EXAMEN TEÓRICO

BEATRIZ ENCISO VILLARREAL

FRONT END DEVELOPER

Creación de microservicios o monolitos:

- ¿Qué son los microservicios y cuál es la diferencia con los monolitos?

Es un nuevo enfoque de programación, modular y una serie de servicios ejecutandose de forma aútonoma y comunicación entre sí.

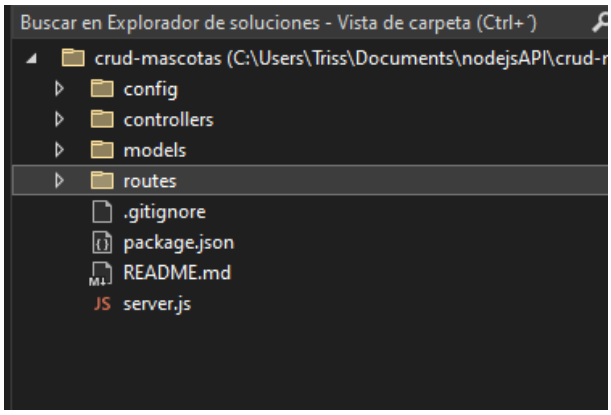
A diferencia de los monolitos, los microservicios son elementos independientes que funcionan en conjunto para llevar a cabo las mismas tareas. Cada uno de esos elementos o procesos es un microservicio.

- ¿Cuáles son las ventajas y desventajas de utilizar microservicios o monolitos?

Monolito		Microservicios	
Ventajas	**Desventajas**	**Ventajas**	**Desventajas**
Deploys sencillos	Dificil mantenimiento	Escalabilidad	Curva de aprendizaje alta
Pruebas sencillas	No escalable	Fácil de mantener	Alta latencia
Menor latencia	Código no reutilizable	Programación modular	Posible complejidad en pruebas.
Consitencia de datos	Desarrollo dificil	Contenedores	Alto consumo de memoria.
Automatización		Desarrollo sencillo	

- ¿Cómo se crea un microservicio o monolito en Node.js o PHP?

1. Se realiza la estructura base para trabajar microservicios.
2. Se establecen las rutas, modelos, controladores para procesar las peticiones HTTP.
3. Se realizan pruebas de las url's a usar como API.



- ¿Cómo se implementan servicios RESTful en Node.js o PHP?

Una API RESTful es un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE.

A grandes rasgos, el proceso para crear una API seria el siguiente:

1. Instalación de dependencias.
2. Configurar archivo js principal.
3. Configuración datos DB'S
4. Conectar con la DB.
5. Crear modelos.
6. Crear controladores.
7. Creación de rutas.
8. Probar url's en postman.

- ¿Cómo se implementa un enrutamiento de API en Node.js o PHP?

Desde el archivo routes.js nosotros podremos hacer uso de "Router" para crear manejadores de rutas.
Una instancia Router es un sistema de middleware y direccionamiento completo.

```
module.exports = app => {
  const pets = require("../controllers/pets.controller.js");

  var router = require("express").Router();

  //Creación/Ingreso de una nueva mascota.
  router.post("/", pets.create);

  //Listado de elementos existentes.
  router.get("/", pets.findAll);

  //Búsqueda de mascotas por ID.
  router.get("/:id", pets.findOne);

  //Actualización de mascotas por ID.
  router.put("/:id", pets.update);

  //Eliminación de una mascota por ID.
  router.delete("/:id", pets.delete);

  //Eliminación del listado existente de mascotas.
  router.delete("/", pets.deleteAll);

  app.use('/api/pets', router);
};
```

- ¿Cómo se manejan los errores en Node.js o PHP?

  En nodeJS, especificamente en express podemos hacer uso de:

- Middlewares.
- Manejador de errores

Pruebas y depuración:

- ¿Cómo se realizan pruebas unitarias en Node.js o PHP?

Existen distintas herramientas que podemos integrar a nodejs para realizar las pruebas, tales como mocha, jest, jasmine, etc.

Jasmine es un marco de trabajo basado en javascript y nos permite escribir nuestros propios scripts de test, todo esto sin afectar el flujo de trabajo.

En resumen, la idea de las pruebas unitarias es probar y verificar de manera aislada el funcionamiento de partes específicas de una aplicación, de tal manera que al ejecutar todas las pruebas unitarias se puede asegurar que todas las partes del código están funcionando de manera esperada.

Por lo general, hay que escribir mucho código para validar nuestras funcionalidades y así poder liberar y asegurar el código creado.

Por convención se crea una carpeta sobre el proyecto _test_ para incluir los scripts a probar, para correr las pruebas ya dependerá de la herramienta utilizado pero comunmente se corren desde terminal para que nos lance el resultado de la prueba.

npm run test

- ¿Cómo se realizan pruebas de integración en Node.js o PHP?

Al igual que las pruebas unitarias podemos seguir trabajando bajo un marco javascript, este tipo de prueba valida que diversos módulos utilizados en la aplicación funcionen de manera correcta.

Comunmente estas pruebas se realizan después de las pruebas unitarias además de ser más costosas de crearlas.

Al igual de las pruebas unitarias dependerá de la herramienta que se esté ocupando para el proceso de testing pero la forma de probarlo es de igual forma ejecutar scripts por terminal para que nos lance respuesta correcta ó incorrecta.

npm run test

- ¿Cómo se depura una aplicación en Node.js o PHP?

Existen diferentes formas para estar realizando debbugin y también podemos ayudarnos de plugins dependiendo el editor que estemos ocupando, pero en general podriamos utilizar:

- Debugger: Este servicio viene incluido y se corre el script desde terminal para obtener una respuesta.

  Ejemplo: node debug script.js

- Puntos de ruptura
- Plugins en editores.

  Ejemplo: VSCode ESLint

- Linters de código:

  Ejemplo: ESLint.

  https://eslint.org/

Bases de datos:

- ¿Qué bases de datos se pueden usar con Node.js o PHP?

  De cualquier tipo relacional y no relacional.
  Relacionales más comunes son mysql, postgresql.
  No relacionales: mongo DB

- ¿Cómo se conecta una base de datos en Node.js o PHP?

  1) Inicialmente hay que instalar la DB que ocupemos (mysql, mongoose, postgresql, etc)
  2) Hay que configurar las variables de nuestra DB

```
module.exports = {
  HOST: "localhost",
  USER: "root",
  PASSWORD: "Interview123",
  DB: "crud"
};
```

3) Conectar nodejs con la base de datos.

```
const mysql = require("mysql2");
const dbConfig = require("../config/db.config.js");

var connection = mysql.createPool({
  host: dbConfig.HOST,
  user: dbConfig.USER,
  password: dbConfig.PASSWORD,
  database: dbConfig.DB
});

module.exports = connection;
```

- ¿Cómo se realiza una consulta a la base de datos en Node.js o PHP?

Dentro de nodejs y en este caso con mysql podremos establecer diversos query's de consulta a la DB, estos archivos comunemente van dentro de la carpeta model.

Ejemplo:

```
sql.query(`SELECT * FROM pets WHERE id = ${id}`, (err, res) => {
  if (err) {
    console.log("Error: ", err);
    result(err, null);
    return;
  }

  if (res.length) {
    console.log("Mascota encontrada: ", res[0]);
    result(null, res[0]);
    return;
  }
  result({ kind: "Registro no encontrado" }, null);
});
```

Herramientas y tecnologías:

- ¿Qué herramientas de construcción de aplicaciones se pueden usar con Node.js o PHP?
    1. Webpack (construcción para simplificar front end)
    2. Brunch (SPA's)
    3. ParcelJS
    4. Gulp

Entre otros =)

- ¿Qué herramientas de gestión de dependencias se pueden usar con Node.js o PHP?

Para nodeJs existen:
Npm

1. Bower
2. Require JS

Para PHP:

1. Composer

- ¿Qué marcos de trabajo se pueden utilizar en Node.js o PHP?

En nodeJs el marco de trabajo es basado en javascript, tales como:

1. EspressJS (MVC)
2. Socket IO (Aplicaciones en tiempo real)
3. NestJS (Api rest)

- ¿Cómo se utiliza Docker para desplegar aplicaciones en Node.js o PHP?

Como contenedor, podremos crear la imagen para que nuestra aplicación viva dentro del mismo y el deply resulte más sencillo y eficaz.

Los contenedores permiten empaquetar las aplicaciones y ejecutarlas en entornos aislados.

EXAMEN PRACTICO

Crear un microservicio o monolito desde cero:

- Crear un microservicio o monolito utilizando Node.js o PHP y una base de datos de su elección. Se debe crear una API RESTful que permita realizar operaciones CRUD en la base de datos.

CRUD MASCOTAS.

El proyecto está basado en nodejs y con las siguientes dependencias:

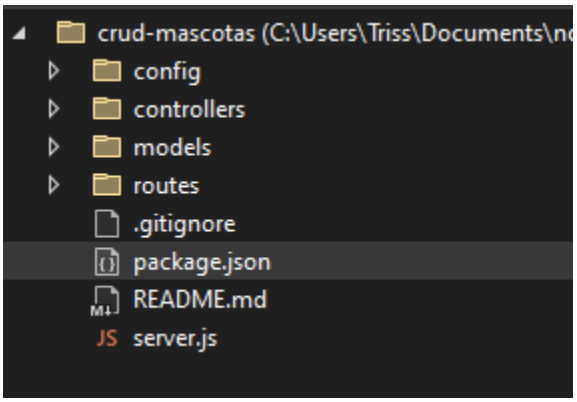
1. Cors
2. Express
3. Mysql2
4. Nodemon

Para correr el proyecto habrá que tener instalado nodejs y con la instrucción:

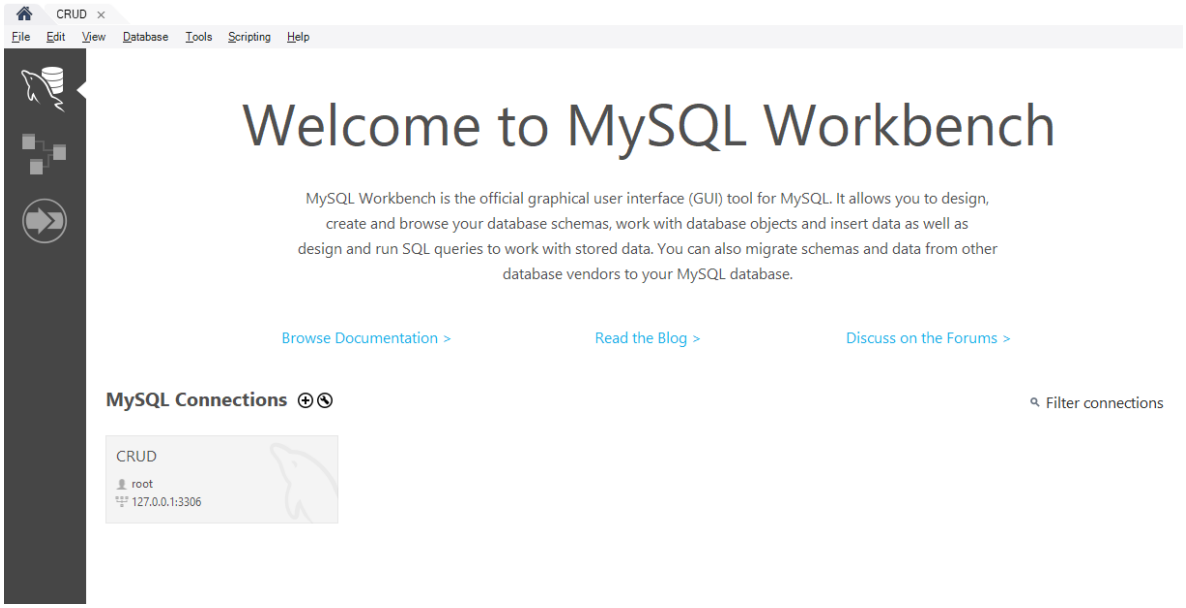
- npm i

se instalarán las dependencias correspondientes.

La estructura del proyecto es la siguiente:



El gestor DB que utiliza es Mysql

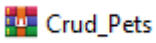


Por lo tanto anexamos el script que se puede importar para validar la funcionalidad de la API.

El query que se ocupó para la creación inicial de la tabla es:

```
CREATE TABLE IF NOT EXISTS `mascotas` (
 id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 name varchar(255) NOT NULL,
 description varchar(255),
 breed varchar(255),
 age int(5),
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Y el archivo de la exportación de la DB con estructura y datos es:



La configuración de la base de datos quedó de la siguiente forma y se encuentra en el archivo config/db.config.js

HOST: "localhost",

USER: "root",

PASSWORD: "Interview123",

DB: "crud"

Por lo tanto, los **endpoints** que se crearon son los siguientes:

1. **/*CREACION DE NUEVAS MASCOTAS*/**

URL: http://localhost:8080/api/pets/

Url de Ejemplo: http://localhost:8080/api/pets

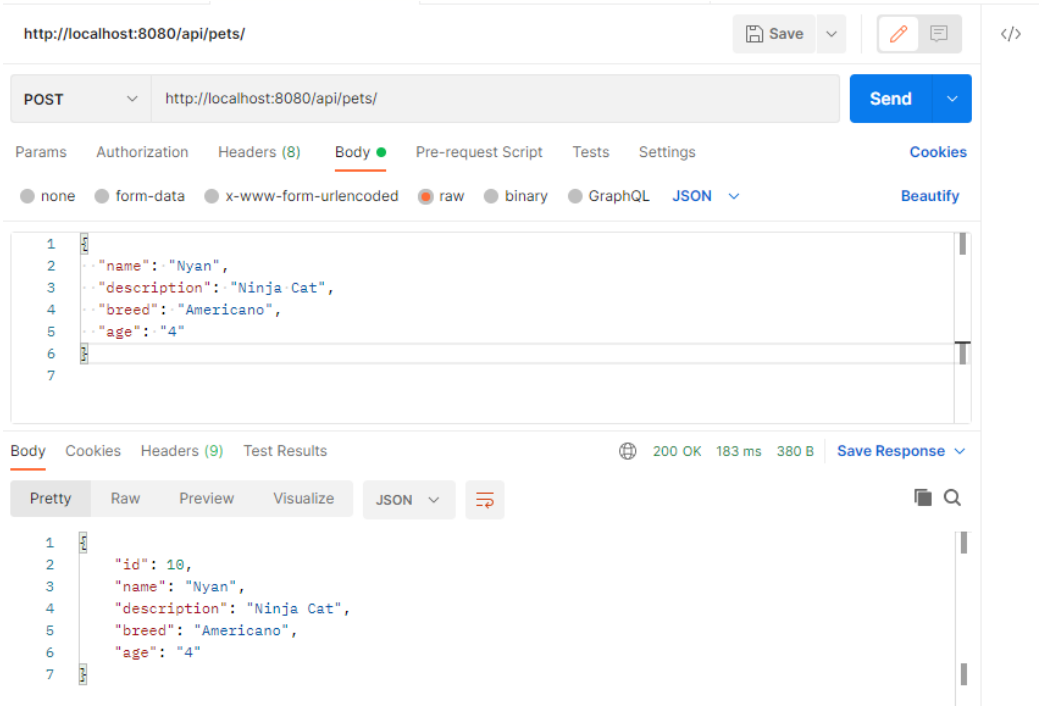
Método: **POST**

Descripción: Ingresará los datos de una nueva mascota.

Datos de entrada:

```
{
 "name": "Nyan",
 "description": "Ninja Cat",
 "breed": "Americano",
 "age": "4"
}
```

Estatus de prueba: **EXITOSO**.



2. **/*LISTADO DE MASCOTAS*/**

URL: http://localhost:8080/api/pets/

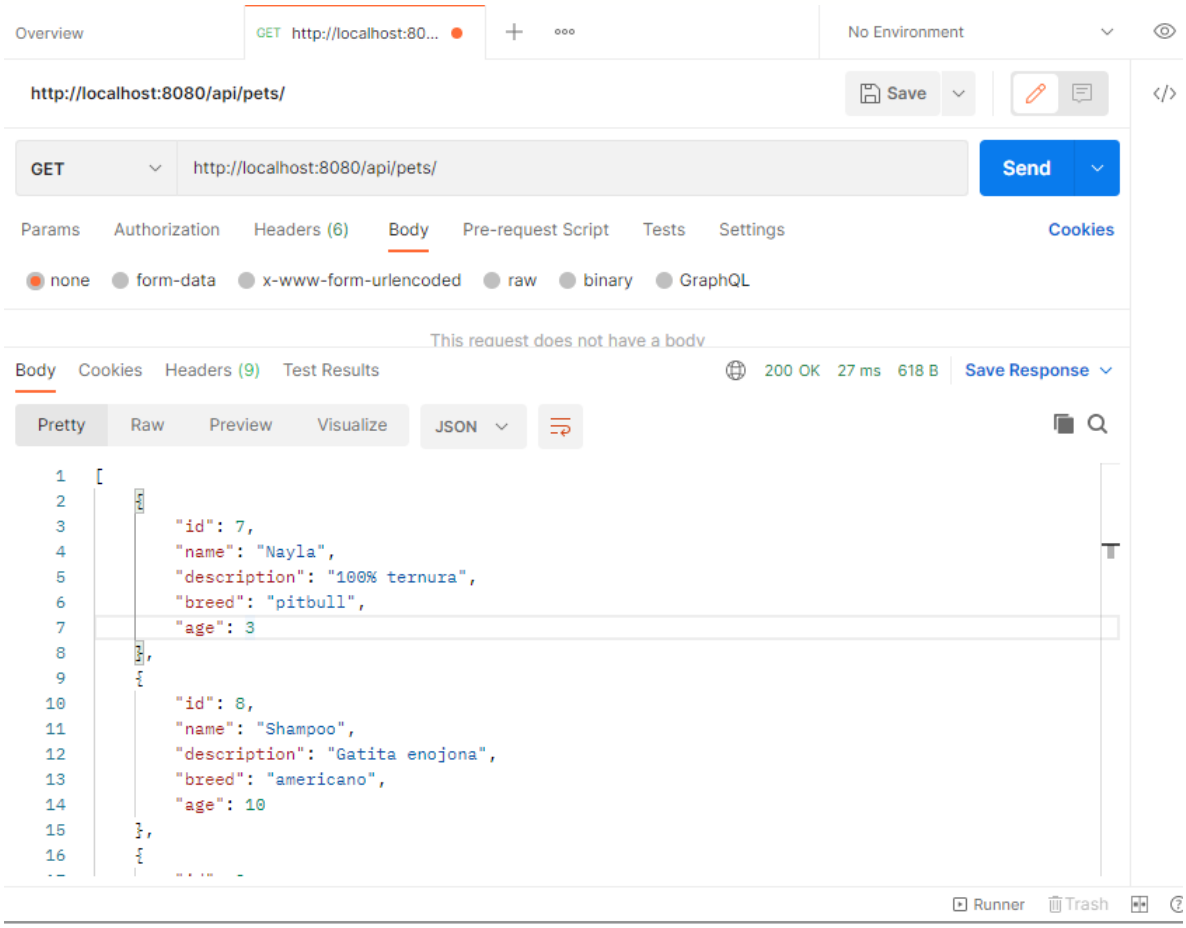
Url de Ejemplo: http://localhost:8080/api/pets

Método: **GET**

Descripción: Enlistar los datos existentes en la tabla mascotas.

Datos de entrada: Ninguno.

Estatus de prueba: **EXITOSO**.



3. /*BUSQUEDA DE UNA MASCOTA EN ESPECIFICO*/

URL:http://localhost:8080/api/pets/:id

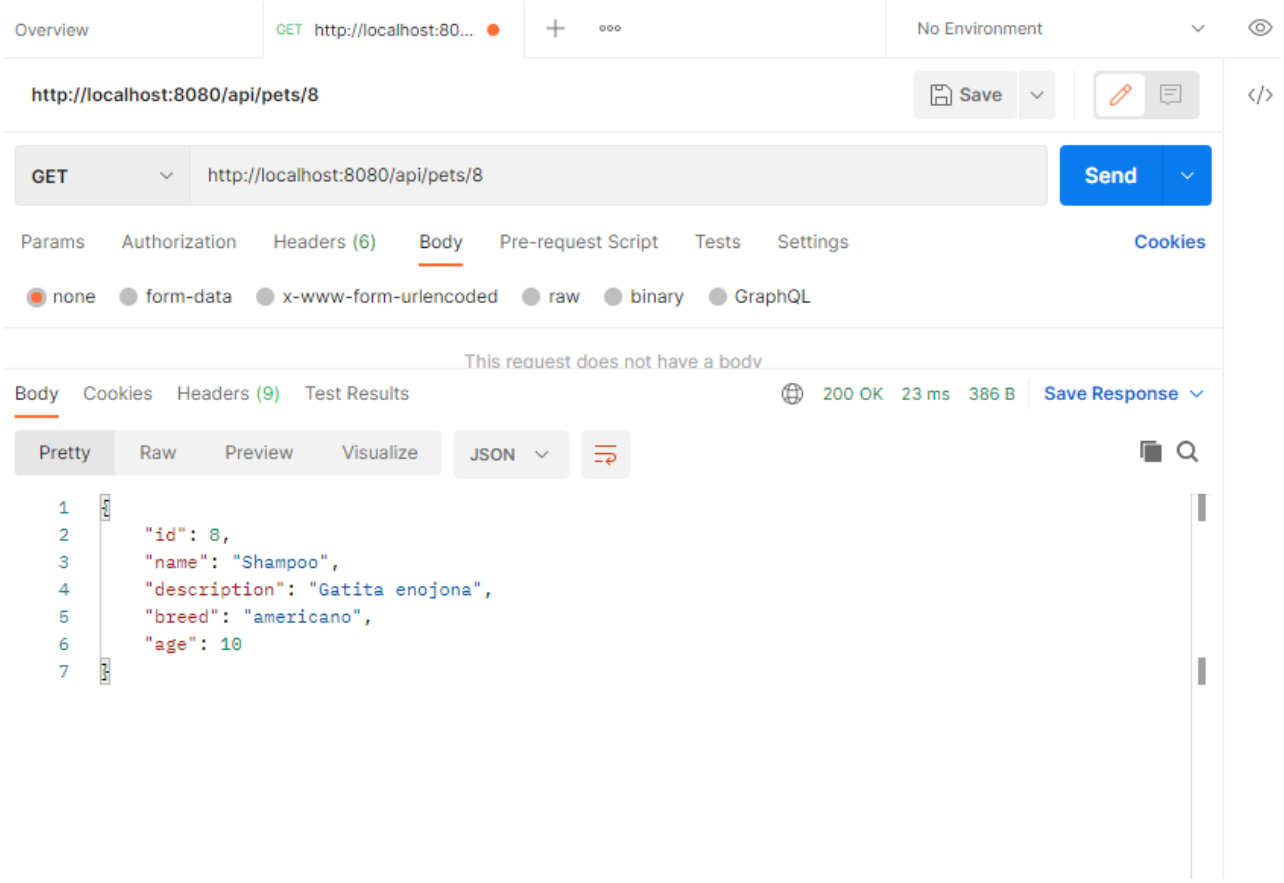
Url de Ejemplo: http://localhost:8080/api/pets/8

Método: **GET**

Descripción: Búsqueda de mascotas por ID.

Datos de entrada: Ninguno.

Estatus de prueba: **EXITOSO**.



4. /*ACTUALIZACION DE INFORMACION DE UNA MASCOTA */

URL: http://localhost:8080/api/pets/:id

Url de Ejemplo: http://localhost:8080/api/pets/9

Método: **PUT**

Descripción: Actualizacíón de mascotas por ID.

Estatus de prueba: **EXITOSO**.

Datos de entrada:

{

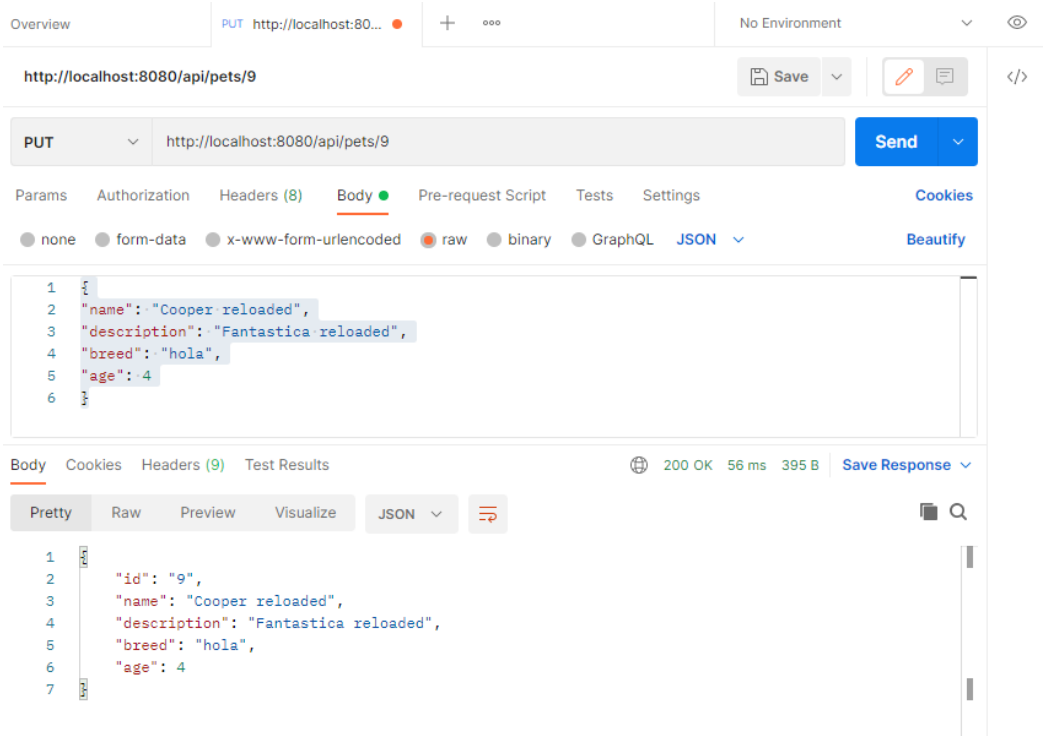
"name": "Cooper reloaded",

"description": "Fantastica reloaded",

"breed": "hola",

"age": 4

}



5. /*BORRADO DE UNA MASCOTA EN ESPECIFICO*/

URL: http://localhost:8080/api/pets/:id

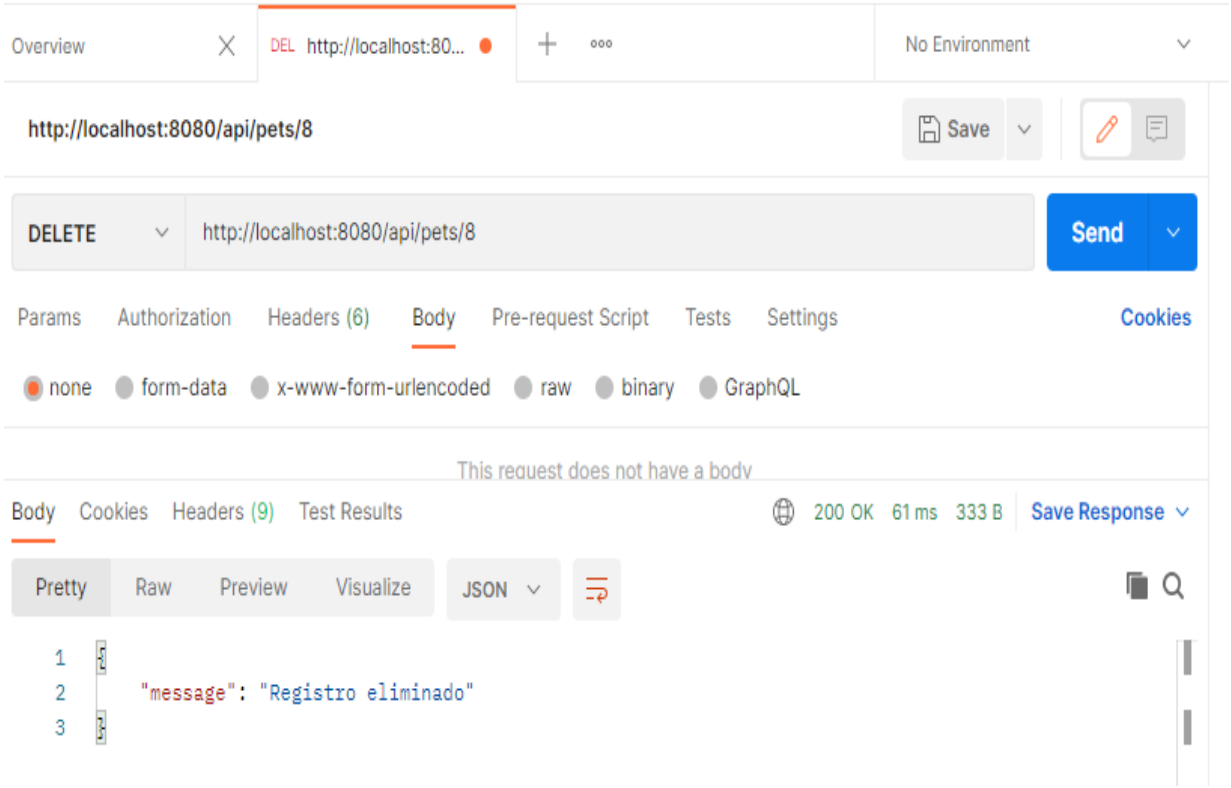
Método: **DELETE**

Url de Ejemplo: http://localhost:8080/api/pets/8

Descripción: Eliminación de una mascota por ID.

Datos de entrada: Ninguno.

Estatus de prueba: **EXITOSO**.



6. /*BORRADO DE LA LISTA COMPLETA DE MASCOTAS */

URL: http://localhost:8080/api/pets

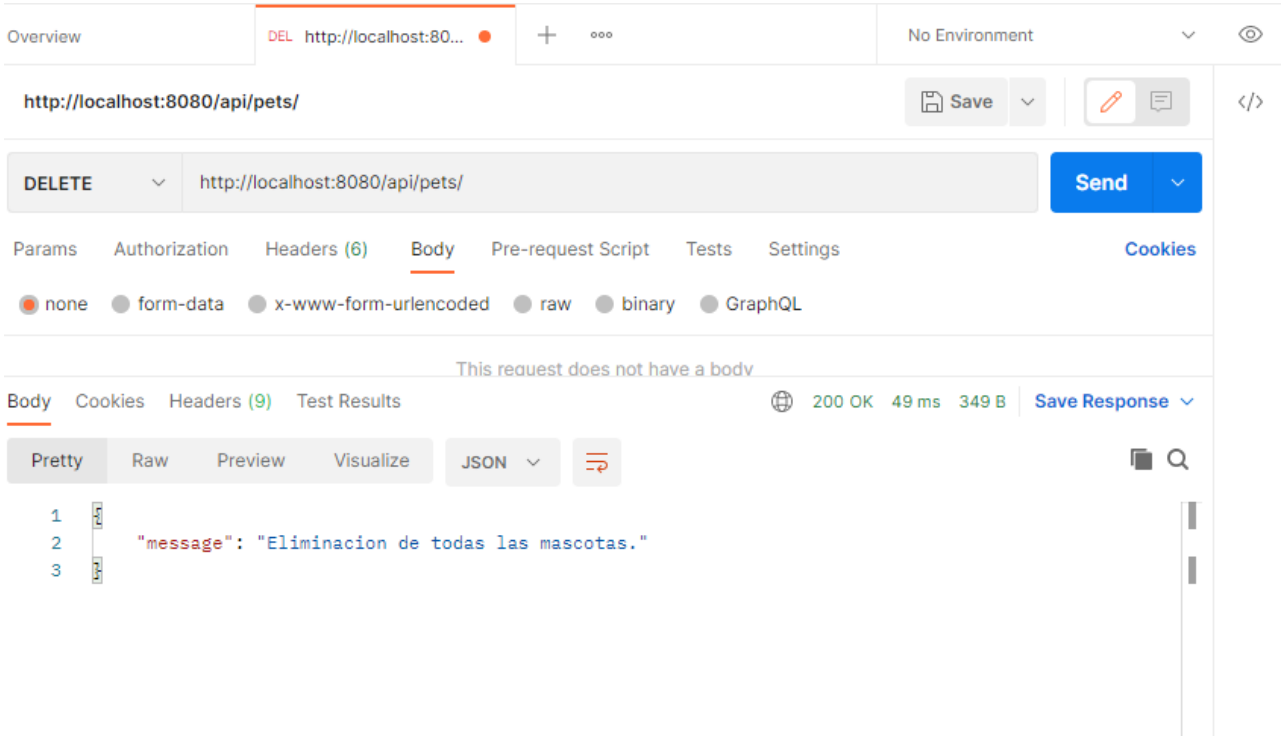
Url de Ejemplo: http://localhost:8080/api/pets/

Método: **DELETE**

Descripción: Eliminación del listado de mascotas en DB.

Datos de entrada: Ninguno.

Estatus de prueba: **EXITOSO**.



¡GRACIAS POR SU ATENCIÓN!