# **Documento de Pruebas del Software 'Pet Services'**

## **Introducción**

Este documento describe las pruebas realizadas en el software 'Pet Services' para validar su correcto funcionamiento. Se han utilizado herramientas como Postman, la consola del navegador (DevTools) y phpMyAdmin para probar la API, la integración con la base de datos y la correcta visualización de los datos en la interfaz web.

## **2. Objetivo**

El objetivo de estas pruebas es asegurar que el software 'Pet Services' cumpla con los requerimientos establecidos, permitiendo la gestión de clientes, servicios y citas, con una correcta comunicación entre el frontend, backend y la base de datos.

## **3. Herramientas utilizadas**

* **Postman**: Para probar las solicitudes HTTP hacia la API.
* **Consola del navegador (DevTools de Edge)**: Para inspeccionar solicitudes de red y errores en el frontend.
* **phpMyAdmin (XAMPP)**: Para verificar la manipulación de datos en la base de datos MySQL.

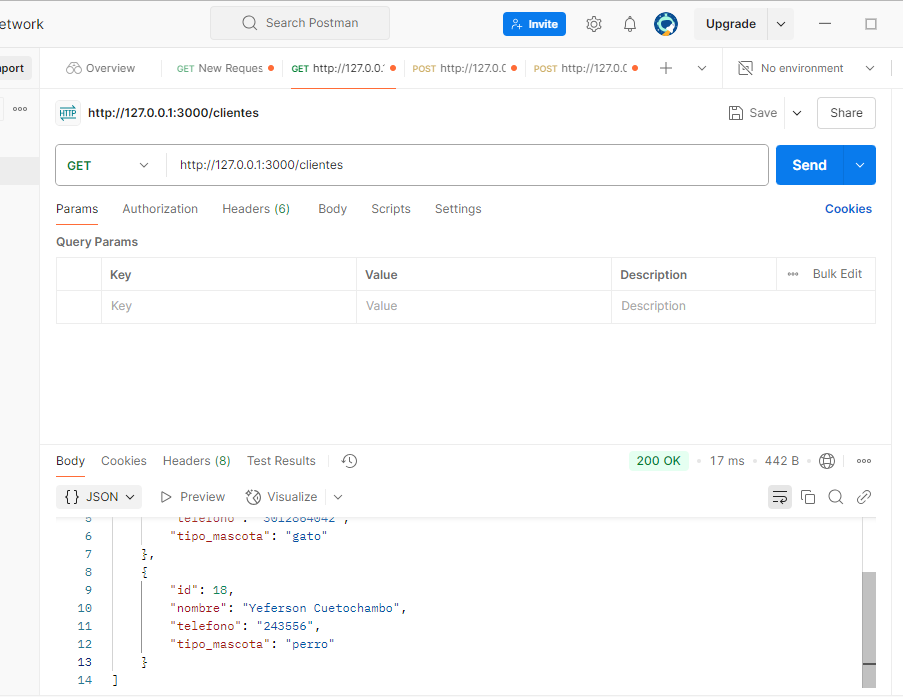
## **4. Pruebas realizadas**

### 4.1 Pruebas con Postman

**Objetivo**: Verificar que la API responde correctamente a las solicitudes.

#### 4.1.1 Prueba de obtención de clientes

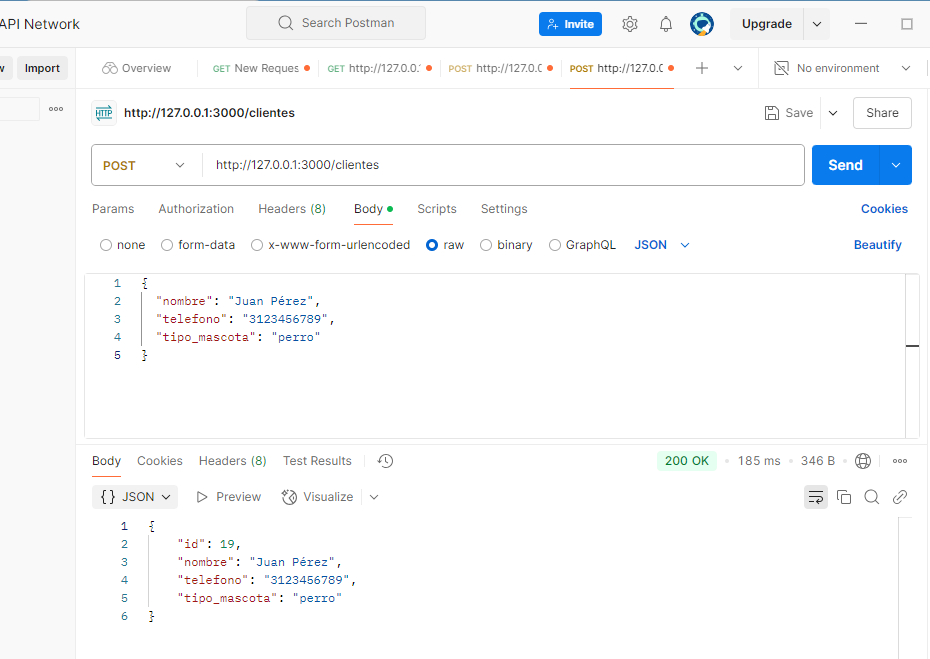
* **Método**: GET
* **URL**: http://localhost:3000/clientes
* **Resultado esperado**: Lista de clientes almacenados en la base de datos.
* **Resultado obtenido**: Se obtuvo la lista de clientes, incluyendo:



* **Estado**: ✅ Aprobado

#### 4.1.2 Prueba de registro de un cliente

* **Método**: POST
* **URL**: http://localhost:3000/clientes
* **Cuerpo de la solicitud**:



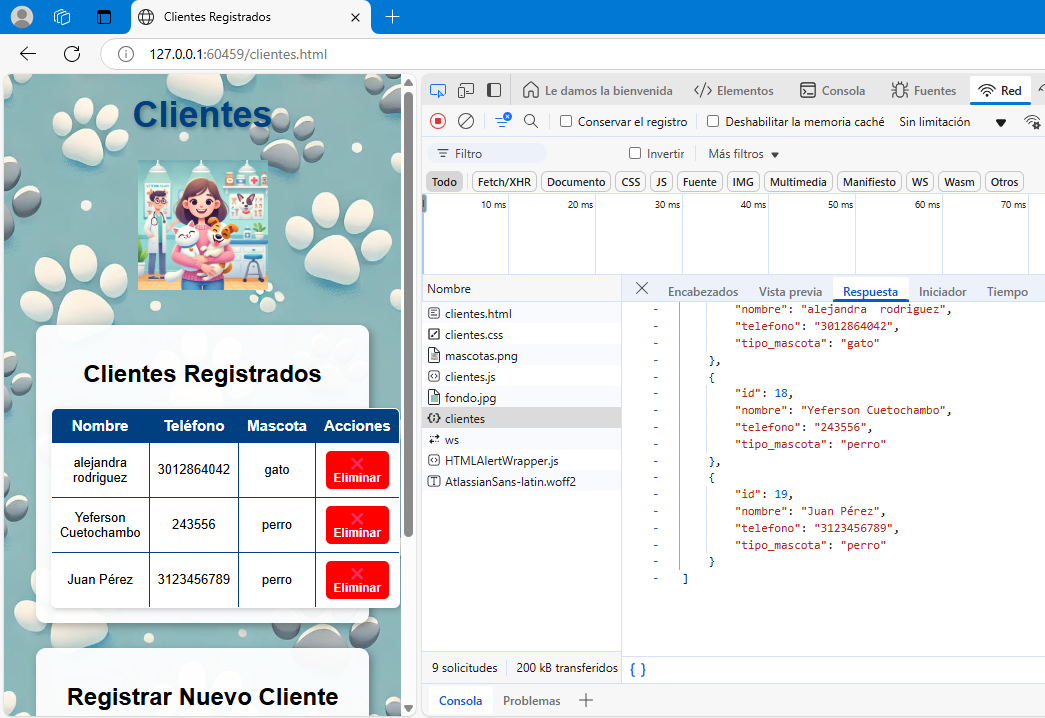
* **Estado**: ✅ Aprobado

### 4.2 Pruebas con la consola del navegador

**Objetivo**: Verificar que las peticiones del frontend al backend se ejecutan correctamente.

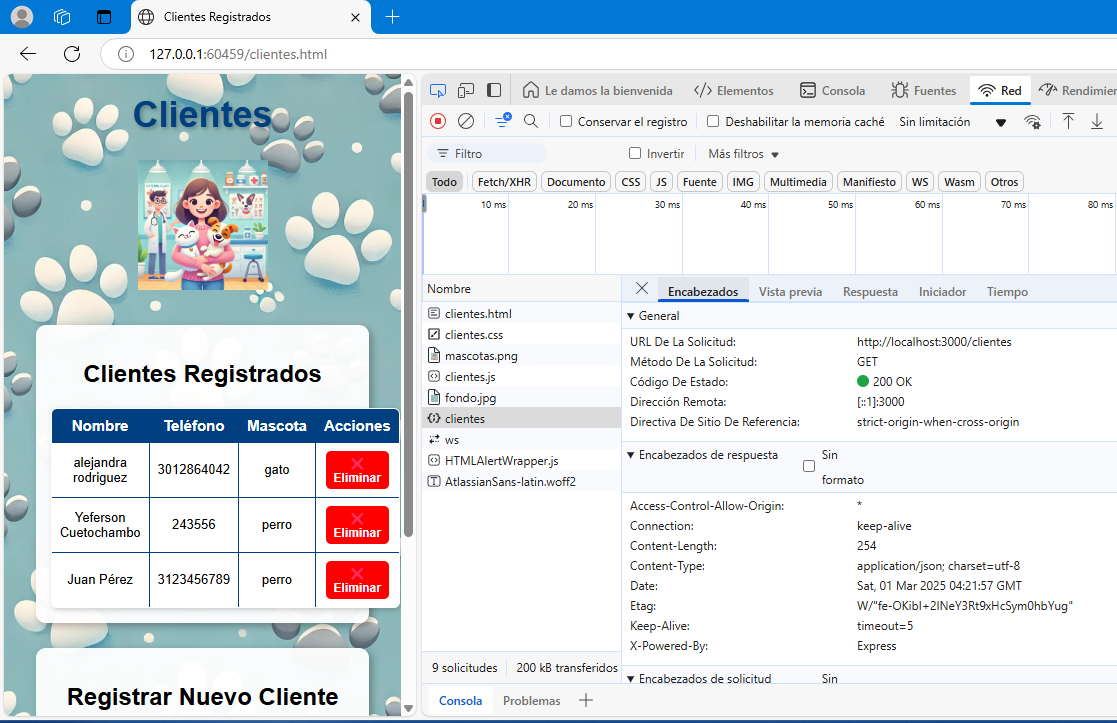
#### 4.2.1 Verificación de solicitudes en la pestaña "Red"

* **Acción realizada**: Se accedió a la pestaña "Red" en DevTools y se revisó la solicitud GET a http://localhost:3000/clientes.
* **Resultado esperado**: Código de estado 200 OK con los datos de los clientes.
* **Resultado obtenido**: Se confirmó la respuesta con los datos obtenidos correctamente.
* **Estado**: ✅ Aprobado



#### 4.2.2 Verificación de errores en la consola

* **Acción realizada**: Se revisó la consola en busca de errores.
* **Resultado obtenido**: No se encontraron errores críticos, solo advertencias menores.
* **Estado**: ✅ Aprobado



### 4.3 Pruebas con phpMyAdmin

**Objetivo**: Confirmar que los datos se almacenan correctamente en la base de datos.

#### 4.3.1 Consulta de clientes

* **Acción realizada**: Se ejecutó la consulta SQL:

SELECT \* FROM clientes;

* **Resultado esperado**: Lista de clientes almacenados.
* **Resultado obtenido**: Se confirmaron los clientes registrados:
* | id | nombre | telefono | tipo\_mascota |
* |----|---------------|-----------|-------------|
* | 18 | Yeferson C. | 243556 | perro |

| 19 | Juan Pérez | 3123456789 | perro |

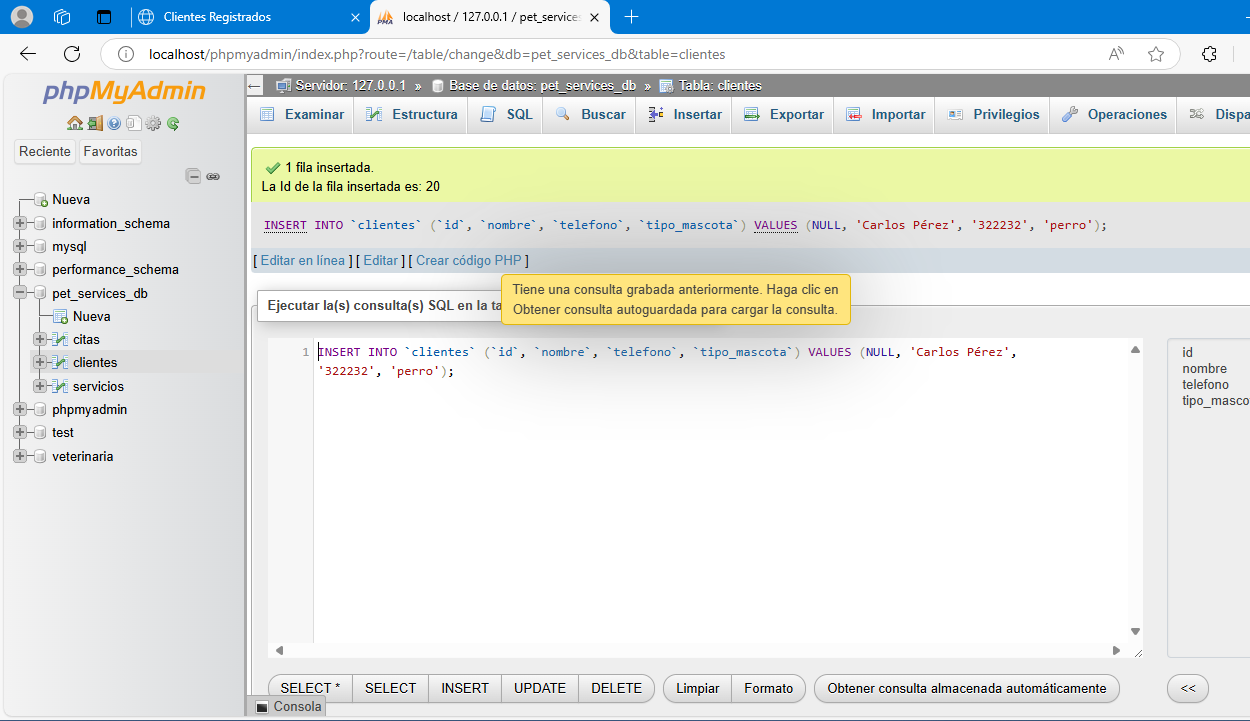
* **Estado**: ✅ Aprobado

#### 4.3.2 Inserción manual de un cliente

* **Acción realizada**: Se insertó un cliente manualmente con:
* INSERT INTO clientes (id, nombre, telefono, tipo\_mascota)

VALUES (NULL, 'Carlos Pérez', '3222323', 'perro');

* **Resultado esperado**: Cliente insertado con ID autoincrementado.
* **Resultado obtenido**: Se confirmó la inserción y su aparición en la consulta SELECT \* FROM clientes;.
* **Estado**: ✅ Aprobado



## **5. Conclusiones y Recomendaciones**

|  |  |  |
| --- | --- | --- |
| **Prueba** | **Estado** | **Observaciones** |
| Obtener clientes (Postman) | ✅ Aprobado | Los datos se obtienen correctamente. |
| Obtener clientes (Consola) | ✅ Aprobado | API responde correctamente. |
| Revisar errores en consola | ✅ Aprobado | No hay errores críticos. |
| Consultar clientes (DB) | ✅ Aprobado | Datos correctamente almacenados. |
| Insertar cliente (DB) | ✅ Aprobado | Inserción manual exitosa. |

### **Recomendaciones**

1. **Corregir la validación de campos en el backend**, ya que el POST a /clientes da error a pesar de enviar los datos requeridos.
2. **Agregar logs en la API** para detectar errores en las solicitudes.
3. **Habilitar CORS** en el backend si se planea consumir la API desde otro dominio.
4. **Optimizar la interfaz** para mostrar mensajes de error detallados en caso de fallos al registrar clientes.