Universidad de Nariño

Ingeniería de Sistemas

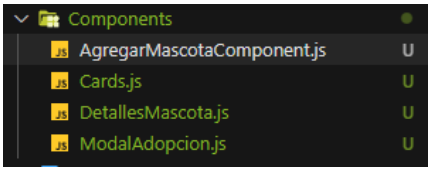
Diplomado de actualización en nuevas tecnologías para el desarrollo de Software

Vanessa Gustin – 219034187

Desarrollar una aplicación Frontend en React que haga uso del Backend desarrollado en el taller anterior.

1. Creación de componentes y métodos asociados a los mismos. (1.5 Ptos).

Creación de todos los componentes necesarios:



Métodos necesarios para el componente AgregarMascotasComponente:

```jsx
// Importaciones de módulos y componentes necesarios
import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import axios from 'axios';

// Definición del componente funcional AgregarMascotaComponent
const AgregarMascotaComponent = () => {
    // URL del servidor donde se gestionan las mascotas
    const url = "http://localhost:8000/mascotas";

    // Estados para almacenar los datos del formulario y mensajes de éxito/error
    const [nombre, setNombre] = useState('');
    const [raza, setRaza] = useState('');
    const [temperamento, setTemperamento] = useState('');
    const [descripcion, setDescripcion] = useState('');
    const [imagen, setImagen] = useState('');
    const [mensajeExito, setMensajeExito] = useState('');
    const [error, setError] = useState('');

    // Función para manejar el evento de agregar una mascota
    const handleAgregarMascota = async () => {
        try {
            // Realiza una solicitud POST al servidor con los datos del formulario
            const response = await axios.post(`${url}/agregar`, {
                nombre,
                raza,
                temperamento,
                descripcion,
                imagen,
            });

            // Verifica si la mascota se agregó con éxito (código de estado HTTP 201)
            if (response.status === 201) {
                // Actualiza el estado para mostrar un mensaje de éxito
                setMensajeExito('Mascota agregada exitosamente');
                // Limpia los campos del formulario después de agregar la mascota
                setNombre('');
                setRaza('');
                setTemperamento('');
                setDescripcion('');
                setImagen('');
            } else {
                // Si la solicitud no fue exitosa, muestra un mensaje de error
                setError('Error al agregar la mascota');
            }
        } catch (error) {
            // Manejo de errores: muestra un mensaje de error y registra detalles en la consola
            setError('Error al agregar la mascota');
            console.error('Error al agregar la mascota:', error);
        }
    };
```

Métodos necesarios para el componente Cards:

```jsx
// Importaciones de estilos y módulos necesarios
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle.min.js';
import React, { useEffect, useState } from "react";
import axios from 'axios';
import ModalAdopcion from "./ModalAdopcion";
import { Link } from 'react-router-dom';

// Definición del componente funcional MascotasComponent
const MascotasComponent = () => {
    // URLs de los endpoints del servidor
    const url = "http://localhost:8000/mascotas";
    const urlAdopcion = "http://localhost:8000/solicitudAdopcion";
    const urlAdminAuth = "http://localhost:8000/user/auth"; // Endpoint para autenticación del administrador

    // Estados para almacenar datos de mascotas, modal, estado de administrador y credenciales del administrador
    const [mascotas, setMascotas] = useState([]);
    const [id, setId] = useState('');
    const [showModal, setShowModal] = useState(false);
    const [isAdmin, setIsAdmin] = useState(false);
    const [adminUsername, setAdminUsername] = useState('');
    const [adminPassword, setAdminPassword] = useState('');
    const [error, setError] = useState('');

    // Efecto de montaje para obtener las mascotas al cargar el componente
    useEffect(() => {
        getMascotas();
    }, []);

    // Función para obtener las mascotas desde el servidor
    const getMascotas = async () => {
        try {
            const respuesta = await axios.get(`${url}/obtener`);
            setMascotas(respuesta.data);
        } catch (error) {
            console.error('Error al obtener mascotas:', error);
        }
    };

    // Función para manejar la adopción de una mascota
    const handleAdoptar = (mascotaId) => {
        setId(mascotaId);
        setShowModal(true);
    };

    // Función para cerrar el modal de adopción
    const handleCloseModal = () => {
        setShowModal(false);
    };
```

```
// Función para confirmar la adopción de una mascota
const handleConfirmAdopcion = async (nombreAdoptante, correoAdoptante) => {
    try {
        // Agrega la solicitud de adopción
        const responseAdopcion = await axios.post(`${urlAdopcion}/agregar`, {
            id_mascota: id,
            nombre_solicitante: nombreAdoptante,
            correo_solicitante: correoAdoptante
        });

        if (responseAdopcion.status === 201) {
            console.log('Adopción exitosa');

            // Elimina la mascota después de la adopción
            const responseEliminar = await axios.delete(`${url}/eliminar/${id}`);

            if (responseEliminar.status === 200) {
                console.log('Mascota eliminada');
                // Realiza acciones adicionales si es necesario
                getMascotas(); // Recargar la lista de mascotas después de la adopción y eliminación
            } else {
                console.log('Error al eliminar la mascota');
            }
        } else {
            console.log('Error al adoptar');
        }
    } catch (error) {
        console.error('Error en la adopción:', error);
    }

    // Cierra el modal
    setShowModal(false);
};

// Función para manejar el inicio de sesión del administrador
const handleAdminLogin = async () => {
    try {
        const response = await axios.post(urlAdminAuth, {
            usuario: adminUsername,
            contrasena: adminPassword
        });

        if (response.status === 200) {
            console.log('Autenticación exitosa');
            setIsAdmin(true);
        } else {
            console.log('Autenticación fallida');
            setError('Nombre de usuario o contraseña incorrectos');
        }
```

Métodos necesarios para el componente DetallesMascota:

```jsx
// IMPORTACIONES
import 'bootstrap/dist/css/bootstrap.min.css';  // Importa los estilos de Bootstrap
import React, { useState, useEffect } from 'react';  // Importa React, useState y useEffect
import { useParams, Link } from 'react-router-dom';  // Importa useParams y Link de react-router-dom
import axios from 'axios';  // Importa la biblioteca Axios para hacer solicitudes HTTP

// COMPONENTE FUNCIONAL DetallesMascota
const DetallesMascota = () => {
    // Extrae el parámetro 'id' de los parámetros de la URL
    const { id } = useParams();

    // Estado para almacenar los detalles de la mascota
    const [mascota, setMascota] = useState(null);

    // Efecto de montaje para obtener los detalles de la mascota según su ID
    useEffect(() => {
        // Función asincrónica para obtener detalles de la mascota
        const obtenerDetallesMascota = async () => {
            try {
                // Realiza una solicitud GET al servidor para obtener detalles de la mascota
                const respuesta = await axios.get(`http://localhost:8000/mascotas/obtener/${id}`);

                // Actualiza el estado con los detalles de la mascota
                setMascota(respuesta.data);
            } catch (error) {
                // Maneja errores en la consola si ocurren problemas al obtener detalles
                console.error('Error al obtener detalles de la mascota:', error);
            }
        };

        // Llama a la función para obtener detalles de la mascota
        obtenerDetallesMascota();
    }, [id]);  // El efecto se ejecuta cuando cambia el ID en los parámetros de la URL

```

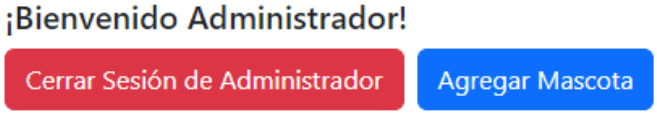
Métodos necesarios para el componente ModalAdopcion:

```jsx
// IMPORTACIONES
import React, { useState } from "react";  // Importa React y useState

// COMPONENTE FUNCIONAL ModalAdopcion
const ModalAdopcion = ({ showModal, handleClose, handleConfirm }) => {
    // Estados locales para el nombre y correo del adoptante
    const [nombreAdoptante, setNombreAdoptante] = useState("");
    const [correoAdoptante, setCorreoAdoptante] = useState("");

    // Función para manejar el clic en el botón de confirmar adopción
    const handleConfirmClick = () => {
        // Llama a la función handleConfirm del componente padre con los datos del adoptante
        handleConfirm(nombreAdoptante, correoAdoptante);

        // Reinicia los estados del nombre y correo del adoptante después de confirmar
        setNombreAdoptante("");
        setCorreoAdoptante("");
    };

```

2. Funcionalidad de Login y Búsqueda (1.5 Ptos)

```
85    // Función para manejar el inicio de sesión del administrador
86    const handleAdminLogin = async () => {
87        try {
88            const response = await axios.post(urlAdminAuth, {
89                usuario: adminUsername,
90                contrasena: adminPassword
91            });
92
93            if (response.status === 200) {
94                console.log('Autenticación exitosa');
95                setIsAdmin(true);
96            } else {
97                console.log('Autenticación fallida');
98                setError('Nombre de usuario o contraseña incorrectos');
99            }
100       } catch (error) {
101           console.error('Error en la autenticación:', error);
102           setError('Error en la autenticación');
103       }
104   };
105
106   // Función para cerrar la sesión del administrador
107   const handleAdminLogout = () => {
108       setIsAdmin(false);
109       setAdminUsername('');
110       setAdminPassword('');
111       setError('');
112   };
113
```

Solo el administrador puede agregar nuevas mascotas (Se debe crear el admin desde el Backend):



3. Uso de HTML 5 y JavaScript (Se debe desarrollar una estructura ordenada, con código legible y documentado). (1 Pto.).

Parte de la renderización de los componentes para evidenciar el uso de HTML y CSS con Bootstrap:

```jsx
// Renderización del componente
return (
    <div className="container mt-5">
        <h2>Agregar Nueva Mascota</h2>

        {/* Muestra mensajes de éxito y error */}
        {mensajeExito && (
            <div className="alert alert-success" role="alert">
                {mensajeExito}
            </div>
        )}
        {error && (
            <div className="alert alert-danger" role="alert">
                {error}
            </div>
        )}

        {/* Formulario para ingresar los datos de la nueva mascota */}
        <form>
            <div className="mb-3">
                <label htmlFor="nombre" className="form-label">Nombre:</label>
                <input
                    type="text"
                    className="form-control"
                    id="nombre"
                    value={nombre}
                    onChange={(e) => setNombre(e.target.value)}
                />
            </div>
            <div className="mb-3">
                <label htmlFor="raza" className="form-label">Raza:</label>
                <input
                    type="text"
                    className="form-control"
                    id="raza"
                    value={raza}
                    onChange={(e) => setRaza(e.target.value)}
                />
            </div>
            <div className="mb-3">
                <label htmlFor="temperamento" className="form-label">Temperamento:</label>
                <input
                    type="text"
                    className="form-control"
                    id="temperamento"
                    value={temperamento}
                    onChange={(e) => setTemperamento(e.target.value)}
```

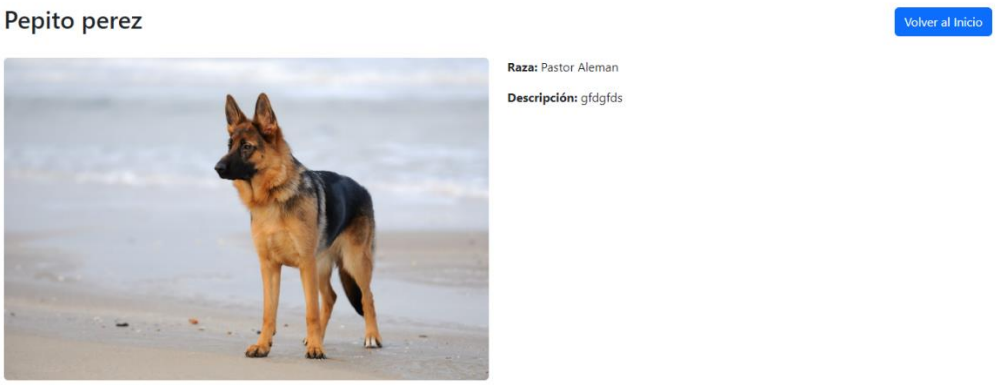
```jsx
    // Renderización del componente
    return (
        <div className="container mt-5">
            {/* Condicionalmente renderiza contenido para administradores o usuarios regulares */}
            {isAdmin ? (
                <div className="mb-3">
                    <h5>¡Bienvenido Administrador!</h5>
                    <button className="btn btn-danger" onClick={handleAdminLogout}>
                        Cerrar Sesión de Administrador
                    </button>

                    {/* Botón para agregar mascotas */}
                    <Link to="/agregar-mascota" className="btn btn-primary ms-2">
                        Agregar Mascota
                    </Link>
                </div>
            ) : (
                <div className="d-flex">
                    {/* Campos y botón para iniciar sesión como administrador */}
                    <div className="mb-3 me-2">
                        <input
                            type="text"
                            className="form-control"
                            placeholder="Usuario"
                            value={adminUsername}
                            onChange={(e) => setAdminUsername(e.target.value)}
                        />
                    </div>
                    <div className="mb-3 me-2">
                        <input
                            type="password"
                            placeholder="Contraseña"
                            className="form-control"
                            value={adminPassword}
                            onChange={(e) => setAdminPassword(e.target.value)}
                        />
                    </div>
                    <div>
                        <button className="btn btn-primary" onClick={handleAdminLogin}>
                            Ingresar como Administrador
                        </button>
                    </div>
                </div>
            )}
```

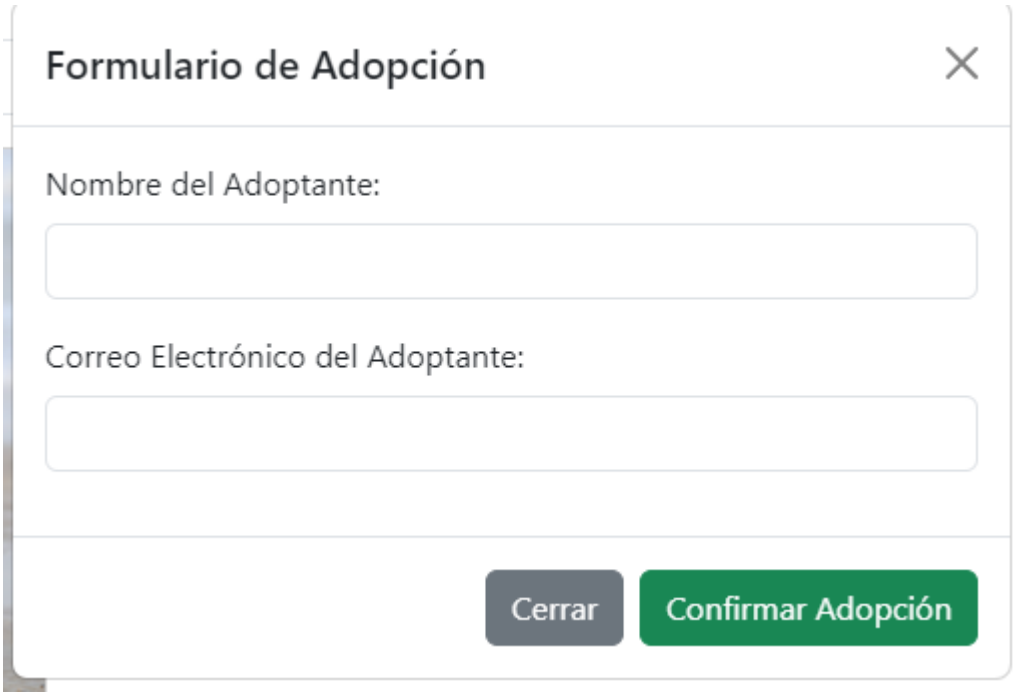
```
    // Renderización del componente
    return (
        <div className="container mt-5">
            {/* Encabezado con título y botón para volver al inicio */}
            <div className="mb-4 d-flex justify-content-between align-items-center">
                <h2>{mascota ? mascota.nombre : 'Detalles de la Mascota'}</h2>
                <Link to="/" className="btn btn-primary">
                    Volver al Inicio
                </Link>
            </div>

            {/* Contenido principal que muestra la información de la mascota */}
            {mascota ? (
                <div className="row">
                    {/* Columna izquierda con la imagen de la mascota */}
                    <div className="col-md-6">
                        <img
                            src={mascota.imagen}
                            className="img-fluid rounded"
                            alt={`Imagen de ${mascota.nombre}`}
                        />
                    </div>

                    {/* Columna derecha con detalles como raza y descripción */}
                    <div className="col-md-6">
                        <p><strong>Raza:</strong> {mascota.raza}</p>
                        <p><strong>Descripción:</strong> {mascota.descripcion}</p>
                    </div>
                </div>
            ) : (
                // Muestra un mensaje de carga si los detalles aún no se han cargado
                <p>Cargando detalles...</p>
            )}
        </div>
    );
};
```

```jsx
    // Renderización del componente
    return (
        <div className={`modal ${showModal ? "show" : ""}`} tabIndex="-1" role="dialog" style={{ display: showModal ? "block" : "none" }}>
            <div className="modal-dialog">
                <div className="modal-content">
                    {/* Encabezado del modal con el título y botón de cierre */}
                    <div className="modal-header">
                        <h5 className="modal-title">Formulario de Adopción</h5>
                        <button type="button" className="btn-close" onClick={handleClose}></button>
                    </div>

                    {/* Cuerpo del modal con el formulario para el nombre y correo del adoptante */}
                    <div className="modal-body">
                        <div className="mb-3">
                            <label htmlFor="nombreAdoptante" className="form-label">Nombre del Adoptante:</label>
                            <input
                                type="text"
                                id="nombreAdoptante"
                                className="form-control"
                                value={nombreAdoptante}
                                onChange={(e) => setNombreAdoptante(e.target.value)}
                            />
                        </div>
                        <div className="mb-3">
                            <label htmlFor="correoAdoptante" className="form-label">Correo Electrónico del Adoptante:</label>
                            <input
                                type="text"
                                id="correoAdoptante"
                                className="form-control"
                                value={correoAdoptante}
                                onChange={(e) => setCorreoAdoptante(e.target.value)}
                            />
                        </div>
                    </div>

                    {/* Pie del modal con botones de cerrar y confirmar adopción */}
                    <div className="modal-footer">
                        <button type="button" className="btn btn-secondary" onClick={handleClose}>
                            Cerrar
                        </button>
                        <button type="button" className="btn btn-success" onClick={handleConfirmClick}>
                            Confirmar Adopción
                        </button>
                    </div>
                </div>
            </div>
        </div>
    );
};
```

4. Estilos CSS (Uso de Bootstrap), se debe generar una interface ordenada estructurada y agradable para el usuario final. (1 Pto.)

Evidencia de la página construida:





Agregar Nueva Mascota

Nombre:

Raza:

Temperamento:

Descripción:

URL de la Imagen:

Agregar Mascota

Volver al Inicio

¡Bienvenido Administrador!

Cerrar Sesión de Administrador | Agregar Mascota

Nombre: Pepito perez

Raza: Pastor Aleman

Jugueton

Adoptar | Detalles

Usuario | Contraseña | Ingresar como Administrador

Nombre: Pepito perez

Raza: Pastor Aleman

Jugueton

Adoptar | Detalles