Taller Final

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

Backend

Tal y como se detalló en el informe pasado, el primer paso es construir la base de datos y la conexión con la misma.

Archivo de conexión con la base de datos:

```
// importar la biblioteca Sequelize
import { Sequelize } from "sequelize";

//definir y exportar la constante que sostendrá la conexion a la base de datos
```

```
export const db = new Sequelize('adopcion', 'adopcion', '123', {
  dialect:'mysql',
  host:'localhost'
});
```

La base de datos proporciona el almacenamiento para los campos de información definidos en los modelos, los modelos a su vez, le sirven de puente a los controladores con la base de datos, para realizar las diferentes operaciones disponibles.

Las rutas se definen de tipo diferente de acuerdo a la acción a realizar, las de tipo get para obtener datos, post para ingresar información, put para actualizar y delete en caso de eliminar.

Modelos

Usuario

Los campos más relevantes son el nombre, usuario, telefonica, contraseña y rol, de la siguiente forma:

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

//definicion de la variable que construirá la tabla
export const usuarios = db.define('usuarios',{
  //definicion de los atributos
  pk:{
     //tipo de dato
     type:Sequelize.INTEGER,
     //no se permite vacio
     allowNull:false,
     //es la llave primaria
     primaryKey:true,
     //es autoincrementado
     autoIncrement:true
  },
  nombres:{
```

```
    //tipo de dato
    type:Sequelize.STRING,
    //no se permite vacio
    allowNull:false
},
telefono:{
    //tipo de dato
    type:Sequelize.STRING,
    //no se permite vacio
    allowNull:true
},
usuario:{
    //tipo de dato
    type:Sequelize.STRING,
    //no se permite vacio
    allowNull:false
},
passwd:{
    //tipo de dato
    type:Sequelize.STRING,
    //no se permite vacio
    allowNull:false
},
rol:{
    //tipo de dato
    type:Sequelize.STRING,
    //no se permite vacio
    allowNull:false
},
foto:{
    //tipo de dato
    type:Sequelize.STRING,
    //se permite vacio
    allowNull:true
```

```
  },
  edad:{
     //tipo de dato
     type:Sequelize.INTEGER,
     //se permite vacio
     allowNull:true
  },
});
```

Mascota

```
//importar Sequelize
import { Sequelize } from "sequelize";
//importar la variable de conexion
import { db } from "../database/conexion.js";

//definicion del objeto que comunicara con la tabla
export const mascotas = db.define('mascotas',{
   //definicion de los atributos
   pk:{
        //tipo de dato
        type:Sequelize.INTEGER,
        //no se permite vacio
        allowNull:false,
        //es la llave primaria
        primaryKey:true,
        //es autoincrementado
        autoIncrement:true
   },
   nombre:{
         //tipo de dato
         type:Sequelize.STRING,
         //no se permite vacio
         allowNull:false
   },
   foto:{
        //tipo de dato
        type:Sequelize.STRING,
        //se permite vacio
        allowNull:true
```

```
    },
    descripcion:{
        //tipo de dato
        type:Sequelize.STRING,
        //se permite vacio
        allowNull:true
    },
    rasa:{
        //tipo de dato
        type:Sequelize.STRING,
        //se permite vacio
        allowNull:true
    },
    edad:{
        //tipo de dato
        type:Sequelize.INTEGER,
        //se permite vacio
        allowNull:true
    },
    tipo_mascota:{
        //tipo de dato
        type:Sequelize.CHAR,
        //no se permite vacio
        allowNull:false
    },
    estado:{
        //tipo de dato
        type:Sequelize.INTEGER,
        //no se permite vacio
        allowNull:true
    }
});
```

Solicitud

```
//importar Sequelize
import { Sequelize } from "sequelize";
//importar la variable de conexion
import { db } from "../database/conexion.js";
//importar el modelo mascotas
import { mascotas } from "./mascotaModelo.js";
```

```
//importar el modelo usuarios
import { usuarios } from "./usuarioModelo.js"

//definicion del objeto que comunicara con la tabla
const solicitudes = db.define('solicitudes',{
   //definicion de los atributos
   pk:{
       //tipo de dato
       type: Sequelize.INTEGER,
       //no se permite vacio
       allowNull: false,
       //es la llave primaria
       primaryKey: true,
       //es autoincrementado
       autoIncrement: true
   },
   mascotaPK:{
       //tipo de dato
       type: Sequelize.INTEGER,
       //no se permite vacio
       allowNull: false
   },
   adoptante:{
        //tipo de dato
        type: Sequelize.INTEGER,
        //no se permite vacio
        allowNull: false
   },
   estado:{
       //tipo de dato
       type: Sequelize.CHAR,
       //no se permite vacio
       allowNull: false
   },
   fecha_inicio:{
       //tipo de dato
       type: Sequelize.DATE,
       //no se permite vacio
       allowNull: false
   },
```

```
    fecha_fin:{
        //tipo de dato
        type: Sequelize.DATE,
        //no se permite vacio
        allowNull: true
    }
});
```

La relación entre las 3 tablas se hace por medio de la solicitud, estableciendo que una solicitud pertenece a un usuario y se adopta a una mascota, de la siguiente manera:

```
// definir la relación de solicitud con mascota
solicitudes.belongsTo(mascotas, { foreignKey: 'mascotaPK' });
//la relacion con usuario
solicitudes.belongsTo(usuarios, { foreignKey: 'adoptante' });
```

Controladores

Los controladores se fundamentan en los principios ya explicados en informes anteriores, se busca gestionar el modelo que permita interactuar con la base de datos, haciendo uso de métodos.

Usuario

```
//importar el modelo de mascotas
import { usuarios } from "../modelos/usuarioModelo.js";

//listar todas los usuarios
const listarUsuarios = (req, res) => {
    usuarios.findAll().then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
encontrar ningun registro"+e});
    });

    return;
}

//buscar un usuario por id
const buscarUsuario= (req , res) => {
    const id = parseInt(req.params.id);
```

```
    if(id == null){
        res.status(400).json({tipo:'error', mensaje: "El id no puede estar
vacio"});
        return;
    }

    usuarios.findByPk(id).then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
encontrar el registro"});
    });

    return;
}

//crear un usuarios en la tabla
const crearUsuario = (req, res) => {
    //los campos nombres, usuario y passwd son obligatorios
    if(!req.body.nombres){
        res.status(400).json({tipo:'error', mensaje: "el campo nombres es
requerido"});
        return;
    }

    if(!req.body.usuario){
        res.status(400).json({tipo:'error',mensaje: "el usuario es
requerido"});
        return;
    }

    if(!req.body.passwd){
        res.status(400).json({tipo:'error',mensaje: "el passwd es
requerido"});
        return;
    }

    if(!req.body.rol){
```

```
        res.status(400).json({tipo:'error',mensaje: "el rol es
requerido"});
        return;
    }

    //definicion del dataset
    const dataset = {
        nombres: req.body.nombres,
        edad: req.body.edad,
        usuario: req.body.usuario,
        passwd: req.body.passwd,
        foto: req.body.foto,
        rol: req.body.rol,
        telefono: req.body.telefono
    }

    //creacion del usuario en la base de datos
    usuarios.create(dataset).then((r)=>{
        res.status(200).json({
            tipo:'success',
            mensaje: "Usuario registrado con exito"
        });
    }).catch((e)=>{
        res.status(500).json({
            tipo:'error',
            mensaje: "No se ha podido registrar el usuario"+e});
    });

    return;
}

//eliminar una usuario por id
const eliminarUsuario = (req , res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({tipo:'error', mensaje: "El id no puede estar
vacio"});
        return;
    }
```

```
    usuarios.destroy({
        where:{pk: id} //recordemos que en la base de datos, el campo de la
llave primaria es pk, no id
    }).then((r) => {
        res.status(200).json({tipo:'success', mensaje: "usuario eliminado
exitosamente"});
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
eliminar el registro"});
    });

    return;
}

//actualizar un usuario por id
const actualizarUsuario = (req , res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({tipo:'error', mensaje: "El id no puede estar
vacio"});
        return;
    }

    if(!req.body.nombres && !req.body.edad && !req.body.foto &&
!req.body.passwd && !req.body.usuario && !req.body.rol){
        res.status(400).json({tipo:'error', mensaje: "No se ha encontrado
ningun dato para actualizar"});
        return;
    }

    const nombres = req.body.nombres;
    const edad = req.body.edad;
    const passwd = req.body.passwd;
    const usuario = req.body.usuario;
    const foto = req.body.foto;
    const rol = req.body.rol;
    const telefono = req.body.telefono;
```

```
    usuarios.update({
        nombres: nombres,
        edad:edad,
        passwd:passwd,
        usuario:usuario,
        foto:foto,
        rol:rol,
        telefono:telefono
    },{
        where:{pk: id} //recordemos que en la base de datos, el campo de la
llave primaria es pk, no id
    }).then((r) => {
        res.status(200).json({tipo:'success', mensaje: "Usuario actualizado
exitosamente"});
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
actualizar el registro"});
    });

    return;
}

export { crearUsuario, listarUsuarios, buscarUsuario, eliminarUsuario,
actualizarUsuario };
```

Mascota

```
//importar el modelo de mascotas
import { mascotas } from "../modelos/mascotaModelo.js";

//listar todas las mascotas
const listarMascotas = (req, res) => {
    mascotas.findAll().then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
encontrar ningun registro"+e});
    });
```

```
    return;
}

//buscar una mascota por id
const buscarMascota = (req , res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({tipo:'error', mensaje: "El id no puede estar
vacio"});
        return;
    }

    mascotas.findByPk(id).then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
encontrar el registro"});
    });

    return;
}

//crear una mascota en la tabla
const crearMascota = (req, res) => {
    //los campos nombre, tipo mascota y estado son obligatorios
    if(!req.body.nombre){
        res.status(400).json({tipo:'error',mensaje: "el campo nombre es
requerido"});
        return;
    }
    if(!req.body.tipo_mascota){
        res.status(400).json({tipo:'error',mensaje: "el tipo_mascota es
requerido"});
        return;
    }

    //definicion del dataset
    const dataset = {
        nombre: req.body.nombre,
```

```
        edad: req.body.edad,
        tipo_mascota: req.body.tipo_mascota,
        estado: req.body.estado,
        foto: req.body.foto,
        descripcion:req.body.descripcion,
        rasa:req.body.rasa
    }

    //creacion de la mascota en la base de datos
    mascotas.create(dataset).then((r)=>{
        res.status(200).json({
            tipo:'success',
            mensaje: "Mascota registrada con exito"
        });
    }).catch((e)=>{
        res.status(500).json({
            tipo:'error',
            mensaje: "No se ha podido registrar la mascota"+e});
    });

    return;
}

//eliminar una mascota por id
const eliminarMascota = (req , res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({tipo:'error', mensaje: "El id no puede estar
vacio"});
        return;
    }

    mascotas.destroy({
        where:{pk: id} //recordemos que en la base de datos, el campo de la
llave primaria es pk, no id
    }).then((r) => {
        res.status(200).json({tipo:'success', mensaje: "Mascota eliminada
exitosamente"});
    }).catch((e) => {
```

```
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
eliminar el registro"});
    });

    return;
}

//actualizar una mascota por id
const actualizarMascota = (req , res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({tipo:'error', mensaje: "El id no puede estar
vacio"});
        return;
    }

    if(!req.body.rasa && !req.body.nombre && !req.body.edad &&
!req.body.tipo_mascota && !req.body.estado && !req.body.foto &&
!req.body.descripcion){
        res.status(400).json({tipo:'error', mensaje: "No se ha encontrado
ningun dato para actualizar"});
        return;
    }

    const nombre = req.body.nombre;
    const edad = req.body.edad;
    const tipo_mascota = req.body.tipo_mascota;
    const estado = req.body.estado;
    const foto = req.body.foto;
    const descripcion = req.body.descripcion;
    const rasa = req.body.rasa;

    mascotas.update({
        nombre: nombre,
        edad:edad,
        tipo_mascota:tipo_mascota,
        estado:estado,
        foto:foto,
        descripcion:descripcion,
```

```
        rasa:rasa

    },{
        where:{pk: id} //recordemos que en la base de datos, el campo de la
llave primaria es pk, no id
    }).then((r) => {
        res.status(200).json({tipo:'success', mensaje: "Mascota actualizada
exitosamente"});
    }).catch((e) => {
        res.status(500).json({tipo:'error', mensaje: "No se ha podido
actualizar el registro"});
    });

    return;
}

export { crearMascota, listarMascotas, buscarMascota, eliminarMascota,
actualizarMascota };
```

Solicitud

```
//importamos el modelo de solicitudes
import { solicitudes } from "../modelos/solicitudModelo.js";

//listar todas las solicitudes
const listarSolicitudes = (req, res) => {
    solicitudes.findAll().then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido consultar las
solicitudes"});
    });

    return;
}

//buscar solicitud por id
const buscarSolicitud = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
```

```
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    solicitudes.findByPk(id).then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido encontrar el
registro"});
    });

    return;
}

//buscar solicitudes de un usuario
const buscarSolicitudUser = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    solicitudes.findAll({
        where:{adoptante:id}
    }).then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido encontrar el
registro"});
    });

    return;
}

//buscar solicitudes de una mascota
const buscarSolicitudMascota = (req, res) => {
    const id = parseInt(req.params.id);
```

```
    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    solicitudes.findAll({
        where:{mascotaPK:id}
    }).then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido encontrar el
registro"});
    });

    return;
}

//eliminar solicitud por id
const eliminarSolicitud = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    solicitudes.destroy({
        where:{ pk: id}
    }).then((r)=>{
        res.status(200).json({mensaje: "Registro eliminado con exito !"});
    }).catch((e) => {
        res.status(500).json({mensaje: "No se a podido remover el registro
de la base de datos"});
    });

    return;
}
```

```
//crear una solicitud
const crearSolicitud = (req, res) => {
   //validar que vengan los campos requeridos
   if(!req.body.mascotaPK){
       res.status(400).json({mensaje: "El campo de id (mascotaPK) de la
mascota es requerido"});
       return;
   }
   if(!req.body.adoptante){
       res.status(400).json({mensaje: "El id del adoptante (adoptante) es
requerido"});
       return;
   }
   if(!req.body.estado){
       res.status(400).json({mensaje: "El estado de la solicitud es
requerido"});
       return;
   }
   if(!req.body.fecha_inicio){
       res.status(400).json({mensaje: "La fecha de inicio es requerida"});
       return;
   }

   //todo el dataset
   const dataset = {
       mascotaPK: req.body.mascotaPK,
       adoptante: req.body.adoptante,
       estado: req.body.estado,
       fecha_inicio: req.body.fecha_inicio,
       fecha_fin: req.body.fecha_fin
   }

   //creacion de la solicitud en la base de datos
   solicitudes.create(dataset).then((r) => {
       res.status(200).json({mensaje: "Solicitud registrada con exito!"});
   }).catch((e) => {
       res.status(500).json({mensaje:" Error, no se ha podido crear el
registro en la base de datos: "+e});
   });
```

```
    return;
}

//buscar solicitud por id
const actualizarSolicitud = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    if(!req.body.adoptante && !req.body.mascotaPK && !req.body.estado &&
!req.body.fecha_inicio && !req.body.fecha_fin){
        res.status(400).json({mensaje: "No se a detectado ningun campo para
actualizar"});
        return;
    }

    const mascotaPK = req.body.mascotaPK;
    const adoptante = req.body.adoptante;
    const estado = req.body.estado;
    const fecha_inicio = req.body.fecha_inicio;
    const fecha_fin = req.body.fecha_fin;

    solicitudes.update({
        mascotaPK:mascotaPK,
        adoptante:adoptante,
        estado: estado,
        fecha_inicio:fecha_inicio,
        fecha_fin: fecha_fin
    },{
        where: {pk:id}
    }).then((r) => {
        res.status(200).json({mensaje: "Solicitud actualizada
exitosamente"});
    }).catch((e) => {
```

```
        res.status(500).json({mensaje: "No se pudo alterar el registro en
la base de datos"});
    });

    return;
}

export { listarSolicitudes, crearSolicitud, buscarSolicitudMascota,
buscarSolicitudUser, buscarSolicitud, eliminarSolicitud,
actualizarSolicitud }
```

Es importante notar los 2 métodos adicionales para el controlador de solicitud, estos 2 métodos son:

- buscarSolicitudMascota, que permite encontrar las solicitudes que corresponden a una determinada mascota.
- buscarSolicitudUser, que permite encontrar las solicitudes que corresponden a un usuario determinado.

Rutas

Las rutas permiten acceder a los métodos del controlador haciendo uso del navegador, por medio de la app.

Usuario

```
import express from "express";
import { crearUsuario, listarUsuarios, buscarUsuario, eliminarUsuario,
actualizarUsuario} from "../controladores/usuarioControlador.js";

//crear la instancia de tipo router
const usuariosRouter = express.Router();

//rutas
//DE TIPO GET
usuariosRouter.get('/', (req, res)=> {
    listarUsuarios(req, res);
});

usuariosRouter.get('/buscar/:id',(req, res) => {
    buscarUsuario(req, res);
});
```

```
//DE TIPO POST
usuariosRouter.post('/crear', (req, res)=>{
   crearUsuario(req, res);
});

//Tipo DELETE
usuariosRouter.delete('/eliminar/:id', (req, res) => {
   eliminarUsuario(req, res);
});

//Tipo PUT
usuariosRouter.put('/actualizar/:id', (req, res) => {
   actualizarUsuario(req, res);
});

export { usuariosRouter };
```

Mascota

```
import express from "express";
import { buscarMascota, crearMascota, listarMascotas, eliminarMascota,
actualizarMascota } from "../controladores/mascotaControlador.js";

//crear la instancia de tipo router
const mascostasRouter = express.Router();

//rutas
//DE TIPO GET
mascostasRouter.get('/', (req, res)=> {
   listarMascotas(req, res);
});

mascostasRouter.get('/buscar/:id',(req, res) => {
   buscarMascota(req, res);
});

//DE TIPO POST
mascostasRouter.post('/crear', (req, res)=>{
   crearMascota(req, res);
});
```

```
//Tipo DELETE
mascostasRouter.delete('/eliminar/:id', (req, res) => {
   eliminarMascota(req, res);
});

//Tipo PUT
mascostasRouter.put('/actualizar/:id', (req, res) => {
   actualizarMascota(req, res);
});

export { mascostasRouter };
```

Solicitud

```
import express from "express";
import { listarSolicitudes, crearSolicitud, buscarSolicitud,
eliminarSolicitud, actualizarSolicitud, buscarSolicitudUser,
buscarSolicitudMascota } from "../controladores/solicitudControlador.js";

//crear la instancia de tipo router
const solicitudesRouter = express.Router();

//rutas
//Tipo GET
solicitudesRouter.get('/', (req, res) => {
   listarSolicitudes(req, res);
});
solicitudesRouter.get('/buscar/:id', (req, res) => {
   buscarSolicitud(req, res);
});

solicitudesRouter.get('/buscar/mascota/:id', (req, res) => {
   buscarSolicitudMascota(req, res);
});

solicitudesRouter.get('/buscar/usuario/:id', (req, res) => {
```

```
    buscarSolicitudUser(req, res);
});

//tipo POST
solicitudesRouter.post('/crear', (req, res) => {
    crearSolicitud(req, res);
});

//tipo DELETE
solicitudesRouter.delete('/eliminar/:id', (req, res) => {
    eliminarSolicitud(req, res);
});

//tipo PUT
solicitudesRouter.put('/actualizar/:id', (req, res) => {
    actualizarSolicitud(req, res);
});

export { solicitudesRouter };
```

Finalmente, el archivo app.js permite levantar la app a servicio.

```
//importar express
import express from "express";
import { mascostasRouter } from "../rutas/mascotasRouter.js";
import { db } from "../database/conexion.js";
import { solicitudesRouter } from "../rutas/solicitudesRouter.js";
import { usuariosRouter } from "../rutas/usuariosRouter.js";
//importar cors
import cors from 'cors';

//Crear la instancia de express
const app = express();

// Middleware para procesar datos JSON en el cuerpo de las solicitudes
app.use(express.json());

//cors para que se puede consultar desde el front
app.use(cors());
```

```
// Middleware para procesar datos de formularios en el cuerpo de las
solicitudes
app.use(express.urlencoded({ extended: true }));

//definir la constante que contendra el puerto por el cual correrá el
servidor
const PORT = 9000;

db.authenticate().then(()=>{
   console.log("La base de datos ha sido cargada con exito");
}).catch((r) => {
   console.log("Error al cargar la base de datos: "+e);
});

//definir las rutas
app.get("/", (req, res) => {res.send("Hola Desde Backend MySQL");});

//definir las rutas para mascotas
app.use('/mascotas', mascostasRouter);

//definir las rutas para solicitudes
app.use('/solicitudes', solicitudesRouter);

//definir las rutas para solicitudes
app.use('/usuarios', usuariosRouter);

//si fue posible conectarse a la base de datos
db.sync().then(() => {
   // puerto de ecucha, recibe el numero de puerto y un función
   app.listen(PORT, ()=>{
       console.log(`Servidor inicializado en el puerto: ${PORT}`);
   });
}).catch((e) => {
   console.log("No se pudo sincronizar con la base de datos: "+e);
});
```

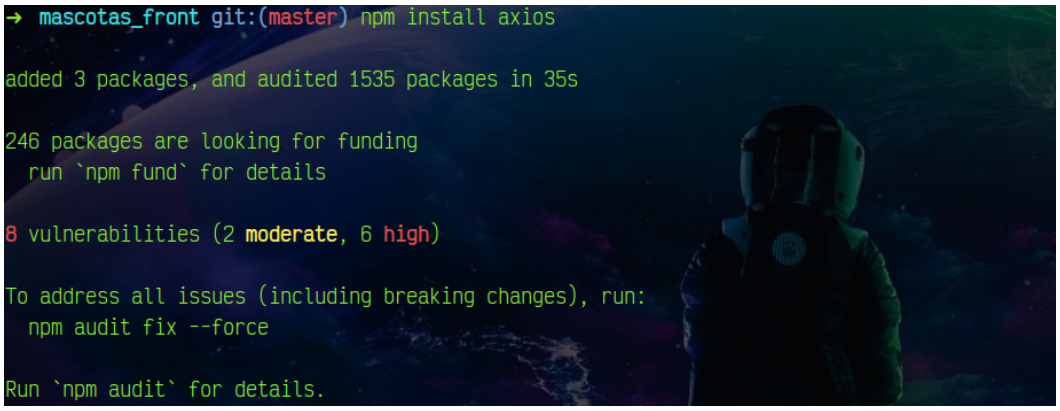
Nótese el uso de cors para lograr acceder a los datos desde otra app como una api.

Posición del backend en local
/home/segundo/Proyectos/Diplomado/Unidad_dos/Backend/TallerUnidad2Backend

FrontEnd

Ahora, se crea el proyecto de React que soportara el frontend:
npx create-react-app nombre_app <- Comando para crear el Proyecto

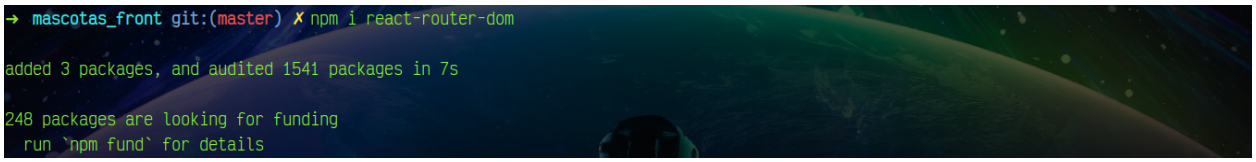
Para conectar con el Backend: **npm install axios**



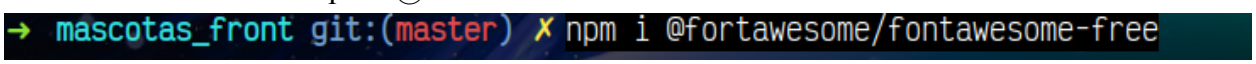
Guia de axios: https://www.freecodecamp.org/espanol/news/como-usar-axios-con-react/#axios

npm install bootstrap

Para trabajar con rutas: **npm i react-router-dom**



Instalar fontawesome: npm i @fortawesome/fontawesome-free



Trabajar con Fontawesome

```
//importar los iconos de fontanswre
import '@fortawesome/fontawesome-free/css/all.min.css';
```

Trabajar con Bootstrap:
en el index.js, para que se apliquen los estilos para toda la app:

```
//importar bootstrap para toda la app
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle';
```

Colocamos un icono y un boton para probar el funcionamiento en App.js

```
import './App.css';

function App() {
 return (
   <div className="App">
     <div
       class="container"
     >
       <a className='btn btn-success' href='#'>
         <i class="fas fa-solid fa-trash"> </i>
       </a>
     </div>


   </div>
 );
}

export default App;
```

npm start <- arrancar el proyecto

Resultado:

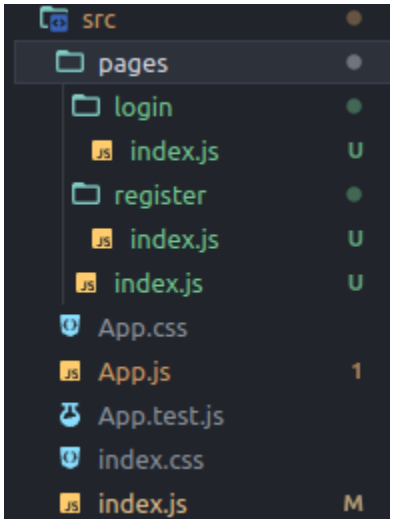
Podemos confirmar que el proyecto ya está listo para trabajar y los iconos y bootstrap ya están funcionando.

Para trabajar con las rutas hacemos las siguientes modificaciones en el index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

//importar los iconos de fontanswre
import '@fortawesome/fontawesome-free/css/all.min.css';

//bootstrap
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle';

//rutas
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

//importar nuestras propias paginas
import { Login, Register } from './pages';
const router = createBrowserRouter([
 {
   path:"/",
   element: <App />
 },
 {
   path:"/login",
   element: <Login />
 },
 {
   path:"/register",
   element: <Register />
 }
]);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
```

```
    <RouterProvider router={router} />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Ahora creamos la carpeta pages con la siguiente estructura:



contenido de login/index.js

```
const Login = () => {
    return(
        <h1>Login</h1>
    );
}

export default Login;
```

contenido de register/index.js

```
const Register = () => {
    return (
        <h1>Register</h1>
    );
}
```

```
export default Register;
```

contenido de pages/index.js

```
import Login from './login';
import Register from './register';

export {Login, Register};
```

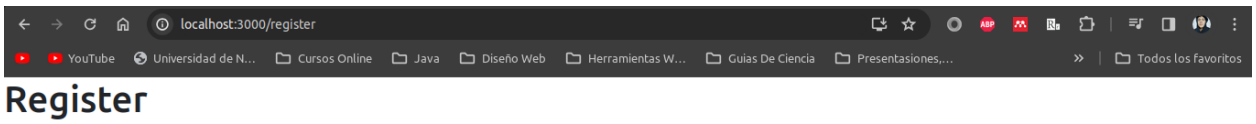
Trabajamos de esta manera porque es más cómodo importar las páginas después, pueden importarse en una sola línea sin problemas:

```
//importar nuestras propias paginas
import { Login, Register } from './pages';
```

Obtenemos lo siguiente:
http://localhost:3000/

http://localhost:3000/login

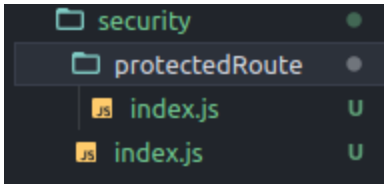
http://localhost:3000/register



Repetimos el proceso para una página “welcome”.

Sistema de Login

Lo primero es definir una carpeta security con la siguiente estructura:



el contenido de protectedRote/index.js es:

```
import { useState } from 'react';
import { Outlet, Navigate } from 'react-router-dom';

const ProtectedRoute = () => {
   const [isAuth, setIsAuth] = useState(true);

   return isAuth? <Outlet />: <Navigate to='/login' />;
}

export default ProtectedRoute;
```

Se hace uso del security/index.js para facilitar la importación, según lo visto anteriormente, se reescribe el archivo src/index.js de la siguiente manera:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

//importar los iconos de fontanswre
import '@fortawesome/fontawesome-free/css/all.min.css';

//bootstrap
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle';

//rutas
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

//importar nuestras propias paginas
import { Login, Register, Welcome} from './pages';
import {ProtectedRoute} from './security';
```

```
const router = createBrowserRouter([
 {
   path:"/",
   element: <App />
 },
 {
   path:"/login",
   element: <Login />
 },
 {
   path:"/register",
   element: <Register />
 },
 {
   path:"/",
   element: <ProtectedRoute />,
   children:[
     {
       path:"/welcome",
       element:<Welcome />,
     }
   ]
 }
]);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
   <RouterProvider router={router} />
 </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Nótese que las rutas protegidas por el momento es solamente welcome, como el isAuth es verdadero, entonces deja entrar de forma correcta:

http://localhost:3000/welcome

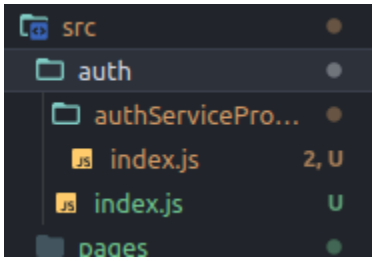
Sin embargo, si hacemos el siguiente cambio en security/protectedRoute/index.js:

```
   const [isAuth, setIsAuth] = useState(false);
```

Nótese que la ruta nos redirigirá a la página de login.

Ahora, se debe crear un estado global para manejar la variable de auth, lo haremos mediante un elemento AuthServiceProvider, de la siguiente forma:

Creamos la carpeta src/auth y seguimos la siguiente estructura:



Dentro del authSeviceProvider/index.js tenemos:

```
import React, { useContext, createContext, useState, useEffect } from
"react";

const AuthContext = createContext({
   isAutenticated: false
});

const AuthServiceProvider = ({children}) => {
   const[isAutenticated, setIsAutenticated] = useState(false);

   return (
       <AuthContext.Provider value={{isAutenticated}}>
           {children}
       </AuthContext.Provider>
   );
}

const useAuth = () => useContext(AuthContext);
```

```
export {AuthServiceProvider, useAuth};
```

useAuth ahora es un hooke que utilizaremos para manejar el inicio de sesión, lo implementamos en el protectedRoute definido anteriormente de la siguiente manera:

```
import { useState, useEffect } from 'react';
import { Outlet, Navigate } from 'react-router-dom';
import { useAuth } from '../../providers';

const ProtectedRoute = () => {
 const auth = useAuth();
 const [authChecked, setAuthChecked] = useState(false);

 useEffect(() => {
   const checkAuthentication = async () => {
     await auth.checkAuth();
     setAuthChecked(true);
   };

   checkAuthentication();
 }, []);

 if (!authChecked) {
   return null;
 }

 return auth.isAutenticated ? <Outlet /> : <Navigate to='/login' />;
};

export default ProtectedRoute;
```

cambiamos el src/index.js para hacer la implementación de la siguiente forma:

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
   <AuthServiceProvider>
     <RouterProvider router={router} />
   </AuthServiceProvider>
 </React.StrictMode>
```

```
);
```

El resultado es el mismo que el ejercicio hecho con el protected route, solo que ahora tenemos el hooke useAuth que nos permite controlar la sesión.

Ahora, para mantener la sesión iniciada se usa un token que se guarda en el navegador, estos tokens deberían ser guardados o gestionados con la base de datos, para que este sea único para cada usuario, sin embargo, para este ejercicio usaremos únicamente el nombre de usuario, de forma que se reescribe el AuthService provider de la siguiente forma.

```
import React, { useContext, createContext, useState, useEffect} from
"react";

const AuthContext = createContext({
   isAutenticated: false,
   getAccessToken: () => {},
   saveUser: (usuario) => {},
   getRefreshToken: () => {},
   signout: () => {}
});

const AuthServiceProvider = ({children}) => {
   const[isAutenticated, setIsAutenticated] = useState(false);
   const[accessToken, setAccessToken] = useState("");

   function checkAuth(){
       if(accessToken){
           //si el usuario está autenticado
           saveUser(accessToken);
       }else{
           //si el usuario no está autenticado
           const token = getRefreshToken();
           if(token){
               saveUser(token);
           }else{
               setIsAutenticated(false);
           }
       }
   }
```

```
    useEffect(()=>{
        checkAuth();
    });

    function getAccessToken(){
        return accessToken;
    }

    function getRefreshToken(){
        const token = localStorage.getItem("token");
        if(token){
            return token;
        }

        return null;
    }

    function saveUser(usuario){
        setAccessToken(usuario);

        localStorage.setItem("token", usuario);

        setIsAutenticated(true);
    }

    function signout() {
        localStorage.removeItem("token");
        setAccessToken("");
    }

    return (
        <AuthContext.Provider value={{isAutenticated, getAccessToken,
saveUser, getRefreshToken, signout}}>
            {children}
        </AuthContext.Provider>
    );
}

const useAuth = () => useContext(AuthContext);
```

```
export {AuthServiceProvider, useAuth};
```

Ahora las sesiones que se inicien, podrán permanecer aún cuando se reinicie el navegador.

El login requiere otras validaciones adicionales, por lo cual se adjunta el código completo que soporta dichas validaciones, cabe resaltar que, al ser un ejercicio de práctica, meramente educativo, no se es tan riguroso con las validaciones.

AuthServiceProvider:

```
import React, { useContext, createContext, useState } from "react";
import axios from "axios";

const AuthContext = createContext({
   user:{},
   isAutenticated: false,
   getAccessToken: () => {},
   saveUser: (usuario) => {},
   getRefreshToken: () => {},
   finalizarSesion: () => {},
   iniciarSesion: async (username, passwd) => {},
   checkAuth: async () => {},
});

const AuthServiceProvider = ({children}) => {
   const[isAutenticated, setIsAutenticated] = useState(false);
   const[accessToken, setAccessToken] = useState("");
   const [refreshToken, setRefreshToken] = useState("");
   const[user, setUser] = useState({});

   const url = "http://localhost:9000/usuarios";

   async function checkAuth(){
       if(accessToken){
           //si el usuario está autenticado
           const u = await getUser(accessToken);
           if(u){
               saveUser(u);
           }
       }else{
           //si el usuario no está autenticado
```

```
        const token = getRefreshToken();
        if(token){
            const u = await getUser(token);
            if(u){
                saveUser(u);
            }
        }else{
            setIsAutenticated(false);
        }
    }

    return;
}

const getUser = async (username) => {
    const res = await axios.get(`${url}/buscar/nombre/${username}`);
    if(res.data !== undefined){
        return res.data;
    }

    return null;
}

function getAccessToken(){
    return accessToken;
}

function getRefreshToken(){
    if (refreshToken) {
        return refreshToken;
    }

    const token = localStorage.getItem("token");
    if(token){
        setRefreshToken(token);
        return token;
    }

    return null;
}
```

```
    function saveUser(usuario){
        setAccessToken(usuario.usuario);

        localStorage.setItem("token", usuario.usuario);

        setUser(usuario);
        setIsAutenticated(true);
    }

    async function iniciarSesion(username, passwd){
        const u = await getUser(username);
        if(u){
            if(u.passwd === passwd){
                saveUser(u);
                return;
            }else{
                setIsAutenticated(false);
                return "passwd";
            }
        }else{
            setIsAutenticated(false);
            return "usuario";
        }
    }

    function finalizarSesion(){
        localStorage.removeItem("token");

        setAccessToken("");
        setRefreshToken("");
        setUser(undefined);
        setIsAutenticated(false);
    }

    return (
        <AuthContext.Provider value={{user, isAutenticated, getAccessToken,
saveUser, getRefreshToken, finalizarSesion, iniciarSesion, checkAuth}}>
            {children}
        </AuthContext.Provider>
```

```
    );
}

const useAuth = () => useContext(AuthContext);

export {AuthServiceProvider, useAuth};
```

Pages/login:

```
import { useState } from 'react';
import {DefaultLayout} from '../../common'
import {useAuth} from '../../auth';
import {Navigate} from 'react-router-dom';

const Login = () => {

    const [username, setUsername] = useState("");
    const [passwd, setPasswd] = useState("");
    const [usError, setUsError] = useState("form-control");
    const [passError, setPassError] = useState("form-control");
    const [errUsMg, setErrUsMg] = useState("");
    const [errPasswdMg, setErrPasswdMg] = useState("");

    const auth = useAuth();

    if(auth.isAutenticated){
        return <Navigate to='/welcome'></Navigate>;
    }

    const iniciarSesion = async () => {
        if(username !== ""){
            if(passwd !== ""){
                const res = await auth.iniciarSesion(username, passwd);
                if(res==="usuario"){
                    setUsError("form-control is-invalid");
                    setErrUsMg("El Usuario no Existe!");
                }

                if(res==="passwd"){
                    setPassError("form-control is-invalid");
                    setErrPasswdMg("Contraseña Incorrecta!");
```

```
                }
            }else{
                setPassError("form-control is-invalid");
                setErrPasswdMg("La Contraseña es Requerida!");
            }
        }else{
            setUsError("form-control is-invalid");
            setErrUsMg("El Usuario es Requerido!");
        }
    }

    return(
        <DefaultLayout>
            <div className="container mt-5 pt-5">
                <div
                    className="row justify-content-center
align-items-center g-2"
                >
                    <div className="col-12 col-md-8">
                        <div className="card">
                            <h3 className="card-title text-center
mt-5">Login</h3>
                            <div className="card-body">
                                <div className="row
justify-content-center">
                                    <div className="col-12">
                                        <div className="row">
                                            <div className="input-group
has-validation">
                                                <span

                                                    id="inputGroupUsername"
                                                >Usuario @</span>
                                                <input
                                                    type="text"

                                                    id="validationUsername"
```

```
aria-describedby="inputGroupUsername validationUsernameFeedback"
                                                    required
                                                    value={username}
                                                    onChange={(e) => {

setUsError("form-control");

setUsername(e.target.value);
                                                    }}
                                                />
                                                <div

id="validationUsernameFeedback"

className="invalid-feedback"
                                                >
                                                    {errUsMg}
                                                </div>
                                            </div>
                                        </div>
                                        <div className="mb-4 row mt-2">
                                            <div className="input-group
has-validation">
                                                <span

className="input-group-text"
                                                    id="inputGroupUsername"
                                                >Contraseña </span>
                                                <input
                                                    type="password"

className={`${passError}`}
                                                    id="validationPasswd"

aria-describedby="inputGroupPasswd validationPasswdFeedback"
                                                    required
                                                    value={passwd}
                                                    onChange={(e) => {
```

```
setPassError("form-control");

setPasswd(e.target.value);
                                                    }}
                                                />
                                                <div

id="validationPasswdFeedback"

className="invalid-feedback"
                                                >
                                                    {errPasswdMg}
                                                </div>
                                            </div>
                                        </div>
                                        <div className="row">
                                            <button
                                                className="btn btn-success
w-80"

                                                onClick={(e) =>
iniciarSesion()}
                                            >
                                                Login
                                            </button>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </DefaultLayout>
    );
}

export default Login;
```

Vista de la Lista de Mascotas

Por conveniencia esta lista de mascotas se verá en la vista de las mascotas.

Mascotas

pages/mascotas/index.js

```
import { useAuth, useMascotas } from "../../providers";
import { DefaultLayout } from "../../common";
import { useEffect, useState } from "react";
import { MascotaComponent, MascotasFormComponent } from
"../../components";

const Mascotas = () => {
   //providers
   const auth = useAuth();
   const mascotasProvider = useMascotas();

   //listas
   const [mascotas, setMascotas] = useState([]);

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

   const getMascotas = async() => {
       const res = await mascotasProvider.getMascotas();
       if(res){
           setMascotas(res);
       }
   }

   return(
       <DefaultLayout>
           <div className="mt-5 pt-4">
               <div className="container">
                   {auth.user.rol==="administrador"?(
                       <div className="row">
                           <div className="col">
                               <button
                                   type="button"
```

```
                                    className="btn btn-outline-success
w-100 mb-3"
                                    data-bs-toggle="modal"

data-bs-target={`#mascota-formulario-Agregar-`}
                                >
                                    <i className="fa-solid fa-add"></i>
                                </button>
                            </div>
                        </div>
                    ):""
                    }
                    <div className="row row-cols-1 row-cols-sm-2
row-cols-lg-3">
                        {
                            mascotas.map((mascota) => (
                                <div className="col mb-5"
key={`maascota${mascota.pk}`}>
                                    <MascotaComponent
                                        {...mascota}
                                    />
                                </div>
                            ))
                        }
                    </div>
                </div>
            </div>

            <MascotasFormComponent
                {...{funcion:"Agregar"}}
            />

        </DefaultLayout>
    );
}

export default Mascotas;
```

Se construye un componente que se encarga de renderizar las características de cada mascota:

MacotaComponent

components/mascota/index.js

```
import { Link } from "react-router-dom";
import { useAuth, useSolicitudes } from "../../providers";
import { useState } from "react";
import { MascotasFormComponent } from '../'

const MascotaComponent = ({pk, nombre, foto, rasa, edad, tipo_mascota,
estado}) => {
   const [mensaje, setMensaje] = useState("");
   const [alertType, setAlertType] = useState("");

   const user = useAuth().user;
   const solicitudesProvider = useSolicitudes();


   const enviarSolicitud = async() => {
       if(pk && user){
           const hoy = new Date();
           const fecha_inicio = hoy.toISOString().split('T')[0];
           const dataset = {
               mascotaPK:pk,
               adoptante:user.pk,
               estado:'P',
               fecha_inicio:fecha_inicio
           }
           const res = await solicitudesProvider.enviarSolicitud(dataset);

           if(res === "success"){
               setMensaje("Solicitud Enviada con Exito");
               setAlertType("alert alert-success");
           }else{
               setMensaje("No se ha Podido realizar la Solicitud");
               setAlertType("alert alert-danger");
           }
       }else{
           setMensaje("No se ha Podido realizar la Solicitud");
           setAlertType("alert alert-danger");
       }
```

```
    }

    return (
        <div className="card h-100">
            {
                foto?(
                    <img src={foto} height={180} className="card-img-top"
alt="Foto de la mascota" />
                ):null
            }
            <div className="card-body">
                <h5 className="card-title">{nombre}</h5>
                <hr />
                <p className="card-text text-secondary">
                    {
                        `${tipo_mascota==='P'?"Perro":"Gato"}`+
                        ` ${rasa?`de raza ${rasa}`:" que no registra
raza"}`+
                        `${edad?` con ${edad} años de edad`:" y no registra
edad"}`+
                        "."
                    }
                </p>
                <hr />
                {
                    estado===1?(
                        <div className="row">
                            <div className="alert alert-success
text-center" role="alert">
                                Esta mascota ya fue adoptada
                            </div>
                            < hr />
                        </div>
                    ):""
                }
                {
                    user.rol==="administrador"?(
                        <div className="row">
                            <div className="col">
                                <Link
```

```
                                    className="btn btn-outline-secondary
w-100 mb-2"
                                    to={`/mascota/${pk}`}
                                >
                                    <i className="fa-solid
fa-list"></i>
                                </Link>
                            </div>
                        </div>
                    ):""
                }
                <div className="row row-cols-1 row-cols-sm-2">
                    {
                        user.rol==="administrador"?(
                        <>

                        </>):""
                    }{
                        user.rol==="usuario"?(
                            <>
                                <div className="col">
                                    <Link
                                        className="btn
btn-outline-secondary w-100"
                                        to={`/mascota/${pk}`}
                                    >
                                        Detalles
                                    </Link>
                                </div>
                                <div className="col">
                                    <button
                                        type="button"
                                        className="btn btn-outline-success
w-100"
                                        data-bs-toggle="modal"

data-bs-target={`#mascota-adopcion${pk}`}
                                    >
                                       Adoptar
                                    </button>
```

```
                                </div>
                            </>
                        ):(
                            <>
                                <div className="col">
                                    <button
                                        type="button"
                                        className="btn btn-outline-warning
w-100"
                                        data-bs-toggle="modal"

data-bs-target={`#mascota-formulario-Editar-${pk}`}
                                    >
                                       <i className="fa-solid fa-edit"></i>
                                    </button>
                                </div>
                                <div className="col">
                                    <button
                                        type="button"
                                        className="btn btn-outline-danger
w-100"
                                        data-bs-toggle="modal"

data-bs-target={`#mascota-formulario-Eliminar-${pk}`}
                                    >
                                       <i className="fa-solid
fa-trash"></i>
                                    </button>
                                </div>
                            </>
                        )
                    }
                </div>
            </div>

            <div
                className="modal fade"
                id={`mascota-adopcion${pk}`}
                tabIndex="-1"
                aria-labelledby={`modalAdoptarMactota${pk}`}
```

```
                aria-hidden="true"
            >
                <div className="modal-dialog">
                    <div className="modal-content">
                        <div className="modal-header">
                            <h1
                                className="modal-title fs-5"
                                id={`modalTituloAdoptarMascota${pk}`}
                            >
                                    Adoptar
                            </h1>
                            <button
                                type="button"
                                className="btn-close"
                                data-bs-dismiss="modal"
                                aria-label="Close"
                            ></button>
                        </div>
                        <div className="modal-body">
                            <div
                                className={`${alertType}`}
                                role="alert"
                            >
                                {mensaje}
                            </div>
                            {
                                estado===1?(
                                    <p className="text-center">
                                        Lo sentimos {nombre} ya fue
adoptado
                                    </p>
                                ):(
                                    <p className="text-center">
                                         ¿Desea adoptar a {nombre} de
{edad} años?
                                    </p>
                                )
                            }
                        </div>
                        <div className="modal-footer">
```

```
                            <button
                                type="button"
                                className="btn btn-secondary"
                                data-bs-dismiss="modal"
                            >Cancelar</button>
                            {
                                estado===0?(
                                    <button
                                        type="button"
                                        className="btn btn-success"
                                        onClick={(e) => enviarSolicitud()}
                                    >Adoptar</button>
                                ):""
                            }
                        </div>
                    </div>
                </div>
            </div>

            <MascotasFormComponent
                {...{pk, funcion:"Editar"}}
            />

            <MascotasFormComponent
                {...{pk, funcion:"Eliminar"}}
            />

        </div>
    );
}

export default MascotaComponent;
```

Buscador

Para implementar el buscador se necesita implementar la siguiente función:

```
const buscar = async (busqueda) => {
        if(busqueda && mascotas){
            setBusqueda(mascotas.filter(mascota => (

(mascota.nombre.toLowerCase().indexOf(busqueda.toLowerCase())) !== -1) ||
```

```
(mascota.rasa.toLowerCase().indexOf(busqueda.toLowerCase()) !== -1) ||
                    (mascota.edad.toString().indexOf(busqueda) !== -1)
                )
            );
        }else{
            getMascotas();
        }
    }
```

ahora manejamos una constante auxiliar busqueda, asi:

```
const [mascotas, setMascotas] = useState([]);
    const [busqueda, setBusqueda] = useState([]);
```

debe cargarse en el getMascotas:

```
const getMascotas = async() => {
        const res = await mascotasProvider.getMascotas();
        if(res){
            setMascotas(res);
            setBusqueda(res);
        }
    }
```

y reemplazada en el .map que recorre las mascotas:

```
{
                            busqueda?(
                                busqueda.map((mascota) => (
                                    <div className="col mb-5"
key={`maascota${mascota.pk}`}>
                                            <MascotaComponent
                                                {...mascota}
                                            />
                                    </div>
                                ))
                            ):(
                                <div className="col mb-5" >
```

```
                                    <p>No hay ninguna Mascota
Registrada</p>
                                </div>
                            )
                        }
```

MascotaDetail

```
import { useEffect, useState } from "react";
import { DefaultLayout } from "../../common";
import { useMascotas } from "../../providers";
import { useParams } from "react-router-dom";
const { Link } = require("react-router-dom");

const MascotaDetail = () => {
   const mascotasService = useMascotas();
   const [mascota, setMascota] = useState({});
   const pk = useParams().id;

   useEffect(()=> {
       getMascota();
   }, []);

   const getMascota  = async () => {
       const res = await mascotasService.buscarMascota(pk);
       setMascota(res);
   }

   return(
       <DefaultLayout>
           <div className="container  mt-5 pt-5">
               <div className="card">
                   <div className="card-body">
                       {
                           mascota?(
                               <div className="row row-cols-1
row-cols-md-2">
                                   <div className="col">
                                       <img
                                           src={mascota.foto}
```

```
                                        className="img-fluid
rounded-top"
                                        style={{maxHeight:350}}
                                        alt={`Imagen
${mascota.nombre}`}
                                    />
                                </div>
                                <div className="col">
                                    <h5>
                                        Nombre: {

mascota.nombre?mascota.nombre:"No definido"
                                                }
                                    </h5>
                                    <p className="text-secondary">
                                        {

mascota.tipo_mascota==='P'?"Perro":"Gato"
                                        }
                                    </p>
                                    <p>Raza: {

mascota.rasa?mascota.rasa:"No definida"
                                            }
                                    </p>
                                    <p>Edad: {

mascota.edad?mascota.edad:"No definida"
                                        } años
                                    </p>
                                    <hr />
                                    <p>{

mascota.descripcion?mascota.descripcion:"No hay descripcion"
                                    }</p>
                                </div>
                            </div>
                        ):(
                            <div className="row">
```

```
                                    <p>No se ha podido encontrar la
mascota!</p>
                                </div>
                            )
                        }
                        <div className="row">
                            <div className="col-sm-12 col-lg-10"></div>
                            <div className="col-sm-12 col-lg-2">
                                <Link className="btn btn-outline-dark
w-100" to="/mascotas">
                                    Regresar
                                </Link>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </DefaultLayout>
    );
}

export default MascotaDetail;
```

Al aislar cada mascota en un componente, podemos acceder a su detail page mediante la llave primaria, pues el componente es un segmento apartado para cada elemento.

Formulario Para mascotas

Finalmente pasamos a la construcción del formulario para las mascotas, como este formulario se utilizará para agregar, editar y eliminar mascotas, lo construiremos como un formulario dinámico y será un componente.

```
import { useEffect, useState } from "react";
import { useMascotas } from "../../providers";
import { mostrarAlerta } from "../../functions";

const MascotasFormComponent = ({pk, funcion}) => {
    //providers
    const mascotasService = useMascotas();
```

```
//la mascota si es que existe
const [mascota, setMascota] = useState({});

//atributos
const [nombre, setNombre] = useState("");
const [foto, setFoto] = useState("");
const [descripcion, setDescripcion] = useState("");
const [rasa, setRasa] = useState("");
const [edad, setEdad] = useState(0);
const [tipo, setTipo] = useState('P');
const [estado, setEstado] = useState(0);

const [alert, setAlert] = useState('');
const [alertType, setAlertType] = useState('');

//control de errores
const msg = "El Valor Digitado no es Válido";

useEffect(()=> {
    if(pk){
        obetenerMascota();
    }
},[]);

const obetenerMascota = async () => {
    try {
        const res = await mascotasService.buscarMascota(pk);

        if (res) {
            setMascota(res);
            setNombre(res.nombre);
            setFoto(res.foto);
            setDescripcion(res.descripcion);
            setRasa(res.rasa);
            setEdad(res.edad);
            setTipo(res.tipo_mascota);
            setEstado(res.estado);
        }
    } catch (error) {
```

```
            console.error("Error al obtener la mascota:", error);
        }
    };

    const actualizarMascota = async () => {
        if(validarCampos()){
            const dataset = {
                nombre:nombre,
                foto:foto,
                descripcion:descripcion,
                rasa:rasa,
                edad:edad,
                tipo_mascota:tipo,
                estado:estado
            }
            const res = await mascotasService.actualizarMascota(pk,
dataset);

            if(res === "success"){
                mostrarAlerta("Mascota Actualizada con Exito","success");
            }else{
                mostrarAlerta("No se ha Podido Actualizar la
Mascota","error");
            }
        }
    }

    const agregarMascota = async () => {
        if(validarCampos()){
            const dataset = {
                nombre:nombre,
                foto:foto,
                descripcion:descripcion,
                rasa:rasa,
                edad:edad,
                tipo_mascota:tipo,
                estado:estado
            }

            const res = await mascotasService.crearMascota(dataset);
```

```
        if(res === "success"){
            mostrarAlerta("Mascota Creada con Exito","success");
        }else{
            mostrarAlerta("No se ha Podido crear la Mascota", "error");
        }
    }
}

const eliminarMascota = async () =>{
    const res = await mascotasService.eliminarMascota(pk);

    if(res === "success"){
        mostrarAlerta("Mascota Eliminada Exito","success");
    }else{
        mostrarAlerta("No se ha Podido Eliminar la Mascota", "error");
    }
}

//cuando un campo está vacio
const campoVacio = (valor) => {
    if(valor===""){
        return true;
    }
    return false;
}

const validarEdad = (edad) => {
    if(!edad){
        return false;
    }

    if(edad < 0){
        return false;
    }

    return true;
}

const validarCampos = () => {
```

```
        if(nombre === "" || foto === "" || descripcion === "" || rasa ===
"" || edad < 0 || !edad){
            return false;
        }

        return true;
    }

    return(
        <div
            className="modal fade"
            id={`mascota-formulario-${funcion}-${pk?pk:""}`}
            tabIndex="-1"

aria-labelledby={`modalForumularioMactota-${funcion}-${pk?pk:""}`}
            aria-hidden="true"
        >
            <div className="modal-dialog">
                <div className="modal-content">
                    <div className="modal-header">
                        <h1
                            className="modal-title fs-5"

id={`modalTituloFormularioMascota-${funcion}-${pk?pk:""}`}
                        >
                                {funcion} Mascota
                        </h1>
                        <button
                            type="button"
                            className="btn-close"
                            data-bs-dismiss="modal"
                            aria-label="Close"
                        ></button>
                    </div>
                    <div className="modal-body">
                        {
                            alertType!==""?(
                                <div className="row">
                                    <div className={`alert
alert-${alertType} text-center`} role="alert">
```

```
                                        {alert}
                                    </div>
                                    < hr />
                                </div>
                            ):""
                        }
                        {
                            funcion==="Eliminar"?(
                                <p className="text-center"> ¿Está seguro de
querer eliminar a {mascota.nombre?mascota.nombre:""} ?</p>
                            ):(
                                <div className="row">
                                    <div className="mb-3 input-group
has-validation">
                                        <label

htmlFor={`nombreFormularioMascota-${funcion}-${pk?pk:""}`}
                                            className="form-label w-100"
                                        >Nombre</label>
                                        <input
                                            type="text"
                                            className={`form-control
${campoVacio(nombre)?"is-invalid":"is-valid"}`}

id={`nombreFormularioMascota-${funcion}-${pk?pk:""}`}

aria-describedby={`nombreGroup-${funcion}-${pk?pk:""}
nombreFeedback-${funcion}-${pk?pk:""}`}
                                            required
                                            placeholder="Firulais"
                                            value={nombre}
                                            onChange={(e) =>
{setNombre(e.target.value); }}
                                        />
                                        <div

id={`nombreFeedback-${funcion}-${pk?pk:""}`}
                                            className="invalid-feedback"
                                        >
                                            {msg}
```

```
                                        </div>
                                    </div>
                                    <div className="mb-3 input-group
has-validation">
                                        <label

htmlFor={`fotoFormularioMascota-${funcion}-${pk?pk:""}`}
                                            className="form-label w-100"
                                        >Url Foto</label>
                                        <input
                                            type="text"
                                            className={`form-control
${campoVacio(foto)?"is-invalid":"is-valid"}`}

id={`fotoFormularioMascota-${funcion}-${pk?pk:""}`}

placeholder="https://una/imagen/de/internet.jpg"

aria-describedby={`fotoGroup-${funcion}-${pk?pk:""}
fotoFeedback-${funcion}-${pk?pk:""}`}
                                            required
                                            value={foto}
                                            onChange={(e) =>
{setFoto(e.target.value);}}
                                        />
                                        <div

id={`fotoFeedback-${funcion}-${pk?pk:""}`}
                                            className="invalid-feedback"
                                        >
                                            {msg}
                                        </div>
                                    </div>
                                    <div className="mb-3">
                                        <label

htmlFor={`descripcionFormularioMascota-${funcion}-${pk?pk:""}`}
                                            className="form-label w-100"
                                        >Descripción</label>
                                        <textarea
```

```
                                        className={`form-control
${campoVacio(descripcion)?"is-invalid":"is-valid"}`}

id={`descripcionFormularioMascota-${funcion}-${pk?pk:""}`}

aria-describedby={`descGroup-${funcion}-${pk?pk:""}
descFeedback-${funcion}-${pk?pk:""}`}
                                        rows="3"
                                        placeholder="Es un perro muy
bonito..."
                                        value={descripcion}
                                        onChange={(e) =>
{setDescripcion(e.target.value);}}
                                    ></textarea>
                                    <div

id={`descFeedback-${funcion}-${pk?pk:""}`}
                                        className="invalid-feedback"
                                    >
                                        {msg}
                                    </div>
                                </div>
                                <div className="mb-3">
                                    <label

htmlFor={`tipoFormularioMascota-${funcion}-${pk?pk:""}`}
                                        className="form-label"
                                    >Tipo de Mascota</label>
                                    <select
                                        className="form-select
form-select mb-3 is-valid"

id={`tipoFormularioMascota-${funcion}-${pk?pk:""}`}
                                        value={tipo}
                                        onChange={(e) =>
setTipo(e.target.value)}
                                    >
                                        <option value='P'
>Perro</option>
                                        <option value='G'>Gato</option>
```

```
                                    </select>
                                </div>
                                <div className="mb-3">
                                    <label

htmlFor={`razaFormularioMascota-${funcion}-${pk?pk:""}`}
                                        className="form-label"
                                    >Raza</label>
                                    <input
                                        type="text"
                                        className={`form-control
${campoVacio(rasa)?"is-invalid":"is-valid"}`}

id={`razaFormularioMascota-${funcion}-${pk?pk:""}`}

aria-describedby={`rasaGroup-${funcion}-${pk?pk:""}
rasaFeedback-${funcion}-${pk?pk:""}`}
                                        placeholder="Pastor Aleman"
                                        value={rasa}
                                        onChange={(e) =>
setRasa(e.target.value)}
                                    />
                                    <div

id={`rasaFeedback-${funcion}-${pk?pk:""}`}
                                        className="invalid-feedback"
                                    >
                                        {msg}
                                    </div>
                                </div>
                                <div className="mb-3">
                                    <label

htmlFor={`edadFormularioMascota-${funcion}-${pk?pk:""}`}
                                        className="form-label"
                                    >Edad en Años</label>
                                    <input
                                        type="number"
                                        className={`form-control
${!validarEdad(edad)?"is-invalid":"is-valid"}`}
```

```
id={`edadFormularioMascota-${funcion}-${pk?pk:""}`}

aria-describedby={`edadGroup-${funcion}-${pk?pk:""}
edadFeedback-${funcion}-${pk?pk:""}`}
                                        value={edad}
                                        onChange={(e) =>
setEdad(e.target.value)}
                                    />
                                    <div

id={`edadFeedback-${funcion}-${pk?pk:""}`}
                                        className="invalid-feedback"
                                    >
                                        {msg}
                                    </div>
                                </div>
                                <div className="mb-3">
                                    <label

htmlFor={`estadoFormularioMascota-${funcion}-${pk?pk:""}`}
                                        className="form-label"
                                    >Estado de la Mascota</label>
                                    <select
                                        className="form-select
form-select mb-3 is-valid"

id={`estadoFormularioMascota-${funcion}-${pk?pk:""}`}
                                        value={estado}
                                        onChange={(e) =>
setEstado(e.target.value)}
                                    >
                                        <option value={0} >Sin
Adoptar</option>
                                        <option value={1}
>Adoptado</option>
                                    </select>
                                </div>
                            </div>
                        )
```

```
                        }{
                            funcion===="Editar"?(
                                <div className="modal-footer">
                                    <button
                                        type="button"
                                        className="btn btn-secondary"
                                        data-bs-dismiss="modal"
                                        onClick={(e) => {
                                            setAlertType("");
                                            setAlert("");
                                        }}
                                    >Cancelar</button>
                                    <button
                                            type="button"
                                            className="btn btn-success"
                                            onClick={(e) =>
actualizarMascota()}
                                    >Guardar Cambios</button>
                                </div>
                            ):""
                        }{
                            funcion===="Agregar"?(
                                <div className="modal-footer">
                                    <button
                                        type="button"
                                        className="btn btn-secondary"
                                        data-bs-dismiss="modal"
                                        onClick={(e) => {
                                            setAlertType("");
                                            setAlert("");
                                        }}
                                    >Cancelar</button>
                                    <button
                                            type="button"
                                            className="btn btn-success"
                                            onClick={(e) =>
agregarMascota()}
                                    >Crear Mascota</button>
                                </div>
                            ):""
```

```
                        }{
                            funcion===="Eliminar"?(
                                <div className="modal-footer">
                                    <button
                                        type="button"
                                        className="btn btn-success"
                                        data-bs-dismiss="modal"
                                        onClick={(e) => {
                                            setAlertType("");
                                            setAlert("");
                                        }}
                                    >Cancelar</button>
                                    <button
                                            type="button"
                                            className="btn btn-danger"
                                            onClick={(e) =>
eliminarMascota()}
                                    >Eliminar Mascota</button>
                                </div>
                            ):""
                        }
                    </div>
                </div>
            </div>
        </div>
    );
}

export default MascotasFormComponent;
```

Solicitudes

Solicitudes recorre el array de solicitudes y por cada solicitud crea un componente solicitud:

```
import { useEffect, useState } from "react";
import { DefaultLayout } from "../../common";
import { useAuth, useSolicitudes } from "../../providers";
import { SolicitudComponent } from "../../components";

const Solicitudes = () => {
    const user = useAuth().user;
```

```
    const solicitudesService = useSolicitudes();

    const [solicitudes, setSolicitudes] = useState([]);

    useEffect(() => {
        inicializar();
    } ,[]);

    const inicializar = async () => {
        if(user.rol === "administrador"){
            const res = await solicitudesService.getSolicitudes();
            if(res){
                setSolicitudes(res);
            }
        }else if(user.rol === "usuario"){
            const res = await
solicitudesService.buscarSolicitudesUsuario(user.pk);
            if(res){
                setSolicitudes(res);
            }
        }
    }

    return (
        <DefaultLayout>
            <div className="mt-5 pt-4">
                <div className="container">
                    <div className="row mt-2 mb-2">
                        <h4 className="text-center"> Solicitudes de
Adopción </h4>
                    </div>
                    <div className="row">
                        {
                            solicitudes?(
                                solicitudes.map((solicitud, index) => (
                                    <SolicitudComponent
                                    key={solicitud.pk}
                                        {...{index, solicitud}}
                                    />
                                ))
```

```
                            ):""
                        }
                    </div>
                </div>
            </div>
        </DefaultLayout>
    );
}

export default Solicitudes;
```

Solicitud:

```
import { useEffect, useState } from "react";
import { useAuth, useMascotas, useSolicitudes, useUsuarios } from
"../../providers";
import { mostrarAlerta } from "../../functions";

const { Link } = require("react-router-dom");

const SolicitudComponent = ({index, solicitud}) => {
    const mascotasService = useMascotas();
    const solicitudesService = useSolicitudes();
    const usersProvider = useUsuarios();
    const user = useAuth().user;

    const [mascota, setMascota] = useState({});
    const [accion, setAccion] = useState('');
    const [persona, setPersona] = useState({});

    useEffect(() => {
        getPersona();
        getMascota();
    }, []);

    const getMascota = async () => {
        if(solicitud.mascotaPK){
            const res = await
mascotasService.buscarMascota(solicitud.mascotaPK);
            if(res){
```

```
                setMascota(res);
            }
        }
    }

    const getPersona = async () => {
        if(solicitud.adoptante){
            const res = await
usersProvider.buscarUsuario(solicitud.adoptante);
            if(res){
                setPersona(res);
            }
        }
    }

    const realizarAccion = async () => {
        if(accion==="Eliminar"){
            await cancelarSolicitud();

            return;
        }

        if(accion==="Rechazar"){
            await rechazarSolicitud();

            return;
        }

        if(accion==="Aceptar"){
            await aceptarSolicitud();

            return;
        }
    }

    const aceptarSolicitud = async () => {
        if(solicitud.pk && solicitud.estado === 'P'){
            const res = await
mascotasService.actualizarMascota(solicitud.mascotaPK, {estado:1});
            if(res === "success"){
```

```
                const hoy = new Date();
                const fecha_fin = hoy.toISOString().split('T')[0];

                const dataset = {
                    estado:"A",
                    fecha_fin:fecha_fin
                }

                const sol = await
solicitudesService.actualizarSolicitud(solicitud.pk, dataset);

                if(sol === "success"){
                    mostrarAlerta("Solicitud aceptada con
Exito","success");
                }else{
                    mostrarAlerta("No se puede Adoptar esa
Mascota","error");
                }
            }else{
                mostrarAlerta("No se puede Adoptar esa Mascota","error");
            }
        }
    }

    const rechazarSolicitud = async () => {
        if(solicitud.pk && solicitud.estado === 'P'){
            const hoy = new Date();
                const fecha_fin = hoy.toISOString().split('T')[0];

                const dataset = {
                    estado:"R",
                    fecha_fin:fecha_fin
                }

            const res = await
solicitudesService.actualizarSolicitud(solicitud.pk, dataset);

            if(res === "success"){
                mostrarAlerta("Solicitud rechazada con Exito","success");
            }else{
```

```
                mostrarAlerta("No se ha Podido rechazar la
Solicitud","error");
            }
        }
    }

    const cancelarSolicitud = async () => {
        if(solicitud.pk && solicitud.estado === 'P'){
            const res = await
solicitudesService.eliminarSolicitud(solicitud.pk);

            if(res === "success"){
                mostrarAlerta("Solicitud eliminada con Exito","success");
            }else{
                mostrarAlerta("No se ha Podido eliminar la
Solicitud","error");
            }
        }
    }

    return (
        <div className={`    card mb-2
                             ${solicitud.estado==='P'?"bg-secondary":""}
                             ${solicitud.estado==='A'?"bg-success":""}
                             ${solicitud.estado==='R'?"bg-danger":""}
                         `}
        >
            <div className="card-header">
                <h4 className="card-title text-center"> {mascota.nombre}
</h4>
            </div>
            <div className="card-body">
                {
                    solicitud?(
                        <div>
                            <div className={`${mascota?"row":"d-none"}`}>
                                <p className="text-center">
                                    Tipo de Mascota:
{mascota.tipo_mascota==='G'?"Gato":""}
                                    {mascota.tipo_mascota==='P'?"Perro":""}
```

```
                                </p>
                                {
                                    user.rol==="administrador"?"":(
                                        <hr />
                                    )
                                }
                            </div>
                            <div
classNAme={`${user.rol==="administrador"?"row":"d-none"}`}>
                                {
                                    <p className="text-center">
                                        {
                                            persona?"Adoptante:
"+persona.nombres:""
                                        }
                                    </p>
                                }
                                <hr />
                            </div>
                            <div className="row row-cols-1 row-cols-md-2">
                                <div className="col mb-2 mt-2">
                                    <p>#{index+1}</p>
                                </div>
                                <div className="col mb-2">
                                    <p> Estado:
                                        {
                                            solicitud.estado === 'A'?"
Aceptada":""
                                        }
                                        {
                                            solicitud.estado === 'P'?"  En
Proceso":""
                                        }
                                        {
                                            solicitud.estado === 'R'?"
Rechazada":""
                                        }
                                    </p>
                                </div>
                                <div className="col mb-2">
```

```
                                    <p>Fecha de Inicio:
                                        {
                                            " "+solicitud.fecha_inicio
                                        }
                                    </p>
                                </div>
                                <div className="col mb-2">
                                    <p>Fecha de Finalización:
                                        {
                                            solicitud.fecha_fin?"
"+solicitud.fecha_fin:" En Proceso"
                                        }
                                    </p>
                                </div>
                            </div>
                            <div className={`row
${user.rol==="usuario"?"row-cols-1 row-cols-sm-2":""}`}>
                                <div className="col mb-3">
                                    {
                                        mascota?(
                                            <Link
                                                className="btn
btn-outline-warning w-100"

to={`/mascota/${mascota.pk}`}
                                            >
                                                Ver Mascota
                                            </Link>
                                        ):"Mascota no encontrada"
                                    }
                                </div>
                                <div className={`col
${user.rol==="usuario"?"":"d-none"}`}>
                                    {
                                        user.rol==="usuario"?(
                                            <button
                                                type="button"
                                                data-bs-toggle="modal"
                                                className={`btn btn-danger
w-100 ${solicitud.estado==='P'?"":"disabled"}`}
```

```
data-bs-target={`#modalSolicitud${solicitud.pk}`}
                                                onClick={(e) =>
setAccion("Eliminar")}
                                            >
                                                Eliminar Solicitud
                                            </button>
                                        ):"Mascota no encontrada"
                                    }
                                </div>
                            </div>
                            {
                                user.rol==="administrador" &&
solicitud.estado==="P"?(
                                    <div className="row row-cols-1
row-cols-sm-3">
                                        <div className="col mb-2">
                                            <button
                                                data-bs-toggle="modal"
                                                className={`btn btn-success
w-100`}

data-bs-target={`#modalSolicitud${solicitud.pk}`}
                                                onClick={(e) =>
setAccion("Aceptar")}
                                            >
                                                Aceptar Solicitud
                                            </button>
                                        </div>
                                        <div className="col mb-2">
                                            <button
                                                data-bs-toggle="modal"
                                                className={`btn btn-danger
w-100`}

data-bs-target={`#modalSolicitud${solicitud.pk}`}
                                                onClick={(e) =>
setAccion("Rechazar")}
                                            >
                                                Rechazar Solicitud
```

```
                                        </button>
                                    </div>
                                    <div className="col">
                                        <button
                                            data-bs-toggle="modal"
                                            className={`btn btn-danger
w-100`}

data-bs-target={`#modalSolicitud${solicitud.pk}`}
                                            onClick={(e) =>
setAccion("Eliminar")}
                                        >
                                            Eliminar Solicitud
                                        </button>
                                    </div>
                                </div>
                            ):""
                        }
                    </div>
                ):""
            }
        </div>

        <div
            className="modal fade"
            id={`modalSolicitud${solicitud.pk}`}
            tabIndex="-1"
            aria-labelledby={`modalLabelSolicitud${solicitud.pk}`}
            aria-hidden="true"
        >
            <div className="modal-dialog">
                <div className="modal-content">
                    <div className="modal-header">
                        <h1
                            className="modal-title fs-5"
                            id={`modalLabelSolicitud${solicitud.pk}`}
                        >
                            {accion?accion:""} Solicitud
                        </h1>
                        <button
```

```
                                type="button"
                                className="btn-close"
                                data-bs-dismiss="modal"
                                aria-label="Close"
                            ></button>
                        </div>
                        <div className="modal-body">
                            {
                                accion==="Eliminar"?(<p>
                                    ¿Está seguro de querer eliminar la
solicitud de la
                                    mascota {mascota?mascota.nombre:""}?
                                </p>):""
                            }{
                                accion==="Rechazar"?(<p>
                                    ¿Está seguro de querer rechazar la
solicitud de la
                                    mascota {mascota?mascota.nombre:""}?
                                </p>):""
                            }{
                                accion==="Aceptar"?(<p>
                                    ¿Está seguro de querer aceptar la
solicitud de la
                                    mascota {mascota?mascota.nombre:""}?
                                </p>):""
                            }
                        </div>
                        <div className="modal-footer">
                            <button
                                type="button"
                                className={`btn
                                            ${accion==="Aceptar"?"
btn-danger":" btn-success"}
                                        `}
                                data-bs-dismiss="modal"
                            >Atrás</button>
                            <button
                                type="button"
                                className={`btn
```

```
                                        ${accion==="Aceptar"?"
btn-success":" btn-danger"}
                                    `}
                                onClick={(e) => realizarAccion()}
                            >
                                {
                                    accion==="Eliminar"?"Confirmar
Eliminación":""
                                }
                                {
                                    accion==="Rechazar"?"Confirmar
Rechazo":""
                                }
                                {
                                    accion==="Aceptar"?"Confirmar
Aceptación":""
                                }
                            </button>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
}

export default SolicitudComponent;
```

Register:

Finalmente se adjunta la página de registro:

```
import { useState } from "react";
import { DefaultLayout } from "../../common";
import { useAuth, useUsuarios } from "../../providers";
import { mostrarAlerta } from "../../functions";
import { Navigate } from "react-router-dom";
const { Link }  = require('react-router-dom');

const Register = () => {
    const usuariosService = useUsuarios();
    const auth = useAuth();
```

```
    const [nombres, setNombres] = useState("");
    const [edad, setEdad] = useState(0);
    const [telefono, setTelefono] = useState("");
    const [userName, setUserName] = useState("");
    const [passwd, setPasswd] = useState("");
    const [alert, setAlert] = useState("");

    if(auth.isAutenticated){
        return <Navigate to="/mascotas" />;
    }

    const registrarUsuario = async () => {
        if(validarCampos()){
            const repetido = await
usuariosService.buscarUsuarioNombre(userName);
            console.log(repetido);
            if(!repetido){
                const dataset = {
                    nombres:nombres,
                    edad:edad,
                    telefono:telefono,
                    usuario:userName,
                    passwd:passwd,
                    rol:"usuario"
                }

                const res = await usuariosService.crearUsuario(dataset);
                if(res === "success"){
                    await auth.iniciarSesion(userName, passwd);

                    mostrarAlerta("Usuario creado con Exito","success");
                }else{
                    mostrarAlerta("No se ha podido registrar","error");
                }
            }else{
                setAlert("El nombre de usuario: "+userName+" ya fue
tomado");
            }
        }else{
```

```
            setAlert("Asegurese de diligenciar todos los campos
correctamente");
        }
    }

    const validarCampos = () => {
        const nombreReg =
/^([A-Za-z]{3,50})(([\s])+([A-Za-z]{3,50})){0,3}$/;
        const edadReg = /^[0-9]{1,3}$/;
        const telReg =  /^([3]{1})([0-9]{9})$/;
        const userNameReg = /^([A-Za-z0-9\\_\\-]{3,80})(@mascotas.com)$/;
        const passwdReg = /^([A-Za-z0-9\\_\\-]{3,80})$/;

        if(!nombreReg.test(nombres)){
            return false;
        }

        if(!edadReg.test(edad) || edad < 18 || edad > 90){
            return false;
        }

        if(!telReg.test(telefono)){
            return false;
        }

        if(!userNameReg.test(userName)){
            return false;
        }

        if(!passwdReg.test(passwd)){
            return false;
        }

        return true;
    }

    const errorLog = (e, campo) => {
        if(campo==="nombre"){
            const reg = /^([A-Za-z]{3,50})(([\s])+([A-Za-z]{3,50})){0,3}$/;
            if(reg.test(e.target.value)){
```

```
            e.target.className = "form-control is-valid";
            setAlert("");
            return;
        }
    }

    if(campo==="edad"){
        if(e.target.value < 18 || e.target.value > 90){
            e.target.className = "form-control is-invalid";
            return;
        }

        const reg = /^[0-9]{1,3}$/;
        if(reg.test(e.target.value)){
            e.target.className = "form-control is-valid";
            return;
        }
    }

    if(campo==="tel"){
        const reg = /^([3]{1})([0-9]{9})$/;
        if(reg.test(e.target.value)){
            e.target.className = "form-control is-valid";
            return;
        }
    }

    if(campo==="userName"){
        const reg = /^([A-Za-z0-9\\_\\-]{3,80})(@mascotas.com)$/;
        if(reg.test(e.target.value)){
            e.target.className = "form-control is-valid";
            return;
        }
    }

    if(campo==="passwd"){
        const reg = /^([A-Za-z0-9\\_\\-]{3,80})$/;
        if(reg.test(e.target.value)){
            e.target.className = "form-control is-valid";
            return;
```

```
            }
        }

        e.target.className = "form-control is-invalid";
        return;
    }

    return (
        <DefaultLayout>
            <div className="container mt-5 pt-5">
                <div
                    className="row justify-content-center
align-items-center g-2"
                >
                    <div className="col-12 col-md-8">
                        <div className="card">
                            <h3 className="card-title text-center
mt-5">Resgitrarse</h3>
                            <div className="card-body">
                                {
                                    alert!==""?(
                                        <div className="row
justify-content-center">
                                            <div className="alert
alert-danger text-center" role="alert">
                                                {alert}
                                            </div>
                                        </div>
                                    ):""
                                }
                                <div className="row
justify-content-center">
                                    <div className="col-12">
                                        <div className="row">
                                            <div className="input-group
has-validation mb-2">
                                                <span

className="input-group-text"
                                                    id="inputGroupNombres"
```

```
                                                >Nombres y Apellidos</span>
                                                <input
                                                    type="text"

className={`form-control`}
                                                    id="validationNombres"

aria-describedby="inputGroupNombres validationNombresFeedback"
                                                    required
                                                    placeholder="Jhon Doe"
                                                    value={nombres}
                                                    onChange={(e) => {

setNombres(e.target.value);
                                                        errorLog(e,
"nombre");
                                                        setAlert("");
                                                    }}
                                                />
                                                <div

id="validationNombresFeedback"

className="invalid-feedback"
                                                >
                                                    Nombre:<br />
                                                        Solo letras y
espacios (A-Z a-z) <br />
                                                        Deben ser más de 3
letras por Nombre o Apellido<br/>
                                                        Cada nombre o
apellido separado por 1 o mas espcios <br />

                                                </div>
                                            </div>
                                            <div className="input-group
has-validation mb-2">
                                                <span

className="input-group-text"
```

```
                                        id="inputGroupEdad"
                                    >Edad</span>
                                    <input
                                        type="number"

className={`form-control`}
                                        id="validationEdad"

aria-describedby="inputGroupEdad validationEdadFeedback"
                                        required
                                        value={edad}
                                        onChange={(e) => {

setEdad(e.target.value);
                                            errorLog(e,
"edad");
                                            setAlert("");
                                        }}
                                    />
                                    <div

id="validationEdadFeedback"

className="invalid-feedback"
                                    >
                                       Edad: numero entre 18 y
90
                                    </div>
                                </div>
                                <div className="input-group
has-validation mb-2">
                                    <span

className="input-group-text"
                                        id="inputGroupTel"
                                    >Celular</span>
                                    <input
                                        type="text"

className={`form-control`}
```

```
                                                id="validationTel"

aria-describedby="inputGroupTel validationTelFeedback"
                                                required
                                                value={telefono}

placeholder="3147856561"
                                                onChange={(e) => {

setTelefono(e.target.value);
                                                    errorLog(e, "tel");
                                                    setAlert("");
                                                }}
                                            />
                                            <div

id="validationTelFeedback"

className="invalid-feedback"
                                            >
                                                Numero de 10 digitos
iniciado en 3
                                            </div>
                                        </div>
                                        <div className="input-group
has-validation">
                                            <span

className="input-group-text"
                                                id="inputGroupUsername"
                                            >Usuario @</span>
                                            <input
                                                type="text"

className={`form-control`}
                                                id="validationUsername"

aria-describedby="inputGroupUsername validationUsernameFeedback"
                                                required
                                                value={userName}
```

```
                                                onChange={(e) => {

setUserName(e.target.value);

errorLog(e,"userName");
                                                        setAlert("");
                                                }}
                                            />
                                            <div

id="validationUsernameFeedback"

className="invalid-feedback"
                                            >
                                                Usuario: <br />
                                                    letras,guines (-) o
guion bajo (_) y numeros sin espacios <br />
                                                    terminado en
@mascotas.com
                                            </div>
                                        </div>
                                    </div>
                                    <div className="mb-4 row mt-2">
                                        <div className="input-group
has-validation">
                                            <span

className="input-group-text"
                                                id="inputGroupUsername"
                                            >Contraseña </span>
                                            <input
                                                type="password"

className={`form-control`}
                                                id="validationPasswd"

aria-describedby="inputGroupPasswd validationPasswdFeedback"
                                                required
                                                value={passwd}
                                                onChange={(e) => {
```

```
setPasswd(e.target.value);
                                                    errorLog(e,
"passwd");
                                                    setAlert("");
                                                }}
                                            />
                                            <div
id="validationPasswdFeedback"
                                            >
                                                Contraseña:
                                                 entre 3 y 80
caracteres, letras, numeros o guiones sin espacios
                                            </div>
                                        </div>
                                    </div>
                                    <div className="row">
                                        <button
                                            className={`btn btn-success
w-80`}
                                            onClick={(e) =>
registrarUsuario()}
                                        >
                                            Registrarse
                                        </button>
                                    </div>
                                    <div className='row text-center
mt-3'>
                                        <p> ¿Ya tienes una cuenta ?
<Link to="/register">Iniciar Sesión</Link></p>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
```

```
            </div>
        </DefaultLayout>
    );
}

export default Register;
```