

## Taller Final

Diplomado de actualización en nuevas tecnologías para el desarrollo de Software.

Ingeniería de Sistemas.

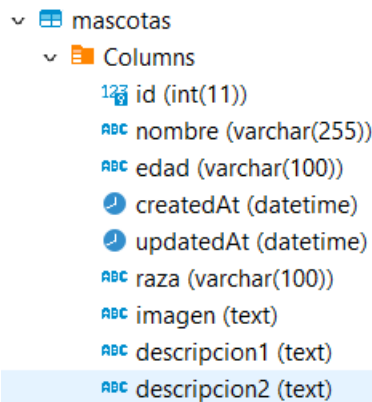
Universidad de Nariño.

Darwin Estiven Diaz Espinosa

Código: 219036069

### Modificaciones en la base de datos:

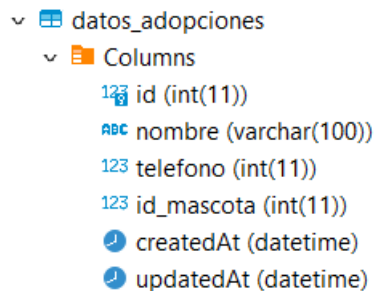
Se creó los campos que faltan en la tabla **mascotas** y quedaría así:



A screenshot of a database schema viewer. It shows a tree view with 'mascotas' expanded, and then 'Columns' expanded. The columns listed are: id (int(11)), nombre (varchar(255)), edad (varchar(100)), createdAt (datetime), updatedAt (datetime), raza (varchar(100)), imagen (text), descripcion1 (text), and descripcion2 (text). The 'descripcion2' row is highlighted in blue.

123	id (int(11))
ABC	nombre (varchar(255))
ABC	edad (varchar(100))
🕒	createdAt (datetime)
🕒	updatedAt (datetime)
ABC	raza (varchar(100))
ABC	imagen (text)
ABC	descripcion1 (text)
ABC	descripcion2 (text)

Se creó también la tabla **datos\_adopcion** con sus respectivos atributos, la cual nos servirá para guardar los datos de las personas que vayan a adoptar cada mascota, ésta quedó así:



A screenshot of a database schema viewer. It shows a tree view with 'datos\_adopciones' expanded, and then 'Columns' expanded. The columns listed are: id (int(11)), nombre (varchar(100)), telefono (int(11)), id\_mascota (int(11)), createdAt (datetime), and updatedAt (datetime).

123	id (int(11))
ABC	nombre (varchar(100))
123	telefono (int(11))
123	id_mascota (int(11))
🕒	createdAt (datetime)
🕒	updatedAt (datetime)

## Modificaciones en el BackEnd:

En el controlador **mascotasController.js**:

Importamos el modelo **adopcionesMascotas** en el cual se tiene la estructura de la tabla **datos\_adopciones**, mas adelante del informe se verá explicado.

```
import { adopcionesMascotas } from "../modelos/adopcionesModelo.js";
```

Agregamos los campos que creamos nuevos en cada función para crear, validar y eliminar:

```
// Crear un nuevo recurso
const crear = (req, res) => {
  // Validar si el campo 'nombre' está proporcionado
  if (!req.body.nombre) {
    res.status(400).json({
      mensaje: "El nombre no puede estar vacío."
    });
    return;
  }

  // Preparar el conjunto de datos con la información del cuerpo de la solicitud
  const dataset = {
    nombre: req.body.nombre,
    edad: req.body.edad,
    imagen: req.body.imagen,
    raza: req.body.raza,
    descripcion1: req.body.descripcion1,
    descripcion2: req.body.descripcion2
  };

  // Utilizar Sequelize para crear el recurso en la tabla 'mascotas'
  mascotas.create(dataset)
    .then((resultado) => {
      res.status(200).json({
        tipo: "success",
        mensaje: "Registro creado correctamente"
      });
    })
    .catch((err) => {
      res.status(500).json({
        tipo: "error",
        mensaje: `Error al crear el registro ::: ${err}`
      });
    });
};
```

Así mismo en cada una de las funciones.

También se creó una función nueva llamada **crearAdopcion()** la cual es utilizada para guardar los registros de los datos de los usuarios que van a adoptar las mascotas en la base de datos, ésta quedó así:

```

// Crear adopción
const crearAdopcion = (req, res) => {
  // Validar si los campos 'nombre' y 'telefono' están proporcionados
  if (!req.body.nombre || !req.body.telefono) {
    res.status(400).json({
      mensaje: "El nombre o el teléfono no pueden estar vacíos."
    });
    return;
  }

  // Preparar el conjunto de datos con la información del cuerpo de la solicitud
  const dataset = {
    id_mascota: req.body.id_mascota,
    nombre: req.body.nombre,
    telefono: req.body.telefono,
  };

  // Utilizar Sequelize para crear el recurso en la tabla 'adopcionesMascotas'
  adopcionesMascotas.create(dataset)
    .then((resultado) => {
      res.status(200).json({
        tipo: "success",
        mensaje: "Registro creado correctamente"
      });
    })
    .catch((err) => {
      res.status(500).json({
        tipo: "error",
        mensaje: `Error al crear el registro ::: ${err}`
      });
    });
};

```

Esta función se encarga de crear un nuevo registro de adopción en la base de datos utilizando los datos proporcionados en la solicitud. Realiza validaciones para asegurarse de que los campos esenciales estén presentes y maneja adecuadamente los casos de éxito y error durante el proceso de creación.

Exportamos junto con las demás funciones la función **crearAdopcion**:

```

// Exportar las funciones del controlador
export { crear, buscarId, buscar, actualizar, eliminar, crearAdopcion };

```

Este es el nuevo modelo creado **adopcionesModelo**:

```
// Importar Sequelize y la instancia de la base de datos desde el archivo de conexión
import Sequelize from "sequelize";
import { db } from "../database/conexion.js";

// Definir el modelo 'adopcionesMascotas'
const adopcionesMascotas = db.define("datos_adopciones", {
  // Columna 'id'
  id: {
    type: Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true,
    allowNull: false,
  },
  // Columna 'nombre'
  nombre: {
    type: Sequelize.STRING,
    allowNull: true
  },
  // Columna 'telefono'
  telefono: {
    type: Sequelize.INTEGER,
    allowNull: true
  },
  // Columna 'id_mascota'
  id_mascota: {
    type: Sequelize.INTEGER,
    allowNull: true
  },
});

// Exportar el modelo 'adopcionesMascotas'
export { adopcionesMascotas };
```

Este código define el modelo **adopcionesMascotas**, que representa la estructura de la tabla "datos\_adopciones" en la base de datos. Cada propiedad del modelo corresponde a una columna en la tabla, especificando su tipo de datos, si permite nulos, etc. Este modelo se utilizará para interactuar con los datos de adopciones de mascotas en la base de datos.

## Rutas:

Se creó una nueva ruta para lo relacionado con las adopciones como se ve a continuación:

```
// Importar express para crear el enrutador
import express from "express";
// Se importan las funciones controladoras desde el archivo mascotasController.js.
import { crear, buscarId, buscar, actualizar, eliminar, crearAdopcion } from "../controladores/mascotasController.js";

// Crear un enrutador de express
const routerMascotas = express.Router();

// Ruta de bienvenida para la API de mascotas
routerMascotas.get("/", (req, res) => {
  res.send("Bienvenido a Mascotas");
});

// Ruta para crear una nueva mascota
routerMascotas.post("/crear", (req, res) => {
  crear(req, res);
});

// Ruta para crear una nueva adopción de mascota
routerMascotas.post("/adopcion/crear", (req, res) => {
  crearAdopcion(req, res);
});

// Ruta para buscar una mascota por ID
routerMascotas.get("/buscar/:id", (req, res) => {
  buscarId(req, res);
});

// Ruta para buscar todas las mascotas
routerMascotas.get("/buscar", (req, res) => {
  buscar(req, res);
});

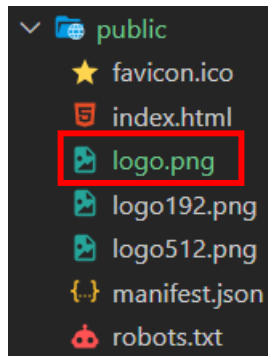
// Ruta para actualizar una mascota por ID
routerMascotas.put("/actualizar/:id", (req, res) => {
  actualizar(req, res);
});

// Ruta para eliminar una mascota por ID
routerMascotas.delete("/eliminar/:id", (req, res) => {
  eliminar(req, res);
});

// Exportar el enrutador para su uso en otros archivos
export { routerMascotas };
```

## Modificaciones en el FrontEnd:

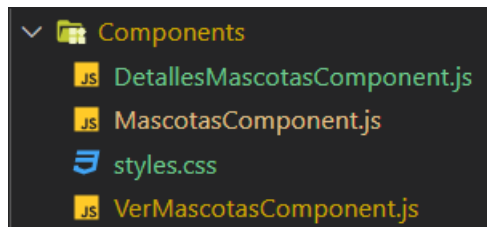
Se agrego una imagen para el logo en la carpeta public:



Esta imagen:



Se tienen 3 componentes y un archivo .css para los estilos de los botones de la paginación:



**NOTA: DE LOS COMPONENTES (MascotasComponent.js y VerMascotasComponent.js) NO PEGO CAPTURA YA QUE CONTIENEN MUCHAS LINEAS DE CÓDIGO, PERO EN EL PROYECTO ESTÁN COMENTADAS Y EXPLICADAS DETALLADAMENTE LAS LINEAS DE CÓDIGO, AL IGUALMENTE QUE TODOS LOS DEMAS ARCHIVOS.**

## Detalles MascotasComponent.js:

```
// IMPORTACIONES
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom"; // Hook para obtener parámetros de la URL
import axios from "axios"; // Librería para realizar solicitudes HTTP
import { Link } from "react-router-dom"; // Componente para navegar entre rutas
import './styles.css'; // Estilos del componente

// CUERPO DEL COMPONENTE
const VerMascotasComponent = () => {
  // Obtiene el parámetro de la URL usando el hook useParams de React Router
  const { id } = useParams();

  const url = "http://localhost:8000/mascotas";
  const [mascotas, setMascotas] = useState([null]); // Estado para almacenar la información de la mascota

  // Efecto de montaje: se ejecuta al cargar el componente o cuando el parámetro id cambia
  useEffect(() => {
    getMascotaById(id); // Llama a la función para obtener la información de la mascota por su ID
  }, [id]);

  // Función asincrónica para obtener la información de la mascota por su ID
  const getMascotaById = async (id) => {
    try {
      const respuesta = await axios.get(`${url}/buscar/${id}`);
      console.log(respuesta.data);
      setMascotas(respuesta.data); // Actualiza el estado con la información de la mascota
    } catch (error) {
      console.error("Error al obtener la mascota:", error);
    }
  };
};
```

```

// RENDERIZACIÓN DEL COMPONENTE
return (
  <div className="App">
    <br />
    {/ * Sección para mostrar el logo o imagen de la página */}
    <div>
      
    </div>
    <br />

    {/ * Verifica si hay información de mascotas antes de renderizar */}
    {mascotas && (
      <div className="container">
        <br />
        {/ * Sección para mostrar los detalles de la mascota */}
        <div>
          <h2>{mascotas.nombre} ({mascotas.raza})</h2>
          <p style={{ textAlign: 'justify' }}>{mascotas.descripcion2}</p>
        </div>

        {/ * Botón para regresar a la página principal */}
        <div className="d-flex justify-content-between mt-3">
          <Link to="/" className="btn btn-dark ms-auto">
            <i className="fas fa-reply"></i> Regresar
          </Link>
        </div>
        <br />
      </div>
    )}
  </div>
);

// EXPORTACIÓN DEL COMPONENTE
export default VerMascotasComponent;

```



## App.js

```
// Importaciones necesarias para el enrutamiento
import { BrowserRouter, Routes, Route } from 'react-router-dom';

// Importaciones de componentes creados
import MascotasComponent from './Components/MascotasComponent.js';
import VerMascotasComponent from './Components/VerMascotasComponent.js';
import DetallesMascotasComponent from './Components/DetallesMascotasComponent.js';

// Estilos de la aplicación
import './App.css';

// Función principal que define la estructura de las rutas
function App() {
  return (
    <BrowserRouter>
      <Routes>
        {/* Ruta para mostrar la lista de mascotas */}
        <Route path="/" element={<VerMascotasComponent></VerMascotasComponent>}></Route>

        {/* Ruta para crear o editar mascotas */}
        <Route path="/crear" element={<MascotasComponent></MascotasComponent>}></Route>


        {/* Ruta para ver detalles de una mascota específica */}
        <Route path="/detalles/:id" element={<DetallesMascotasComponent></DetallesMascotasComponent>}></Route>
      </Routes>
    </BrowserRouter>
  );
}


// Exporta la función principal para ser utilizada en otros archivos
export default App;
```

**NOTA:** EN EL PROYECTO TODO EL CÓDIGO ESTA COMENTADO Y EXPLICADO DETALLADAMENTE.

## Capturas de la ejecución:


Pantalla principal con la paginación funcionando correctamente:






**Toby**  
**Edad:** 11 meses  
**Raza:** Lobo Siberiano  
Perro grande, juguetón y amoroso.

[i Detalles](#) [Adoptar](#)



**Nina**  
**Edad:** 1 año  
**Raza:** Pincher  
Perra pequeña


[i Detalles](#) [Adoptar](#)




**Luna**  
**Edad:** 3 años  
**Raza:** Criolla  
Cachorra juguetona

[i Detalles](#) [Adoptar](#)


[Anterior](#) [1](#) [2](#) [Siguiente](#)





**Rambo**  
**Edad:** 10 meses  
**Raza:** American bully  
Perro de tamaño estándar.

[i Detalles](#) [Adoptar](#)



**Romeo**  
**Edad:** 15 meses  
**Raza:** Pug  
Perro bajo y de aspecto cuadrado.

[i Detalles](#) [Adoptar](#)

[Anterior](#) [1](#) [2](#) [Siguiente](#)

Aquí se pueden ver los datos en la base de datos:

mascotas Enter a SQL expression to filter results (use Ctrl+Space)									
			createdAt	updatedAt	raza	imagen	descripcion1	descripcion2	
1	10	Toby	11	2023-12-29 20:26:23	2023-12-30 17:49:04	Lobo Siberiano	<a href="https://upload.wikimedia.org/wiki">https://upload.wikimedia.org/wiki</a>	Perro grande, juguetón y amoroso.	El perro lobo checoslovaco (tambi
2	11	Nina	1	2023-12-29 20:38:23	2023-12-30 18:00:26	Pincher	<a href="https://images.hola.com/imagen">https://images.hola.com/imagen</a>	Perra pequeña	El pinscher miniatura es una raza c
3	13	Luna	3	2023-12-30 02:19:58	2023-12-30 18:01:48	Criolla	<a href="https://www.adopta.mx/wp-conte">https://www.adopta.mx/wp-conte</a>	Cachorra juguetona	Se conoce como perro criollo o m
4	18	Rambo	10	2023-12-30 17:23:34	2024-01-03 03:08:35	American bully	<a href="https://static.wixstatic.com/media">https://static.wixstatic.com/media</a>	Perro de tamaño estándar.	American Bully es una raza de per
5	22	Romeo	15	2024-01-03 03:06:49	2024-01-03 03:08:30	Pug	<a href="https://cdn.wamiz.fr/cdn-cgi/ima">https://cdn.wamiz.fr/cdn-cgi/ima</a>	Perro bajo y de aspecto cuadrado.	El pug1 (también conocido como

Captura de la pagina de detalles, esta página se abre al momento de darle clic en el botón **Detalles**.



### Toby (Lobo Siberiano)

El perro lobo checoslovaco (también referido como PLC) es una raza canina relativamente nueva, y cuyo linaje original se remonta a un experimento llevado a cabo en 1955 en Checoslovaquia. Después de comenzar la gestación del linaje de los 48 ejemplares de pastor alemán con cuatro lobos europeos, se elaboró un plan para crear un híbrido que tuviera el temperamento, la mentalidad y la capacidad de entrenamiento del pastor alemán, junto con la fuerza, la constitución física y la resistencia de los lobos. Tienen una apariencia muy similar a los lobos de los Cárpatos.

[Regresar](#)



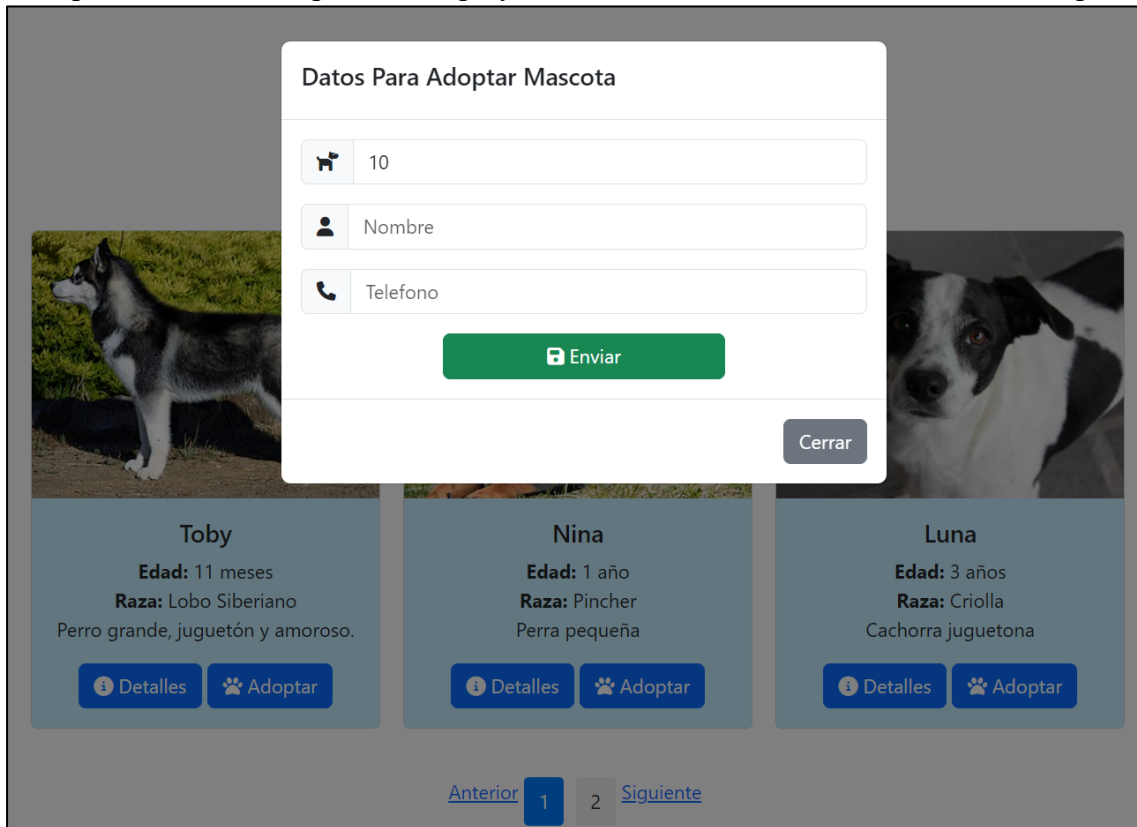
### Rambo (American bully)

American Bully es una raza de perro tipo Terrier de tamaño estándar originaria de Estados Unidos.<sup>2</sup> No se utilizaban como perros de pelea aún cuando estos eventos fueron prohibidos en 1976; actualmente son criados como perros de compañía. Es reconocida principalmente por el United Kennel Club (desde 2013), Club Híbrido Canino Americano y por el ABKC (American Bully Kennel Club), en Estados Unidos, y por Alianz Canine y la CRCPE (Conservación de las razas caninas puras en España), en España, ambas entidades reconocidas oficialmente por el Ministerio de Agricultura, Ganadería y Medio Ambiente de España.

[Regresar](#)


Al darle en el botón **Regresar**, vuelve a la pantalla principal.


Al darle en el botón **Adoptar** se despliega el formulario para llenar los datos de la persona que va a adoptar la mascota, el primer campo ya trae el id de la mascota de la cual va a adoptar:





The screenshot shows a modal titled "Datos Para Adoptar Mascota" overlaid on a list of animals. The modal contains three input fields: the first has a dog icon and the value "10"; the second has a person icon and the placeholder "Nombre"; the third has a phone icon and the placeholder "Telefono". Below the fields is a green "Enviar" button and a grey "Cerrar" button. The background shows three animal cards: Toby (Siberian Husky, 11 months), Nina (Pincher, 1 year), and Luna (Criolla, 3 years). Navigation links "Anterior", "1", "2", and "Siguiete" are at the bottom.

**Datos Para Adoptar Mascota**

 10

 Nombre

 Telefono

 Enviar

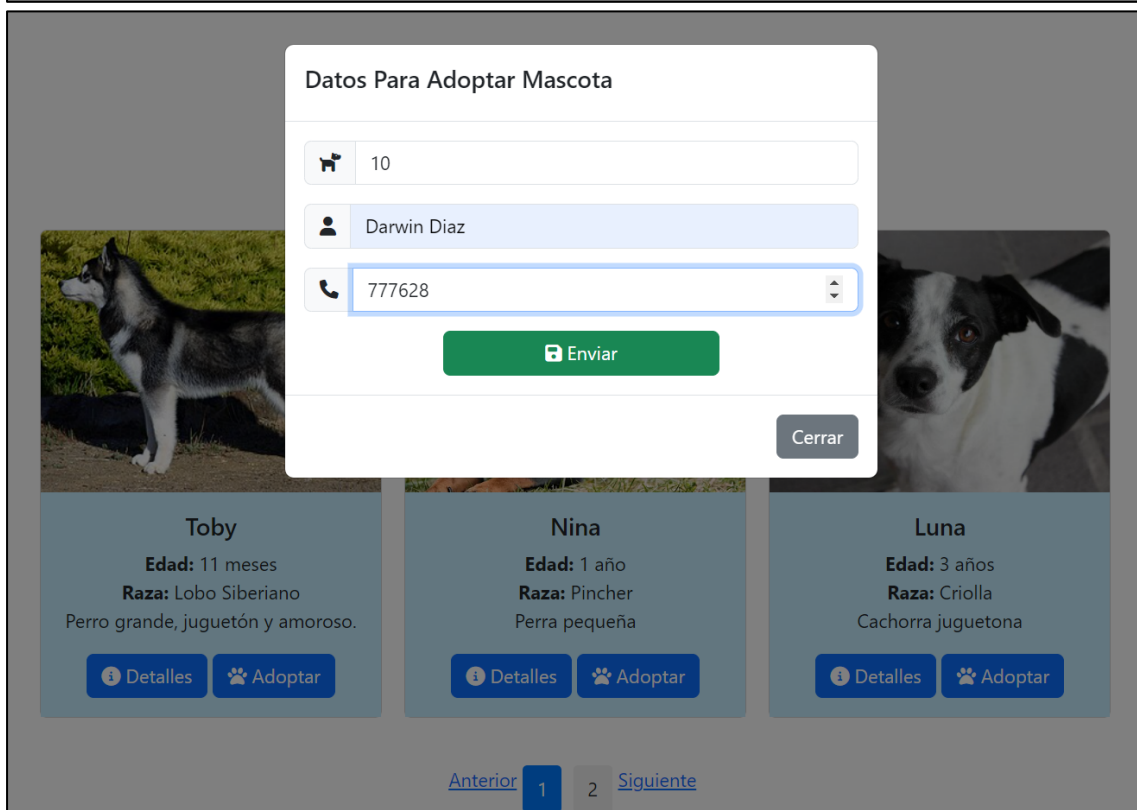
Cerrar

**Toby**  
**Edad:** 11 meses  
**Raza:** Lobo Siberiano  
Perro grande, juguetón y amoroso.  
[Detalles](#) [Adoptar](#)

**Nina**  
**Edad:** 1 año  
**Raza:** Pincher  
Perra pequeña  
[Detalles](#) [Adoptar](#)


**Luna**  
**Edad:** 3 años  
**Raza:** Criolla  
Cachorra juguetona  
[Detalles](#) [Adoptar](#)


[Anterior](#) **1** 2 [Siguiete](#)





This screenshot shows the same modal as above, but with the "Nombre" field filled with "Darwin Diaz" and the "Telefono" field filled with "777628". The "Enviar" button remains green, indicating the form is valid.

**Datos Para Adoptar Mascota**

 10

 Darwin Diaz

 777628

 Enviar

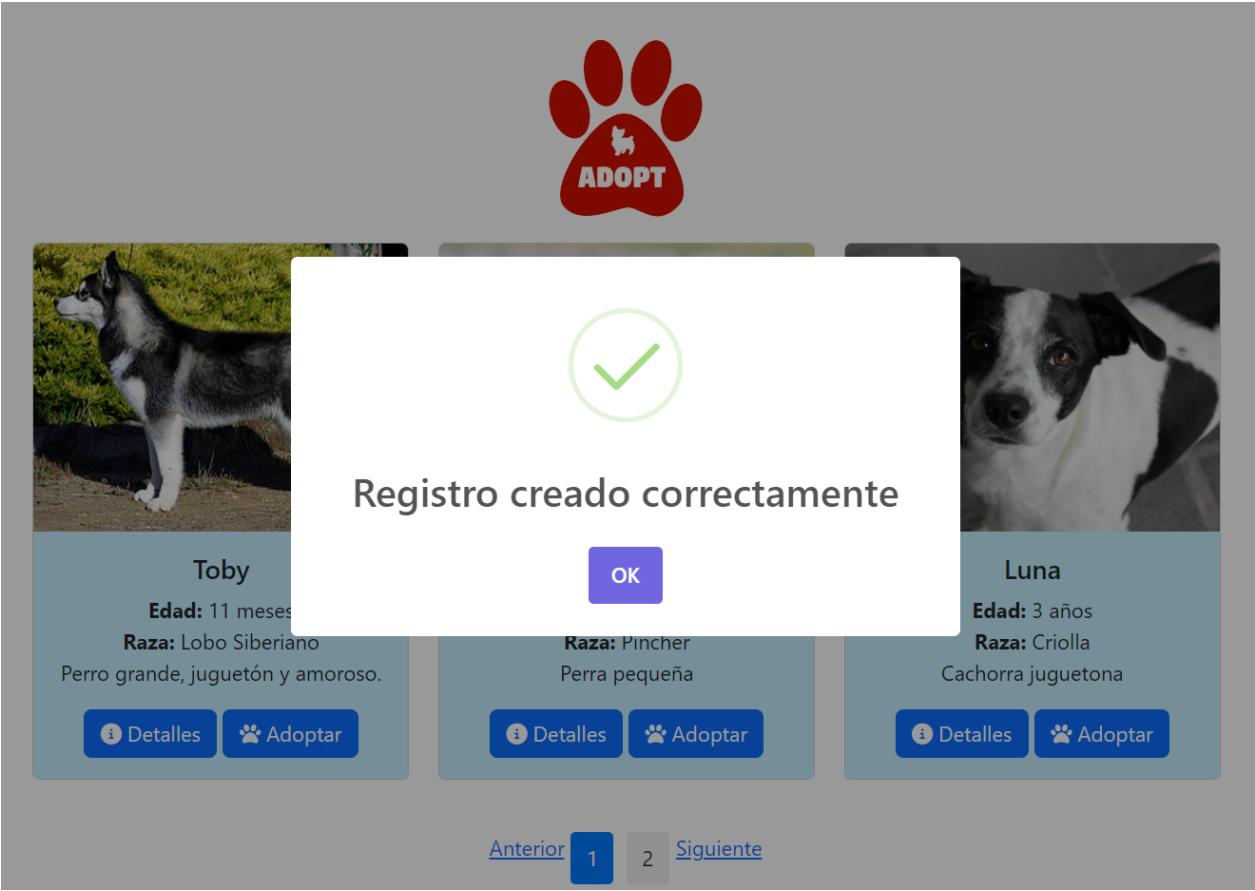
Cerrar

**Toby**  
**Edad:** 11 meses  
**Raza:** Lobo Siberiano  
Perro grande, juguetón y amoroso.  
[Detalles](#) [Adoptar](#)

**Nina**  
**Edad:** 1 año  
**Raza:** Pincher  
Perra pequeña  
[Detalles](#) [Adoptar](#)

**Luna**  
**Edad:** 3 años  
**Raza:** Criolla  
Cachorra juguetona  
[Detalles](#) [Adoptar](#)

[Anterior](#) **1** 2 [Siguiete](#)



Aquí se puede ver los registros en la tabla de adopciones:

datos_adopciones <i>Enter a SQL expression to filter results (use Ctrl+Space)</i>						
Grilla	id	nombre	telefono	id_mascota	createdAt	updatedAt
1	7	Sandra	7.256.036	13	2024-01-02 19:55:13	2024-01-02 19:55:13
2	9	Darwin Diaz	777.628	10	2024-01-03 03:32:28	2024-01-03 03:32:28

## AGREGACIONES PARA EL TALLER FINAL

### Agregaciones en el BackEnd:

MascotasController.js:

```
// Importar los módulos y dependencias necesarios
import { Op } from "sequelize"; // consultas de Sequelize.
import { usuarios } from "../modelos/usuariosModelo.js";
```

Se modificó la función buscar y quedó así:

```
// Buscar todos los recursos
const buscar = (req, res) => {
  const terminoBusqueda = req.query.termino;
  console.log(terminoBusqueda)

  // Verificar si se proporciona un término de búsqueda
  if (!terminoBusqueda) {
    // Si no se proporciona, realizar la búsqueda normal de todos los recursos
    mascotas.findAll()
      .then((resultado) => {
        res.status(200).json(resultado);
      })
      .catch((err) => {
        res.status(500).json({
          mensaje: `No se encontraron registros ::: ${err}`
        });
      });
  } else {
    // Si se proporciona un término de búsqueda, realizar la búsqueda filtrada
    mascotas.findAll({
      where: {
        // Puedes ajustar estas condiciones según tus necesidades
        [Op.or]: [
          { nombre: { [Op.like]: `%%${terminoBusqueda}%` } },
          { raza: { [Op.like]: `%%${terminoBusqueda}%` } },
        ]
      }
    })
      .then((resultado) => {
        res.status(200).json(resultado);
      })
      .catch((err) => {
        res.status(500).json({
          mensaje: `No se encontraron registros con el término de búsqueda ::: ${err}`
        });
      });
  }
};
```

Esta función busca en la tabla de mascotas y recupera los registros según el término de búsqueda proporcionado. Si no se proporciona ningún término, devuelve todos los registros. Utiliza

Sequelize (ORM) para construir la consulta y manejar las condiciones de búsqueda. Además, captura errores y devuelve respuestas adecuadas tanto en caso de éxito como de error.

Se creo una nueva función llamada “autenticarUsuario” para el login del admin:

```
const autenticarUsuario = async (req, res) => {
  // Obtener el nombre de usuario y contraseña del cuerpo de la solicitud
  const { username, password } = req.body;
  console.log(username);
  console.log(password);

  try {
    // Buscar un usuario en la base de datos que coincida con el nombre de usuario y la contraseña proporcionados
    const usuarioEncontrado = await usuarios.findOne({ where: { username, password } });
    console.log(usuarioEncontrado);

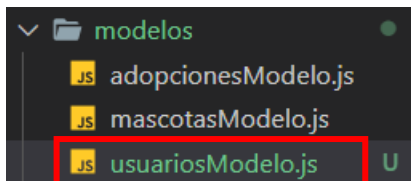
    if (usuarioEncontrado) {
      // Autenticación exitosa: El usuario existe en la base de datos con las credenciales proporcionadas
      res.status(200).json({ success: true });
    } else {
      // Credenciales incorrectas: El usuario no fue encontrado con las credenciales proporcionadas
      res.status(401).json({ success: false, mensaje: "Credenciales incorrectas" });
    }
  } catch (error) {
    // Manejar errores: En caso de error durante la búsqueda en la base de datos
    console.error("Error en autenticarUsuario:", error);
    res.status(500).json({ success: false, mensaje: "Error interno del servidor" });
  }
};
```

Esta función está diseñada para autenticar a un usuario. Recibe el nombre de usuario y la contraseña desde el cuerpo de la solicitud. Luego, utiliza Sequelize para buscar un usuario en la base de datos que coincida con estas credenciales. Si se encuentra el usuario, devuelve una respuesta de éxito; de lo contrario, devuelve un mensaje de "Credenciales incorrectas". La función maneja errores, proporcionando un mensaje de error interno del servidor en caso de problemas durante la autenticación.

Se exportan las funciones del controlador añadiendo la nueva:

```
export { crear, buscarId, buscar, actualizar, eliminar, crearAdopcion, autenticarUsuario };
```

Se crea un nuevo modelo llamado “usuariosModelo.js” para la tabla de usuarios:



## Tabla usuarios en la base de datos con sus respectivos atributos:

- ▼ usuarios
  - ▼ Columns
    - 123 id\_user (int(11))
    - ABC username (varchar(100))
    - ABC password (varchar(100))
    - 🕒 createdAt (datetime)
    - 🕒 updatedAt (datetime)

## Modelo usuariosModelo.js:

```
// Importar Sequelize y la instancia de la base de datos desde el archivo de conexión
import Sequelize from "sequelize";
import { db } from "../database/conexion.js";

// Definir el modelo 'usuarios'
const usuarios = db.define("usuarios", {
  // Columna 'id'
  id_user: {
    type: Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true,
    allowNull: false,
  },
  // Columna 'usuario'
  username: {
    type: Sequelize.STRING,
    allowNull: true,
  },
  // Columna 'contraseña'
  password: {
    type: Sequelize.STRING,
    allowNull: true,
  },
});

// Exportar el modelo 'adopcionesMascotas'
export { usuarios };
```

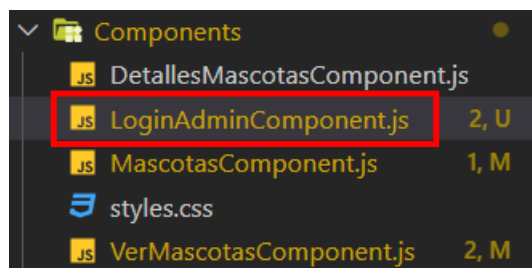


**Nueva ruta en el archivo mascotasRouter.js para el login:**

```
//Ruta login
routerMascotas.post("/login", async (req, res) => {
  try {
    // Intenta autenticar al usuario al llamar a la función 'autenticarUsuario'
    await autenticarUsuario(req, res);
  } catch (error) {
    // Manejar errores: Si hay algún error, imprímelo en la consola y devuelve una respuesta de error interno del servidor
    console.error("Error en el servidor:", error);
    res.status(500).json({ success: false, message: "Error interno del servidor" });
  }
});
```

**Agregaciones en el FrontEnd:**

**Nuevo componente para el login:**



## LoginAdminComponent.js:

```
import React, { useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom"; // Cambiado de useHistory
import { Link } from "react-router-dom"; // Componente para navegar entre rutas
import { mostrarAlerta } from "../functions.js"; // Importa una función mostrarAlerta desde un archivo functions.js

const LoginAdminComponent = () => {
  // Definición de la URL para las solicitudes
  const url = "http://localhost:8000/mascotas";

  // Hook de navegación
  const navigate = useNavigate();

  // Estado para almacenar los datos del formulario (nombre de usuario y contraseña)
  const [formData, setFormData] = useState({
    username: "",
    password: "",
  });

  // Función para manejar cambios en los campos del formulario
  const handleChange = (e) => {
    setFormData({
      ...formData,
      [e.target.name]: e.target.value,
    });
  };

  // Función para manejar el envío del formulario
  const handleSubmit = async (e) => {
    e.preventDefault();

    try {
      // Enviar una solicitud POST al servidor con los datos del formulario
      const response = await axios.post(`${url}/login`, formData);

      if (response.data.success) {
        // Autenticación exitosa, redirigir a la página /crear
        navigate("/crear");
      } else {
        // Mostrar mensaje de error
        mostrarAlerta("Username o password incorrectas");
      }
    } catch (error) {
      // Manejar errores
      mostrarAlerta("Username o password incorrectas");
      if (error.response) {
        // El error tiene una respuesta del servidor
        console.log("Detalles de la respuesta del servidor:", error.response);
      }
    }
  };
};
```

```

// Este componente representa la interfaz de usuario para la página de inicio de sesión de un administrador.
return (
  <div className="App">
    <br />
    {/} Sección para mostrar el logo o imagen de la página {/}
    <div>
      
    </div>

    {/} Contenedor principal {/}
    <div className="container mt-5">
      {/} Fila centralizada {/}
      <div className="row justify-content-center">
        {/} Columna de ancho medio {/}
        <div className="col-md-6">
          {/} Tarjeta que contiene el formulario {/}
          <div className="card">
            {/} Encabezado de la tarjeta con título centrado {/}
            <div className="card-header">
              <h2 className="text-center">Login</h2>
            </div>

            {/} Cuerpo de la tarjeta que contiene el formulario de inicio de sesión {/}
            <div className="card-body">
              {/} Formulario {/}
              <form onSubmit={handleSubmit}>
                {/} Campo de entrada para el nombre de usuario {/}
                <div className="form-group">
                  <label htmlFor="username">Username:</label>
                  <input
                    type="text"
                    className="form-control"
                    id="username"
                    name="username"
                    value={formData.username}
                    onChange={handleChange}
                    required
                  />
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
)

```



Se modificó las variables para la paginación, ya que, al buscar, la paginación cambia:

```
// Calcula el número total de páginas requeridas para la paginación
const pageCount = Math.ceil(mascotas.length / mascotasPerPage);

// Calcula el índice de inicio de las mascotas en la página actual
const offset = pageNumber * mascotasPerPage;
```

Se agregó el buscador y el botón de cerrar sesión cuando este Logeado:

```
<form class="d-flex" role="search" style={{ flex: '1' }}>
  <input
    class="form-control me-2"
    type="search"
    placeholder="Buscar por nombre o raza"
    aria-label="Search"
    style={{ width: '100%' }}
    value={busqueda}
    onChange={(e) => setBusqueda(e.target.value)}
  />
  <button class="btn btn-success" type="button" onClick={getMascotas}>
    Buscar
  </button>
</form>

<Link to={`/${}`} className="btn btn-danger ms-2">
  <i className="fas fa-sign-out-alt"></i> Cerrar Sesión
</Link>
```

En el componente VerMascotasComponent.js se hicieron las mismas modificaciones anteriores ya que el buscador es el mismo, solo se agrega el icono para login con la redirección de la página correspondiente, en este caso “/login”:

```
<Link to="/login">
  
</Link>
```



En el archivo App.js se agregó la ruta para el login:


```
/* Ruta para ir al login */
<Route path="/login" element={<LoginAdminComponent></LoginAdminComponent>}></Route>
```

**NOTA: EN EL PROYECTO TODO EL CÓDIGO ESTA COMENTADO Y EXPLICADO DETALLADAMENTE.**

## CAPTURAS DE LA EJECUCIÓN:


Página principal:

Buscar




**Toby**  
**Edad:** 11 meses  
**Raza:** Lobo Siberiano  
Perro grande, juguetón y amoroso.

[Detalles](#)[Adoptar](#)



**Nina**  
**Edad:** 1 año  
**Raza:** Pincher  
Perra pequeña

[Detalles](#)[Adoptar](#)



**Luna**  
**Edad:** 3 años  
**Raza:** Criolla  
Cachorra juguetona

[Detalles](#)[Adoptar](#)

[Anterior](#)

12

[Siguiete](#)

Buscar



**Rambo**  
**Edad:** 10 meses  
**Raza:** American bully  
Perro de tamaño estándar.

[Detalles](#)[Adoptar](#)



**Romeo**  
**Edad:** 15 meses  
**Raza:** Pug  
Perro bajo y de aspecto cuadrado.

[Detalles](#)[Adoptar](#)



**Mono**  
**Edad:** 3 meses  
**Raza:** Pug  
Un perrito hermoso


[Detalles](#)[Adoptar](#)

[Anterior](#)


12


[Siguiete](#)

El buscador funciona correctamente tanto en esta pagina como en la pagina del administrador:



Buscar







**Rambo**  
**Edad:** 10 meses  
**Raza:** American bully  
Perro de tamaño estándar.


[Detalles](#) [Adoptar](#)

[Anterior](#) [1](#) [Siguiente](#)



Buscar



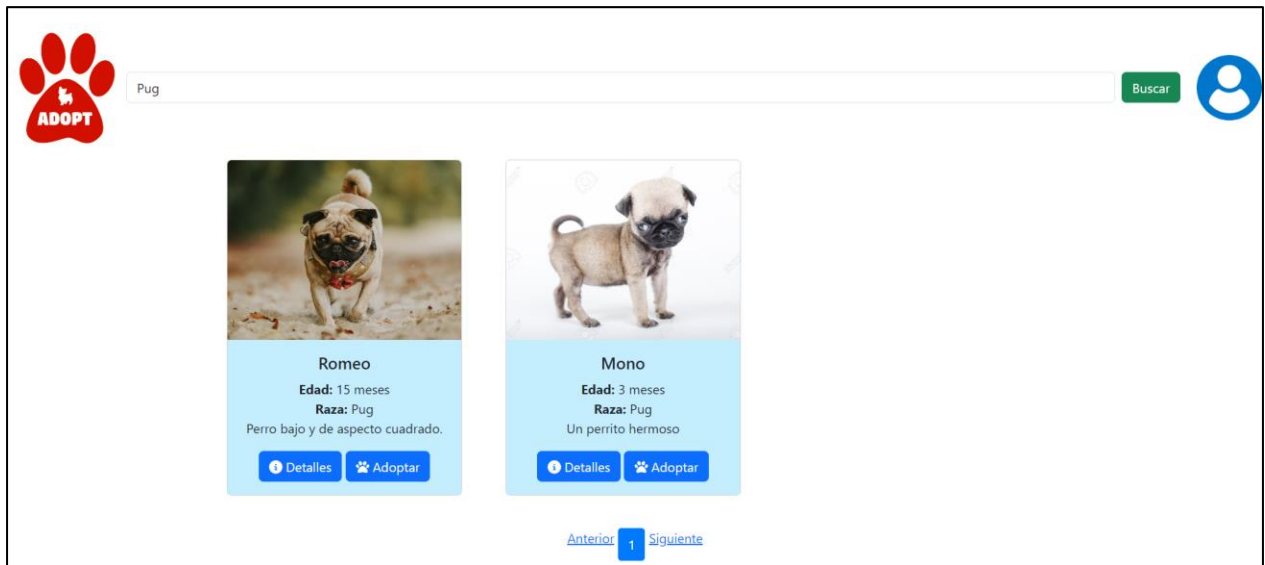


**Nina**  
**Edad:** 1 año  
**Raza:** Pincher  
Perra pequeña

[Detalles](#) [Adoptar](#)

[Anterior](#) [1](#) [Siguiente](#)

Igualmente al buscar por raza:



Al darle este boton:



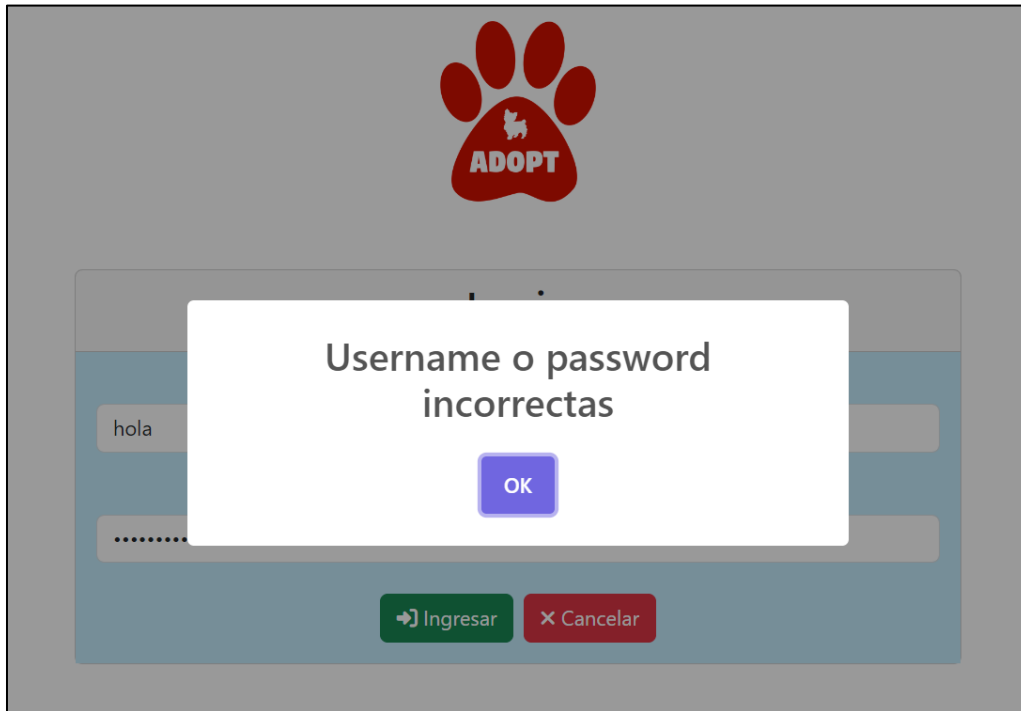
Se despliega esta pantalla del login:

A login screen design. At the top center is a large red paw print logo with the word 'ADOPT' in white. Below the logo is a light gray header bar with the word 'Login' in bold black text. The main content area has a light blue background. It contains two white input fields. The first field is labeled 'Username:' and the second is labeled 'Password:'. At the bottom of the form are two buttons: a green button with a white right-pointing arrow and the text 'Ingresar', and a red button with a white 'X' and the text 'Cancelar'.

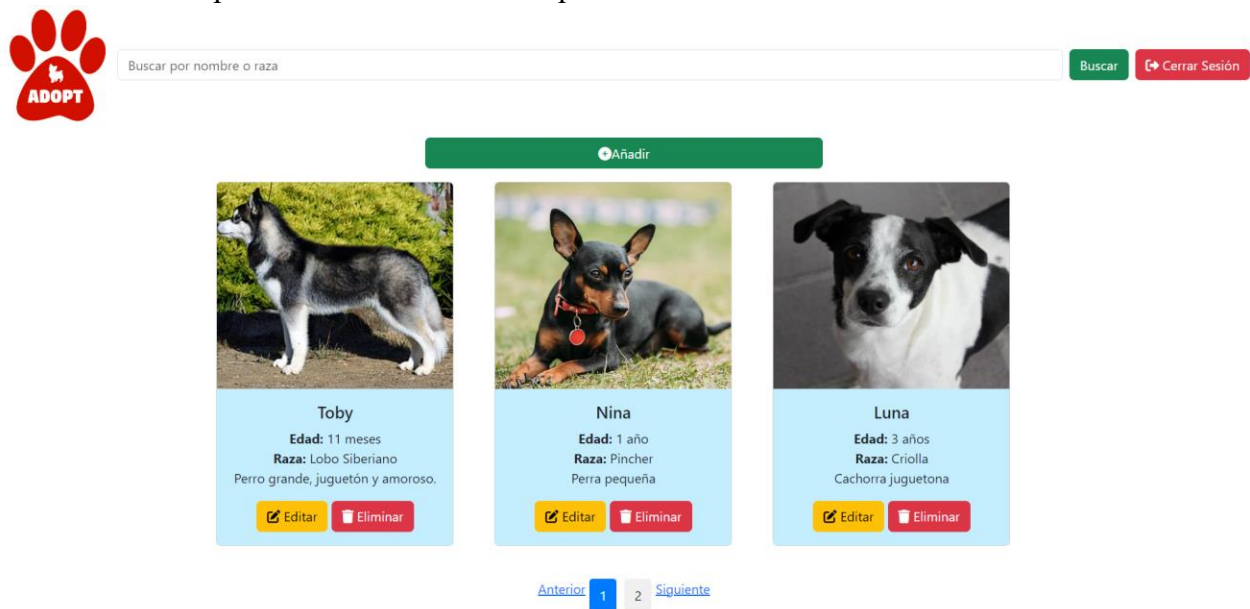


Al darle en el boton **cancelar** regresa a la pagina principal es decir a la pagina de la captura anterior.

Al darle al boton **ingresar** estando las credenciales (username o password) incorrectas arrojará esta alerta:



Al darle al boton **ingresar** estando las credenciales (username o password) correctas se redirige a la pantalla del admin donde aparece el CRUD:



Al darle el boton **Añadir** se despliega el modal para llenar todos los datos de la mascota a registrar:

The screenshot shows the 'Registrar Mascota' modal form overlaid on the website. The form contains the following fields and options:

- Nombre:** Mono
- Edad:** 3 meses
- Raza:** Pug
- Imagen:** <https://previews.123rf.com/images/bernardbodo/bernarc>
- Descripción:** Un perrito hermoso
- Detalles:** rro, su historia, personalidad, alimentación y mucho más.

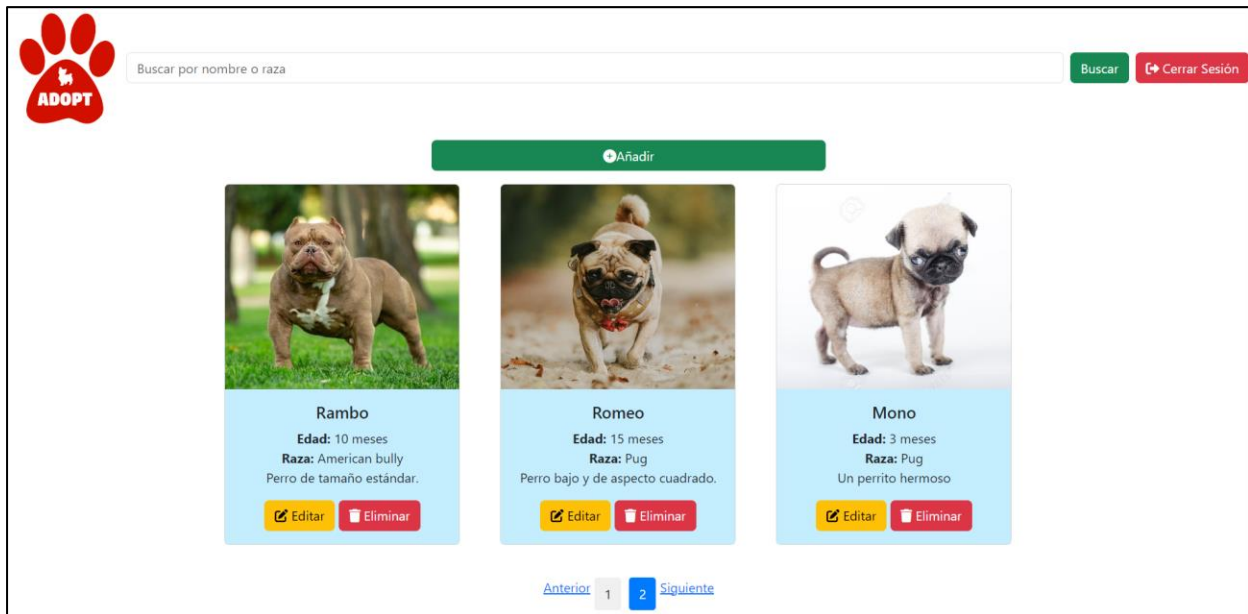
Buttons: **Guardar** (green), **Cerrar** (grey).

Background content (visible through the modal):


- Toby:** Edad: 11 meses, Raza: Lobo Siberiano, Perro grande, juguetón y amoroso.
- Luna:** Edad: 3 años, Raza: Criolla, Cachorra juguetona.

Navigation: [Anterior](#) 1 2 [Siguiente](#)

Aquí se va la mascota ya registrada:




Evidencia del buscador por nombre de la mascota:



[Buscar](#) [Cerrar Sesión](#)


[+ Añadir](#)



**Luna**  
**Edad:** 3 años  
**Raza:** Criolla  
Cachorra juguetona


[✎ Editar](#) [🗑 Eliminar](#)

[Anterior](#) [1](#) [Siguiente](#)



[Buscar](#) [Cerrar Sesión](#)

[+ Añadir](#)




**Toby**  
**Edad:** 11 meses  
**Raza:** Lobo Siberiano  
Perro grande, juguetón y amoroso.

[✎ Editar](#) [🗑 Eliminar](#)

[Anterior](#) [1](#) [Siguiente](#)

Busqueda por raza con su respectiva paginación:




Pincher

Buscar

Cerrar Sesión

Añadir




**Nina**  
Edad: 1 año  
Raza: Pincher  
Perra pequeña

Editar

Eliminar

[Anterior](#) [1](#) [Siguiente](#)




Pug

Buscar

Cerrar Sesión


Añadir



**Romeo**  
Edad: 15 meses  
Raza: Pug  
Perro bajo y de aspecto cuadrado.

Editar

Eliminar



**Mono**  
Edad: 3 meses  
Raza: Pug  
Un perrito hermoso

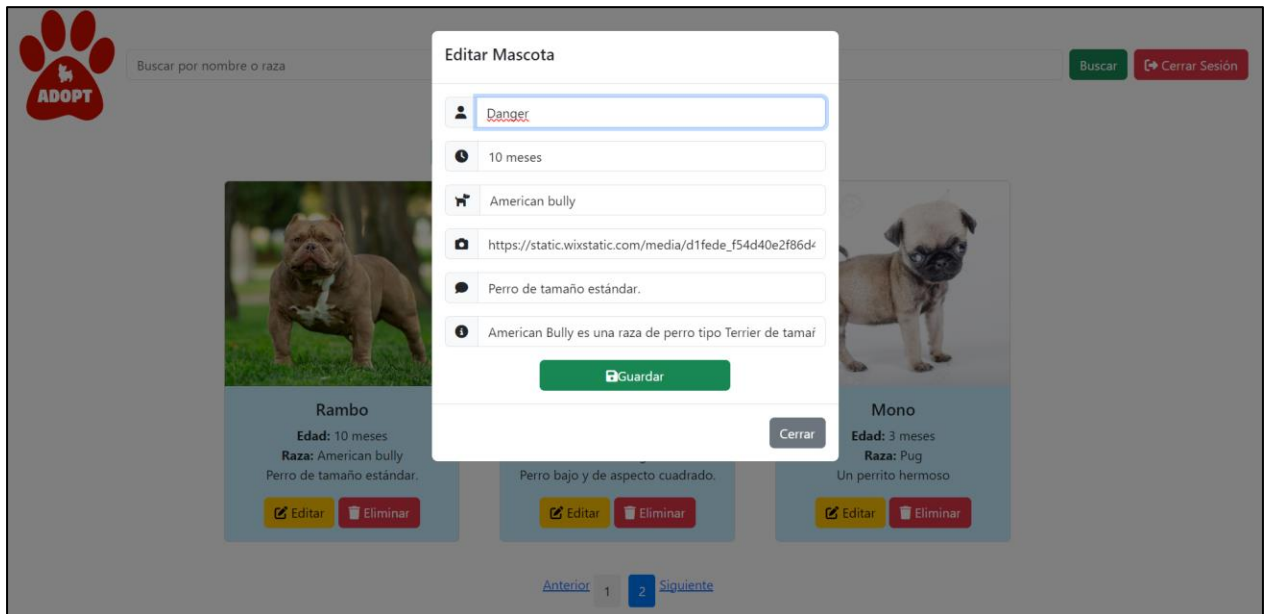
Editar

Eliminar

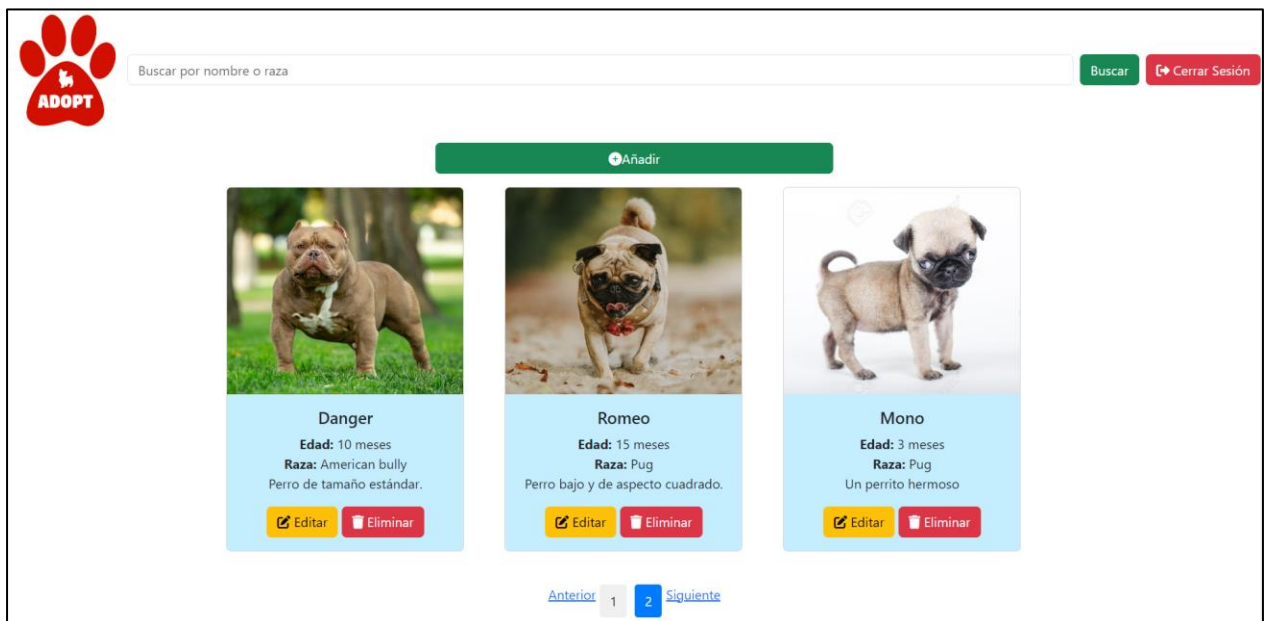
[Anterior](#) [1](#) [Siguiente](#)

Boton Editar:

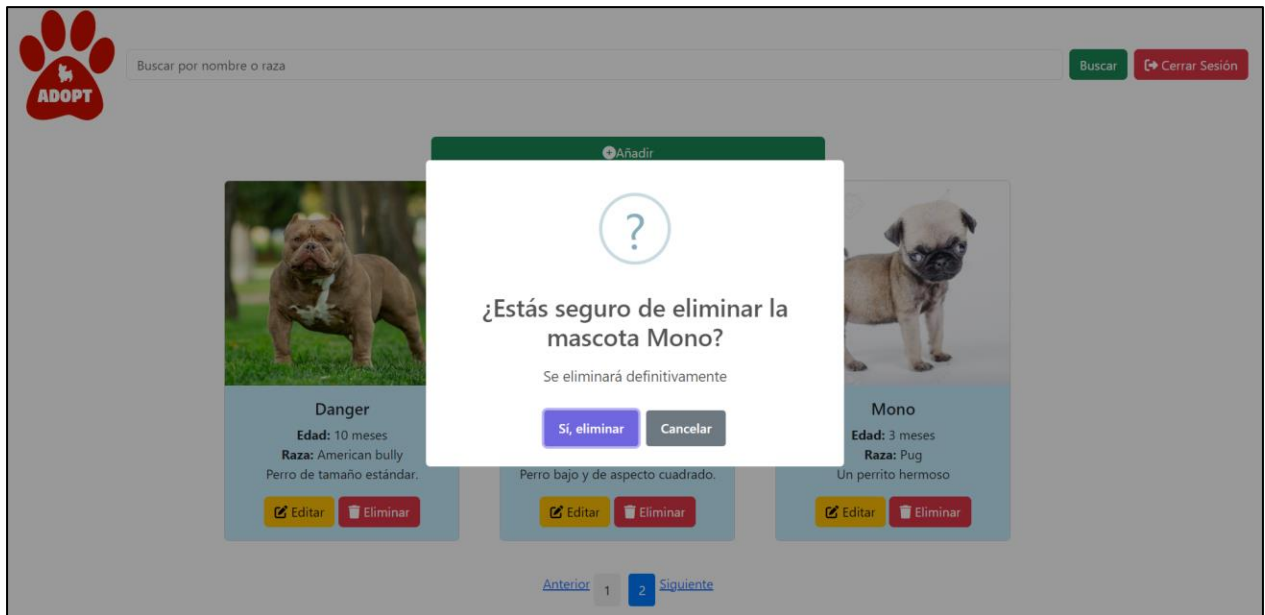
Vamos a cambiar el nombre de Rambo a Danger:



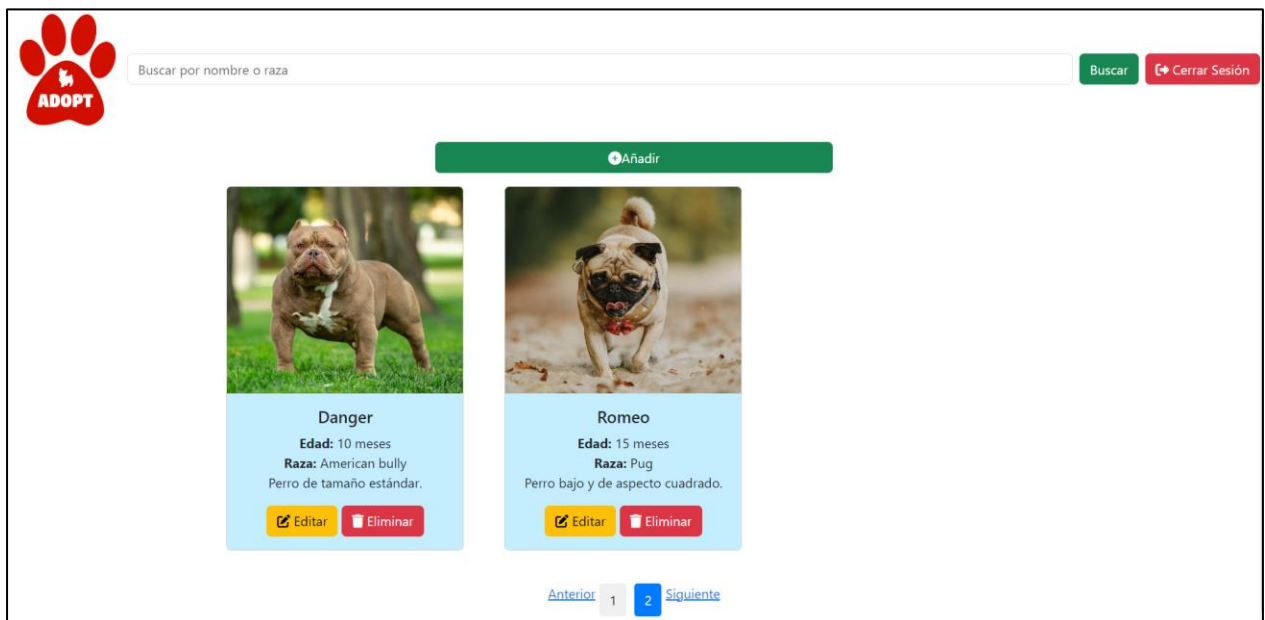
Aquí se visualiza el cambio:



Boton eliminar:



Se eliminó la mascota:



Al darle en el boton Cerrar Sesión hace un logout y vuelve a la pagina de inicio:

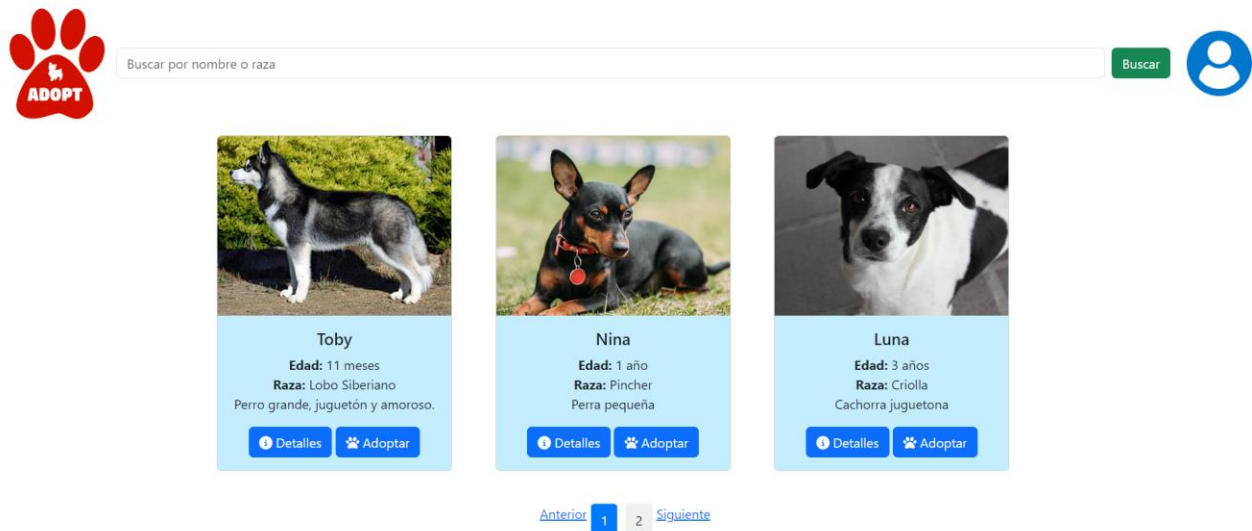


Tabla **usuarios** de la base de datos:

usuarios   Enter a SQL expression to filter results (use Ctrl+Space)					
Grilla	id_user	username	password	createdAt	updatedAt
1	1	admin	admin123	[NULL]	[NULL]

Tabla **mascotas** de la base de datos:

mascotas   Enter a SQL expression to filter results (use Ctrl+Space)									
Grilla	id	nombre	createdAt	updatedAt	raza	imagen	descripcion1	descripcion2	
1	10	Toby	2023-12-29 20:26:23	2023-12-30 17:49:04	Lobo Siberiano	https://upload.wikimedia.org/wiki	Perro grande, juguetón y amoroso.	El perro lobo checoslovaco (tambi	
2	11	Nina	2023-12-29 20:38:23	2023-12-30 18:00:26	Pincher	https://images.hola.com/imagen	Perra pequeña	El pinscher miniatura es una raza c	
3	13	Luna	2023-12-30 02:19:58	2023-12-30 18:01:48	Criolla	https://www.adopta.mx/wp-conte	Cachorra juguetona	Se conoce como perro criollo o m	
4	18	Danger	2023-12-30 17:23:34	2024-01-03 22:21:44	American bully	https://static.wixstatic.com/media	Perro de tamaño estándar.	American Bully es una raza de peri	
5	22	Romeo	2024-01-03 03:06:49	2024-01-03 03:08:30	Pug	https://cdn.wamiz.fr/cdn-cgi/ima	Perro bajo y de aspecto cuadrado.	El pug1 (también conocido como	

Tabla **datos\_adopciones** de la base de datos:

datos_adopciones   Enter a SQL expression to filter results (use Ctrl+Space)						
Grilla	id	nombre	telefono	id_mascota	createdAt	updatedAt
1	7	Sandra	7.256.036	13	2024-01-02 19:55:13	2024-01-02 19:55:13
2	9	Darwin Diaz	777.628	10	2024-01-03 03:32:28	2024-01-03 03:32:28