

DESARROLLO DE APLICACIONES EMPRESARIALES

ESTUDIANTES

CRISTIAN DAVID DIAZ AZA

EDWARD MAURICIO CARVAJAL DELGADO

OSCAR MAURICIO ESPARZA ZAMBRANO

WILLIAM JOSÉ CASTELLANOS ALTUVE

JUAN DAVID GAMBOA OROZCO

JUAN SEBASTIÁN MERCHÁN DUARTE

A191

DOCENTE

JULIAN BARNEY JAIMES RINCÓN

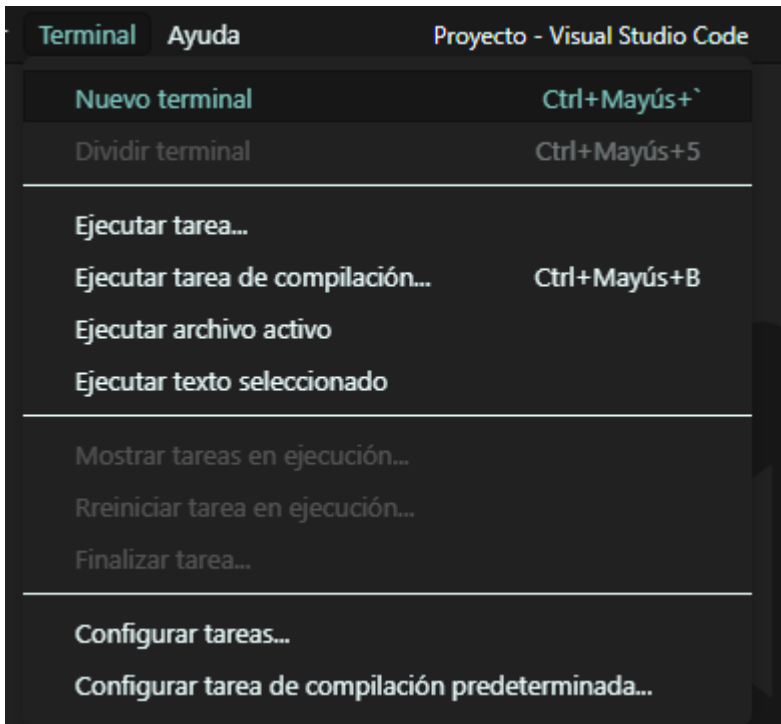
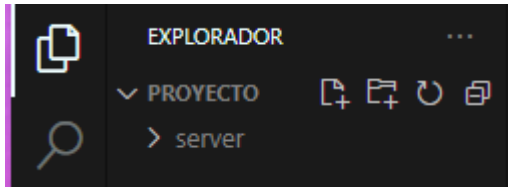
UNIDADES TECNOLÓGICAS DE SANTANDER

TECNOLOGÍA EN DESARROLLO DE SISTEMAS INFORMÁTICOS

BUCARAMANGA, REAL DE MINAS

2022

Comenzaremos creando una carpeta llamada server.



```
PS C:\Users\ADMIN\Desktop\Proyecto> npm init -y
```

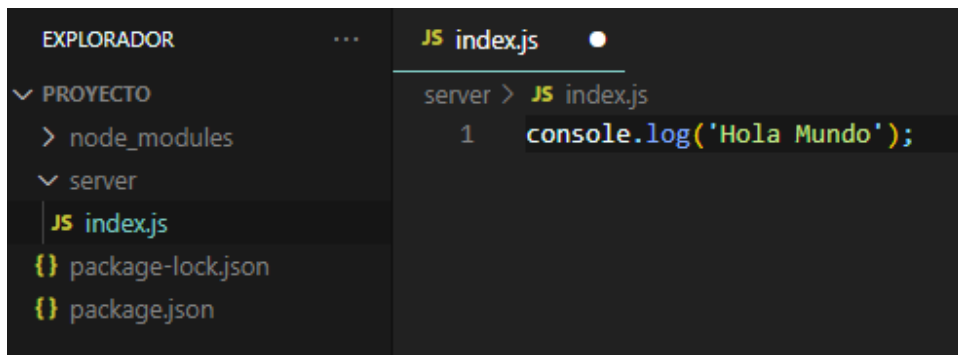
Con este código crearemos un package.json

```
PS C:\Users\ADMIN\Desktop\Proyecto> npm i express mysql2 morgan
```

Instalamos las dependencias de express para poder crear el servidor, el de msq2 que es el mismo de MySQL pero este resiste promesas y el de Morgan para poder ver los mensajes por consola.

```
PS C:\Users\ADMIN\Desktop\Proyecto> npm i nodemon -D
```

Instalamos esto para no tener que estar reiniciando el código del servidor a cada rato por cualquier modificación.



Procederemos a crear en la carpeta server un index.js donde colocaremos

```
Console.log ('Hola Mundo');
```

Y lo ejecutaremos y se nos mostrara así en la consola

```
PS C:\Users\ADMIN\Desktop\Proyecto> node .\server\index.js
Hola Mundo
PS C:\Users\ADMIN\Desktop\Proyecto>
```



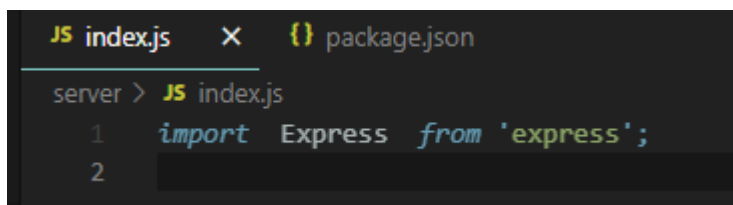
```
1 {
2   "name": "proyecto",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "dev": "nodemon server/index.js"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {}
13 }
```

Colocamos esto para que a la hora de modificar el código se nos actualice automáticamente. Para ello lo vamos a correr de la siguiente manera en el terminal.

```
PS C:\Users\ADMIN\Desktop\Proyecto> npm run dev
```

```
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server/index.js`
Hola Mundo 1
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node server/index.js`
Hola Mundo 2
[nodemon] clean exit - waiting for changes before restart
█
```

Como se puede apreciar le hicimos un cambio y se actualizo automáticamente.



```
server > JS index.js
1 import Express from 'express';
2
```

Para poder importar express el nodemon no nos lo reconoce por lo tanto, realizamos esto en el package.json

```
> node_modules
  server
    JS index.js
    {} package-lock.json
    {} package.json
  1 {
  2   "name": "proyecto",
  3   "version": "1.0.0",
  4   "description": "",
  5   "main": "index.js",
  6   "type": "module",
  ▶ Debug
```

Ahora lo que haremos es hacer que el servidor nos devuelva algunas respuestas para esto haremos lo siguiente.

Crearemos un archivo llamado config.js y podremos lo siguiente

```
> node_modules
  server
    JS config.js
    JS index.js
  1 export const PORT = 3000;
```

Y luego lo llamaremos en nuestro índice de la siguiente forma

```
import express from 'express';
import { PORT } from './config.js';

const app= express();

app.listen(PORT)
console.log('el servidor se esta corriendo en el puerto PORT');
```

Y si colocamos en un navegador localhost:3000 nos saldría lo siguiente

```
< > ↺ 🌐 localhost:3000

Cannot GET /
```

Lo que haremos ahora es crear la base de datos, la crearemos a través de cmd con Docker.

```
C:\Users\ADMIN>docker pull mysql
```

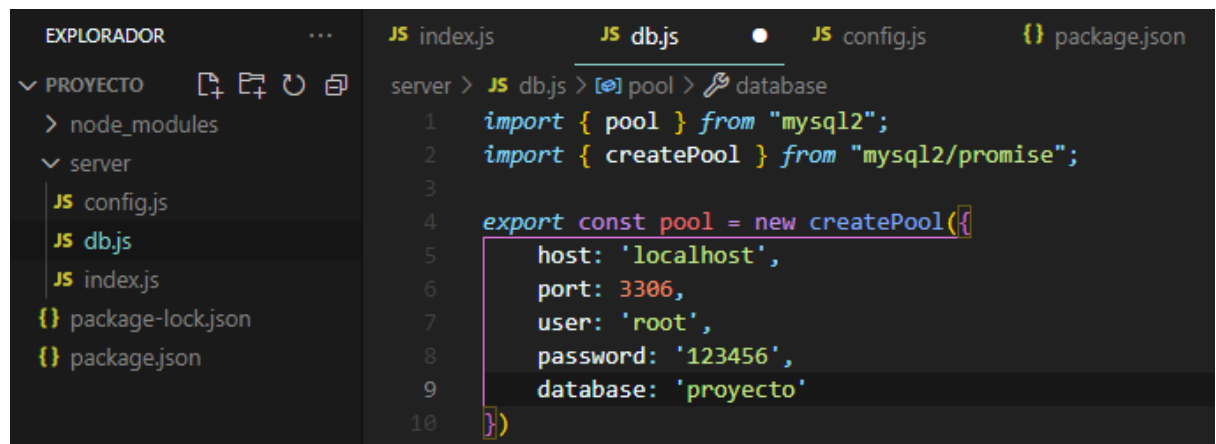
Esto es para que descargue la imagen de MySQL en Docker, una vez descargada le daremos el siguiente comando

```
C:\Users\ADMIN>docker run --name db -e MYSQL_ROOT_PASSWORD=123456 -e MYSQL_DATABASE=proyecto -p 3306:3306 -d mysql
```

Donde le colocaremos el nombre db, contraseña de 123456 nombre de la base de datos proyecto y la correremos en el puerto 3306 que es donde corre normal mente.

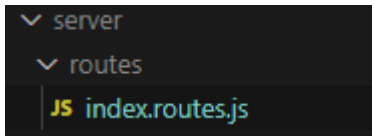
Si le damos el comando Docker ps nos mostrara la base de datos

```
C:\Users\ADMIN>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
ed789d3d1cd7   mysql    "docker-entrypoint.s..." 21 seconds ago Up 19 seconds 0.0.0.0:3306->3306
```



Creamos un archivo llamado db.js donde nos conectaremos al base de datos.

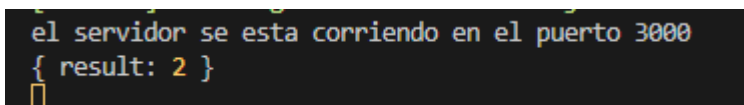
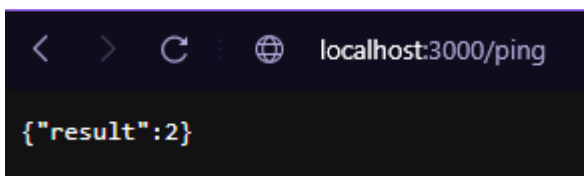
Creamos en la carpeta server una llamada routers y crearemos un archivo llamado index.routes.js



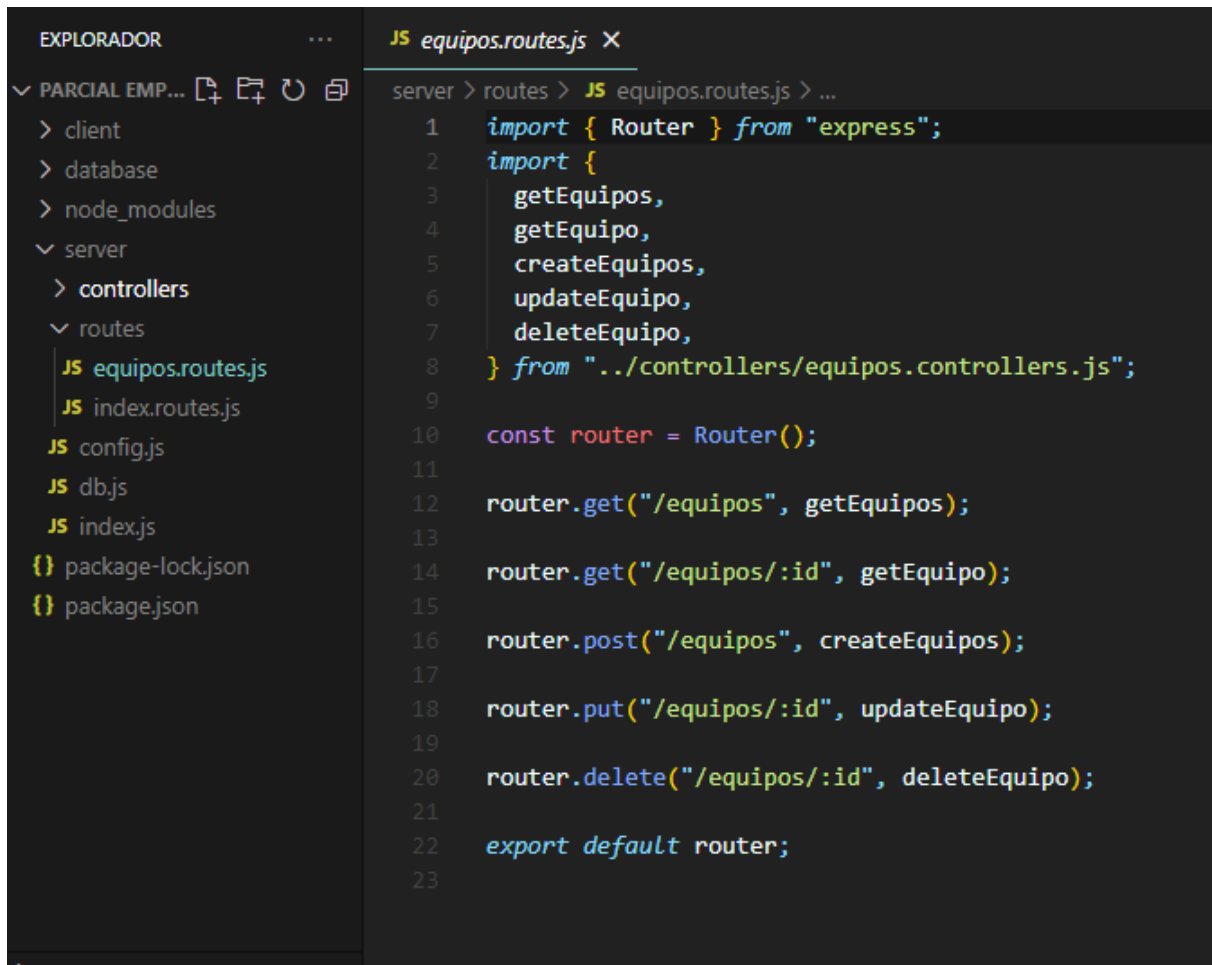
```
JS db.js      JS index.routes.js X  JS index.js
server > routes > JS index.routes.js > [⌕] default
1  import { Router } from "express";
2  import { pool } from "../db.js";
3
4  const router = Router();
5
6  router.get("/ping", async (req, res) => {
7    const [rows] = await pool.query("SELECT 1 + 1 as result");
8    console.log(rows[0]);
9    res.json(rows[0]);
10 });
11
12 export default router;
```

el archivo index.routes.js es el que nos mostrara en el local host los archivos de la base de datos a través de consultas.

En este caso le pedimos que sumemos 1+1 y nos los muestre en una columna llamada resultado



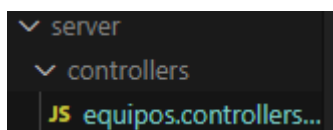
Creemos el siguiente archivo llamado equipos.routes.js que es ahí donde crearemos las rutas para las tareas del crud



The image shows a VS Code editor with two panels. The left panel is the 'EXPLORADOR' (File Explorer) showing a project structure with folders like 'client', 'database', 'node_modules', 'server', 'controllers', and 'routes'. The 'server' folder is expanded, showing 'equipos.routes.js' and 'index.routes.js'. The right panel shows the code inside 'equipos.routes.js'.

```
server > routes > JS equipos.routes.js > ...
1  import { Router } from "express";
2  import {
3    getEquipos,
4    getEquipo,
5    createEquipos,
6    updateEquipo,
7    deleteEquipo,
8  } from "../controllers/equipos.controllers.js";
9
10 const router = Router();
11
12 router.get("/equipos", getEquipos);
13
14 router.get("/equipos/:id", getEquipo);
15
16 router.post("/equipos", createEquipos);
17
18 router.put("/equipos/:id", updateEquipo);
19
20 router.delete("/equipos/:id", deleteEquipo);
21
22 export default router;
23
```

Creamos una carpeta en server llamada controllers y crearemos un archivo llamado equipos.routes.js que es el que se encarga de realizar todas las peticiones del crud



This image is a close-up of the file explorer in VS Code, showing the 'server' folder expanded to reveal a new file named 'equipos.controllers.js'.

```
server
└── controllers
    └── JS equipos.controllers...
```


server > controllers > JS equipos.controllers.js > ...

```
1  import { pool } from "../db.js";
2
3  export const getEquipos = async (req, res) => {
4    try {
5      const [result] = await pool.query(
6        "SELECT * FROM equipos ORDER BY createAt ASC"
7      );
8      res.json(result);
9    } catch (error) {
10     return res.status(500).json({ mensaje: error.mensaje });
11   }
12 };
13
14 export const getEquipo = async (req, res) => {
15   try {
16     const [result] = await pool.query("SELECT * FROM equipos WHERE id = ?", [
17       req.params.id,
18     ]);
19
20     if (result.length === 0)
21       return res.status(404).json({ mensaje: "Equipo no encontrado" });
22
23     res.json(result[0]);
24   } catch (error) {
25     return res.status(500).json({ mensaje: error.mensaje });
26   }
27 };
28
29 export const createEquipos = async (req, res) => {
30   try {
31     const { eq1, eq2, description } = req.body;
32     const [result] = await pool.query(
33       "INSERT INTO equipos(eq1, eq2, description) VALUES (?, ?, ?)",
34       [eq1, eq2, description]
35     );
36     res.json({
37       id: result.insertId,
38       eq1,
39       eq2,
40       description,
41     });
42   } catch (error) {
43     return res.status(500).json({ mensaje: error.mensaje });
44   }
45 };
46
47 export const updateEquipo = async (req, res) => {
48   try {
49     const { eq1, eq2, description } = req.body;
50     const [result] = await pool.query("UPDATE equipos SET ? WHERE id = ?", [
51       req.body,
```

```

49     const { eq1, eq2, description } = req.body;
50     const [result] = await pool.query("UPDATE equipos SET ? WHERE id = ?", [
51       req.body,
52       req.params.id,
53     ]);
54     res.json(result);
55   } catch (error) {
56     return res.status(500).json({ mensaje: error.mensaje });
57   }
58 };
59
60 export const deleteEquipo = async (req, res) => {
61   try {
62     const [result] = await pool.query("DELETE FROM equipos WHERE id = ?", [
63       req.params.id,
64     ]);
65
66     if (result.affectedRows === 0)
67       return res.status(404).json({ mensaje: "Equipo no encontrado" });
68
69     return res.sendStatus(204);
70   } catch (error) {
71     return res.status(500).json({ mensaje: error.mensaje });
72   }
73 };
74

```

lo siguiente que haremos es llamarlo en el index.js para que podamos hacer peticiones

```

import indexRoutes from "../routes/index.routes.js";
import equiposRoutes from "../routes/equipos.routes.js";

```

```

app.use(indexRoutes);
app.use(equiposRoutes);

```

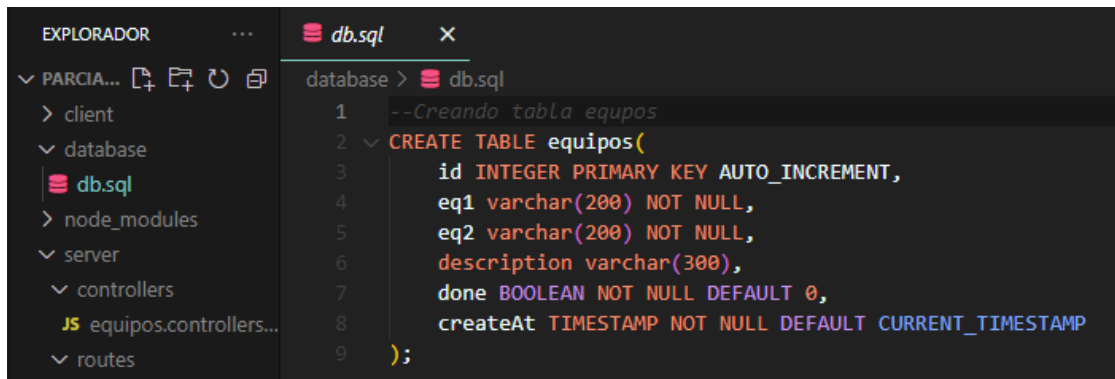
Agregamos esto también al index.js

```

app.use(express.json());

```

Ahora creamos una carpeta llamada database donde crearemos el archivo db.sql



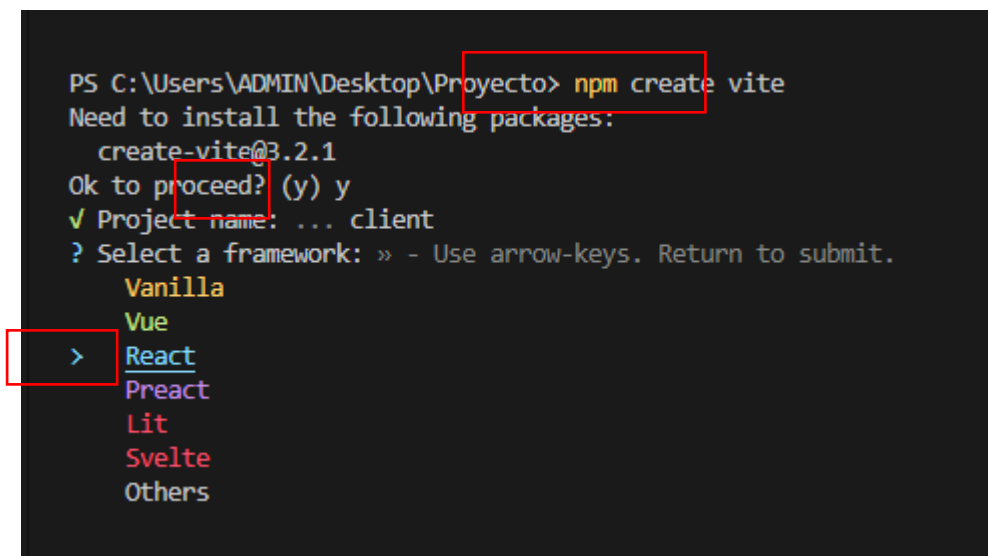
```
EXPLORADOR  ...  db.sql  X
PARCIA...  database > db.sql
> client
> database
  db.sql
> node_modules
> server
  controllers
  JS equipos.controllers...
  routes
1  --Creando tabla equipos
2  CREATE TABLE equipos(
3      id INTEGER PRIMARY KEY AUTO_INCREMENT,
4      eq1 varchar(200) NOT NULL,
5      eq2 varchar(200) NOT NULL,
6      description varchar(300),
7      done BOOLEAN NOT NULL DEFAULT 0,
8      createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
9  );
```

Y este seria todo el backend de nuestro aplicativo.

Ahora comenzaremos con el fronted.

Comenzamos poniendo el comando npm create vite en la terminal.

Le daremos que yes y seleccionaremos react.



```
PS C:\Users\ADMIN\Desktop\Proyecto> npm create vite
Need to install the following packages:
  create-vite@3.2.1
Ok to proceed? (y) y
√ Project name: ... client
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
  > React
  Preact
  Lit
  Svelte
  Others
```

```
PS C:\Users\ADMIN\Desktop\Proyecto> npm create vite
Need to install the following packages:
  create-vite@3.2.1
Ok to proceed? (y) y
✓ Project name: ... client
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
> JavaScript
  TypeScript
```

Ahora entraremos a la carpeta client y le instalaremos las dependencias

```
PS C:\Users\ADMIN\Desktop\Proyecto> cd client
PS C:\Users\ADMIN\Desktop\Proyecto\client> npm install
[#####.....] - idealTree:@babel/generator: timing idealTree:node_modules/@babel/generator
```

Y los corremos con el **NPM RUNV DEV.**

y se nos abrirá así

```
VITE v3.0.0 ready in 413 ms
→ Local: http://127.0.0.1:5173/
→ Network: use --host to expose
```

Luego nos vamos a la carpeta client abrimos y entramos al archivo App.jsx.
borramos todos de ahí y pondremos lo siguiente.

```
App.jsx X
client > src > App.jsx > ...
1  import { Route, Routes } from "react-router-dom";
2  import EquipoPage from "../pages/EquipoPage";
3  import EquipoForm from "../pages/EquipoForm";
4  import NotFound from "../pages/NotFound";
5  import { EquipoContextProvider } from "../context/EquipoProvider";
6
7  import NavBar from "../components/NavBar";
8  import FooterBar from "../components/FooterBar";
9
10 function App() {
11   return (
12     <div className="bg-zinc-900 min-h-screen">
13       <NavBar />
14       <div className="container mx-auto py-5 px-10">
15         <EquipoContextProvider>
16           <Routes>
17             <Route path="/" element={<EquipoPage></EquipoPage>} />
18             <Route path="/new" element={<EquipoForm></EquipoForm>} />
19             <Route path="/edit/:id" element={<EquipoForm></EquipoForm>} />
20             <Route path="*" element={<NotFound></NotFound>} />
21           </Routes>
22         </EquipoContextProvider>
23       </div>
24       <FooterBar />
25     </div>
26   );
27 }
28
29 export default App;
30
```

Ahora en la terminal pondremos este comando para instalar estas dependencias.

```
PS C:\Users\ADMIN\Desktop\Proyecto\client> npm install react-router-dom@6
[.....] | idealTree:client: sill idealTree buildDeps
```

Ahora lo que haremos es llamarla en el main.jsx

```
main.jsx X
client > src > main.jsx
1  import React from "react";
2  import ReactDOM from "react-dom/client";
3  import App from "./App";
4  import "./index.css";
5  import { BrowserRouter } from "react-router-dom";
6
7  ReactDOM.createRoot(document.getElementById("root")).render(
8    <React.StrictMode>
9      <BrowserRouter>
10        <App />
11      </BrowserRouter>
12    </React.StrictMode>
13  );
14
```

Ahora en la carpeta Pages creamos equipospage.jsx y tendrá el siguiente código.

```
pages
├── EquipoForm.jsx
├── EquipoPage.jsx
└── NotFound.jsx
```

```
EquipoPage.jsx X
client > src > pages > EquipoPage.jsx > ...
1 import { useEffect } from "react";
2 import EquipoCard from "../components/EquipoCard";
3 import { useEquipos } from "../context/EquipoProvider";
4
5 function EquipoPage() {
6   const { equipos, loadEquipo } = useEquipos();
7
8   useEffect(() => {
9     loadEquipo();
10   }, []);
11
12   function renderMain() {
13     if (equipos.length === 0)
14       return (
15         <div className="flex text-white mt-screen mt-6 mb-72">
16           <h1 className="text-5xl">No equipo aun</h1>
17         </div>
18       );
19     return equipos.map((equipo) => (
20       <EquipoCard equipo={equipo} key={equipo.id} />
21     ));
22   }
23
24   return (
25     <div>
26       <h1 className="text-5xl text-white font-bond text-center">Partidos</h1>
27       <div className="grid grid-cols-2 gap-3"> {renderMain()} </div>
28       <footer></footer>
29     </div>
30   );
31 }
32
33 export default EquipoPage;
34
```

Ahora en la misma carpeta Pages creamos equiposform.jsx

```
pages
├── EquipoForm.jsx
├── EquipoPage.jsx
└── NotFound.jsx
```

EquipoForm.jsx X

client > src > pages > EquipoForm.jsx > ...

```
1  import { Form, Formik } from "formik";
2  import { useEquipos } from "../context/EquipoProvider";
3  import { useParams, useNavigate } from "react-router-dom";
4  import { useEffect, useState } from "react";
5
6  function EquipoForm() {
7    const { createEquipo, getEquipo, updateEquipo } = useEquipos();
8    const [equipo, setEquipo] = useState({
9      eq1: "",
10     eq2: "",
11     description: "",
12   });
13   const params = useParams();
14   const navigate = useNavigate();
15
16   useEffect(() => {
17     const loadEquipo = async () => {
18       if (params.id) {
19         const equipo = await getEquipo(params.id);
20         console.log(equipo);
21         setEquipo({
22           eq1: equipo.eq1,
23           eq2: equipo.eq2,
24           description: equipo.description,
25         });
26       }
27     };
28     loadEquipo();
29   }, []);
30
31   return (
32     <div>
33       <Formik
34         initialValues={equipo}
35         enableReinitialize={true}
36         onSubmit={async (values, actions) => {
37           console.log(values);
38           if (params.id) {
39             await updateEquipo(params.id, values);
40           } else {
41             await createEquipo(values);
```



```

41     await createEquipo(values);
42   }
43   navigate("/");
44   setEquipo({
45     eq1: "",
46     eq2: "",
47     description: "",
48   });
49   }}
50   >
51   ({ handleChange, handleSubmit, values, isSubmitting }) => (
52     <Form
53       onSubmit={handleSubmit}
54       className="bg-slate-400 max-w-sm rounded-md p-4 mx-auto mt-10"
55     >
56       <h1 className="text-xl font-bold uppercase text-center">
57         {params.id ? "Editar Equipo" : "Crear Equipo"}
58       </h1>
59       <label className="block mt-2 mb-2">Equipo 1</label>
60       <input
61         type="text"
62         name="eq1"
63         placeholder="Escribe el nombre del Equipo 1"
64         className="px-2 py-1 rounded-md w-full"
65         onChange={handleChange}
66         value={values.eq1}
67       />
68       <label className="block mt-2 mb-2">Equipo 2</label>
69       <input
70         type="text"
71         name="eq2"
72         placeholder="Escribe el nombre del Equipo 2"
73         className="px-2 py-1 rounded-md w-full"
74         onChange={handleChange}
75         value={values.eq2}
76       />
77       <label className="block mt-2 mb-2">Description</label>
78       <textarea
79         name="description"
80         rows="3"
81         placeholder="Escribe una descripcion"

```

```

81         placeholder="Escribe una descripcion"
82         className="px-2 py-1 rounded-md w-full"
83         onChange={handleChange}
84         value={values.description}
85     ></textarea>
86     <button
87         type="submit"
88         disabled={isSubmitting}
89         className="block bg-green-500 px-2 py-1 mt-2 w-full rounded-md"
90     >
91         {isSubmitting ? "Guardando..." : "Guardar"}
92     </button>
93 </Form>
94 )}
95 </Formik>
96 </div>
97 );
98 }
99
100 export default EquipoForm;
101

```

Creamos otro archivo en la capeta Pages llamado. NotFound.jsx

```

NotFound.jsx X
client > src > pages > NotFound.jsx > NotFound
1  function NotFound() {
2      return (
3          <div className="flex justify-center px-20 py-5 text-white mt-10">
4              <h1>Noy Found</h1>
5          </div>
6      );
7  }
8
9  export default NotFound;
10

```

Ahora crearemos una carpeta llamada components y crearemos el navbar.jsx

```

v components
  EquipoCard.jsx
  FooterBar.jsx
  NavBar.jsx

```

```
NavBar.jsx X
client > src > components > NavBar.jsx > ...
1  import { Link } from "react-router-dom";
2
3  function NavBar() {
4    return (
5      <div className="bg-neutral-800 flex justify-between px-20 py-4">
6        <Link to="/" className="text-white font-bold">
7          <h1>Organizador de Partidos de Futbol Uts</h1>
8        </Link>
9        <ul className="flex gap-x-1">
10         <li>
11           <Link to="/" className="bg-sky-300 px-2 py-1 rounded-md">
12             Inicio
13           </Link>
14         </li>
15         <li>
16           <Link to="/new" className="bg-amber-300 px-2 py-1 rounded-md">
17             Crear Partidos
18           </Link>
19         </li>
20       </ul>
21     </div>
22   );
23 }
24
25 export default NavBar;
26
```

Nos vamos a la terminal e instalamos lo siguiente.

```
PS C:\Users\ADMIN\Desktop\Proyecto\client> npm i formik
```

Ahora creamos una carpeta api dentro de SRC y creamos un archivo llamado equipo.api.js

```
src
└─ api
   JS equipo.api.js
```

Ahora procederemos a instalar axios con el siguiente comando en la terminal.

```
PS C:\Users\ADMIN\Desktop\Proyecto\client> npm i axios
[#####.....] / idealTree:client: timing idealTree:#root Completed in 866ms
```

Ahora pondremos el siguiente código en equipo.api.js

```
import axios from "axios";

export const getEquiposRequest = async () =>
  await axios.get("http://localhost:4000/equipos");

export const getEquipoRequest = async (id) =>
  await axios.get(`http://localhost:4000/equipos/${id}`);

export const createEquiposRequest = async (equipo) =>
  await axios.post("http://localhost:4000/equipos", equipo);

export const updateEquipoRequest = async (id, newEquipo) =>
  await axios.put(`http://localhost:4000/equipos/${id}`, newEquipo);

export const deleteEquipoRequest = async (id) =>
  await axios.delete(`http://localhost:4000/equipos/${id}`);

export const toggleEquipoRequest = async (id, done) =>
  await axios.put(`http://localhost:4000/equipos/${id}`, { done });
```

Ahora lo que haremos es llamar en el índice al cors. Y el índice definitivo nos quedaría así

```
JS index.js  X
server > JS index.js > ...
1  import express from "express";
2  import cors from "cors";
3  import { dirname, join } from "path";
4  import { fileURLToPath } from "url";
5  import { PORT } from "../config.js";
6
7  import indexRoutes from "../routes/index.routes.js";
8  import equiposRoutes from "../routes/equipos.routes.js";
9
10 const app = express();
11 const __dirname = dirname(fileURLToPath(import.meta.url));
12 console.log(__dirname);
13
14 //El cors se le puede poner al backend y con este no da error de cors a
15 app.use(cors());
16 app.use(express.json());
17
18 app.use(express.static(join(__dirname, "../client/dist")));
19
20 app.use(indexRoutes);
21 app.use(equiposRoutes);
22
23 app.listen(PORT);
24 console.log(`El servidor esta corriendo en el puerto ${PORT}`);
25
```

En la carpeta components creamos un archivo llamado equipocard.jsx que nos ayudara a ordenar

EquipoCard.jsx

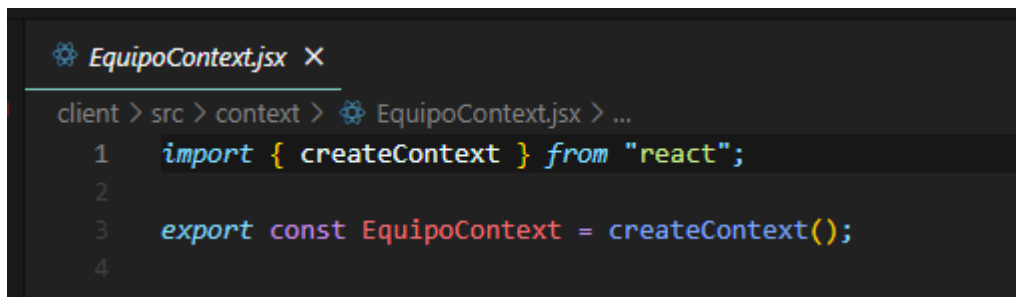
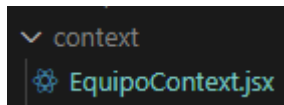
client > src > components > EquipoCard.jsx > ...

```

1  import { useEquipos } from "../context/EquipoProvider";
2  import { useNavigate } from "react-router-dom";
3
4  function EquipoCard({ equipo }) {
5    const { deleteEquipo, toggleEquipoDone } = useEquipos();
6    const navigate = useNavigate();
7
8    const handleDone = async () => {
9      await toggleEquipoDone(equipo.id);
10   };
11
12   return (
13     <div className="bg-zinc-700 text-white rounded-md p-4 mt-6">
14       <header className="flex justify-between">
15         <h2 className="text-lg font-bold">{equipo.eq1}</h2>
16         <p className="text-lg font-bold">VS</p>
17         <h2 className="text-lg font-bold">{equipo.eq2}</h2>
18         <span>{equipo.done == 1 ? "✓" : "✗"}</span>
19       </header>
20       <div className="flex gap-x-2 justify-center mt-2 mb-3">
21         <p className="text-md">{equipo.description}</p>
22         <p className="text-md">{equipo.createAt}</p>
23       </div>
24       <div className="flex gap-x-2 justify-center">
25         <button
26           className="bg-blue-500 px-2 py-1 text-black rounded-md"
27           onClick={() => navigate(`/edit/${equipo.id}`)}
28         >
29           Editar
30         </button>
31         <button
32           className="bg-red-500 px-2 py-1 text-black rounded-md"
33           onClick={() => deleteEquipo(equipo.id)}
34         >
35           Eliminar
36         </button>
37         <button
38           className="bg-green-500 px-2 py-1 text-black rounded-md"
39           onClick={() => handleDone(equipo.done)}
40         >
41           Partido Terminado
42         </button>
43       </div>
44     </div>
45   );
46 }
47
48 export default EquipoCard;

```

Creamos la carpeta context y creamos el archivo equipocontext.jsx que nos permite crear un componente que puede tener mas componentes dentro.

A screenshot of a code editor with a dark theme. The file 'EquipoContext.jsx' is open, and the editor shows the following code:

```
1 import { createContext } from "react";
2
3 export const EquipoContext = createContext();
4
```

Luego creamos en la misma carpeta un archivo llamado equipoprovider

⚙️ EquipoProvider.jsx ✕

client > src > context > ⚙️ EquipoProvider.jsx > ...

```
1  import { useContext, useState } from "react";
2  import {
3    getEquiposRequest,
4    getEquipoRequest,
5    createEquiposRequest,
6    updateEquipoRequest,
7    deleteEquipoRequest,
8    toggleEquipoRequest,
9  } from "../api/equipo.api";
10 import { EquipoContext } from "../EquipoContext";
11
12 export const useEquipos = () => {
13   const context = useContext(EquipoContext);
14   if (!context) {
15     throw new Error(
16       "El hook useEquipos deberia estar dentro de EquipoContextProvider"
17     );
18   }
19   return context;
20 };
21
22 export const EquipoContextProvider = ({ children }) => {
23   const [equipos, setEquipos] = useState([]);
24
25   async function loadEquipo() {
26     const response = await getEquiposRequest();
27     setEquipos(response.data);
28     // console.log(response.data);
29   }
30
31   const getEquipo = async (id) => {
32     try {
33       const response = await getEquipoRequest(id);
34       return response.data;
35     } catch (error) {
36       console.log(error);
37     }
38   };
39 }
```



```

39
40 const createEquipo = async (equipo) => {
41   try {
42     await createEquiposRequest(equipo);
43     // setEquipos([...equipos, response.data]);
44   } catch (error) {
45     console.log(error);
46   }
47 };
48
49 const updateEquipo = async (id, newEquipo) => {
50   try {
51     const response = await updateEquipoRequest(id, newEquipo);
52     console.log(response);
53   } catch (error) {
54     console.log(error);
55   }
56 };
57
58 const deleteEquipo = async (id) => {
59   try {
60     const response = await deleteEquipoRequest(id);
61     setEquipos(equipos.filter((equipo) => equipo.id !== id));
62   } catch (error) {
63     console.log(error);
64   }
65 };
66
67 const toggleEquipoDone = async (id) => {
68   try {
69     const equipoFound = equipos.find((equipo) => equipo.id === id);
70     await toggleEquipoRequest(id, equipoFound.done === 0 ? true : false);
71     setEquipos(
72       equipos.map((equipo) =>
73         equipo.id === id ? { ...equipo, done: !equipo.done } : equipo
74       )
75     );
76   } catch (error) {
77     console.log(error);
78   }
79 };
80
81 return (
82   <EquipoContext.Provider
83     value={{
84       equipos,
85       loadEquipo,
86       getEquipo,
87       createEquipo,
88       updateEquipo,
89       deleteEquipo,

```

```

81     return (
82       <EquipoContext.Provider
83         value={{
84           equipos,
85           loadEquipo,
86           getEquipo,
87           createEquipo,
88           updateEquipo,
89           deleteEquipo,
90           toggleEquipoDone,
91         }}
92       >
93         {children}
94       </EquipoContext.Provider>
95     );
96   };
97

```

Y con esto habría avacado toda la programación del fronted del aplicativo, ahora lo que haremos es agregar estilos de css con tailwind para eso necesitaremos instalas sus dependencias.

```

PS C:\Users\ADMIN\Desktop\Proyecto\client> npm install -D tailwindcss
[.....] | idealTree:client: sill idealTree buildDeps

```

Luego ejecutaremos este código para crear un archivo llamado tailwind donde aplicaremos los estilos para el aplicativo. Y en el index.css agregaremos lo siguiente.

```

PS C:\Users\ADMIN\Desktop\Proyecto\client> npx tailwindcss init

Created Tailwind CSS config file: tailwind.config.cjs
PS C:\Users\ADMIN\Desktop\Proyecto\client>

```

```

JS tailwind.config.cjs

```

```
JS tailwind.config.cjs X
client > JS tailwind.config.cjs > ...
1  /** @type {import('tailwindcss').Config} */
2  module.exports = {
3    content: ["/index.html", "/src/**/*.{js,ts,jsx,tsx}"],
4    theme: {
5      extend: {},
6    },
7    plugins: [],
8  };
9
```

```
# index.css 3 X
client > src > # index.css
1  @tailwind base;
2  @tailwind components;
3  @tailwind utilities;
4
```

En las imágenes anteriores se puede presenciar el código ya con el css aplicado, a continuación, mostrare el aplicativo definitivo y funcionando.

