DOCUMENTACIÓN DEL PROYECTO 'CRUD'

Documentación en 'App.js'-client

Frontend

Este es el archivo principal del Frontend de la aplicación. Definiendo la **interfaz** del usuario y **gestiona** la interacción del usuario con el CRUD.

La funcionalidad principal de este archivo es permitir a los usuarios registrar, actualizar, eliminar (**CRUD**) y ver a los usuarios en la base de datos SQL.

- ✓ Este archivo comienza importando dependencias necesarias, incluyendo ***'useState'*** de React para manejar el **estado** y los efectos secundarios.
- ✓ Define un componente funcional 'App' que contiene el formulario de registro de usuarios y la tabla para mostrar los usuarios existentes.
- ✓ Utiliza estados locales para almacenar los datos del usuario; nombres, apellidos, corre, teléfono, etc, que permiten controlar el estado de la interfaz de usuario; modo edición, lista de usuarios, etc.
- ✓ Define funciones para manejar las operaciones del CRUD en la base de datos mediante solicitudes HTTP.
- ✓ Incluye funciones auxiliares para limpiar los campos del formulario y activar el modo de edición cuando se selecciona un usuario para editar.

Línea 10-20

```
//Variables de estado para almacenar los datos a registrar
//inicializadas en un string vacío a excepción de 'telefono','edad','identificación' e 'id'
que comienzan como indefinidos
const[Nombres,setNombres]=useState("");
  const[Apellidos,setApellidos]=useState("");
  const[correo,setcorreo]=useState("");
  const[telefono,settelefono]=useState();
  const[edad,setedad]=useState();
  const[nacimiento,setnacimiento]=useState("");
  const[fecha,setfecha]=useState("");
  const[identificacion,setidentificacion]=useState();
  const[editar,seteditar]=useState(false); //Se inicializa como 'false'
  const[id,setid]=useState();
  const[usuarioslist,setusuarios]=useState([]);
```

Línea 22-42

```
const add=()=>{
   Axios.post("http://localhost:3001/create",{ //Con una llamada 'post' se encarga de enviar
los datos del nuevo usuario en el cuerpo de la solicitud (Creando valores nuevos en la base
de datos)
      Nombres:Nombres,
      Apellidos:Apellidos,
      correo:correo,
      telefono:telefono,
      edad:edad,
```

```
      nacimiento:nacimiento,
      fecha:fecha,
      identificacion:identificacion
   }).then(()=>{ //Encargada de agregar el usuario, actualiza la lista usuarios y limpia los
campos del formulario.
    getusuarios();
    limpiarcampos();
    Swal.fire({ //Alerta de mensaje de registro con librería 'Swal'
      title: "Registro Exitoso!!",
      text: "El empleado "+Nombres+" fue registrado con exito!!",
      icon: "success",
      timer:3000 //Tiempo visible de la alerta
    });
   });
  }
```

Línea 44-65

```
const update=()=>{
    Axios.put("http://localhost:3001/update",{ //Solicitud 'put' a la dirección para enviar
los datos actualizados del usuario
       Nombres:Nombres,
       id:id,
       Apellidos:Apellidos,
       correo:correo,
       telefono:telefono,
       edad:edad,
       nacimiento:nacimiento,
       fecha:fecha,
       identificacion:identificacion
    }).then(()=>{ //Luego de hacer la actualización de datos, se actualizará la lista de
datos y limpia los campos del formulario
     getusuarios();
     limpiarcampos();
     Swal.fire({ //Alerta de éxito usando librería 'Swal'
      title: "Actualizacon Exitosa!!",
      text: "El empleado "+Nombres+" fue actualizado con exito!!",
      icon: "success",
      timer:3000 //Tiempo de duración visible de la alerta
    });
    });
   }
```

Línea 67 – 92

```
const eliminarusuario=(val)=>{ //Función para eliminar un usuario
    Swal.fire({
      title: "Confirmar eliminado",
```

```
      text: "Realmente desea eliminar a "+val.nombres+"?", //Mensaje de confirmación de
eliminación de usuario con el nombre de este
      icon: "warning", //icono de aviso
      showCancelButton: true, //Mostrar botón de cancelar alerta
      confirmButtonColor: "#3085d6", //Color botón de confirmación
      cancelButtonColor: "#d33", //Color botón de cancelar
      confirmButtonText: "Si, Eliminarlo" //Texto de botón confirmación
    }).then((result) => { //Comprobaciones
      if (result.isConfirmed) { //Si la confirmación es verificada se realiza llamado de
'delete' a la base de datos
        Axios.delete(`http://localhost:3001/delete/${val.id}`).then(()=>{ //Luego de la
confirmación de eliminación, se hace actualización de lista usuarios y limpieza de los campos
en el formulario
          getusuarios();
          limpiarcampos();
          Swal.fire({
            title: "Eliminado",
            text: val.nombres+ "fue eliminado", //Alerta de usuario eliminado
            icon: "success", //Icono de tarea exitosa
            timer:3000 //Tiempo de alerta
          });
    });

      }
    });
    }
```

Línea 95-107

```
const limpiarcampos=()=>{//Función para limpiar los campos del formulario, donde todas las
variables vuelven a su valor definido en las constantes
    setNombres("");
    setApellidos("");
    setcorreo("");
    settelefono("");
    setedad("");
    setnacimiento("");
    setfecha("");
    setidentificacion("");
    setid("") //También reestablece el estado del 'id'
    seteditar(false);

   }
```

Línea 109-121

```
const editarusuario=(val)=>{//Función editar usuario
    seteditar(true); //Cambia su valor a true para habilitar la edición
//Los valores de los estados obtendrán un nuevo valor para ser nuevamente declarados
    setNombres(val.nombres);
```

```
    setApellidos(val.apellidos);
    setcorreo(val.correo);
    settelefono(val.telefono);
    setedad(val.edad);
    setnacimiento(val.nacimiento);
    setfecha(val.fecha);
    setidentificacion(val.identificacion);
    setid(val.id);
  }
```

Línea 123 – 128

```
//Función que obtiene la lista de usuarios de la base de datos
const getusuarios=()=>{ //Solicitud 'get' en la ubicación
    Axios.get("http://localhost:3001/usuarios").then((response)=>{
     setusuarios(response.data) //Actualiza 'usuarioslist' con los datos obtenidos
    });
  }
  getusuarios() //Realiza la función 'getusuarios' para obtener la lista de usuarios
(llamarla)
```

Línea 132-211

```
return (

<div className="container"> //Esta es la clase con la cual se alineará el contenido

    <div className="card text-center">
      <div className="card-header">
        REGISTRO USUARIOS
      </div>
      <div className="card-body">
        <div className="input-group mb-3">

//Desde AQUÍ tenemos los campos de entrada para el registro del usuario
//Cada campo tendrá su actualización de valor
          <span className="input-group-text" >Nombres</span>
          <input type="text"
          onChange={(event)=>{
            setNombres(event.target.value)//Actualiza el estado de nombre del usuario
          }}
          className="form-control" value={Nombres} placeholder="Ingrese su nombre"/>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Apellidos</span>
          <input type="text"
           onChange={(event)=>{
            setApellidos(event.target.value)
          }}
          className="form-control" value={Apellidos} placeholder="Ingrese su apellido"/>
```

```
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Correo</span>
          <input type="text"
          onChange={(event)=>{
            setcorreo(event.target.value)
          }}
          className="form-control" value={correo} placeholder="Ingrese su correo"/>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Telefono</span>
          <input type="number"
          onChange={(event)=>{
            settelefono(event.target.value)
          }}
          className="form-control" value={telefono} placeholder="Ingrese su telefono"/>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Edad</span>
          <input type="number"
          onChange={(event)=>{
            setedad(event.target.value)
          }}
          className="form-control" value={edad} placeholder="Ingrese su edad"/>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Lugar Nacimiento</span>
          <input type="text"
          onChange={(event)=>{
            setnacimiento(event.target.value)
          }}className="form-control" value={nacimiento} placeholder="Ingrese su lugar de
nacimiento"/>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Fecha De Nacimiento</span>
          <input type="date"
          onChange={(event)=>{
            setfecha(event.target.value)
          }}
          className="form-control" value={fecha}/>
        </div>
        <div className="input-group mb-3">
          <span className="input-group-text" >Identificacion</span>
          <input type="number"
           onChange={(event)=>{
            setidentificacion(event.target.value)
          }}
          className="form-control" value={identificacion} placeholder="Ingrese su
identificacion"/>
        </div>
```

```
      </div>
      <div className="card-footer text-muted"> //Desde aquí empezamos con los botones para actualizar, cancelar o registrar un Usuario
           {
             editar? //SI se está en este MODO de Edición, tendrá las funciones de actualizar y/o cancelar la edición
               <div>

             <button className='btn btn-warning m-2'  onClick={update}>Actualizar</button>
             <button className='btn btn-info m-2' onClick={limpiarcampos}>Cancelar</button>
             </div>
             :<button className='btn btn-success' onClick={add}>Registrar</button>
           }
```

Continuación; 215- 266

```
</div>
</div>
//Esta será la tabla en donde se podrán encontrar a los Usuarios YA registrados, ordenados por su ID, Nombres y Apellidos, Correo Electrónico, Teléfono , Edad, Lugar de Nacimiento, Identificación Y además las acciones/funciones de eliminar o editar la información del usuario
         <table className="table table-striped">
         <thead>
               <tr>
                 <th scope="col">#</th>
                 <th scope="col">Nombres</th>
                 <th scope="col">Apellidos</th>
                 <th scope="col">Correo</th>
                 <th scope="col">Telefono</th>
                 <th scope="col">Edad</th>
                 <th scope="col">Lugar Nacimiento</th>
                 <th scope="col">Fecha De Dacimiento</th>
                 <th scope="col">Identificacion</th>
                 <th scope="col">Acciones</th>
               </tr>
             </thead>
             <tbody>
//A continuación se añadirán los registros de los estados con sus respectivos valores para mostrarlos en la tabla
                {
                  usuarioslist.map((val,key)=>{
                    return <tr key={val.id}>
                                <th>{val.id}</th>
                                <td>{val.nombres}</td>
                                <td>{val.apellidos}</td>
                                <td>{val.correo}</td>
                                <td>{val.telefono}</td>
                                <td>{val.edad}</td>
                                <td>{val.nacimiento}</td>
```

```
                                <td>{val.fecha}</td>
                                <td>{val.identificacion}</td>
                                <td>
                                    <div className="btn-group" role="group" aria-
label="Basic example">
                                      <button type="button"
                                      onClick={()=>{
                                        editarusuario(val) //Botones para llamar a las
acciones / funciones de edición o eliminar a un usuario
                                      }}
                                      className="btn btn-info">Editar</button>
                                      <button type="button" onClick={()=>{
                                        eliminarusuario(val);
                                      }} className="btn btn-danger">Eliminar</button>
                                    </div>
                                </td>
                            </tr>
                    })
                }


            </tbody>
        </table>
    </div>
  );
}
```

Documentación del código de 'Index.js'-server

Backend

Este el archivo principal del backend de la aplicación. Define los **endpoints API** y maneja las solicitudes **HTTP** entrantes.

La funcionalidad principal de este archivo es proporcionar una interfaz para interactuar con la base de datos MySQL, permitiendo realizar operaciones **CRUD** en la tabla de usuarios.

- ✓ Este archivo comienza importando las dependencias necesarias, incluyendo '**express' , 'mysql'** y **'cors'**.
- ✓ Configura el servidor Express y conecta la aplicación a la base de datos MySQL.
- ✓ Define los **endpoints API** para realizar operaciones **CRUD**.
- ✓ Cada **endpoint** maneja las solicitudes **HTTP** correspondientes y ejecuta consultas SQL en la base de datos para realizar las operaciones necesarias.
- ✓ Utiliza el módulo **'mysql'** para realizar consultas **SQL** y manejar los resultados de la base de datos.
- ✓ Utiliza el **Middleware 'cors'** para permitiar solicitudes de dominios externos.

Línea 1 - 7

```
// Importación de los módulos necesarios
const express= require("express"); //Importación del modulo 'express' para crear el servidor
web
```

```
const app=express(); //Crear una instancia de la aplicación ‘express’
const mysql=require("mysql");//Importación del módulo ‘MySQL’ para interactuar con la base de datos
const cors=require("cors");//Importación del módulo ‘cors’ para habilitar el intercambio de recursos entre diferentes orígenes

app.use(cors()); //Middleware de ‘express’ para habilitar ‘cors’
app.use(express.json());//Middleware de ‘express’ para analizar solicitudes ‘Json’
```

Línea 9 - 14

```
//Creación de una conexión a la base de datos MySQL
const db=mysql.createConnection({
    host:"localhost", //Dirección del servidor de la base de datos
    user:"root", //Nombre de usuario de la base de datos
    password:"", //Contraseña de la base de datos
    database:"usuarios" //Nombre de la base de datos a la que conectarse
});
```

Línea 16 – 35

```
//Endpoint POST para crear un nuevo usuario en la base de datos
app.post("/create",(req,res)=>{ //Obtener los datos del cuerpo de la solicitud
    const Nombres=req.body.Nombres;
    const Apellidos=req.body.Apellidos;
    const correo=req.body.correo;
    const telefono=req.body.telefono;
    const edad=req.body.edad;
    const nacimiento=req.body.nacimiento;
    const fecha=req.body.fecha;
    const identificacion=req.body.identificacion;

//Query para insertar un nuevo usuario en la base de datos

    db.query("INSERT INTO usuarios(nombres,apellidos,correo,telefono,edad,nacimiento,fecha,identificacion) values(?,?,?,?,?,?,?,?)",[Nombres,Apellidos,correo,telefono,edad,nacimiento,fecha,identificacion],
    (err,result)=>{
        if(err){ //En caso de ERROR se mostrará en la consola
            console.log(err);
        }else{
            res.send(result); //En caso de ÉXITO se enviará una respuesta al cliente
        }
    }
    );
});
```

Línea 37 – 47

```
//Endpoint GET para obtener todos los usuarios de la base de datos
app.get("/usuarios",(req,res)=>{
// Consulta SQL para seleccionar todos los usuarios de la tabla ‘usuarios’
    db.query("SELECT * FROM usuarios",
    (err,result)=>{
        if(err){
            console.log(err); //En caso de ERROR se mostrará en la consola
        }else{
            res.send(result); //En caso de ÉXITO se enviará una respuesta al cliente con los
usuarios obtenidos
        }
    }
    );
});
```

Línea 49 – 69

```
//Endpoint PUT para actualizar un usuario en la base de datos
app.put("/update",(req,res)=>{
//Obtener los datos del cuerpo de la solicitud
    const Nombres=req.body.Nombres;
    const id=req.body.id;
    const Apellidos=req.body.Apellidos;
    const correo=req.body.correo;
    const telefono=req.body.telefono;
    const edad=req.body.edad;
    const nacimiento=req.body.nacimiento;
    const fecha=req.body.fecha;
    const identificacion=req.body.identificacion;



//Query para actualizar un usuario en la base de datos
    db.query("UPDATE usuarios SET
nombres=?,apellidos=?,correo=?,telefono=?,edad=?,nacimiento=?,fecha=?,identificacion=? WHERE
id=?",[Nombres,Apellidos,correo,telefono,edad,nacimiento,fecha,identificacion,id],
    (err,result)=>{
        if(err){ //En caso de ERROR se mostrará en la consola
            console.log(err);
        }else{
            res.send(result); //En caso de ÉXITO se enviará una respuesta al cliente
        }
    }
    );
});
```

Línea 71- 83

```
//Endpoint DELETE para eliminar un usuario de la base de datos
app.delete("/delete/:id",(req,res)=>{
```

```
    const id=req.params.id; //Obtención del ‘id’ del usuario a eliminar desde los parámetros
de la URL

//Query para eliminar un usuario de la base d datos por su ‘id’
    db.query("DELETE FROM usuarios WHERE id=?",id,
    (err,result)=>{
        if(err){
            console.log(err); //En caso de ERROR se mostrará en la consola
        }else{
            res.send(result);//En caso de ÉXITO se enviará una respuesta al cliente
        }
    }
    );
});
```

Línea 89 – 91

```
//Iniciar el servidor y escuchar en el puerto 3001
app.listen(3001,()=>{
    console.log("corriendo en el puerto 3001") //Mensaje en consola para mostrar que SI
funciona
})
```