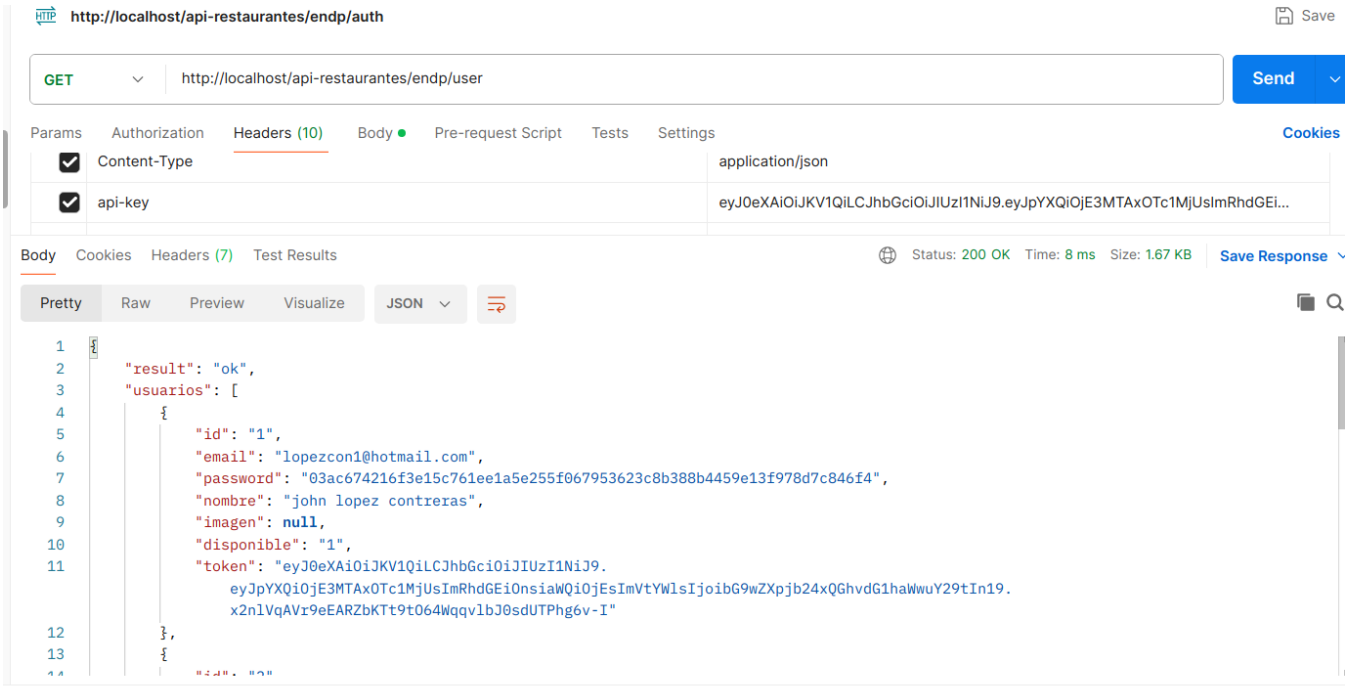


Elimonamos un restaurante por su id

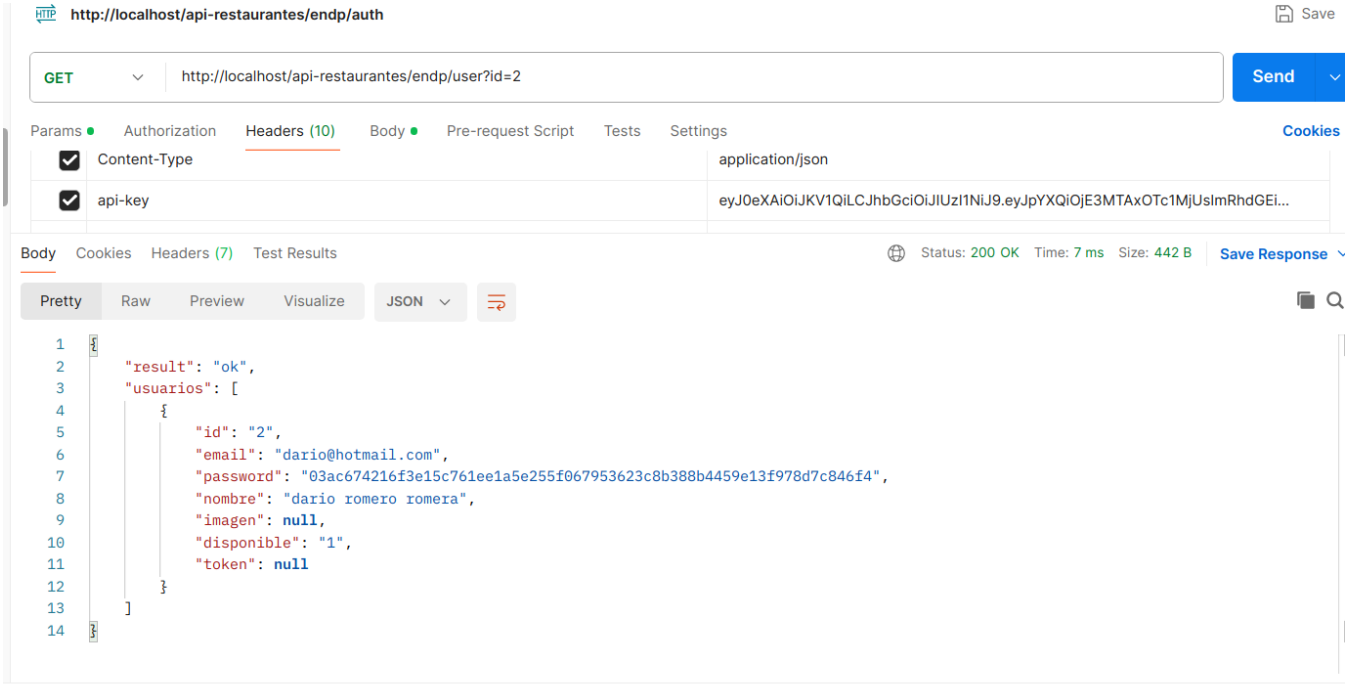




Listamos todos los usuarios



Buscamos un usuario por su id



REPOSITORIO:https://github.com/johnlopez0505/docker-lamp-restaurantes.git