



Universidad de **Nariño**

TALLER UNIDAD 2 BACKEND.

ELABORADO POR:

JULIETH STEPHANIA HIDALGO NARVAEZ

UNIVERSIDAD DE NARIÑO - SEDE IPIALES

FACULTAD DE INGENIERIA DE SISTEMAS

X SEMESTRE

**DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA EL
DESARROLLO DE SOFTWARE.**

IPIALES – NARIÑO 2024



Universidad de **Nariño**

TALLER UNIDAD 2 BACKEND.

ELABORADO POR:

JULIETH STEPHANIA HIDALGO NARVAEZ

PRESENTADO A:

MG. VICENTE AUX REVELO

**UNIVERSIDAD DE NARIÑO - SEDE IPIALES
FACULTAD DE INGENIERIA DE SISTEMAS**

X SEMESTRE

**DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA EL
DESARROLLO DE SOFTWARE.**

IPIALES – NARIÑO 2024

TALLER UNIDAD 1 BACKEND.

1. Base de datos que lleva el registro de la empresa de adopción de mascotas Furry Friends.

1.1. La base de datos incluye las siguientes tablas:

- **Tabla Pets (Mascotas):**

- **Id:** Identificador único de la mascota.
- **Name:** Nombre de la mascota.
- **Age:** Edad de la mascota.
- **Species:** Especie de la mascota (ej. perro, gato).
- **Breed:** Raza de la mascota.
- **Description:** Descripción de la mascota.
- **Status:** Estado de la mascota (disponible, adoptada).
- **Gender:** Género de la mascota.
- **Size:** Tamaño de la mascota.
- **Weight:** Peso de la mascota.
- **Entry_date:** Fecha de ingreso de la mascota.
- **Vaccinated:** si la mascota está vacunada o no.
- **Neutered:** Si está esterilizada o no.
- **Medical_conditions:** Condiciones médicas de la mascota.
- **Available_for_adoption:** si está disponible para adopción.

- **Tabla Adopters (Adoptantes):**

- **Id:** Identificador único del adoptante.
- **First_name:** Primer nombre del adoptante.
- **Last_name:** Apellido del adoptante.
- **email:** Correo electrónico del adoptante.
- **Phone:** Número de teléfono del adoptante.
- **Address:** Dirección del adoptante.
- **Birth_date:** Fecha de nacimiento del adoptante.
- **Occupation:** Ocupación del adoptante.
- **Housing_type:** Tipo de vivienda del adoptante.
- **Has_other_pets:** si tiene otras mascotas.
- **Type_of_other_pets:** Tipo de otras mascotas, si las tiene.
- **Adoption_reason:** Razón para la adopción.

- **Tabla Adoption_requests (solicitudes de adopción):**
 - **Id:** Identificador único de la solicitud.
 - **Pet_id:** Identificador de la mascota solicitada
 - **Adopter_id:** Identificador del adoptante
 - **Request_date:** Fecha en que se realizó la solicitud.
 - **Status:** Estado de la solicitud (pendiente, aprobada, rechazada).
 - **approval_date:** Fecha de aprobación de la solicitud.
 - **Employee_id:** Identificador del empleado que gestiona la solicitud (clave foránea de `employees`).
- **Tabla Employees (Empleados):**
 - **Id:** Identificador único del empleado.
 - **Full_name:** Nombre completo del empleado.
 - **Position:** Posición del empleado (ej. gerente, coordinador).
 - **Phone:** Número de teléfono del empleado.
 - **Email:** Correo electrónico del empleado.
 - **Hire_date:** Fecha de contratación del empleado.
- **Relaciones entre las tablas:**
 - **Relación entre Pets y Adoption_requests:** Una mascota puede tener múltiples solicitudes de adopción, pero cada solicitud corresponde a una sola mascota. La clave foránea es **Pet_id** en la tabla **Adoption_requests**.
 - **Relación entre Adopters y Adoption_requests:** Un adoptante puede realizar múltiples solicitudes de adopción, pero cada solicitud corresponde a un único adoptante. La clave foránea es **Adopter_id** en la tabla **Adoption_requests**.
 - **Relación entre Employees y Adoption_requests:** Un empleado puede gestionar múltiples solicitudes de adopción, pero cada solicitud es gestionada por un solo empleado. La clave foránea es **Employee_id** en la tabla **Adoption_requests**.

DIAGRAMA



Ilustración 1 Diagrama Mascotas

2. En el contexto de nuestra empresa **Furry Friends**, se propone el desarrollo de una aplicación backend utilizando **NodeJS** y **ExpressJS**, esta aplicación será responsable de gestionar el registro y la administración de mascotas, así como las solicitudes de adopción. La implementación permitirá que los usuarios interactúen de manera eficiente con la base de datos previamente establecida.

2.1. Códigos del proyecto:

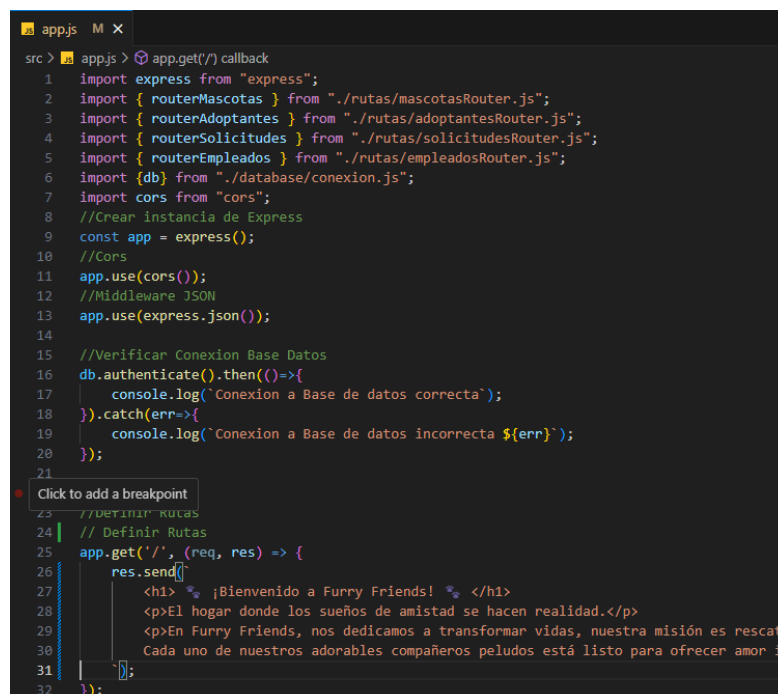
- 2.1.1. **Database – Conexión.Js:** El código establece una conexión a una base de datos MySQL llamada "mascotas" usando Sequelize y luego crea una conexión donde se especifica el nombre de usuario "mascotas2024" y que la base de datos se encuentra en el mismo servidor (localhost), finalmente exporta esta conexión para que otras partes de la aplicación puedan utilizarla, lo que facilita realizar operaciones como agregar, leer, actualizar o eliminar información sobre las mascotas.



```
conexion.js
src > database > conexion.js > ...
1 import Sequelize from "sequelize";
2
3 const db = new Sequelize("mascotas","mascotas","mascotas2024",{
4     dialect: "mysql",
5     host: "localhost"
6 });
7
8 export {db}
```

Ilustración 2: Conexión.js

- 2.1.2. **App.js:** Este fragmento de código establece la estructura básica de la aplicación, asegurando que la conexión a la base de datos esté funcionando correctamente y definiendo la respuesta que se mostrará al acceder a la ruta principal, brindando un mensaje cálido y acogedor a los usuarios.



```
app.js
src > app.js > app.get('/') callback
1 import express from "express";
2 import { routerMascotas } from "../rutas/mascotasRouter.js";
3 import { routerAdoptantes } from "../rutas/adoptantesRouter.js";
4 import { routerSolicitudes } from "../rutas/solicitudesRouter.js";
5 import { routerEmpleados } from "../rutas/empleadosRouter.js";
6 import { db } from "../database/conexion.js";
7 import cors from "cors";
8 //Crear instancia de Express
9 const app = express();
10 //Cors
11 app.use(cors());
12 //Middleware JSON
13 app.use(express.json());
14
15 //Verificar Conexion Base Datos
16 db.authenticate().then(()=>{
17     console.log('Conexion a Base de datos correcta');
18 }).catch(err=>{
19     console.log('Conexion a Base de datos incorrecta ${err}');
20 });
21
22 //Definir Rutas
23 // Definir Rutas
24 app.get('/', (req, res) => {
25     res.send(
26         <h1> 🐾 ¡Bienvenido a Furry Friends! 🐾 </h1>
27         <p>El hogar donde los sueños de amistad se hacen realidad.</p>
28         <p>En Furry Friends, nos dedicamos a transformar vidas, nuestra misión es rescatar
29         Cada uno de nuestros adorables compañeros peludos está listo para ofrecer amor in
30     );
31 });
32
```

2.2. Controladores: Para especificar los controladores los cuales son los encargados de manejar la lógica del proyecto y las interacciones con el modelo tomaremos un ejemplo de unos de ellos.

2.2.1. adoptantesController.js: Este código es un controlador escrito para un entorno Node.js, que gestiona las operaciones de CRUD (Crear, Leer, Actualizar y Eliminar) para un recurso llamado Adoptante, el cual está orientado hacia una aplicación de adopción de mascotas, donde se manejan los datos de las personas que desean adoptar.

El controlador exporta cinco funciones principales:

- **Crear:** para registrar un nuevo adoptante.
- **Buscar:** para obtener todos los registros de adoptantes.
- **Buscar Id:** para buscar un adoptante por su ID.
- **Actualizar:** para actualizar los datos de un adoptante.
- **Eliminar:** para eliminar un adoptante por su ID.

➤ **Crear Adoptantes:** El propósito es buscar un adoptante específico mediante su ID. Si no se proporciona el ID, se devuelve un estado 400. Se utiliza el método `findByPk ()` de Sequelize para buscar el registro con la clave primaria ID. La respuesta devuelve el registro encontrado o un error si no se encuentra o hay algún problema en la consulta.

```
app.js M adoptantesController.js X
src > controladores > adoptantesController.js > ...
1 import { adoptantes } from "../modelos/adoptantesModelo.js";
2
3 // Crear un Adoptante
4 const crear = (req, res) => {
5
6   // Validar campos requeridos
7   if (!req.body.first_name || !req.body.last_name || !req.body.email) {
8     return res.status(400).send({
9       mensaje: "Los campos 'first_name', 'last_name' y 'email' no pueden estar vacíos."
10    });
11  }
12
13   // Construir el dataset para el adoptante
14   const dataset = {
15     first_name: req.body.first_name,
16     last_name: req.body.last_name,
17     email: req.body.email,
18     phone: req.body.phone || null, // Campo opcional
19     address: req.body.address || null, // Campo opcional
20     birth_date: req.body.birth_date || null, // Campo opcional
21     occupation: req.body.occupation || null, // Campo opcional
22     housing_type: req.body.housing_type || null, // Campo opcional
23     has_other_pets: req.body.has_other_pets || null, // Campo opcional
24     type_of_other_pets: req.body.type_of_other_pets || null, // Campo opcional
25     adoption_reason: req.body.adoption_reason || null // Campo opcional
26   };
27
28   // Usar Sequelize para crear el recurso en la base de datos
29   adoptantes.create(dataset)
30     .then((resultado) => {
31       res.status(201).json({
32         mensaje: "Registro de Adoptante creado con éxito",
33         data: resultado
34       });
35     })
36     .catch((err) => {
37       res.status(500).json({
38         mensaje: `Registro de Adoptante no creado: ${err.message}`
39       });
40     });
41 };
```

Ilustración 3 Crear Adoptantes

- **Buscar Adoptante:** Este código define una función llamada `buscar` que busca todos los registros de adoptantes en la base de datos usando el método `findAll()`. Si la consulta es exitosa, devuelve los resultados en formato JSON con un estado 200. Si ocurre un error, devuelve un mensaje de error con un estado 500, indicando que no se encontraron registros y mostrando el error específico.

```
//Buscar Adoptante
const buscar = (req, res) => {
  adoptantes.findAll().then((resultado) => {
    res.status(200).json(resultado);
  }).catch((err) => {
    res.status(500).json({
      mensaje: `No se encontraron registros ::: ${err}`
    });
  });
}
```

Ilustración 4 Buscar Adoptantes

- **Buscar por ID:** Este código define una función llamada `buscarId` que busca un adoptante específico por su ID.

```
//buscar por ID
const buscarId = (req, res) => {
  const id = req.params.id;
  if (id == null) {
    res.status(400).json({
      mensaje: "El id Adoptante no puede estar vacío"
    });
    return;
  }
  else {
    adoptantes.findById(id).then((resultado) => {
      res.status(200).json(resultado);
    }).catch((err) => {
      res.status(500).json({
        mensaje: `No se encontraron registros de Adoptante ::: ${err}`
      });
    });
  }
}
```

Ilustración 5: Buscar Adoptantes por id

- **Actualizar Adoptantes:** Este código define la función `actualizar` que permite actualizar los datos de un adoptante identificado por su ID. Primero, verifica si los campos `first_name`, `last_name` y `email` están vacíos; si es así, responde con un estado 400 indicando que no se encontraron datos para actualizar. Si se proporcionan datos, los campos opcionales como nombre, apellido, correo electrónico, teléfono, dirección, entre otros, se actualizan utilizando el método `update()` de Sequelize. Si la operación es exitosa, responde con un estado 200 y un mensaje de éxito. Si ocurre un error, responde con un estado 500 y un mensaje de error.

```
// Actualizar Adoptante
const actualizar = (req, res) => {
  const id = req.params.id;
  if (!req.body.first_name && !req.body.last_name && !req.body.email) {
    res.status(400).json({
      mensaje: "No se encontraron datos de Adoptante para actualizar"
    });
    return;
  } else {
    const first_name = req.body.first_name || null; // Campo opcional
    const last_name = req.body.last_name || null; // Campo opcional
    const email = req.body.email || null; // Campo opcional
    const phone = req.body.phone || null; // Campo opcional
    const address = req.body.address || null; // Campo opcional
    const birth_date = req.body.birth_date || null; // Campo opcional
    const occupation = req.body.occupation || null; // Campo opcional
    const housing_type = req.body.housing_type || null; // Campo opcional
    const has_other_pets = req.body.has_other_pets || null; // Campo opcional
    const type_of_other_pets = req.body.type_of_other_pets || null; // Campo opcional
    const adoption_reason = req.body.adoption_reason || null; // Campo opcional

    adoptantes.update({
      first_name,
      last_name,
      email,
      phone,
      address,
      birth_date,
      occupation,
      housing_type,
      has_other_pets,
      type_of_other_pets,
      adoption_reason
    }, { where: { id } })
      .then((resultado) => {
        res.status(200).json({
          tipo: 'success',
          mensaje: "Registro de Adoptante actualizado"
        });
      })
      .catch((err) => {
        res.status(500).json({
          tipo: 'error',
          mensaje: `Error al actualizar registro de Adoptante: ${err.message}`
        });
      });
  }
}
```

Ilustración 6: Actualizar Adoptantes

- **Eliminar Adoptante:** Este código define la función `eliminar`, que permite eliminar un registro de adoptante identificado por su ID. Primero, verifica si se ha proporcionado un ID; si no, responde con un estado 203 y un mensaje indicando que se debe ingresar un ID de adoptante. Si el ID está presente, utiliza el método `destroy()` de Sequelize para eliminar el registro correspondiente. Si la eliminación es exitosa, responde con un estado 200 y un mensaje de éxito indicando que el registro fue eliminado. En caso de error, responde con un estado 500 y un mensaje de error.

```
//Eliminar Adoptante
const eliminar = (req, res) => {
  const id = req.params.id;
  if (id == null) {
    res.status(203).json({
      message: "Debe ingresar un ID de Adoptante!",
    });
    return;
  }

  adoptantes.destroy({ where: { id: id } })
    .then((result) => {
      res.status(200).json({
        tipo: 'success',
        mensaje: `Registro con id ${id} Eliminado Correctamente`
      });
    })
    .catch((err) => {
      res.status(500).json({
        tipo: 'error',
        mensaje: `Error al eliminar Registro de Adoptante ::: ${err}`
      });
    });
};
```

Ilustración 7: Eliminar Adoptantes

2.2.2. MODELOS: Muestran las definiciones de los modelos de datos y encontramos los siguientes modelos.

- **adoptantesModelo.js:** Estructura de datos para adoptantes.
- **empleadosModelo.js:** Estructura de datos para empleados.
- **mascotasModelo.js:** Estructura de datos para mascotas.
- **solicitudesModelo.js:** Estructura de datos para solicitudes.

Se tomará como ejemplo el de adoptantes para demostrar cómo se los realiza.

➤ **adoptantesModelo.js:** Este código define un modelo de datos llamado `adoptantes` utilizando Sequelize, que es un ORM para Node.js. Primero, se importa Sequelize y la conexión a la base de datos. El modelo incluye varios atributos que representan las características de un adoptante, como:

- **id:** un identificador único que se auto-incrementa y actúa como clave primaria.
- **first_name y last_name:** campos obligatorios que almacenan el nombre y apellido del adoptante.
- **email:** un campo único y obligatorio que guarda la dirección de correo electrónico del adoptante.
- **phone, address, birth_date, occupation, housing_type, type_of_other_pets y adoption_reason:** campos opcionales que pueden dejarse vacíos.
- **has_other_pets:** un campo booleano que indica si el adoptante tiene otras mascotas.

Además, se establece `timestamps: false`, lo que significa que no se añadirán automáticamente columnas para las fechas de creación y actualización. Al final, el modelo `adoptantes` se exporta para ser utilizado en otras partes de la aplicación.

```
import Sequelize from "sequelize";
import { db } from "../database/conexion.js";

const adoptantes = db.define("adopters", {
  id: {
    type: Sequelize.BIGINT,
    allowNull: false,
    autoIncrement: true,
    primaryKey: true
  },
  first_name: {
    type: Sequelize.TEXT,
    allowNull: false
  },
});
```

Ilustración 8: Modelo Adoptantes.

```

last_name: {
  type: Sequelize.TEXT,
  allowNull: false
},
email: {
  type: Sequelize.TEXT,
  allowNull: false,
  unique: true
},
phone: {
  type: Sequelize.TEXT,
  allowNull: true
},
address: {
  type: Sequelize.TEXT,
  allowNull: true
},
birth_date: {
  type: Sequelize.DATE,
  allowNull: true
},
occupation: {
  type: Sequelize.TEXT,
  allowNull: true
},
housing_type: {
  type: Sequelize.TEXT,
  allowNull: true
},
has_other_pets: {
  type: Sequelize.BOOLEAN, // True = 1: Es decir "Si", False = 0: Es decir "No"
  allowNull: true
},
type_of_other_pets: {
  type: Sequelize.TEXT,
  allowNull: true
},
adoption_reason: {
  type: Sequelize.TEXT,
  allowNull: true
}
}, {
  timestamps: false
});

export { adoptantes }

```

Ilustración 9: Modelo Adoptantes.

2.2.3. RUTAS: Se muestra la configuración de las rutas API y encontramos las siguientes rutas:

- **adoptantesRouter.js:** Rutas para gestionar adoptantes.
- **empleadosRouter.js:** Rutas para gestionar empleados.
- **mascotasRouter.js:** Rutas para gestionar mascotas.
- **solicitudesRouter.js:** Rutas para gestionar solicitudes.

Se tomará como ejemplo el de adoptantes para demostrar cómo se los realiza.

- **adoptantesRouter.js:** Este código configura las rutas para gestionar adoptantes en una aplicación de Express. Se importan las funciones del controlador y se crea un enrutador llamado `routerAdoptantes`. La ruta principal (`/`) proporciona un mensaje sobre las operaciones disponibles. Las rutas específicas permiten crear un adoptante, buscar todos los adoptantes, buscar por ID, actualizar datos y eliminar un adoptante. Al final, el enrutador se exporta para su uso en la aplicación.

```
src > rutas > adoptantesRouter.js > ...
1  import express from "express";
2  import {crear,buscar,buscarId,actualizar,eliminar} from "../controladores/adoptantesController.js";
3
4  const routerAdoptantes = express.Router();
5
6  routerAdoptantes.get('/', (req, res) => {
7    res.send('Gestion de Registros de adoptantes: (Creación, Búsqueda, Actualización y Eliminación)');
8  });
9
10 routerAdoptantes.post('/crear', (req, res) => {
11   crear(req,res);
12 }
13 });
14
15 routerAdoptantes.get('/buscar', (req, res) => {
16   buscar(req,res);
17 }
18 });
19
20 routerAdoptantes.get('/buscarId/:id', (req, res) => {
21   buscarId(req,res);
22 }
23 });
24
25 routerAdoptantes.put('/actualizar/:id', (req, res) => {
26   actualizar(req,res);
27 }
28 });
29
30 routerAdoptantes.delete('/eliminar/:id', (req, res) => {
31   eliminar(req,res);
32 }
33 });
34
35 export {routerAdoptantes}
```

Ilustración 10: Rutas de Adoptantes.

2.2.4. REQUESTS: En el código desarrollado, se implementan correctamente los verbos HTTP según las acciones que se deben realizar con los recursos de mascotas en la API. Para las consultas, se utiliza `GET`, permitiendo buscar mascotas de manera general o por ID sin modificar los datos en el servidor, en el caso de la creación de una nueva mascota, se emplea `POST`, para la actualización de un recurso existente, se usa `PUT`, reemplazando completamente la información de la mascota y finalmente, se implementa `DELETE` para eliminar un recurso, en este caso, una mascota específica. Esto garantiza que se siga un uso adecuado de los verbos HTTP, alineado con las convenciones de REST y permitiendo un manejo eficiente de los datos.

```
//Busqueda de adoptante
###
Send Request
GET http://127.0.0.1:4000/adoptantes/buscar HTTP/1.1
//Busqueda de adoptante por id
###
Send Request
GET http://127.0.0.1:4000/adoptantes/buscarId/2 HTTP/1.1
//Creacion de adoptante
###
Send Request
POST http://127.0.0.1:4000/adoptantes/crear HTTP/1.1
Content-type: application/json
{
  "first_name": "José",
  "last_name": "Martínez",
  "email": "jose.martinez@example.com",
  "phone": "555-123-4567",
  "address": "Boulevard de la Libertad 456",
  "birth_date": "1978-11-30",
  "occupation": "Arquitecto",
  "housing_type": "Casa",
  "has_other_pets": true,
  "type_of_other_pets": "Gato",
  "adoption_reason": "Deseo adoptar un compañero para mi gato"
}
//Actualizacion de adoptante
###
Send Request
PUT http://127.0.0.1:4000/adoptantes/actualizar/2 HTTP/1.1
Content-type: application/json
{
  "first_name": "María",
  "last_name": "Fernández",
  "email": "maria.fernandez@example.com",
  "phone": "098-765-4321",
  "address": "Avenida Siempre Viva 742",
  "birth_date": "1990-03-15",
  "occupation": "Docente",
  "housing_type": "Apartamento",
  "has_other_pets": false,
  "type_of_other_pets": null,
  "adoption_reason": "Quiero ofrecer un hogar a un animal necesitado"
}
//Eliminacion de adoptante
###
Send Request
DELETE http://127.0.0.1:4000/adoptantes/eliminar/2 HTTP/1.1
```

Ilustración 11 Request

3. La empresa Furry Friends realizará pruebas exhaustivas para validar las diferentes operaciones, asegurando que los resultados obtenidos reflejen el correcto funcionamiento de los procesos establecidos que cumplan con los objetivos definidos y estos resultados servirán como evidencia tanto del cumplimiento como de la efectividad de las soluciones implementadas.

3.1. Modulo adoptantes: Para muestra de las pruebas utilizaremos el modulo de adoptantes.

- **Crear Adoptantes:** Aquí creamos a los adoptantes.

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:4000/adoptantes/crear`. The response status is 201 Created, with a size of 401 Bytes and a time of 21 ms. The response body is a JSON object containing a success message and a newly created adopter's details.

```
POST http://127.0.0.1:4000/adoptantes/crear
Status: 201 Created Size: 401 Bytes Time: 21 ms
Response
1 {
2   "mensaje": "Registro de Adoptante creado con éxito",
3   "data": {
4     "id": 5,
5     "first_name": "María",
6     "last_name": "González",
7     "email": "maria.gonzalez@example.com",
8     "phone": "555-987-6543",
9     "address": "Avenida de la Paz 123",
10    "birth_date": "1985-06-15T00:00:00.000Z",
11    "occupation": "Ingeniera",
12    "housing_type": "Apartamento",
13    "has_other_pets": null,
14    "type_of_other_pets": null,
15    "adoption_reason": "Busco un amigo para mis hijos"
16  }
17 }
```

The screenshot shows a database table named 'adopters' with the following columns: first_name, last_name, email, phone, address, birth_date, occupation, housing_type, has_other_pets, type_of_other_pets, and adoption_reason. The table contains 5 rows of data.

	first_name	last_name	email	phone	address	birth_date	occupation	housing_type	has_other_pets	type_of_other_pets	adoption_reason
1	José	Martínez	jose.martinez@example.com	555-123-4567	Boulevard de la Libertad 456	1978-11-30	Arquitecto	Casa	1	Gato	Deseo adoptar un compañero para mi gato
2	Andrés	López	andres.lopez@example.com	555-456-7890	Calle de la Amistad 202	1982-09-12	Chef	Casa	1	Perro	Quiero un compañero para mis paseos
3	Lucía	Rodríguez	lucia.rodriguez@example.com	555-345-6789	Calle del Río 101	1995-03-05	Estudiante	Compartido	[NULL]	[NULL]	Quiero tener una mascota que me acompañe en mis estudios
4	Carlos	Hernández	carlos.hernandez@example.com	555-234-5678	Calle del Sol 789	1990-02-20	Profesor	Casa	1	Perro	Quiero darle un hogar a un animalito necesitado
5	María	González	maria.gonzalez@example.com	555-987-6543	Avenida de la Paz 123	1985-06-15	Ingeniera	Apartamento	[NULL]	[NULL]	Busco un amigo para mis hijos

- **Buscar Adoptantes:** Aquí se busca al adoptante que se tenga en la base de datos.

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:4000/adoptantes/buscarId/3`. The response status is 200 OK, with a size of 353 Bytes and a time of 13 ms. The response body is a JSON object containing the details of the adopter with ID 3.

```
GET http://127.0.0.1:4000/adoptantes/buscarId/3
Status: 200 OK Size: 353 Bytes Time: 13 ms
Response
1 {
2   "id": 3,
3   "first_name": "Lucía",
4   "last_name": "Rodríguez",
5   "email": "lucia.rodriguez@example.com",
6   "phone": "555-345-6789",
7   "address": "Calle del Río 101",
8   "birth_date": "1995-03-05",
9   "occupation": "Estudiante",
10  "housing_type": "Compartido",
11  "has_other_pets": null,
12  "type_of_other_pets": null,
13  "adoption_reason": "Quiero tener una mascota que me acompañe en mis estudios"
14 }
```

- **Actualizar Adoptantes:** Se podría actualizar cualquier dato que ya se tenga en la base de datos.

PUT http://127.0.0.1:4000/adoptantes/actualizar/3 Send

Query Headers Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

1 {
2 "first_name": "Lucía",
3 "last_name": "Rodríguez",
4 "email": "lucirodriguez854@example.com",
5 "phone": "555-345-6789",
6 "address": "Calle del Mar 308",
7 "birth_date": "1995-03-05",
8 "occupation": "Estudiante",
9 "housing_type": "Compartido",
10 "has_other_pets": false,
11 "type_of_other_pets": null,
12 "adoption_reason": "Quiero tener una mascota que me acompañe en mis estudios"
13 }

Status: 200 OK Size: 64 Bytes Time: 31 ms

Response Headers Cookies Results Docs

1 {
2 "tipo": "success",
3 "mensaje": "Registro de Adoptante actualizado"
4 }

1	1	José	Martínez	jose.martinez@example.com	555-123-4567	Boulevard de la Libertad 456	1978-11-30	Arquitecto	Casa	1	Gato	Deseo adoptar un compañero para mi gato
2	2	Andrés	López	andres.lopez@example.com	555-456-7890	Calle de la Amistad 202	1982-09-12	Chef	Casa	1	Perro	Quiero un compañero para mis paseos
3	3	Lucía	Rodríguez	lucirodriguez854@example.com	555-345-6789	Calle del Mar 308	1995-03-05	Estudiante	Compartido	[NULL]	[NULL]	Quiero tener una mascota que me acompañe en mis estudios

- **Eliminar Adoptantes:** Se podría eliminar cualquier adoptante que este en la base de datos.

DELETE http://127.0.0.1:4000/adoptantes/eliminar/5 Send

Query Headers Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

1 {
2 "tipo": "success",
3 "mensaje": "Registro con id 5 Eliminado Correctamente"
4 }

Status: 200 OK Size: 72 Bytes Time: 14 ms

Response Headers Cookies Results Docs

		A:1 first_name	A:2 last_name	A:3 email	A:4 phone	A:5 address	A:6 birth_date	A:7 occupation	A:8 housing_type	A:9 has_other_pets	A:10 type_of_other_pets	A:11 adoption_reason
1	1	José	Martínez	jose.martinez@example.com	555-123-4567	Boulevard de la Libertad 456	1978-11-30	Arquitecto	Casa	1	Gato	Deseo adoptar un compañero para mi gato
2	2	Andrés	López	andres.lopez@example.com	555-456-7890	Calle de la Amistad 202	1982-09-12	Chef	Casa	1	Perro	Quiero un compañero para mis paseos
3	3	Lucía	Rodríguez	lucirodriguez854@example.com	555-345-6789	Calle del Mar 308	1995-03-05	Estudiante	Compartido	[NULL]	[NULL]	Quiero tener una mascota que me acompañe en mis estudios
4	4	Carlos	Hernández	carlos.hernandez@example.com	555-234-5678	Calle del Sol 789	1990-02-20	Profesor	Casa	1	Perro	Quiero darle un hogar a un animalito necesitado

3.3. Modulo Mascotas: Para muestra de las pruebas utilizaremos el módulo de mascotas.

➤ Crear Mascotas: Aquí creamos a las Mascotas.

The screenshot displays a REST client interface with a POST request to `http://127.0.0.1:4000/mascotas/crear`. The request body is a JSON object representing a pet named "Paquita". The response status is 201 Created, with a size of 401 Bytes and a time of 22 ms. The response body is a JSON object containing a success message and the created pet's details.

```
POST http://127.0.0.1:4000/mascotas/crear
```

```
{
  "name": "Paquita",
  "species": "Fresh",
  "age": 5,
  "description": "Perra divertido y cariñoso, perfecta para la vida urbana.",
  "status": "adopted",
  "gender": "Femenino",
  "size": "mediano",
  "weight": 12.0,
  "entry_date": "2024-05-22",
  "vaccinated": true,
  "neutered": false,
  "medical_conditions": "Ninguna",
  "available_for_adoption": false
}
```

Status: 201 Created Size: 401 Bytes Time: 22 ms

```
{
  "mensaje": "Registro de Mascota creado con éxito",
  "data": {
    "id": 5,
    "name": "Paquita",
    "species": "Fresh",
    "breed": null,
    "age": 5,
    "description": "Perra divertido y cariñoso, perfecta para la vida urbana.",
    "status": "adopted",
    "gender": "Femenino",
    "size": "mediano",
    "weight": 12,
    "entry_date": "2024-05-22T00:00:00.000Z",
    "vaccinated": true,
    "neutered": null,
    "medical_conditions": "Ninguna",
    "available_for_adoption": null
  }
}
```

id	name	species	breed	age	description	status	gender	size	weight	entry_date	vaccinated	neutered	medical_conditions
1	Luna	Gato	Siamés	3	Gata tranquila y cariñosa, ideal para interiores.	adopted	femenino	pequeño	4,2	2024-08-20	1	[NULL]	Alergia leve
2	Max	Conejo	Enano	2	Conejo enano, muy activo y curioso.	available	masculino	pequeño	1,8	2024-09-01	[NULL]	[NULL]	Problema dental recurrente
3	Simba	Perro	Pastor Alemán	4	Perro leal y protector, excelente para familias.	available	masculino	grande	30	2024-07-15	1	1	Ninguna
4	Coco	Gato	Persa	2	Gata suave y juguetona, le encanta acurrucarse.	available	femenino	mediano	5,5	2024-09-10	1	1	Ninguna
5	Paquita	Fresh	[NULL]	5	Perra divertido y cariñoso, perfecta para la vida urbana.	adopted	femenino	mediano	12	2024-05-22	1	[NULL]	Ninguna
6	Lola	Gato	Bengali	1	Gata energética y juguetona, le encanta explorar.	available	femenino	pequeño	3,5	2024-09-05	1	[NULL]	Ninguna
7	Toby	Perro	Beagle	3	Perro curioso y amigable, siempre listo para la aventura.	available	masculino	pequeño	10	2024-08-30	1	1	Ninguna

➤ Buscar Mascotas: Permite buscar alguna mascota que se tenga registrada en la base de datos.

The screenshot displays a REST client interface with a GET request to `http://127.0.0.1:4000/mascotas/buscarId/5`. The response status is 200 OK, with a size of 333 Bytes and a time of 7 ms. The response body is a JSON object representing the pet with ID 5.

```
GET http://127.0.0.1:4000/mascotas/buscarId/5
```

Status: 200 OK Size: 333 Bytes Time: 7 ms

```
{
  "id": 5,
  "name": "Paquita",
  "species": "Fresh",
  "breed": null,
  "age": 5,
  "description": "Perra divertido y cariñoso, perfecta para la vida urbana.",
  "status": "adopted",
  "gender": "femenino",
  "size": "mediano",
  "weight": "12.00",
  "entry_date": "2024-05-22",
  "vaccinated": true,
  "neutered": null,
  "medical_conditions": "Ninguna",
  "available_for_adoption": null
}
```

- **Actualizar Mascotas:** Se podría actualizar cualquier dato que ya se tenga en la base de datos.

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:4000/mascotas/actualizar/5`. The request body is a JSON object representing a pet. The response is a 200 OK status with a JSON body indicating success.

Request:

```
PUT http://127.0.0.1:4000/mascotas/actualizar/5
```

Body:

```
{
  "id": 5,
  "name": "Paquita",
  "species": "Fresh",
  "breed": null,
  "age": 5,
  "description": "Perrita hermosa divertida y cariñosa, perfecta para ser tu mejor amiga.",
  "status": "adopted",
  "gender": "femenino",
  "size": "mediano",
  "weight": "12.00",
  "entry_date": "2024-05-22",
  "vaccinated": true,
  "neutered": null,
  "medical_conditions": "Ninguna",
  "available_for_adoption": null
}
```

Status: 200 OK **Size:** 62 Bytes **Time:** 12 ms

Response:

```
{
  "tipo": "success",
  "mensaje": "Registro de Mascota Actualizado"
}
```

- **Eliminar Mascotas:** Se podría eliminar cualquier mascota que este en la base de datos.

The screenshot shows a REST client interface with a DELETE request to `http://127.0.0.1:4000/mascotas/eliminar/6`. The response is a 200 OK status with a JSON body indicating success.

Request:

```
DELETE http://127.0.0.1:4000/mascotas/eliminar/6
```

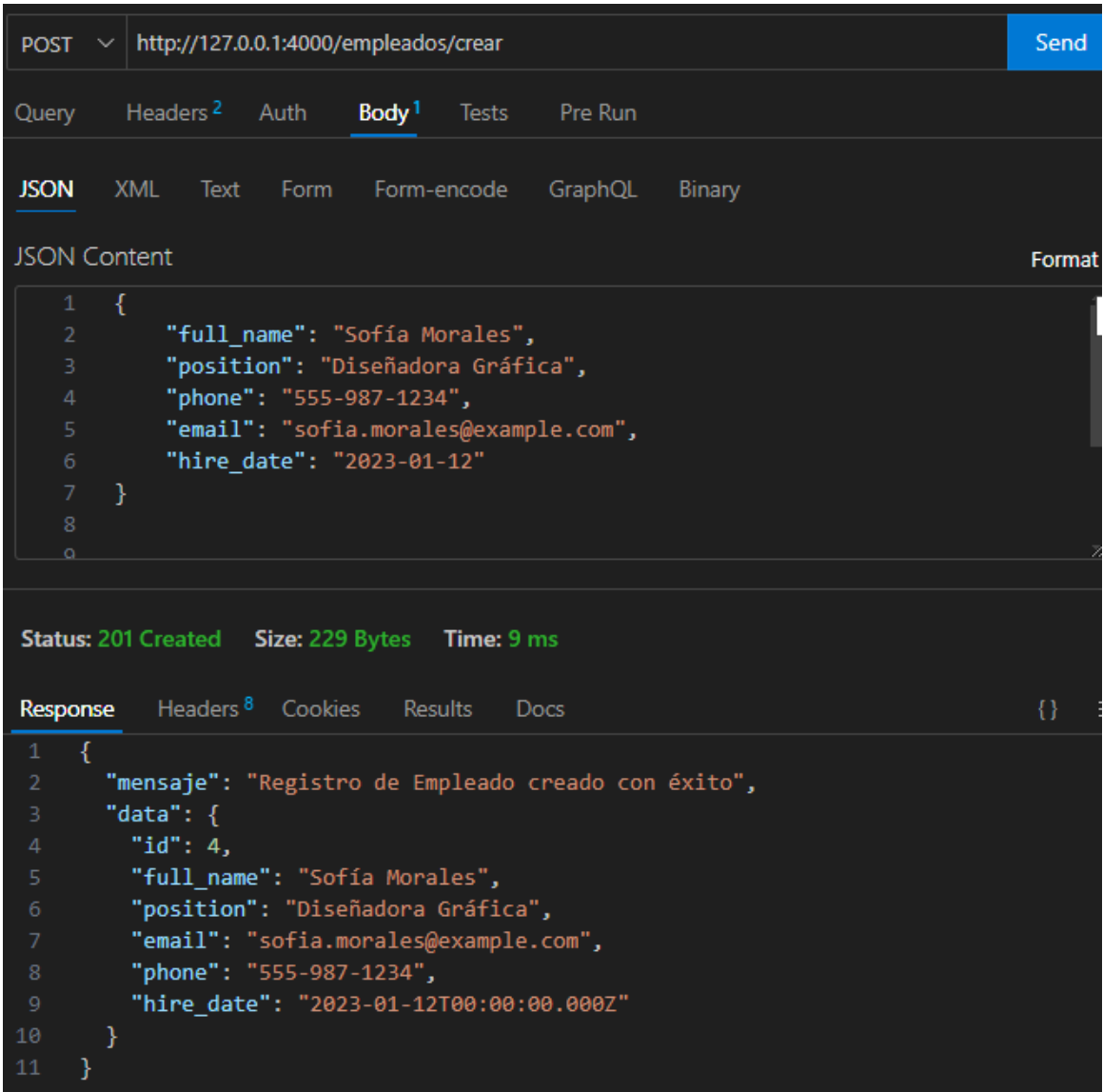
Status: 200 OK **Size:** 72 Bytes **Time:** 9 ms

Response:

```
{
  "tipo": "success",
  "mensaje": "Registro con id 6 Eliminado Correctamente"
}
```

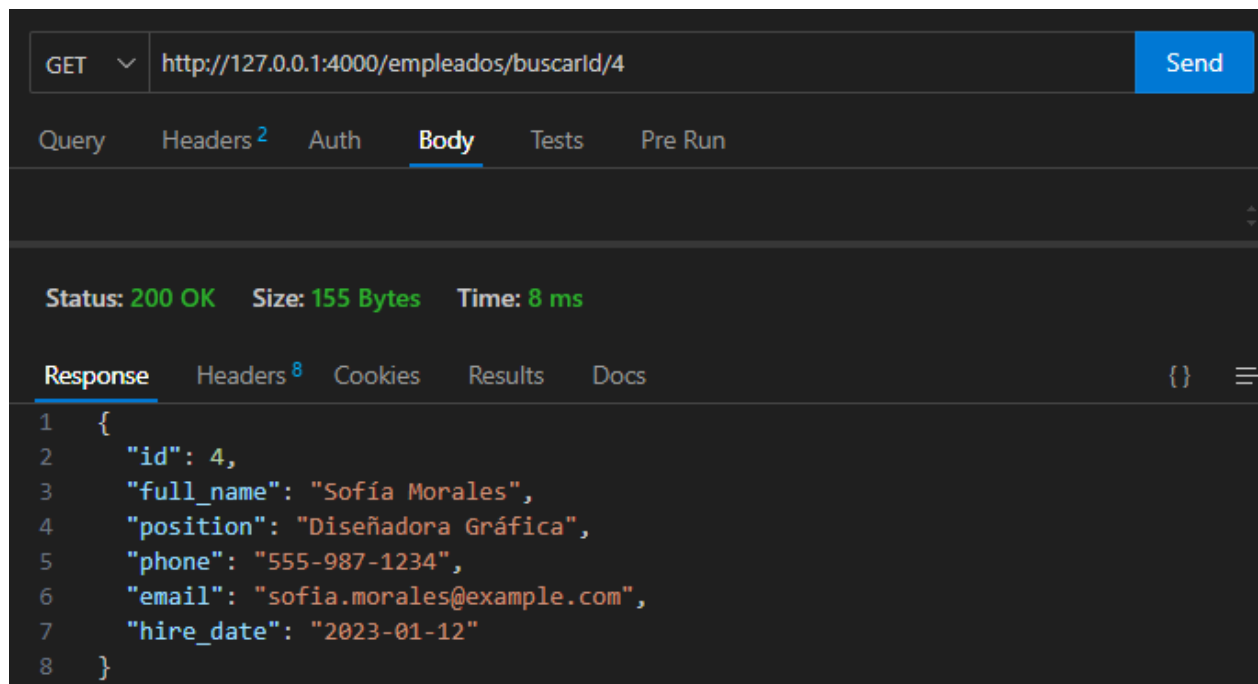
3.4. **Modulo Empleados:** Para muestra de las pruebas utilizaremos el módulo de empleados.

➤ **Crear Empleados:** Aquí creamos a los empleados.



123 id	A-Z full_name	A-Z position	A-Z phone	A-Z email	hire_date
1	Carlos Rodríguez	Gerente de Marketing	555-987-6543	carlos.rodriguez@example.com	2022-03-15
2	Ana Gómez	Desarrolladora de Software	555-321-6789	ana.gomez@example.com	2020-05-20
3	Pedro Sánchez	Asistente Administrativo	555-123-4567	pedro.sanchez@example.com	2019-08-30
4	Sofía Morales	Diseñadora Gráfica	555-987-1234	sofia.morales@example.com	2023-01-12
5	Javier Pérez	Analista de Datos	555-456-7890	javier.perez@example.com	2022-09-01

- **Buscar Empleados:** Permite buscar algún empleado que se tenga registrado en la base de datos.



```
GET http://127.0.0.1:4000/empleados/buscarId/4 Send
```

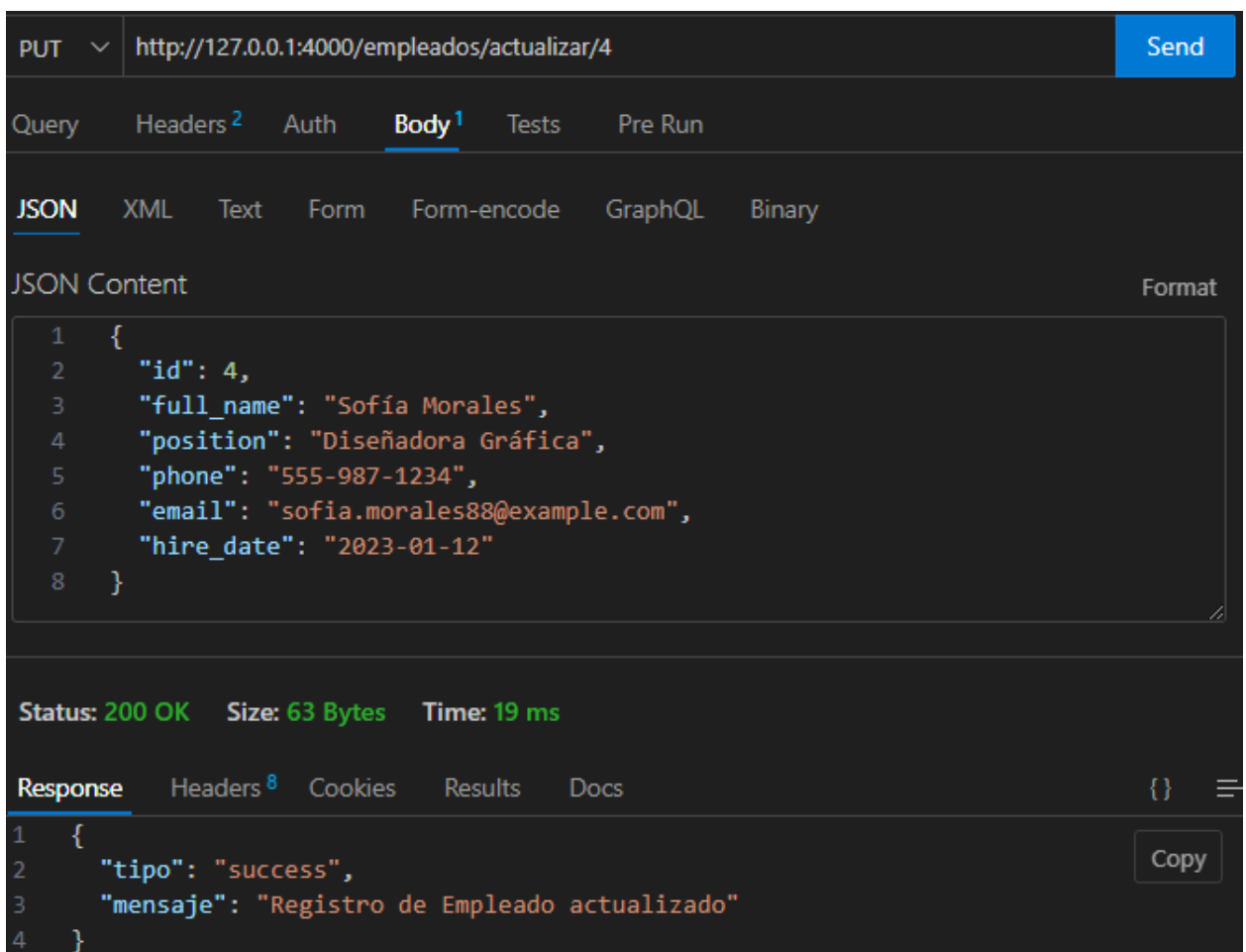
Query Headers² Auth Body Tests Pre Run

Status: 200 OK Size: 155 Bytes Time: 8 ms

Response Headers⁸ Cookies Results Docs {} ≡

```
1 {
2   "id": 4,
3   "full_name": "Sofía Morales",
4   "position": "Diseñadora Gráfica",
5   "phone": "555-987-1234",
6   "email": "sofia.morales@example.com",
7   "hire_date": "2023-01-12"
8 }
```

- **Actualizar Empleados:** Se podría actualizar cualquier dato que ya se tenga en la base de datos.



```
PUT http://127.0.0.1:4000/empleados/actualizar/4 Send
```

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "id": 4,
3   "full_name": "Sofía Morales",
4   "position": "Diseñadora Gráfica",
5   "phone": "555-987-1234",
6   "email": "sofia.morales88@example.com",
7   "hire_date": "2023-01-12"
8 }
```

Status: 200 OK Size: 63 Bytes Time: 19 ms

Response Headers⁸ Cookies Results Docs {} ≡

```
1 {
2   "tipo": "success",
3   "mensaje": "Registro de Empleado actualizado"
4 }
```

Copy

- **Eliminar Empleados:** Se podría eliminar cualquier empleado que este en la base de datos.

DELETE

▼

http://127.0.0.1:4000/empleados/eliminar/4

Send

Query

Headers²

Auth

Body

Tests

Pre Run

Status: 200 OK

Size: 72 Bytes

Time: 15 ms

Response

Headers⁸

Cookies

Results

Docs

{}

≡

1

{

2

"tipo": "success",

3

"mensaje": "Registro con id 4 Eliminado Correctamente"

4

}

Copy

	123 id	A-Z full_name	A-Z position	A-Z phone	A-Z email	🕒 hire_date
1	1	Carlos Rodríguez	Gerente de Marketing	555-987-6543	carlos.rodriguez@example.com	2022-03-15
2	2	Ana Gómez	Desarrolladora de Software	555-321-6789	ana.gomez@example.com	2020-05-20
3	3	Pedro Sánchez	Asistente Administrativo	555-123-4567	pedro.sanchez@example.com	2019-08-30
4	5	Javier Pérez	Analista de Datos	555-456-7890	javier.perez@example.com	2022-09-01

3.5. Modulo Solicitud: Para muestra de las pruebas utilizaremos el módulo de empleados.

- **Crear Solicitud:** Aquí creamos a las solicitudes.

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:4000/solicitudes/crear`. The request body is a JSON object with the following content:

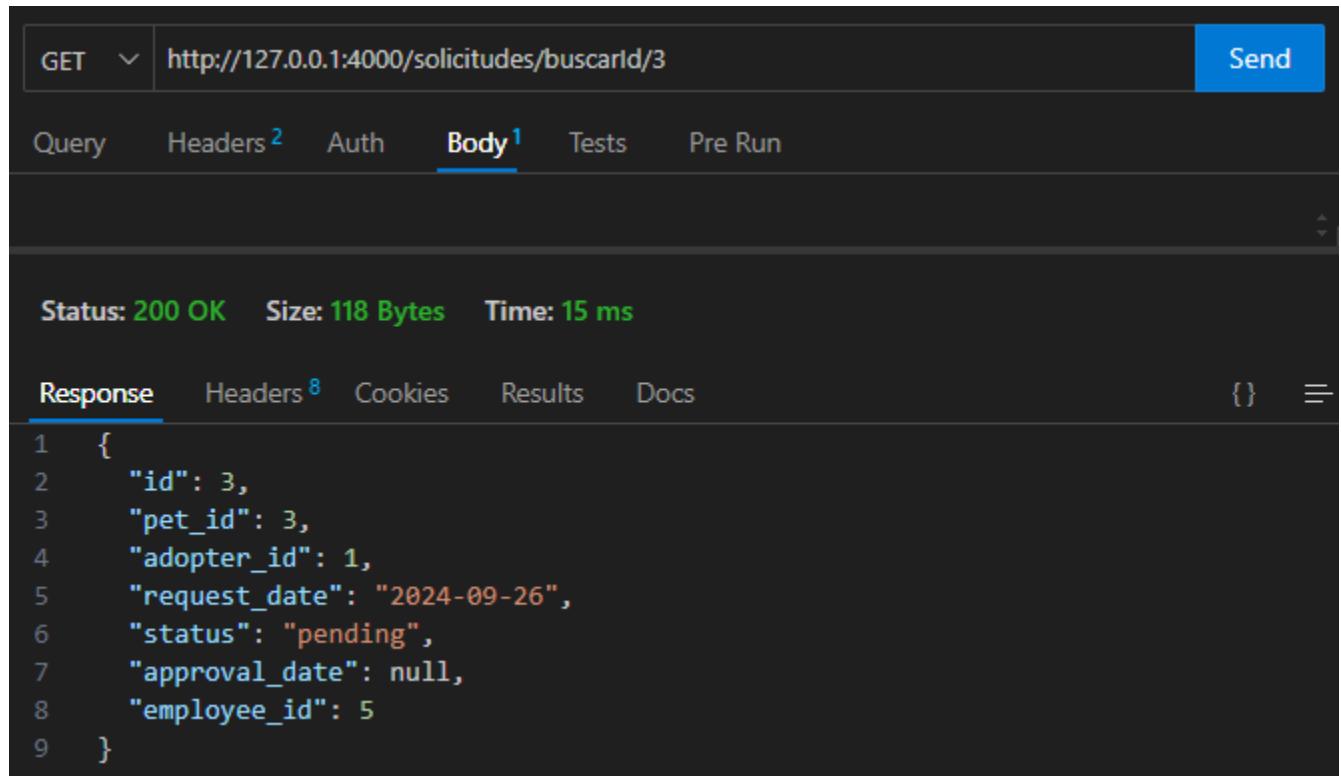
```
{
  "pet_id": 3,
  "adopter_id": 3,
  "employee_id": 3,
  "request_date": "2024-09-26",
  "status": "pending"
}
```

The response status is **201 Created**, with a size of **206 Bytes** and a time of **44 ms**. The response body is a JSON object:

```
{
  "mensaje": "Registro de Solicitud de Adopción creado con éxito",
  "data": {
    "id": 1,
    "pet_id": 3,
    "adopter_id": 3,
    "request_date": "2024-09-26T00:00:00.000Z",
    "status": "pending",
    "approval_date": null,
    "employee_id": 3
  }
}
```

adoption_requests								
Enter a SQL expression to filter results (use Ctrl+Space)								
	123 id	123 pet_id	123 adopter_id	request_date	A-Z status	approval_date	123 employee_id	
1	1	3	3	2024-09-26	pending	[NULL]	3	
2	2	1	2	2024-09-26	pending	[NULL]	1	
3	3	3	1	2024-09-26	pending	[NULL]	5	

- **Buscar Solicitud:** Permite buscar alguna Solicitud que se tenga registrado en la base de datos.



GET ⌵ http://127.0.0.1:4000/solitudes/buscarId/3 Send

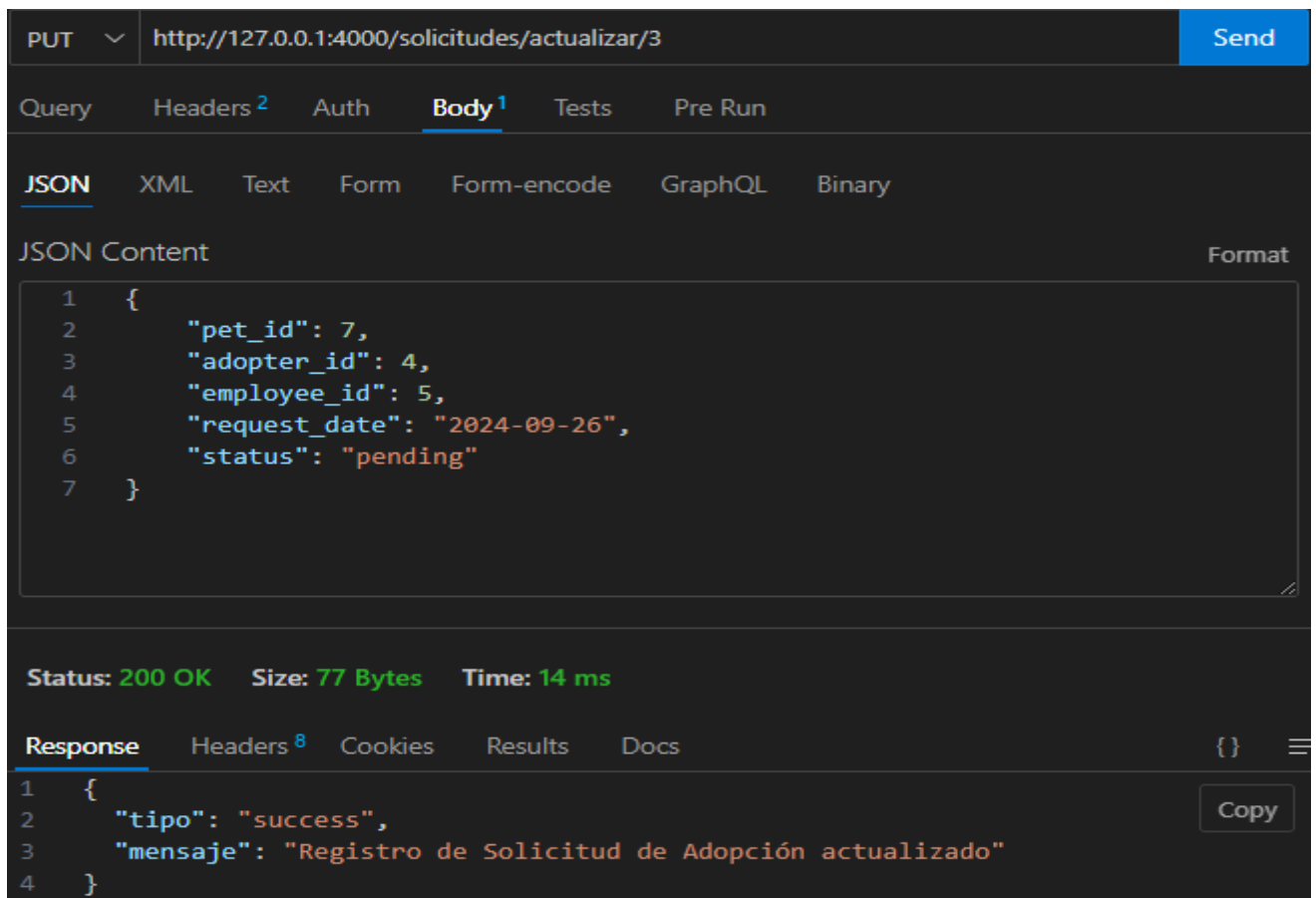
Query Headers² Auth Body¹ Tests Pre Run

Status: 200 OK Size: 118 Bytes Time: 15 ms

Response Headers⁸ Cookies Results Docs {} ≡

```
1 {
2   "id": 3,
3   "pet_id": 3,
4   "adopter_id": 1,
5   "request_date": "2024-09-26",
6   "status": "pending",
7   "approval_date": null,
8   "employee_id": 5
9 }
```

- **Actualizar Solicitud:** Se podría actualizar cualquier solicitud que ya se tenga en la base de datos.



PUT ⌵ http://127.0.0.1:4000/solitudes/actualizar/3 Send

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "pet_id": 7,
3   "adopter_id": 4,
4   "employee_id": 5,
5   "request_date": "2024-09-26",
6   "status": "pending"
7 }
```

Status: 200 OK Size: 77 Bytes Time: 14 ms

Response Headers⁸ Cookies Results Docs {} ≡

```
1 {
2   "tipo": "success",
3   "mensaje": "Registro de Solicitud de Adopción actualizado"
4 }
```

Copy

- **Eliminar Solicitud:** Se podría eliminar cualquier solicitud que este en la base de datos.

DELETE ▼ http://127.0.0.1:4000/solicitudes/eliminar/1 Send

Query Headers ² Auth **Body** Tests Pre Run

Status: 200 OK Size: 72 Bytes Time: 12 ms

Response Headers ⁸ Cookies Results Docs {} ≡

```
1 {
2   "tipo": "success",
3   "mensaje": "Registro con id 1 Eliminado Correctamente"
4 }
```

adoption_requests Enter a SQL expression to filter results (use Ctrl+Space)								
	123 id	123 pet_id	123 adopter_id	🕒 request_date	A-Z status	🕒 approval_date	123 employee_id	
1	2	1	2	2024-09-26	pending	[NULL]	1	
2	3	7	4	2024-09-26	pending	[NULL]	5	