Diplomado de actualización en nuevas tecnologías para el desarrollo de software

Taller Unidad 3 Frontend

Hecho por:

Anthony Danilo Parra

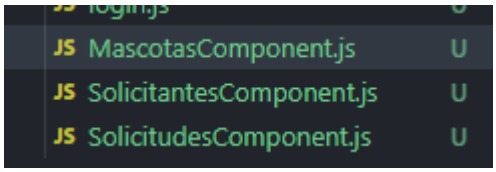
Universidad de Nariño

San Juan de Pasto

2023

1. Creación de componentes y métodos asociados a los mismos.
2. Uso de HTML 5 y JavaScript (Se debe desarrollar una estructura ordenada, con código legible y documentado).

Para la creación de los componentes, se hizo teniendo en cuenta el taller 2 del backend y en cual se creó 3 componentes:



Para el caso de MascotasComponenet.js se siguió el video de la clase, y se añadio nuevas cosas como la paginacion, la creacion de cards, y tambien se incluyeron modales para ver los detalles.

Los metodos asociados son los siguientes:

```
//CUERPO COMPONENTE
const MascotasComponent = () => {
  const url = "http://localhost:8000/mascotas";
  const url2 = "http://localhost:8000/solicitudes";
  const url3 = "http://localhost:8000/solicitantes";
  const [mascotas, setMascotas] = useState([]);
  const [solicitudes, setSolicitudes] = useState([]);
  const [solicitantes, setSolicitantes] = useState([]);
  const [id, setId] = useState("");
  const [nombre, setNombre] = useState("");
  const [tipo, setTipo] = useState("");
  const [edad, setEdad] = useState("");
  const [estado, setEstado] = useState("");
  const [operacion, setOperacion] = useState("");
  const [titulo,setTitulo]=useState("");
  const mascotasPorPagina = 9;
  const [paginaActual, setPaginaActual] = useState(1);
  const [mascotaSeleccionada, setMascotaSeleccionada] = useState(null);
```

El getMascotas(); para traer las mascotas desde el backend

```
const getMascotas = async () => {
  try {
    const respuesta = await axios.get(`${url}/buscar`);
    console.log(respuesta.data);

    // Verifica si la propiedad 'mascotas' es un array antes de establecerla
    if (Array.isArray(respuesta.data.mascotas)) {

      setMascotas(respuesta.data.mascotas);
    } else {
      // Maneja el caso en que 'mascotas' no es un array
      console.error("Formato de respuesta no válido: ", respuesta.data);
    }
  } catch (error) {
    console.error("Error al obtener datos: ", error);
  }
};
```

Lo mismo se hizo para getSolicitudes y getSolicitantes, ya que se necesitan para la informacion de las mascotas

```
const getSolicitudes = async () => {
  try {
    const respuesta = await axios.get(`${url2}/buscar`);
    console.log(respuesta.data);

    if (Array.isArray(respuesta.data.solicitud)) {
      setSolicitudes(respuesta.data.solicitud);
    } else {
      // Maneja el caso en que 'solicitud' no es un array
      console.error("Formato de respuesta no válido: ", respuesta.data);
    }
  } catch (error) {
    console.error("Error al obtener datos de solicitudes: ", error);
  }
};


const getSolicitantes = async () => {
  try {
    const respuesta = await axios.get(`${url3}/buscar`);
    console.log(respuesta.data);

    if (Array.isArray(respuesta.data.persona)) {
      setSolicitantes(respuesta.data.persona);
    } else {
      // Maneja el caso en que 'solicitante' no es un array
      console.error("Formato de respuesta no válido: ", respuesta.data);
    }
  } catch (error) {
    console.error("Error al obtener datos de solicitantes: ", error);
  }
};
```

```
useEffect(() => {
  getMascotas();
  getSolicitudes();
  getSolicitantes();
}, []);
```

El openModal Abre un modal para registrar o editar la información de una mascota.

```
/**
 * @param {number} opcion - Opción que indica si se está registrando (1) o editando (2) la mascota.
 * @param {string} id - Identificador único de la mascota.
 * @param {string} nombre - Nombre de la mascota.
 * @param {string} tipo - Tipo o especie de la mascota.
 * @param {string} edad - Edad de la mascota.
 * @param {string} estado - Estado actual de la mascota.
 */
const openModal = (opcion, id, nombre, tipo, edad, estado) => {
  // Establece los valores de los estados del componente con los parámetros proporcionados
  setId(id || '');
  setNombre(nombre || '');
  setTipo(tipo || '');
  setEdad(edad || '');
  setEstado(estado || '');
  setOperacion(opcion);

  // Configura el título del modal según la opción seleccionada
  if (opcion === 1) {
    setTitulo("Registrar Mascota");
  } else if (opcion === 2) {
    setTitulo("Editar Mascota");
    setId(id);
    setNombre(nombre);
    setTipo(tipo);
    setEdad(edad);
    setEstado(estado);
  }
};
```

En Validar, se valida los campos de nombre, tipo, edad y estado antes de enviar una solicitud para registrar o actualizar la información de una mascota.

```
    }
```

```
    // Envía la solicitud utilizando los parámetros y el método configurados
    enviarSolicitud(metodo, parametros);
  }
};
```

En enviarSolicitud, envía una solicitud HTTP al servidor utilizando el método y parámetros especificados.

```
 *
 * @param {string} metodo - Método HTTP para la solicitud (Ej. "GET", "POST", "PUT", "DELETE").
 * @param {Object} parametros - Parámetros necesarios para la solicitud, incluyendo la URL de la extensión y los datos.
 */
const enviarSolicitud = async (metodo, parametros) => {
  try {
    const respuesta = await axios({
      method: metodo,
      url: parametros.urlExt,
      data: parametros
    });

    // Extrae el tipo y el mensaje de la respuesta y muestra una alerta
    const tipo = respuesta.data.tipo;
    const mensaje = respuesta.data.mensaje;
    mostrarAlerta(mensaje, tipo);

    // Si la respuesta es exitosa, realiza acciones adicionales
    if (tipo === "success") {
      // Cierra el modal con el ID "btnCerrarModal"
      document.getElementById("btnCerrarModal").click();

      // Ejecuta funciones para actualizar datos después de la operación exitosa
      getMascotas();
      getSolicitudes();
      getSolicitantes();
    }
  } catch (error) {
    // Muestra una alerta en caso de error en la solicitud
    mostrarAlerta(`Error en la solicitud`, error);
  }
};
```

En eliminarMascota, muestra un cuadro de confirmación para eliminar una mascota y envía una solicitud de eliminación.

```
 *
 *
 * @param {string} id - Identificador único de la mascota a eliminar.
 * @param {string} nombre - Nombre de la mascota a eliminar.
 */
const eliminarMascota = (id, nombre) => {
  Swal.fire({
    title: '¿Está seguro de eliminar a ' + nombre + '?',
    text: "No podrá revertir esta acción!",
    icon: 'warning',
    showCancelButton: true,
    confirmButtonColor: '#3085d6',
    cancelButtonColor: '#d33',
    confirmButtonText: 'Si, eliminar!'
  }).then((result) => {
    if (result.isConfirmed) {
      // Establece el ID y envía una solicitud de eliminación utilizando el método DELETE
      setId(id);
      enviarSolicitud('DELETE', { urlExt: `${url}/eliminar/${id}` });
    }
  });
};
```

En openDetailsModal, abre un modal para mostrar detalles de una mascota, incluyendo información del solicitante si está disponible.

```
 * @param {string} id - Identificador único de la mascota.
 * @param {string} nombre - Nombre de la mascota.
 * @param {string} tipo - Tipo o especie de la mascota.
 * @param {string} edad - Edad de la mascota.
 * @param {string} estado - Estado de adopción de la mascota.
 * @param {string} idSolicitante - Identificador único del solicitante de adopción (opcional).
 */
const openDetailsModal = (id, nombre, tipo, edad, estado, idSolicitante) => {
  // Establece la mascota seleccionada con la información proporcionada
  setMascotaSeleccionada({
    id,
    nombre,
    tipo,
    edad,
    estado_adopcion: estado,
    idSolicitante,
  });

  // Configura el título del modal
  setTitulo("Detalles de la Mascota");

  // Obtén información del solicitante si hay un ID de solicitante proporcionado
  if (idSolicitante) {
    fetch(`http://localhost:8000/solicitantes/buscar/${idSolicitante}`)
      .then(response => response.json())
      .then(data => {
        // Actualiza el estado de la mascota seleccionada con la información del solicitante
        setMascotaSeleccionada(prevMascota => ({
          ...prevMascota,
          solicitante: data.persona,
        }));
      })
      .catch(error => console.error('Error al obtener información del solicitante:', error));
  }

  // Abre el modal de detalles
  document.getElementById('btnCerrarModal').click(); // Cierra el modal de edición
  document.getElementById('modalDetallesMascota').click(); // Abre el modal de detalles
};
```

El useEffect que se ejecuta cada vez que la variable de estado 'mascotaSeleccionada' se actualiza.

Imprime el estado actualizado en la consola y realiza acciones adicionales, como abrir el modal de detalles.

```
useEffect(() => {
  // Imprime el estado actualizado en la consola
  console.log('Estado actualizado en useEffect:', mascotaSeleccionada);

  // Realiza acciones adicionales aquí después de la actualización del estado
  if (mascotaSeleccionada && mascotaSeleccionada.solicitante) {
    // Por ejemplo, abrir el modal después de que el estado se haya actualizado completamente
    document.getElementById('btnCerrarModal').click(); // Cierra el modal de edición
    document.getElementById('modalDetallesMascota').click(); // Abre el modal de detalles
  }
}, [mascotaSeleccionada]);
```

En rendereizarMascotaPorPagina, renderiza y devuelve un conjunto de filas de tarjetas de mascotas, filtradas según el término de búsqueda.

Cada fila contiene hasta 3 tarjetas de mascotas.

```
 * @returns {Array} - Un arreglo de elementos JSX representando las filas de tarjetas de mascotas.
 */
const renderizarMascotasPorPagina = () => {
  // Arreglo que contendrá las filas de tarjetas de mascotas
  const filas = [];

  // Filtra las mascotas actuales según el término de búsqueda
  const mascotasFiltradas = mascotasActuales.filter((mascota) =>
    mascota.nombre.toLowerCase().includes(busqueda.toLowerCase())
  );

  // Calcula el número total de filas necesarias
  const totalFilas = Math.ceil(mascotasFiltradas.length / 3);

  // Itera a través de las filas y crea tarjetas de mascotas para cada fila
  for (let i = 0; i < totalFilas; i++) {
    // Calcula el índice de inicio y fin para las mascotas en la fila actual
    const inicio = i * 3;
    const fin = inicio + 3;
    // Obtiene las mascotas para la fila actual
    const mascotasFila = mascotasFiltradas.slice(inicio, fin);

    // Crea una fila de tarjetas de mascotas
    filas.push(
      <div key={i} className="row mt-3">
        {mascotasFila.map((mascota) => (
          <div key={mascota.id} className="col-4">
            <div className="card" style={{ marginBottom: '20px', margin: '20px', border: '1px solid #ddd', borderRadius: '8px', boxShadow: '0 4px 8px rgba(0, 0, 0, 0.1)' }}>
              <div className="card-body">
                <h5 className="card-title">{mascota.nombre}</h5>
                <p className="card-text">
                  <strong>Tipo:</strong> {mascota.tipo}<br />
                  <strong>Edad:</strong> {mascota.edad}<br />
                  <strong>Estado:</strong> {mascota.estado_adopcion}
                </p>
                {/* Botón para ver detalles de la mascota */}
                <button
                  onClick={() => openDetailsModal(mascota.id, mascota.nombre, mascota.tipo, mascota.edad, mascota.estado_adopcion)}
                  className="btn btn-info"
                  data-bs-toggle="modal"
                  data-bs-target="#modalDetallesMascota"
                >
                  <i className="fa-solid fa-info"></i> Ver Detalles
                </button>
```

```
                {/* Botón para eliminar la mascota */}
                <button onClick={() => eliminarMascota(mascota.id, mascota.nombre)} className="btn btn-danger ml-2">
                  <i className="fa-solid fa-trash"></i> Eliminar
                </button>
              </div>
            </div>
          </div>
        ))}
      </div>
    );
  }

  // Devuelve el arreglo de filas de tarjetas de mascotas
  return filas;
};
```

Componente principal de la aplicación. El return

Renderiza un conjunto de elementos, incluyendo un botón para añadir mascotas, la lista paginada de mascotas, y un modal para agregar o editar información de mascotas.

```jsx
return (
  <div className="App">

    <div className="container-fluid">
      <div className="row mt-3">
        <div className="col-md-4 offset-md-4">
        <div className="input-group mb-3">
          <span className="input-group-text">
            <i className="fa-solid fa-search"></i>
          </span>
          <input
            type="text"
            className="form-control"
            placeholder="Buscar mascota"
            value={busqueda}
            onChange={(e) => setBusqueda(e.target.value)}
          />
        </div>
          <div className="d-grid mx-auto">
            <button
            onClick={()=>openModal(1)}
              className="btn btn-dark"
              data-bs-toggle="modal"
              data-bs-target="#modalMascotas"
            >
              <i className="fa-solid fa-circle-plus"></i>Añadir
            </button>
          </div>
        </div>
      </div>
      {renderizarMascotasPorPagina()}
      {paginacion()}
    </div>
```

```jsx
<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="nombre"
            className="form-control"
            placeholder="Nombre"
            value={nombre}
            onChange={(e)=>setNombre(e.target.value)}
          ></input>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text">
            <i className="fa-solid fa-gift"></i>
          </span>
          <input
            type="text"
            id="tipo"
            className="form-control"
            placeholder="Tipo"
            value={tipo}
            onChange={(e)=>setTipo(e.target.value)}

          ></input>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text">
            <i className="fa-solid fa-gift"></i>
          </span>
          <input
            type="text"
            id="edad"
            className="form-control"
            placeholder="Edad"
            value={edad}
            onChange={(e)=>setEdad(e.target.value)}
          ></input>
        </div>
```

```
        <div className="input-group mb-3">
          <span className="input-group-text">
            <i className="fa-solid fa-gift"></i>
          </span>
          <input
            type="text"
            id="estado_adopcion"
            className="form-control"
            placeholder="Estado"
            value={estado}
            onChange={(e)=>setEstado(e.target.value)}
          ></input>
        </div>
        <div className="d-grid col-6 mx-auto">
          <button onClick={()=>validar()} className="btn btn-success">
            <i className="fa-solid fa-floppy-disk"></i>Guardar
          </button>
        </div>
      </div>
      <div className="modal-footer">
        <button
          id="btnCerrarModal"
          type="button"
          className="btn btn-secondary"
          data-bs-dismiss="modal"
        >
          Cerrar
        </button>
      </div>
    </div>
  </div>
  <button></button>
</div>
```

Modal para mostrar detalles de una mascota seleccionada, incluyendo información del solicitante si está disponible.

```jsx
<div className="modal fade" id="modalDetallesMascota" tabIndex="-1" aria-labelledby="modalDetallesMascotaLabel" aria-hidden="true">
  <div className="modal-dialog">
    <div className="modal-content">
      <div className="modal-header">
        <h5 className="modal-title" id="modalDetallesMascotaLabel">Detalles de la Mascota</h5>
        <button type="button" className="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div className="modal-body">
        {mascotaSeleccionada && (
          <>
            <p><strong>ID:</strong> {mascotaSeleccionada.id}</p>
            <p><strong>Nombre:</strong> {mascotaSeleccionada.nombre}</p>
            <p><strong>Tipo:</strong> {mascotaSeleccionada.tipo}</p>
            <p><strong>Edad:</strong> {mascotaSeleccionada.edad}</p>
            <p><strong>Estado de Adopción:</strong> {mascotaSeleccionada.estado_adopcion}</p>

            {/* Verifica si hay un solicitante y muestra su información */}
            {solicitudes.map(solicitud => {
              if (solicitud.mascota_id === mascotaSeleccionada.id) {
                // Encuentra el solicitante correspondiente a la solicitud
                const solicitante = solicitantes.find(s => s.id === solicitud.solicitante_id);

                if (solicitante) {
                  return (
                    <div key={solicitud.id}>
                      <h6>Solicitante:</h6>
                      <p><strong>Nombre:</strong> {solicitante.nombre}</p>
                      <p><strong>Correo:</strong> {solicitante.correo}</p>
                      <p><strong>Teléfono:</strong> {solicitante.telefono}</p>
                      <p><strong>Dirección:</strong> {solicitante.direccion}</p>
                      {/* Agrega más detalles del solicitante según tus necesidades */}
                    </div>
                  );
                }
              }
              return null;
            })}
          </>
        )}
      </div>
```

```jsx
      <div className="modal-footer">
        {/* Botón para editar */}
        <button
          onClick={() => openModal(2, mascotaSeleccionada.id, mascotaSeleccionada.nombre, mascotaSeleccionada.tipo, mascotaSeleccionada.edad, mascotaSeleccionada.estado_adopci
          className="btn btn-warning"
          data-bs-toggle="modal"
          data-bs-target="#modalMascotas"
        >
          <i className="fa-solid fa-edit"></i> Editar
        </button>
      </div>
    </div>
  </div>
</div>
```

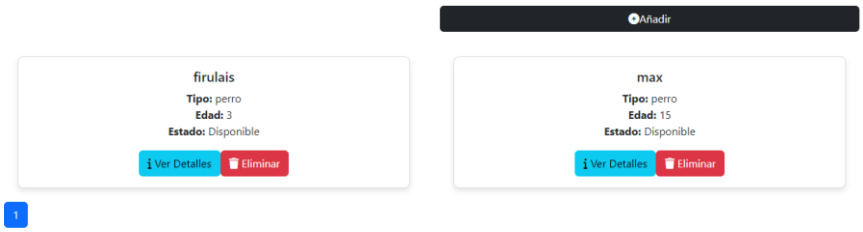
```jsx
};

//EXPORT
export default MascotasComponent;
```

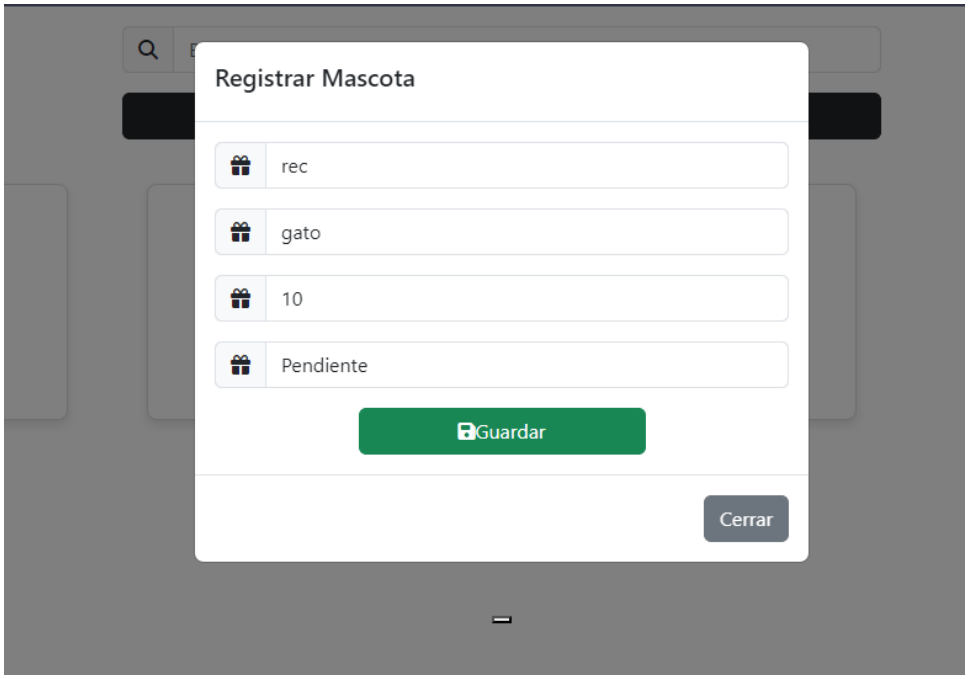
Cabe mencionar, que para los componentes de SolicitudesComponent.js y SolicitantesComponent.js es el mismo procedimiento

3. Estilos CSS (Uso de Bootstrap), se debe generar una interface ordenada estructurada y agradable para el usuario final. Elaborar un informe que detalle el proceso de construcción y la implementación del aplicativo.

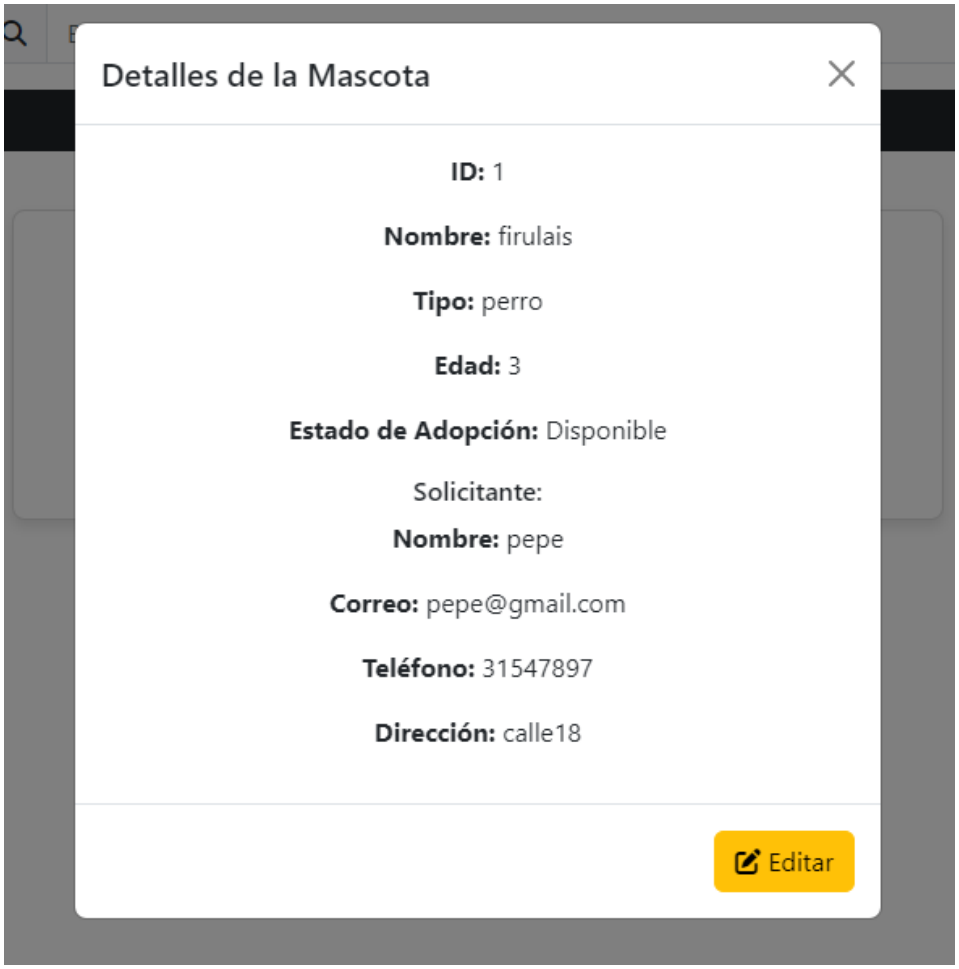
En las siguientes imágenes se puede apreciar, el uso de bootstrap para darle un mejor apariencia



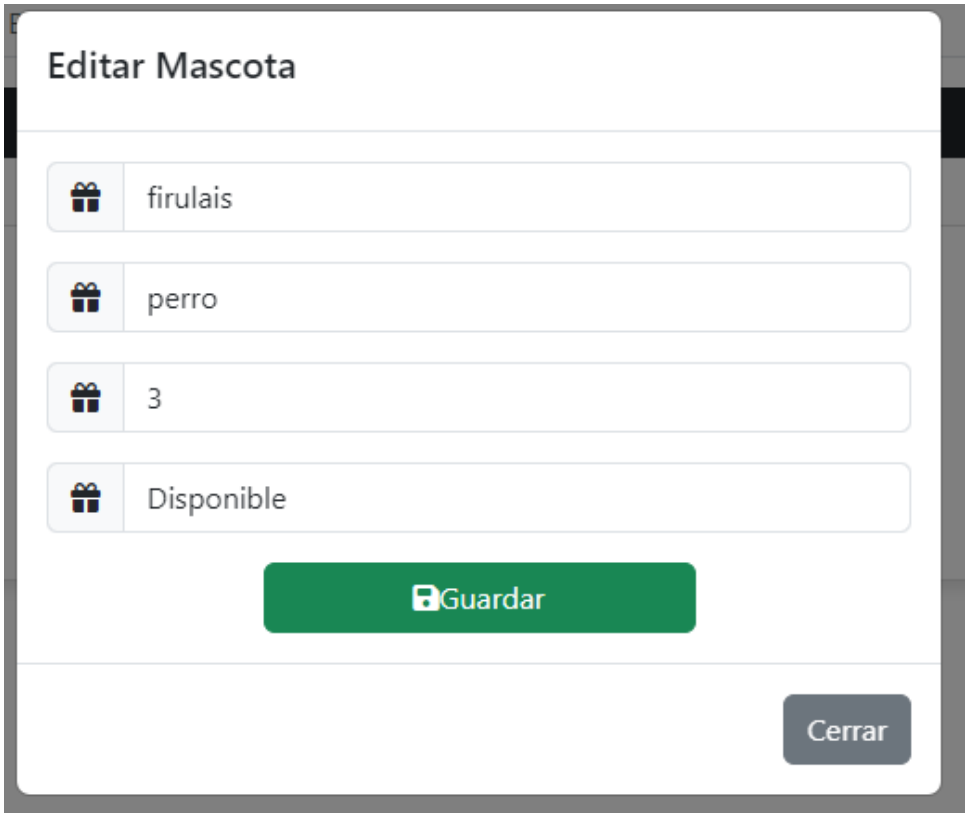
Añadir una nueva mascota



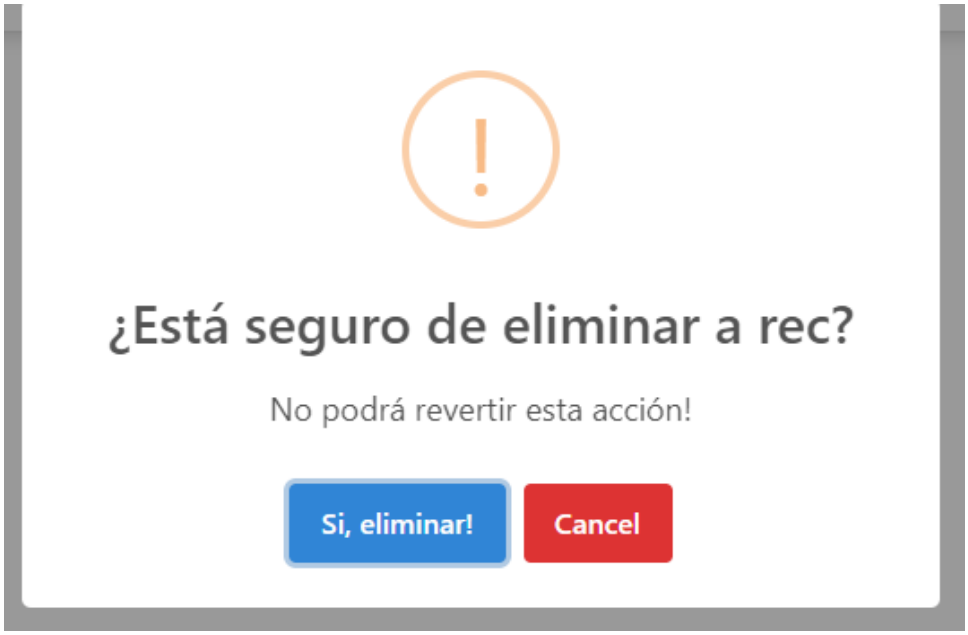
Click en ver detalles

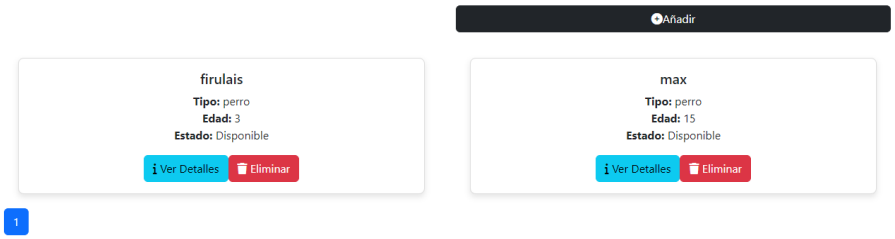


Edicion

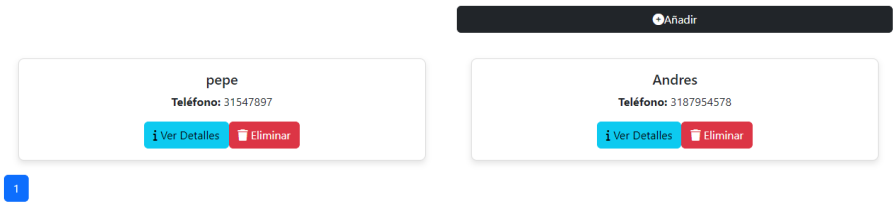


Eliminacion





En solicitantes es lo mismo



Y en solicitudes tambien es lo mismo