

# 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 gitignore 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 Docker
    ports:
      - "${MYSQL_PORT}:3306"    -Mapea el puerto del host al puerto
    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
      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 código

Se cambiaron los parámetros 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_usuario'])){
        $response = array(
            'result' => 'error',
            'details' => 'El campo id del usuario es obligatorio'
        );
        Response::result(400, $response);
        exit;
    }

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

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

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

```

```

        );

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

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

        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 autenticación.
 */
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 modifican las consultas a la base de datos para prevenir inyección sql

```
class AuthModel{  
    public function login($email, $password)  
    {  
        $query = "SELECT id, nombre, email FROM usuarios WHERE email = '$email'";  
        $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

The screenshot shows a Postman interface for a REST client. The URL is `http://localhost/api-restaurantes/endpoint/auth`. The request method is `POST`. The request body is in JSON format with the following content:

```

{
  "email": "lopezcon1@hotmail.com",
  "password": "1234"
}

```

The response status is `201 Created` with a time of `26 ms` and a size of `556 B`. The response body is in JSON format with the following content:

```

{
  "result": "ok",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlbnRhdGEiOjE0NSiaWQiOiJEsImVtYWlsIjoibG9wZXpjb24xQGhvdG1haWwuY29tIn9.IczXglDwEYI1V1LgX9p1_jS0tjbittzf9eaVa2UVpbQ",
  "id": 1,
  "nombre": "john lopez contreras",
  "email": "lopezcon1@hotmail.com",
  "imagen": "http://localhost/api-restaurantes/public/img/"
}

```

Listamos todos los restaurantes



HTTP <http://localhost/api-restaurantes/endpoint/restaurante> Save

GET <http://localhost/api-restaurantes/endpoint/restaurante> Send

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

<input checked="" type="checkbox"/>	Content-Type	application/json
<input checked="" type="checkbox"/>	api-key	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlMTAxOTY4MzMmRhdG...
	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 ⌵ ⌵

```
1  {
2    "result": "ok",
3    "restaurantes": [
4      {
5        "id": "8",
6        "id_usuario": "1",
7        "nombre": "Casa Marcial",
8        "ciudad": "Arriendas",
9        "provincia": "Asturias",
10       "telefono": "985 84 09 91",
11       "imagen": "http://localhost/api-restaurantes/public/img/65edc2afbd06f.JPEG"
12     },
13     {
14       "id": "9"
```

## Buscamos un restaurante por su id

HTTP <http://localhost/api-restaurantes/endpoint/restaurante> Save

GET <http://localhost/api-restaurantes/endpoint/restaurante?id=11> Send

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

<input checked="" type="checkbox"/>	Content-Type	application/json
<input checked="" type="checkbox"/>	api-key	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlMTAxOTY4MzMmRhdG...
	Key	Value

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

Pretty Raw Preview Visualize JSON ⌵ ⌵

```
1  {
2    "result": "ok",
3    "restaurantes": [
4      {
5        "id": "11",
6        "id_usuario": "1",
7        "nombre": "magoga",
8        "ciudad": "Abadía Retuerta LeDomaine",
9        "provincia": "Valladolid",
10       "telefono": "983 68 76 00",
11       "imagen": "http://localhost/api-restaurantes/public/img/65edd4ba630db.JPEG"
12     }
13   ]
```

# Editamos un restaurante por su id

HTTP <http://localhost/api-restaurantes/endpoint/restaurante> Save

**PUT** <http://localhost/api-restaurantes/endpoint/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

```
1 {
2   ....
3   ....
4   ...."nombre": "la casa grandes",
5   ...."ciudad": "jaen",
6   ...."provincia": "jaen",
7   ...."telefono": "983 68 76 00"
8 }
```

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

**Pretty** Raw Preview Visualize **JSON** ⌵

```
1 {
2   "result": "ok actualizacion",
3   "file_img": ""
4 }
```

localhost:8002/index.php?route=/sql&pos=0&db=restaurantDb&table=restaurantes

Servidor: db - Base de datos: restaurantDb - Tabla: restaurantes

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Disparadores

Mostrando filas 0 - 6 (total de 7, La consulta tardó 0.0002 segundos.)

SELECT \* FROM `restaurantes`

Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Opciones extra

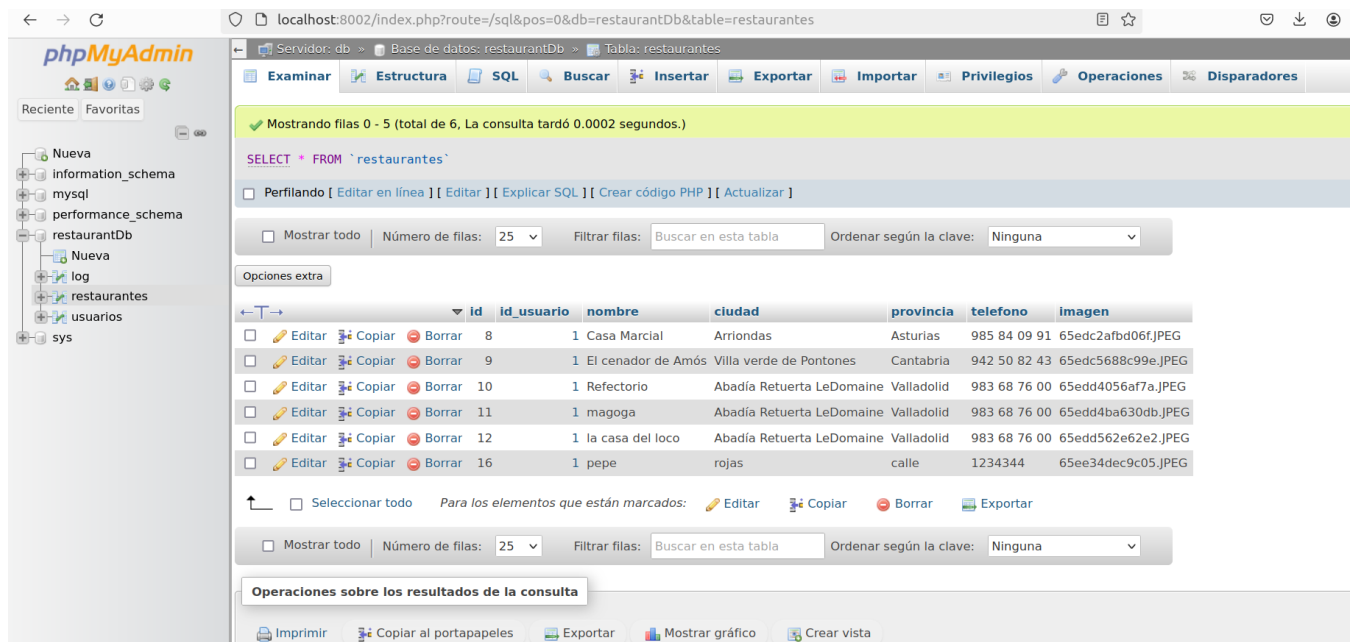
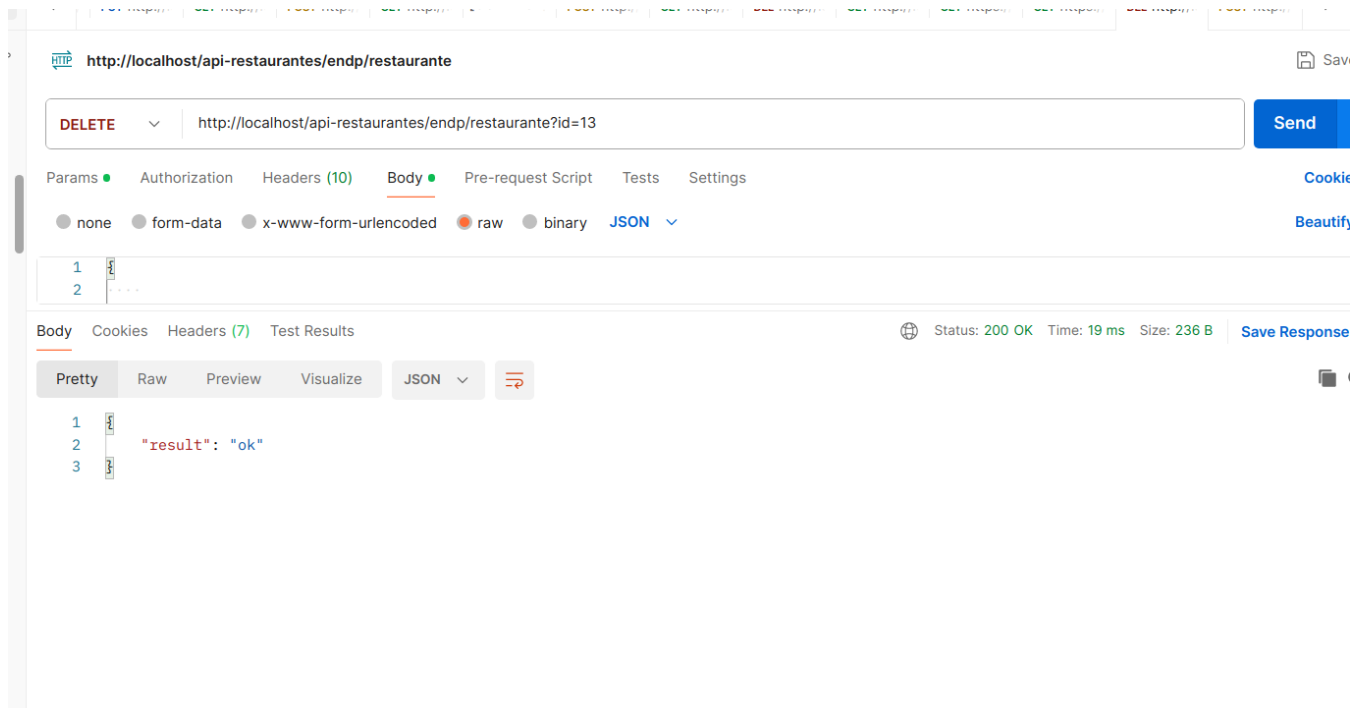
	id	id_usuario	nombre	ciudad	provincia	telefono	imagen
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	8	1	Casa Marcial	Arriendas	Asturias	985 84 09 91	65edc2afbd06f.JPEG
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	9	1	El cenador de Amós	Villa verde de Pontones	Cantabria	942 50 82 43	65edc5688c99e.JPEG
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	10	1	Refectorio	Abadía Retuerta LeDomaine	Valladolid	983 68 76 00	65edd4056af7a.JPEG
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	11	1	magoga	Abadía Retuerta LeDomaine	Valladolid	983 68 76 00	65edd4ba630db.JPEG
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	12	1	la casa del loco	Abadía Retuerta LeDomaine	Valladolid	983 68 76 00	65edd562e62e2.JPEG
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	13	1	la casa grandes	jaen	jaen	983 68 76 00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	16	1	pepe	rojas	calle	1234344	65ee34dec9c05.JPEG

☐ Seleccionar todo | Para los elementos que están marcados: ☐ Editar ☐ Copiar ☐ Borrar ☐ Exportar

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Operaciones sobre los resultados de la consulta

# Elimonamos un restaurante por su id



Listamos todos los usuarios

