

Universidad de Nariño.

Ingeniería de Sistemas.

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

Taller Unidad 3 Frontend

Nombre: Fredilton Daniel Getial Torres

Código: 219036091

Teléfono: 3187863224

1. Uso del Backend desarrollado en el taller anterior(algunas implantaciones, como la de imagen).

Subir imagen al servidor desde el equipo

```
import { mascotas } from "../modelos/mascotasModelo.js";
import { solicitudes } from "../modelos/solicitudesModelo.js";

// Importar la biblioteca 'multer' para manejar la carga de archivos
import multer from 'multer';

// Configuración del almacenamiento de archivos usando multer
const storage = multer.diskStorage({
  // Definir el directorio de destino para almacenar los archivos
  destination: function (req, file, cb) {
    cb(null, 'imagenes'); // La carpeta 'imagenes' debe existir en el servidor
  },
  // Definir el nombre del archivo en el servidor como el nombre original del archivo
  filename: function (req, file, cb) {
    cb(null, file.originalname);
  },
});

const upload = multer({ storage: storage });

// Configurar Express para servir archivos estáticos desde la carpeta 'imagenes'
app.use('/imagenes', express.static('imagenes'));

// Ruta POST para crear una nueva mascota con la imagen asociada
routerMascotas.post('/crearMascota', upload.single('imagen'), (req, res) => {
  crearMascota(req, res);
});
```

Agregar nuevos campos (imagen, raza, detalle)

```
const crearMascota = (req, res) => {
  // Verificar si el nombre, tipo, edad y id_mascota están presentes y no son nulos
  if (!req.body.id_mascota || !req.body.nombre || !req.body.tipo) {
    return res.status(400).json({
      tipo: "success",
      mensaje: "id_mascota, nombre, tipo y edad son campos requeridos y no pueden ser nulos."
    });
  }

  if (!req.file) {
    return res.status(400).json({
      tipo: 'error',
      mensaje: 'Debes cargar una imagen para la mascota.',
    });
  }

  // Obtener el nombre del archivo de imagen cargado desde la solicitud POST
  const imagen = req.file.filename;

  // Crear un objeto dataset con los campos relevantes
  const dataset = {
    id_mascota: req.body.id_mascota,
    nombre: req.body.nombre,
    tipo: req.body.tipo,
    edad: req.body.edad,
    descripcion: req.body.descripcion,
    disponible: req.body.disponible,
    imagen: imagen,
    raza: req.body.raza,
    detalle: req.body.detalle
  };

  // Utilizar Sequelize para crear el recurso
  mascotas.create(dataset)
    .then((resultado) => {
      res.status(200).json({
        tipo: "success",
        mensaje: "Registro creado correctamente",
        resultado: resultado
      });
    })
}
```

2. Creación de componentes y métodos asociados a los mismos.

Componente de administrador (ruta oculta, y modificado lo mirado en clases)

Importaciones y variables de estado

```
// Importar las bibliotecas y componentes necesarios
import React, { useEffect, useState } from "react";
import axios from "axios";
import { mostrarAlerta } from "../functions.js";
import Swal from 'sweetalert2';
import withReactContent from "sweetalert2-react-content";

// Definir el componente principal llamado MascotasComponent
const MascotasComponent = () => {
  // Definir la URL de la API para obtener datos
  const url = "http://localhost:8000/mascotas";

  // Definir un array de tipos de mascotas
  const tiposMascotas = ['gato', 'perro'];




  // Definir variables de estado utilizando
  const [mascotas, setMascotas] = useState([]); // Estado para almacenar la lista de mascotas
  const [id_mascota, setId] = useState(""); // Estado para almacenar el ID de una mascota
  const [nombre, setNombre] = useState(""); // Estado para almacenar el nombre de una mascota
  const [tipo, setTipo] = useState(""); // Estado para almacenar el tipo de una mascota
  const [edad, setEdad] = useState(""); // Estado para almacenar la edad de una mascota
  const [descripcion, setDescripcion] = useState(""); // Estado para almacenar la descripción de una mascota
  const [disponible, setDisponible] = useState(""); // Estado para almacenar la disponibilidad de una mascota
  const [imagen, setImagen] = useState(""); // Estado para almacenar la URL de la imagen de una mascota
  const [raza, setRaza] = useState(""); // Estado para almacenar la raza de una mascota
  const [detalle, setDetalle] = useState(""); // Estado para almacenar detalles adicionales de una mascota
  const [operacion, setOperacion] = useState(""); // Estado para almacenar la operación actual (por ejemplo, "agregar", "editar")
  const [titulo, setTitulo] = useState(""); // Estado para almacenar el título del modal/dialogo
}
```

Mostrar mascotas

```
// Cargar datos de mascotas al montar el componente
useEffect(() => {
  getMascotas();
}, []);

// Definir la función asincrónica getMascotas
const getMascotas = async () => {
  try {
    // Realizar una solicitud GET a la API para buscar mascotas
    const respuesta = await axios.get(`${url}/buscarMascota`);
    // Actualizar el estado de mascotas con los datos de la respuesta
    setMascotas(respuesta.data);
  } catch (error) {
    console.error("Error al obtener mascotas:", error);
  }
};
```

➕ Añadir

 <p>Lucas ID: 1</p> <p>Tipo: perro</p> <p>raza: Pastor Aleman</p> <p>Edad: 8 meses</p> <p>Descripción: Perro muy tranquilo pero le gusta mucho jugar</p> <p>Disponible: Sí</p> <div style="display: flex; justify-content: center; gap: 10px; margin-top: 10px;"> ✎ Editar 🗑 Eliminar </div>	 <p>Mateo ID: 2</p> <p>Tipo: perro</p> <p>raza: Chihuahua</p> <p>Edad: 4 meses</p> <p>Descripción: pequeño en tamaño, grande en corazón</p> <p>Disponible: Sí</p> <div style="display: flex; justify-content: center; gap: 10px; margin-top: 10px;"> ✎ Editar 🗑 Eliminar </div>	 <p>Michi ID: 3</p> <p>Tipo: gato</p> <p>raza: Persa</p> <p>Edad: 2 años</p> <p>Descripción: Es un gato muy dormilon y a la vez muy travieso</p> <p>Disponible: Sí</p> <div style="display: flex; justify-content: center; gap: 10px; margin-top: 10px;"> ✎ Editar 🗑 Eliminar </div>
---	---	---

Funciones abrir modal (agregar y editar)

```
// Función para abrir el modal y configurar los estados según la opción seleccionada
const openModal = (opcion, id_mascota, nombre, tipo, edad, descripcion, disponible, imagen, raza, detalle) => {
  // Reiniciar todos los estados
  setId('');
  setNombre('');
  setTipo('');
  setEdad('');
  setDescripcion('');
  setDisponible('');
  setImagen('');
  setRaza('');
  setDetalle('');
  // Establecer la operación actual
  setOperacion(opcion);
  // Configurar el título del modal según la opción
  if (opcion === 1) {
    setTitulo("Registrar Mascota");
  } else if (opcion === 2) {
    setTitulo("Editar Mascota");
    // Configurar los estados con los valores de la mascota seleccionada para editar
    setId(id_mascota);
    setNombre(nombre);
    setTipo(tipo);
    setEdad(edad);
    setDescripcion(descripcion);
    setDisponible(disponible);
    setImagen(imagen);
    setRaza(raza);
    setDetalle(detalle);
  }
};
```

Validar datos

```
// Función para validar y procesar la información de la mascota
const validar = () => {
  let parametros;
  let metodo;

  // Validar cada uno de los datos de mascota
  if (!id_mascota) {
    console.log("Debe escribir una Id");
    mostrarAlerta("Debe escribir una Id");
  }
}
```


Registrar Mascota



Debe escribir una Id

OK



 Guardar

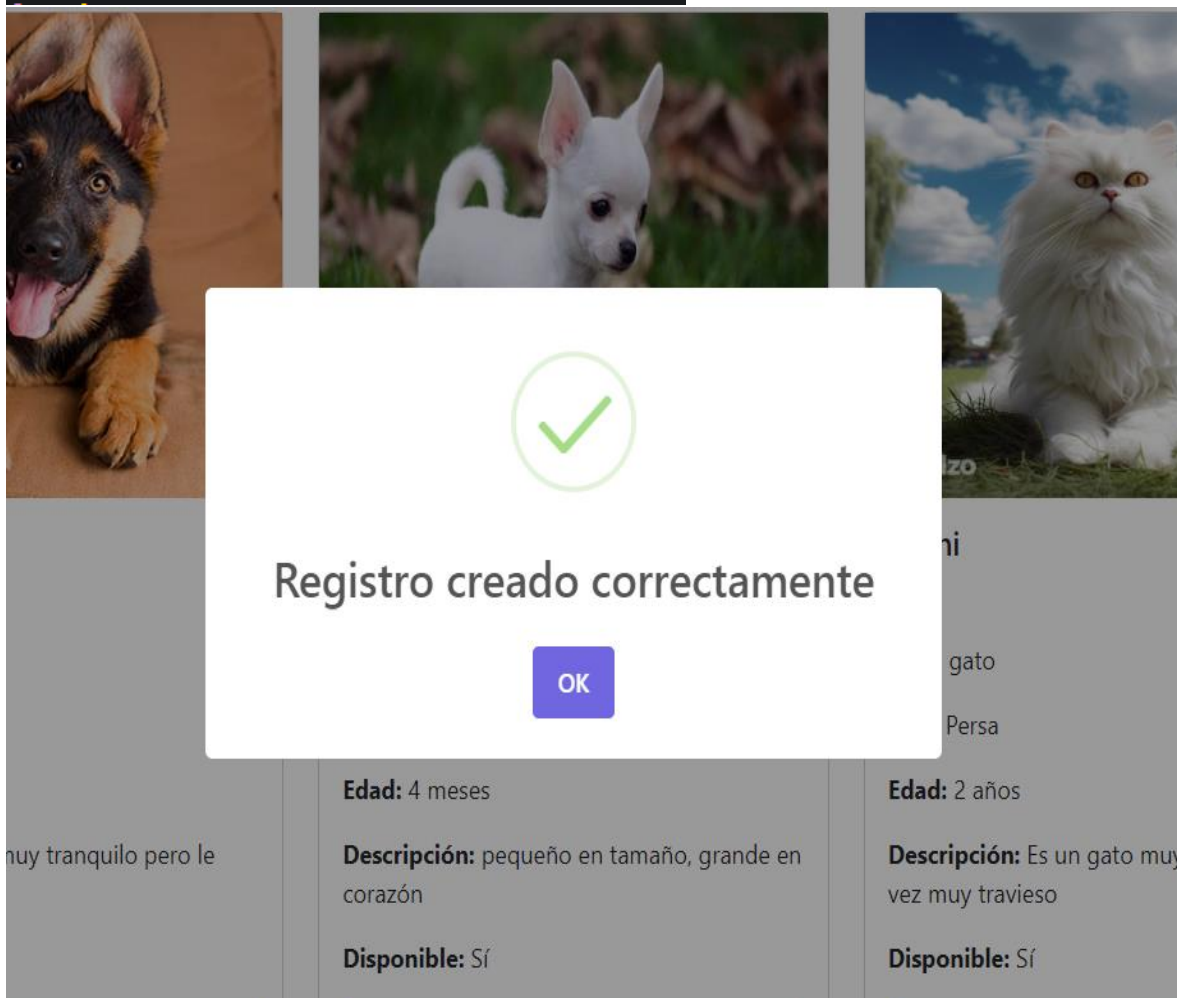
Cerrar

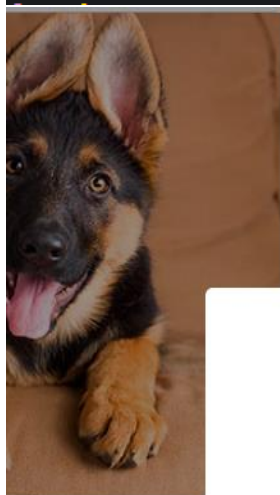
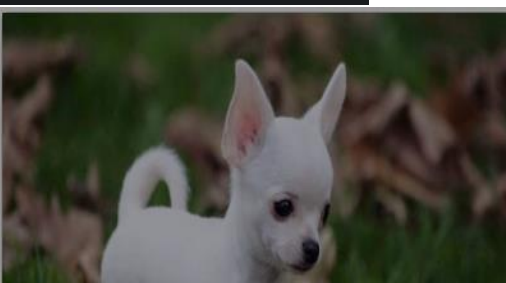

Configurar datos

```
// Configurar los datos para la solicitud según la operación (Registrar o Editar)
if (operacion === 1) {
  const formData = new FormData();
  formData.append('id_mascota', (method) String.trim(): string
  formData.append('nombre', nom
  formData.append('tipo', tipo. Removes the leading and trailing white space and line terminator characters from a string.
  formData.append('edad', edad.trim());
  formData.append('descripcion', descripcion.trim());
  formData.append('disponible', disponible.trim());
  formData.append('imagen', imagen);
  formData.append('raza', raza.trim());
  formData.append('detalle', detalle.trim());
}
```

Método POST (agregar)

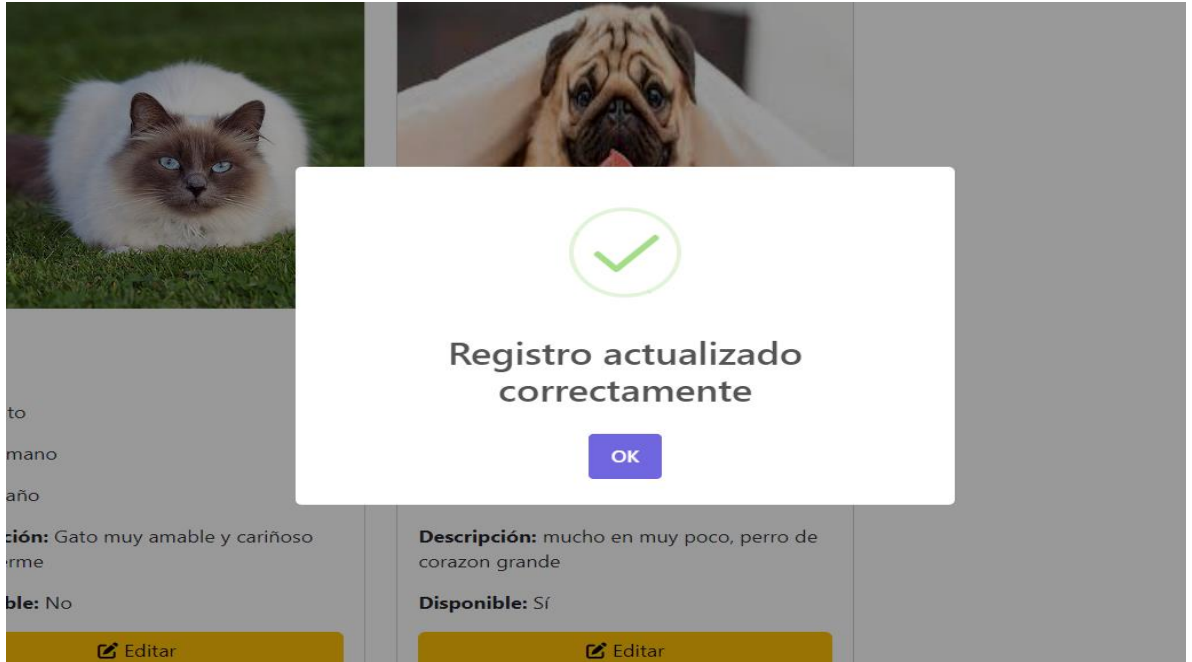
```
parametros = {
  urlExt: `${url}/crearMascota`,
  data: formData,
};
metodo = "POST";
```



		
<div>Registro creado correctamente</div> <div>OK</div>		
huy tranquilo pero le	Edad: 4 meses Descripción: pequeño en tamaño, grande en corazón Disponible: Sí	Edad: 2 años Descripción: Es un gato muy vez muy travieso Disponible: Sí

Método PUT (actualizar)

```
parametros = {  
  urlExt: `${url}/actualizarMascota/${id_mascota}`,  
  data: formData,  
};  
metodo = "PUT";
```



Función enviar solicitud

```
// Función para enviar solicitudes al servidor  
const enviarSolicitud = async (metodo, parametros) => {  
  try {  
    // Realizar la solicitud utilizando axios  
    const response = await axios({  
      method: metodo,  
      url: parametros.urlExt,  
      data: parametros.data,  
      headers: {  
        'Content-Type': 'multipart/form-data',  
      },  
    });  
  
    // Extraer el tipo y mensaje de la respuesta  
    let tipo = response.data.tipo;  
    let mensaje = response.data.mensaje;  
  
    // Mostrar una alerta con el mensaje y tipo obtenidos  
    mostrarAlerta(mensaje, tipo);  
  
    // Si la solicitud es exitosa, cerrar el modal y actualizar la lista de mascotas  
    if (tipo === "success") {  
      document.getElementById("btnCerrarModal").click();  
      getMascotas();  
    }  
  } catch (error) {  
    // Manejar errores, mostrar alerta y log en consola  
    mostrarAlerta(`Error en la solicitud`, error);  
    console.log('Error en la solicitud:', error.response);  
  }  
};
```

Función eliminar registro

```
// Función para confirmar y eliminar una mascota
const eliminarMascota = (id, nombre) => {
  const MySwal = withReactContent(Swal);

  // Mostrar un cuadro de diálogo para confirmar la eliminación
  MySwal.fire({
    title: `¿Estás seguro de eliminar la mascota ${nombre}?`,
    icon: 'question',
    text: 'Se eliminará definitivamente',
    showCancelButton: true,
    confirmButtonText: 'Sí, eliminar',
    cancelButtonText: 'Cancelar'
  }).then((result) => {
    // Verificar la respuesta del usuario
    if (result.isConfirmed) {
      // Configurar el estado de ID y enviar una solicitud DELETE al servidor
      setId(id);
      enviarSolicitud("DELETE", { urlExt: `${url}/eliminarMascota/${id}`, id: id });
    } else {
      // Mostrar una alerta en caso de que el usuario cancele la eliminación
      mostrarAlerta("No se eliminó la mascota", "info");
    }
  });
};
```





No se eliminó la mascota

OK

Edad: 1 años

Descripción: mucho en muy poco, perro de corazon grande



Registro Eliminado

OK

able y cariñoso

Creación de HTML(visualizar en carts)

```
<div className="row mt-3">
  <div className="col-12 col-lg-8 offset-0 offset-lg-2">
    <div className="row row-cols-1 row-cols-md-3 g-4">
      {mascotas.map((mascota, i) => (
        <div key={mascota.id_mascota} className="col">
          <div className="card">
            <img
              src={`http://localhost:8000/imagenes/${mascota.imagen}`}
              alt={`Imagen de ${mascota.nombre}`}
              style={{ width: '312px', height: '300px' }}
            />
            <div className="card-body">
              <h5 className="card-title">{mascota.nombre}</h5>
              <p className="card-text">
                <strong>ID:</strong> {mascota.id_mascota}
              </p>
            </div>
          </div>
        )
      )}
```


Nota: así para todos los registros


Creación de HTML(modal con campos)


```
<div id="modalMascotas" className="modal fade" aria-hidden="true">
  <div className="modal-dialog">
    <div className="modal-content">
      <div className="modal-header">
        <label className="h5">{titulo}</label>
      </div>
      <div className="modal-body">
        <input type="hidden" id="id"></input>
        <div className="input-group mb-3">
          <span className="input-group-text">
            <i className="fa-solid fa-gift"></i>
          </span>
          <input
            type="text"
            id="id_mascota"
            className="form-control"
            placeholder="ID"
            value={id_mascota}
            onChange={(e) => setId(e.target.value)}
            readOnly={operacion === 2} // readOnly solo cuando la operación es editar
            disabled={operacion === 2} // Deshabilitar el campo
          ></input>
        </div>
      </div>
    </div>
  </div>
```


Nota: así para todos los campos


Editar Mascota





















Seleccionar archivo

Carlino.jpeg



Guardar

Cerrar

Componente de mascotas (ruta principal)

Importar componentes necesarios y variables de estado

```
// Importar las bibliotecas y componentes necesarios
import React, { useEffect, useState } from "react";
import axios from "axios";
import { Link } from 'react-router-dom';
import ReactPaginate from "react-paginate";

// Definir el componente principal llamado MascotasComponent
const MascotasComponent = () => {
  // URL de la API para obtener datos de mascotas
  const url = "http://localhost:8000/mascotas";
  // Estados para almacenar la lista de mascotas, la página actual y la cantidad de mascotas por página
  const [mascotas, setMascotas] = useState([]);
  const [currentPage, setCurrentPage] = useState(0);
  const mascotasPerPage = 3;

```

Función mostrar mascotas

```
// Efecto secundario para obtener mascotas al montar el componente
useEffect(() => {
  getMascotas();
}, []);


// Función para obtener mascotas desde la API
const getMascotas = async () => {
  const respuesta = await axios.get(`${url}/buscarMascota`);
  setMascotas(respuesta.data);
};

```

html

```
<div className="row mt-3">
  <div className="col-12 col-lg-8 offset-0 offset-lg-2">
    <div className="row row-cols-1 row-cols-md-3 g-4">
      {currentMascotas.map((mascota) => (
        <div key={mascota.id_mascota} className="col">
          <div className="card">
            <h5 className="card-title text-center">{mascota.nombre}</h5>
            <img
              src={`http://localhost:8000/imagenes/${mascota.imagen}`}
              alt={`Imagen de ${mascota.nombre}`}
              style={{ width: '100%', height: '300px' }}
            />
            <div className="card-body">
              <p className="card-text mb-1">
                <strong>Tipo:</strong> {mascota.tipo}
              </p>
              <p className="card-text mb-1">
                <strong>Edad:</strong> {mascota.edad}
              </p>
              <p className="card-text mb-1">
                <strong>Descripción:</strong> {mascota.descripcion}
              </p>
              <p className="card-text mb-1">
                <strong>Disponible:</strong>{" "}
                {mascota.disponible ? "Sí" : "No"}
              </p>
            </div>
          </div>
        )
      )}
    </div>
  </div>
</div>
```

Lucas



Tipo: perro
Edad: 8 meses
Descripción: Perro muy tranquilo pero le gusta mucho jugar
Disponible: Sí

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


Mateo



Tipo: perro
Edad: 4 meses
Descripción: pequeño en tamaño, grande en corazón
Disponible: Sí

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

Michi



Tipo: gato
Edad: 2 años
Descripción: Es un gato muy dormilón y a la vez muy travieso
Disponible: Sí

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

Redirección de botones

```
<div className="d-flex justify-content-between ">
  <Link to={`/detalles/${mascota.id_mascota}`} className="btn btn-info">
    <i className="fa-solid fa-eye"></i> Detalles
  </Link>
  <Link to={`/adoptar/${mascota.id_mascota}`} className="btn btn-success">
    <i className="fa-solid fa-heart"></i> Adoptar
  </Link>
</div>
```

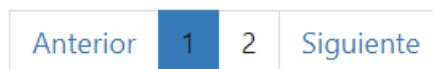
Funciones para paginación

```
// Función para manejar el cambio de página en la paginación
const handleClick = (data) => {
  setCurrentPage(data.selected);
};

// Calcular el índice de inicio de las mascotas en la página actual
const offset = currentPage * mascotasPerPage;
// Crear un subconjunto de mascotas correspondiente a la página actual
const currentMascotas = mascotas.slice(offset, offset + mascotasPerPage);
```

Html

```
<div className="pagination-container text-center mt-4">
  <ReactPaginate
    previousLabel={"Anterior"}
    nextLabel={"Siguiete"}
    breakLabel={"..."}
    pageCount={Math.ceil(mascotas.length / mascotasPerPage)}
    marginPagesDisplayed={2}
    pageRangeDisplayed={5}
    onPageChange={handlePageClick}
    containerClassName={"pagination"}
    subContainerClassName={"pages pagination"}
    activeClassName={"active"}
  />
</div>
```



Html (logo)


```
<div className="container-fluid">
  <div style={{marginLeft: '100px'}}>
    
  </div>
</div>
```



Final




Lucas



Tipo: perro
Edad: 8 meses
Descripción: Perro muy tranquilo pero le gusta mucho jugar
Disponible: Sí

[👁️ Detalles](#)[❤️ Adoptar](#)


Mateo



Tipo: perro
Edad: 4 meses
Descripción: pequeño en tamaño, grande en corazón
Disponible: Sí

[👁️ Detalles](#)[❤️ Adoptar](#)

Michi



Tipo: gato
Edad: 2 años
Descripción: Es un gato muy dormilón y a la vez muy travieso
Disponible: Sí

[👁️ Detalles](#)[❤️ Adoptar](#)

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

Componente de detalles (al presionar el botón detalles)

Importar componentes necesarios y variables de estado

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { useParams, useNavigate } from "react-router-dom";

const DetallesMascota = () => {
  // Obtener el parámetro 'id' de la URL
  const { id } = useParams();

  // Configurar la navegación
  const navigate = useNavigate();

  // Estado para almacenar la información de la mascota y controlar la carga
  const [mascota, setMascota] = useState(null);
  const [loading, setLoading] = useState(true);
}
```

Función mostrar mascotas por Id

```
// Efecto que se ejecuta al montar el componente y cuando 'id' cambia
useEffect(() => {
  const getMascotaDetalles = async () => {
    try {
      // Realizar solicitud GET para obtener detalles de la mascota
      const respuesta = await axios.get(`http://localhost:8000/mascotas/buscarMascota/${id}`);

      // Almacenar detalles en el estado
      setMascota(respuesta.data);
    } catch (error) {
      // Manejar errores durante la obtención de datos
      console.error("Error al obtener detalles de la mascota", error);
    } finally {
      // Establecer el estado de carga como 'false' independientemente del resultado
      setLoading(false);
    }
  }
});
```

Html

```
// Renderizar la información de la mascota en una estructura de tarjeta
return (
  <div className="App">
    <div className="container">
      <div style={{ marginLeft: '100px' }}>
        
      </div>
      <div className="row mt-3">
        <div className="col-12 col-lg-8 offset-0 offset-lg-2">
          <div className="card">
            <div className="card-body">
              <h5 className="card-title">{mascota.nombre}{mascota.raza || 'Raza no disponible'}</h5>
              <br><br>
              <p className="card-text">
                <strong>Detalle:</strong> {mascota.detalle || 'Detalle no disponible'}
              </p>
              <br><br>
              <div className="card-footer text-right d-flex justify-content-end p-3">
                <button className="btn btn-primary" onClick={handleRegresar}>
                  Regresar
                </button>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
)
```

final



Lucas(Pastor Aleman)

Detalle: Los Pastores alemanes son una de las razas más reconocibles del mundo. Tienen un aspecto musculoso y atento y una actitud noble y distante. Son perros ágiles y bien equilibrados de porte orgulloso. Su pelaje puede presentar varios colores (consulta el estándar de la raza) y están formados por un manto superior duro, áspero y lacio, y un manto inferior grueso. La altura ideal del macho adulto es de 63 cm y la de la hembra, 58 cm. El peso puede estar entre 30 y 36 kg.

[Regresar](#)

Componente de adopción (al presionar el botón adoptar)

Importar componentes necesarios y variables de estado

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useParams, useNavigate } from "react-router-dom";
import { mostrarAlerta } from "../functions.js";

// CUERPO COMPONENTE
const AdoptarComponent = () => {
  // URL base para las solicitudes HTTP
  const url = "http://localhost:8000/solicitudes";

  // Obtener el parámetro 'id' de la URL utilizando react-router-dom
  const { id: idMascotaSeleccionada } = useParams();

  // Configurar la navegación utilizando react-router-dom
  const navigate = useNavigate();

  // Estados para almacenar datos del formulario
  const [idMascota, setIdMascota] = useState("");
  const [nombreSolicitante, setNombreSolicitante] = useState("");
  const [correoSolicitante, setCorreoSolicitante] = useState("");

```

Función validar datos y cargar componte

```
// Efecto para establecer el ID de la mascota seleccionada al cargar el componente
useEffect(() => {
  setIdMascota(idMascotaSeleccionada);
}, [idMascotaSeleccionada]);

// Función para validar el formulario antes de enviar la solicitud
const validar = () => {
  if (nombreSolicitante.trim() === "") {
    mostrarAlerta("Debe ingresar un nombre de solicitante");
  } else if (correoSolicitante.trim() === "") {
    mostrarAlerta("Debe ingresar un correo de solicitante");
  } else {
    enviarSolicitud();
  }
};

```

Debe ingresar un nombre de
solicitante

OK

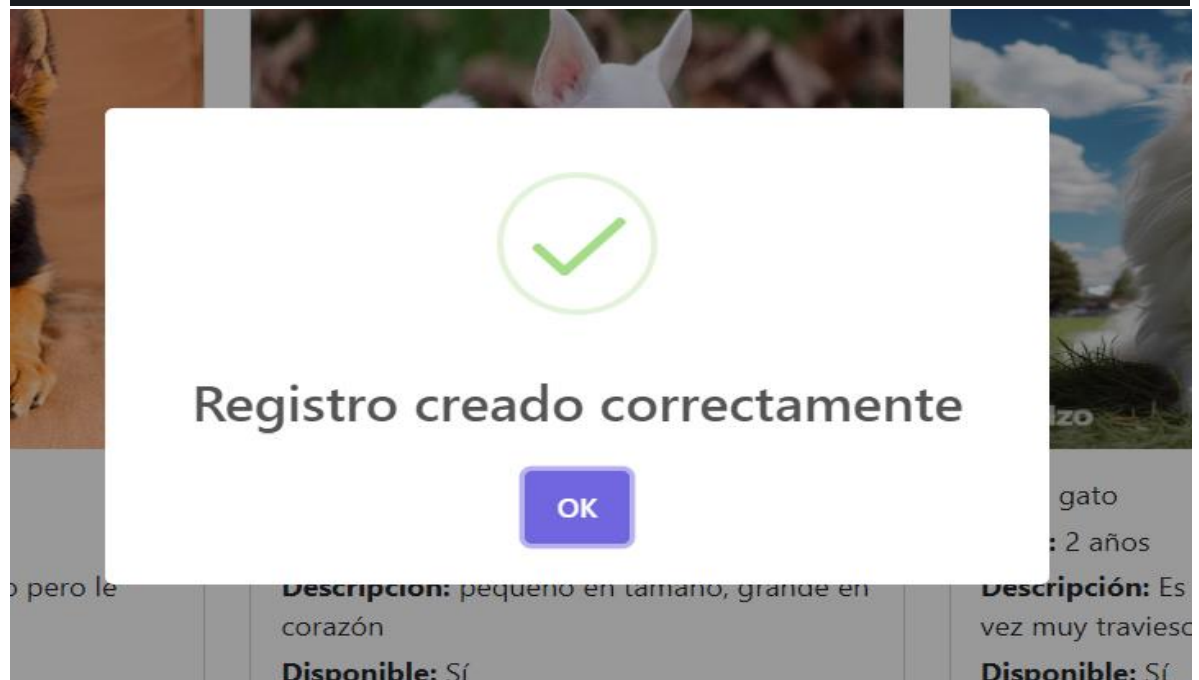
Función enviar solicitud

```
// Función para enviar la solicitud de adopción al servidor
const enviarSolicitud = async () => {
  try {
    // Realizar solicitud POST para crear una solicitud de adopción
    const response = await axios.post(`${url}/crearSolicitud`, {
      idMascota: idMascota.trim(),
      nombreSolicitante: nombreSolicitante.trim(),
      correoSolicitante: correoSolicitante.trim(),
    });

    // Determinar el tipo de alerta según el estado de la solicitud
    const tipo = response.data.resultado.estadoSolicitud === 'Pendiente' ? 'success' : 'error';
    const mensaje = response.data.mensaje;

    // Mostrar la alerta al usuario
    mostrarAlerta(mensaje, tipo);

    // Limpiar el formulario y regresar a la página principal en caso de éxito
    if (tipo === 'success') {
      limpiarFormulario();
      navigate('/');
    }
  } catch (error) {
    // Manejar diferentes tipos de errores durante la solicitud
    if (error.response) {
      mostrarAlerta(error.response.data.mensaje, 'error');
    } else if (error.request) {
      mostrarAlerta('No se recibió respuesta del servidor', 'error');
    } else {
      mostrarAlerta('Error en la configuración de la solicitud', 'error');
    }
  }
};
```





Html


```
// Renderizado del componente con el formulario de adopción
return (
  <div className="container">
    <h2>Formulario de Adopción</h2>
    <div className="input-group mb-3">
      <span className="input-group-text">
        <i className="fa-solid fa-id-card"></i>
      </span>
      <input
        type="text"
        className="form-control"
        placeholder="ID de Mascota"
        value={idMascota}
        disabled // Deshabilitar la edición del campo
      ></input>
    </div>
    <div className="input-group mb-3">
      <span className="input-group-text">
        <i className="fa-solid fa-user"></i>
      </span>
      <input
        type="text"
        className="form-control"
        placeholder="Nombre del Solicitante"
        value={nombreSolicitante}
        onChange={(e) => setNombreSolicitante(e.target.value)}
      ></input>
    </div>
  </div>
)
```

Final

Formulario de Adopción

 1


 Daniel










 daniel@gmail.com

♥ Adoptar

Regresar

Visualización en la base de datos

olicitudes |  Enter a SQL expression to filter results (use Ctrl+Space)

	 id_Solicitud	 idMascota	 nombreSolicitante	 correoSolicitante	 estadoSolicitud
1	1	2 	Fredy	Fredy@gmail.com	Pendiente
2	2	1 	Daniel	daniel@gmail.com	Pendiente
3	3	1 	Daniel	daniel@gmail.com	Pendiente
4	4	2 	Daniel	daniel@gmail.com	Pendiente

Conclusión

Al concluir el desarrollo Frontend en React para la Universidad de Nariño, experimenté un significativo avance en mis habilidades de programación web. La creación de componentes modulares y la implementación de métodos asociados permitieron una estructura eficiente de la aplicación, mejorando mi comprensión de la arquitectura de React. El uso de HTML5 y JavaScript contribuyó a un código claro y legible, mientras que la integración de estilos con Bootstrap proporcionó una interfaz atractiva y responsiva. La documentación detallada en el informe comunicó efectivamente las decisiones de diseño y la lógica de implementación, destacando mejoras y extensiones más allá de los requisitos. Este proyecto fortaleció tanto mis habilidades técnicas como la capacidad para abordar desafíos de desarrollo, preparándome para enfrentar proyectos futuros en el campo del desarrollo de software. La entrega en un repositorio remoto como GitHub subraya mi enfoque en la documentación y presentación efectiva de proyectos.