

React + MySql

Tutorial

Crear carpeta para el proyecto en general

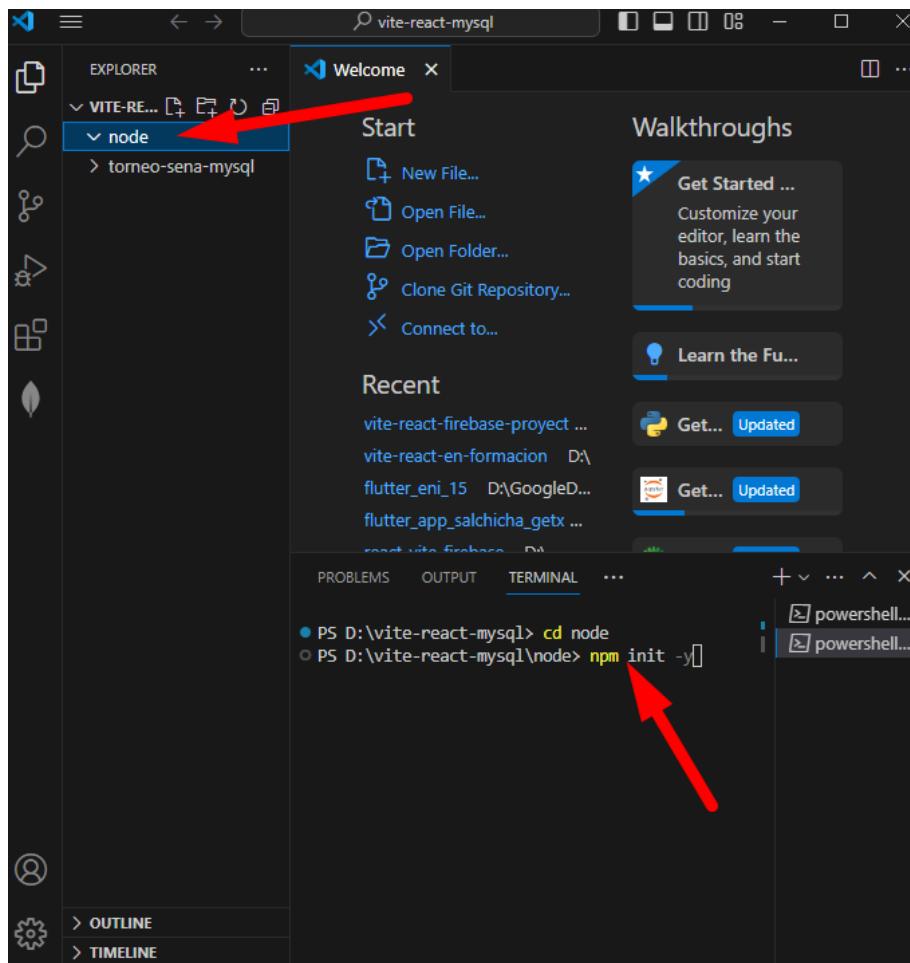
En este ejemplo se llamará **vite-react-mysql**

Dentro de la carpeta debe crear dos carpetas:

1. node (se crea manualmente).
2. Crear proyecto con vite (aquí se crea una carpeta por defecto). En la consola digitar el comando: **npm create vite**.

Back-end de la aplicación con Express

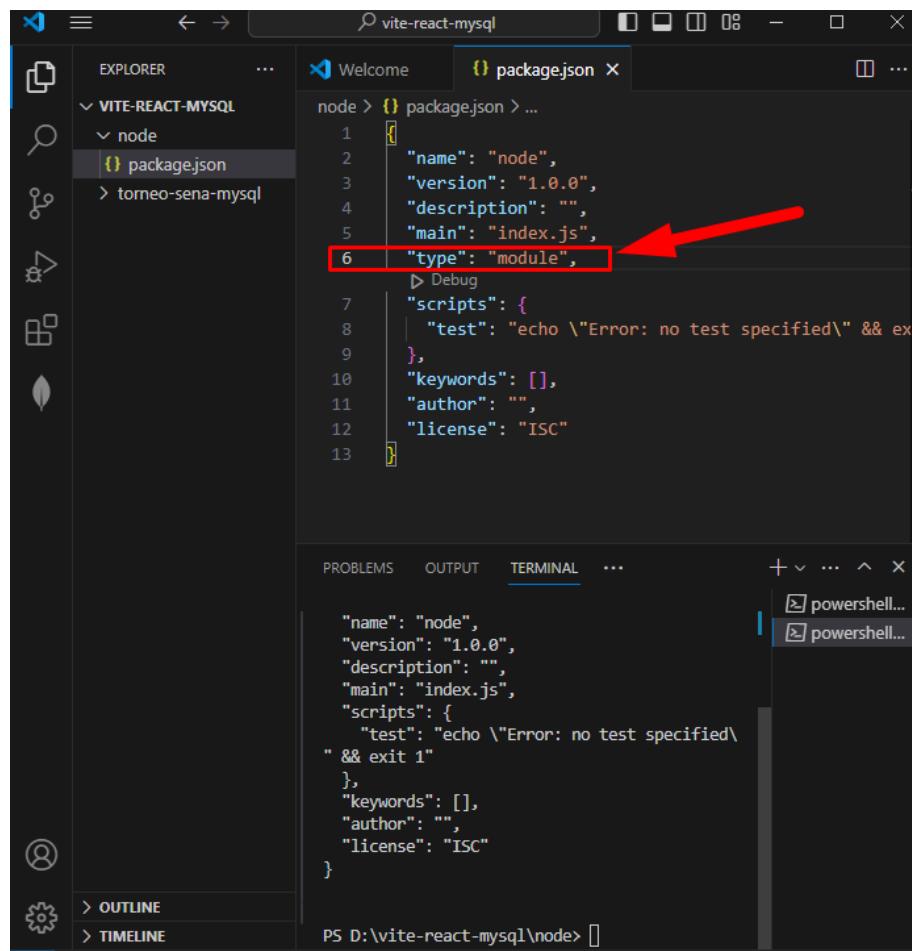
Ingresar a carpeta node y ejecutar **npm init -y**



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left has a tree view with a folder named 'node' selected, indicated by a red arrow. The terminal at the bottom has two lines of command history: 'PS D:\vite-react-mysql> cd node' and 'PS D:\vite-react-mysql\node> npm init -y'. A red arrow points from the text 'Ingresar a carpeta node y ejecutar npm init -y' to the terminal line 'npm init -y'.

```
PS D:\vite-react-mysql> cd node
PS D:\vite-react-mysql\node> npm init -y
```

Se crea un archivo llamado package.json, en él, agregar la propiedad type – module:



```
node > {} package.json > ...
1  {
2    "name": "node",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module", -----^
7    "scripts": {
8      "test": "echo \\\"Error: no test specified\\\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC"
13 }
```

The screenshot shows the VS Code interface with the package.json file open in the center editor. The line `6 "type": "module",` is highlighted with a red rectangular box and a red arrow pointing to the right. The bottom terminal window shows a PowerShell prompt with the command `PS D:\vite-react-mysql\node>`.

Instalación de paquetes

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file, search, file tree, and others. The main area shows the 'package.json' file with the following content:

```
node > {} package.json > ...
1  {
2    "name": "node",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "echo \\"Error: no test specified\\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC"
13 }
```

Below the code editor is a tab bar with PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing a command prompt window with the following text:

```
"version": "1.0.0",
"description": "",
"main": "index.js",
"scripts": {
  "test": "echo \\"Error: no test specified\\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC"
}

PS D:\vite-react-mysql\node> npm i express cors mysql2 sequelize
o [ ..... ] \ idealTree:node: sill idealTree buildDeps
```

A red arrow points from the text 'npm i express cors mysql2 sequelize' in the terminal to the terminal window.

Express es el framework de node js para el backend de aplicaciones web o móviles.

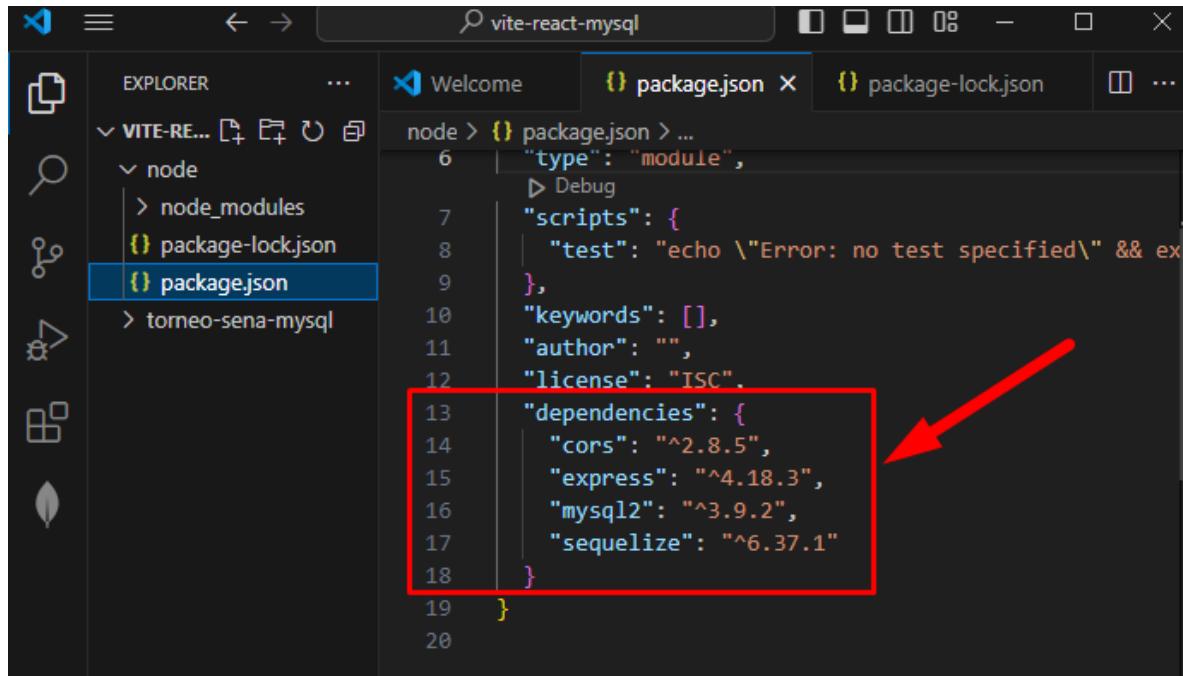
Cors quiere decir Cross-Origin Resource Sharing (Compartir recursos entre orígenes cruzados). Es una política de seguridad implementada por los navegadores web que restringe las solicitudes HTTP entre diferentes dominios, para proteger a los usuarios de ciertos ataques, como solicitudes maliciosas de sitios no autorizados.

Para desarrollar una app con Node.js y React, se necesitan dos servidores, los cuales se ejecutan en dominios diferentes (uno para el frontend y otro para el backend). Por ejemplo, la aplicación React podría estar siendo servida desde <http://localhost:3000> y el servidor Node.js podría estar en <http://localhost:8000>.

Cuando el código en el navegador (frontend) intenta realizar una solicitud HTTP a un servidor en otro dominio (backend), el navegador implementa CORS para asegurarse de que el servidor permita esa solicitud desde un origen diferente. Sin la configuración adecuada de CORS en el servidor, el navegador bloqueará esas solicitudes por motivos de seguridad.

Mysql2 es el controlador de la base de datos MySQL

Sequelize es el ORM para gestionar la base de datos MySQL, un ORM (Object-Relational Mapping) es una técnica de programación que permite trabajar con datos de la base de datos utilizando objetos y métodos en lugar de escribir consultas SQL directamente.

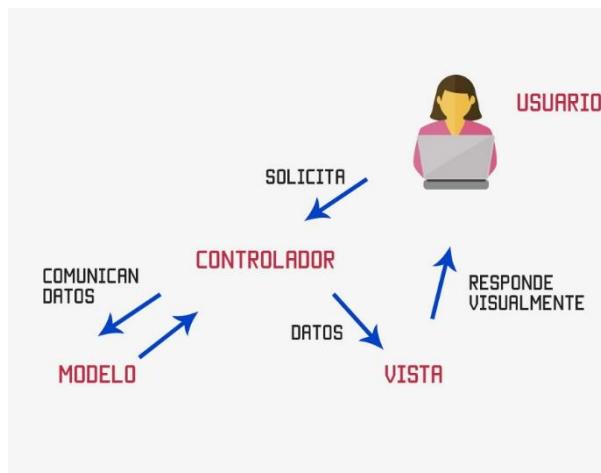


The screenshot shows the VS Code interface with the 'package.json' file open. A red box highlights the 'dependencies' section, which includes 'cors', 'express', 'mysql2', and 'sequelize'. A red arrow points from the text above to this highlighted section.

```
node > {} package.json > ...
6   "type": "module",
7   "scripts": {
8     "test": "echo \\"Error: no test specified\\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "cors": "^2.8.5",
15    "express": "^4.18.3",
16    "mysql2": "^3.9.2",
17    "sequelize": "^6.37.1"
18  }
19 }
20 }
```

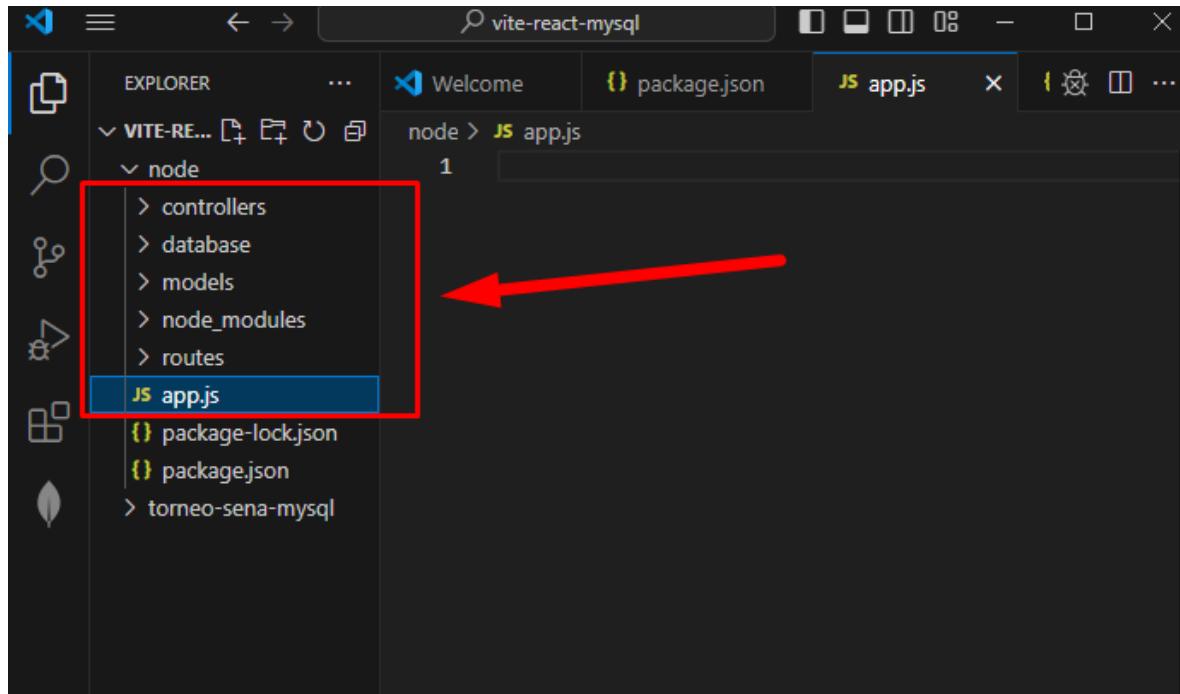
Antes de continuar, ¿qué es MVC (Modelo vista controlador)?

Según la página de código facilito, MVC es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación.

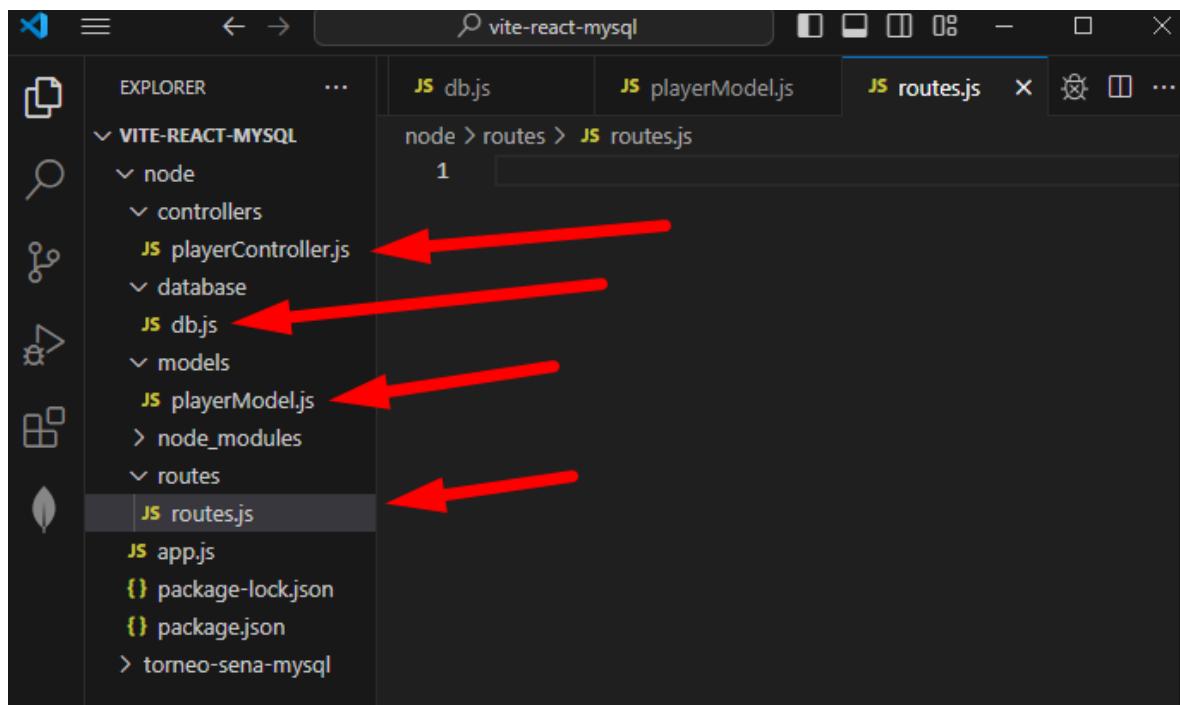


A este proyecto se le suman las RUTAS y la arquitectura SPA (Single-Page Application) de REACT.

Crear estructura de carpetas y archivo app.js:



Crear archivos en carpetas:



Creación de la base de datos y la tabla players:

1. Primero se debe realizar la instalación de la herramienta XAMPP
2. Ejecutar los servidores de apache y mysql
3. Abrir phpMyAdmin

The image consists of two vertically stacked screenshots of the phpMyAdmin interface.

Top Screenshot (Bases de datos page):

- Step 1:** A red arrow points to the "Nueva" (New) button in the sidebar.
- Step 2:** A red arrow points to the "Crear base de datos" (Create Database) button.
- Step 3:** A red arrow points to the "utf8mb4_general_ci" dropdown menu.
- Step 4:** A red box highlights the database name "torneo_react".

Bottom Screenshot (Crear tabla page):

- Step 1:** A red arrow points to the "torneo_react" database in the sidebar.
- Step 2:** A red arrow points to the "Nombre:" (Name:) input field containing "players".
- Step 3:** A red arrow points to the "Número de columnas:" (Number of columns:) dropdown menu set to "7".
- Step 4:** A red box highlights the "Continuar" (Continue) button.

Nombre de la tabla: players

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice	A.I.	Comentarios	Virtualidad
<input type="text" value="id"/>	<input type="text" value="INT"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="PRIMARY"/>	<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="documento"/>	<input type="text" value="INT"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="nombres"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="50"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="apellidos"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="50"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="genero"/>	<input type="text" value="ENUM('F', 'M', 'O')"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="createdat"/>	<input type="text" value="TIMESTAMP"/>	<input type="text"/>	<input type="text" value="CURRENT_TIMESTAMP"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="updatedat"/>	<input type="text" value="TIMESTAMP"/>	<input type="text"/>	<input type="text" value="CURRENT_TIMESTAMP"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>

Estructura

Comentarios de la tabla: Cotejamiento: Motor de almacenamiento: InnoDB

definición de la PARTICIÓN:

Particiones:

Previsualizar SQL Guardar

Estructura de tabla

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	<input type="text" value="id"/>	<input type="text" value="int(11)"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="Ninguna"/>	<input type="text" value="AUTO_INCREMENT"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
2	<input type="text" value="documento"/>	<input type="text" value="int(11)"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="Ninguna"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
3	<input type="text" value="nombres"/>	<input type="text" value="varchar(50)"/>	<input type="text" value="utf8mb4_general_ci"/>	<input type="checkbox"/>	<input type="text" value="Ninguna"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
4	<input type="text" value="apellidos"/>	<input type="text" value="varchar(50)"/>	<input type="text" value="utf8mb4_general_ci"/>	<input type="checkbox"/>	<input type="text" value="Ninguna"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
5	<input type="text" value="genero"/>	<input type="text" value="enum('F', 'M', 'O')"/>	<input type="text" value="utf8mb4_general_ci"/>	<input type="checkbox"/>	<input type="text" value="Ninguna"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
6	<input type="text" value="createdat"/>	<input type="text" value="timestamp"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="current_timestamp()"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
7	<input type="text" value="updatedat"/>	<input type="text" value="timestamp"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text" value="current_timestamp()"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Seleccionar todo Para los elementos que están marcados: Examinar Cambiar Eliminar Primaria Único Índice

Imprimir Planteamiento de la estructura de tabla Mover columnas Normalizar

Agregar 1 columna(s) después de updatedat Continuar

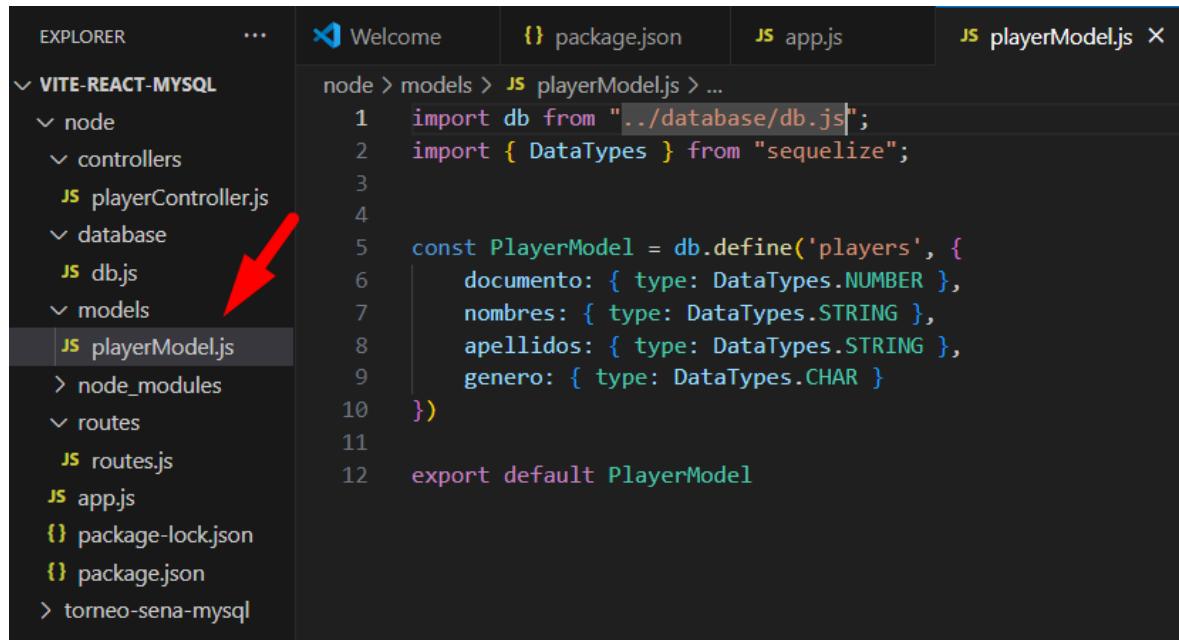
Conexión a la base de datos:

```

node > database > db.js > ...
1 import { Sequelize } from "sequelize";
2
3 //Como parámetros va el nombre de la bd, el usuario de la bd, la contraseña,
4 const db = new Sequelize('torneo_react', 'root', '', {
5   host: 'localhost',
6   dialect: 'mysql'
7 })
8
9 export default db

```

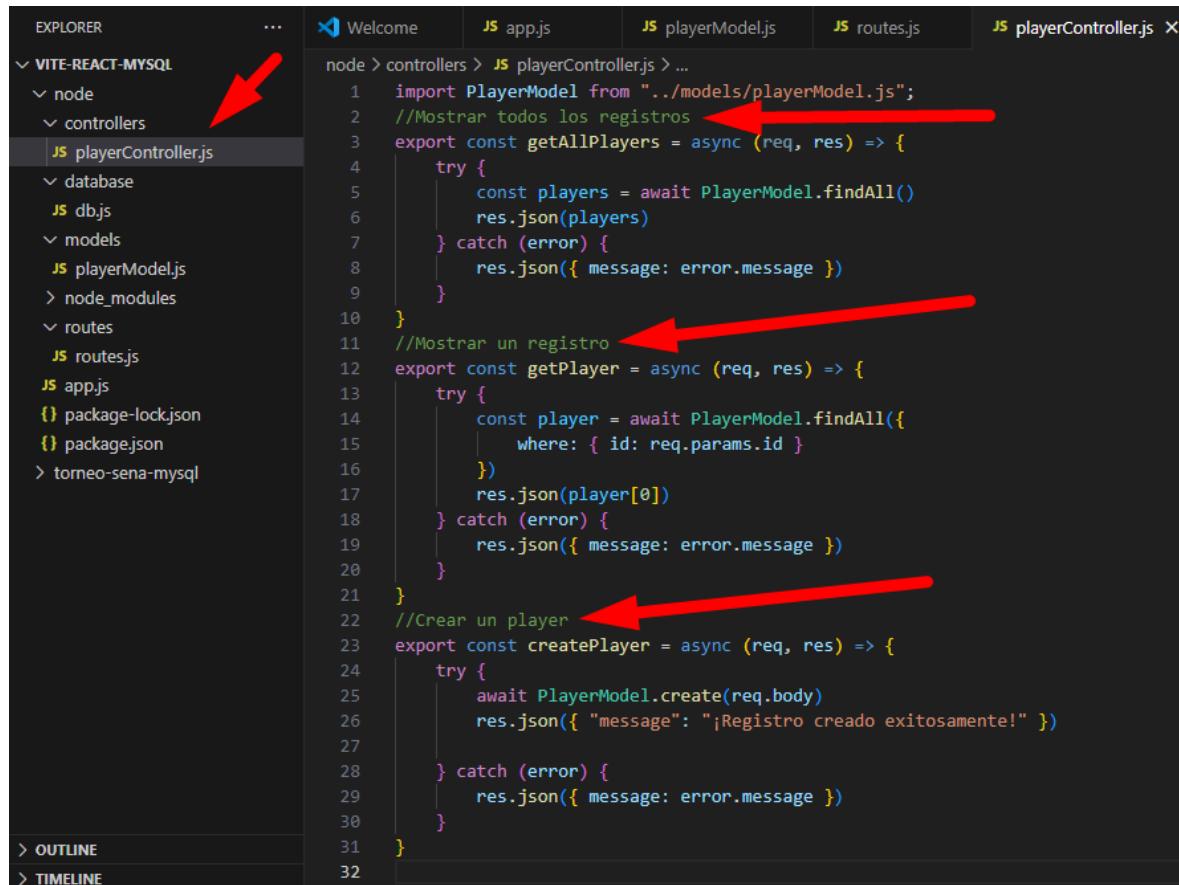
Ahora, crear los modelos (son los que se conectan a las tablas de la base de datos):



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project structure for 'VITE-REACT-MYSQL'. The 'models' folder contains a file named 'playerModel.js', which is currently selected and highlighted in grey. On the right, the main editor window shows the code for 'playerModel.js'. The code defines a Sequelize model for the 'players' table:

```
node > models > JS playerModel.js > ...
1 import db from "../database/db.js";
2 import { DataTypes } from "sequelize";
3
4
5 const PlayerModel = db.define('players', {
6   documento: { type: DataTypes.NUMBER },
7   nombres: { type: DataTypes.STRING },
8   apellidos: { type: DataTypes.STRING },
9   genero: { type: DataTypes.CHAR }
10})
11
12 export default PlayerModel
```

Configurar el controlador



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project structure for 'VITE-REACT-MYSQL'. The 'controllers' folder contains a file named 'playerController.js', which is currently selected and highlighted in grey. On the right, the main editor window shows the code for 'playerController.js'. The code defines three asynchronous functions: 'getAllPlayers', 'getPlayer', and 'createPlayer'. Red arrows highlight each function definition.

```
node > controllers > JS playerController.js > ...
1 import PlayerModel from "../models/playerModel.js";
2 //Mostrar todos los registros
3 export const getAllPlayers = async (req, res) => {
4   try {
5     const players = await PlayerModel.findAll()
6     res.json(players)
7   } catch (error) {
8     res.json({ message: error.message })
9   }
10 }
11 //Mostrar un registro
12 export const getPlayer = async (req, res) => {
13   try {
14     const player = await PlayerModel.findAll({
15       where: { id: req.params.id }
16     })
17     res.json(player[0])
18   } catch (error) {
19     res.json({ message: error.message })
20   }
21 }
22 //Crear un player
23 export const createPlayer = async (req, res) => {
24   try {
25     await PlayerModel.create(req.body)
26     res.json({ "message": "¡Registro creado exitosamente!" })
27   } catch (error) {
28     res.json({ message: error.message })
29   }
30 }
31 }
```

playerController.js

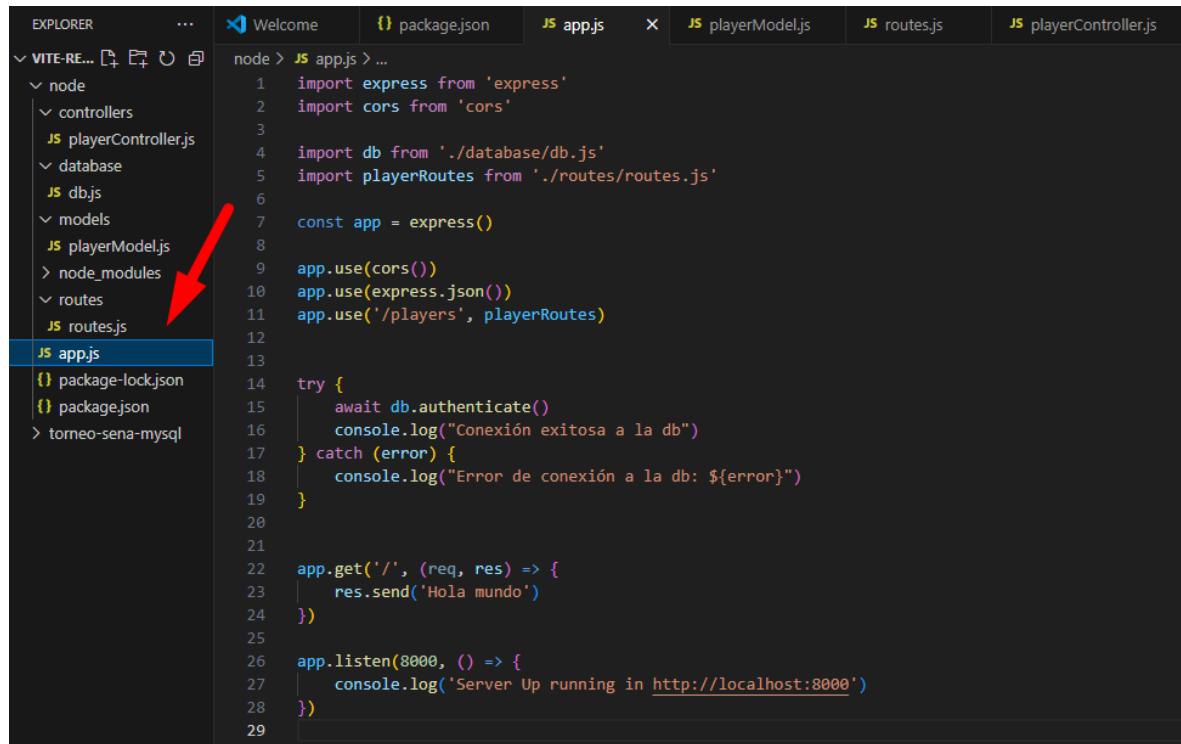
```
34
35 //Actualizar un registro
36 export const updatePlayer = async (req, res) => {
37   try {
38     await PlayerModel.update(req.body, {
39       where: { id: req.params.id }
40     })
41     res.json({ "message": "¡Registro actualizado exitosamente!" })
42   } catch (error) {
43     res.json({ message: error.message })
44   }
45 }
46
47 //Borrar un registro
48 export const deletePlayer = async (req, res) => {
49   try {
50     await PlayerModel.destroy({
51       where: { id: req.params.id }
52     })
53     res.json({ "message": "¡Registro borrado exitosamente!" })
54   } catch (error) {
55     res.json({ message: error.message })
56   }
57 }
```

Configurar las rutas

routes.js

```
1 import express from "express";
2 import { createPlayer, deletePlayer, getAllPlayers, getPlayer, updatePlayer } from "../controllers/
3 const router = express.Router();
4
5 router.get('/', getAllPlayers)
6 router.get('/:id', getPlayer)
7 router.post('/', createPlayer)
8 router.put('/:id', updatePlayer)
9 router.delete('/:id', deletePlayer)
10
11 export default router
```

Configurar app.js



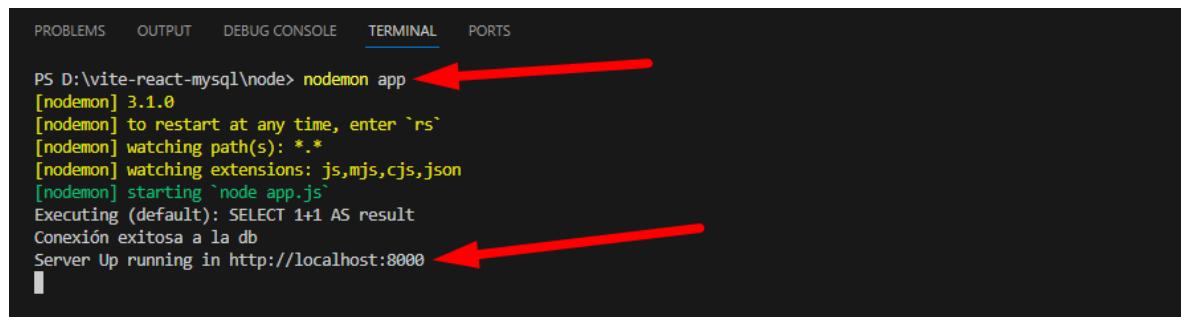
```
node > JS app.js > ...
1 import express from 'express'
2 import cors from 'cors'
3
4 import db from './database/db.js'
5 import playerRoutes from './routes/routes.js'
6
7 const app = express()
8
9 app.use(cors())
10 app.use(express.json())
11 app.use('/players', playerRoutes)
12
13
14 try {
15   await db.authenticate()
16   console.log("Conexión exitosa a la db")
17 } catch (error) {
18   console.log("Error de conexión a la db: ${error}")
19 }
20
21
22 app.get('/', (req, res) => {
23   res.send('Hola mundo')
24 })
25
26 app.listen(8000, () => {
27   console.log('Server Up running in http://localhost:8000')
28 })
```

Instalar **nodemon** para ejecutar el servidor:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS D:\vite-react-mysql\node> npm install -g nodemon
added 33 packages in 4s
4 packages are looking for funding
  run `npm fund` for details
○ PS D:\vite-react-mysql\node>
```

Levantar el servidor con el comando: **nodemon app**

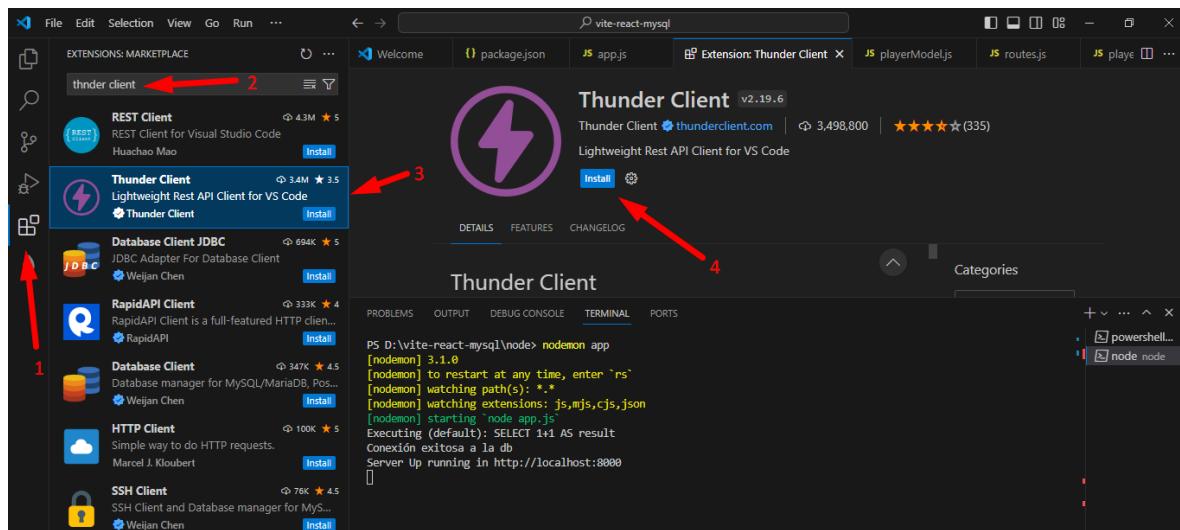


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\vite-react-mysql\node> nodemon app
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Executing (default): SELECT 1+1 AS result
Conexión exitosa a la db
Server Up running in http://localhost:8000
```

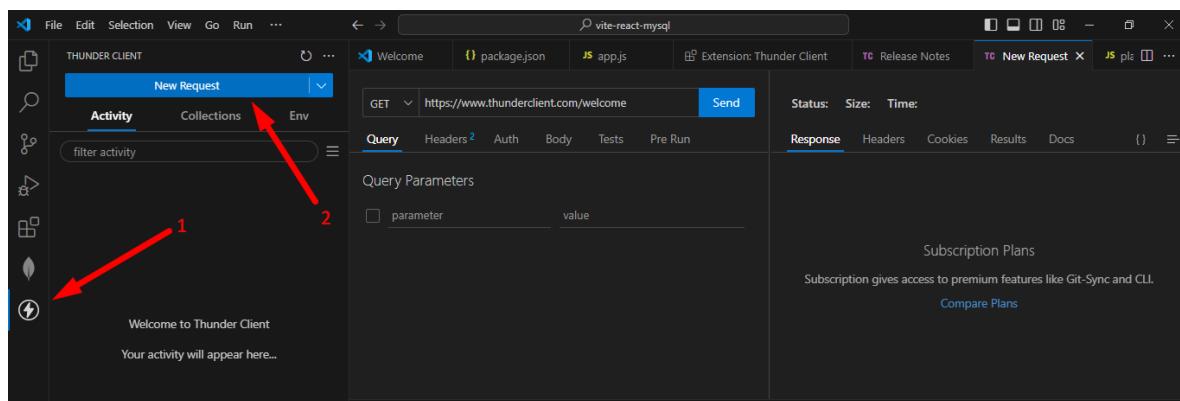
Resultado:



Instalar **Thunder Client** para hacer pruebas en la base de datos sin necesidad de un frontend:

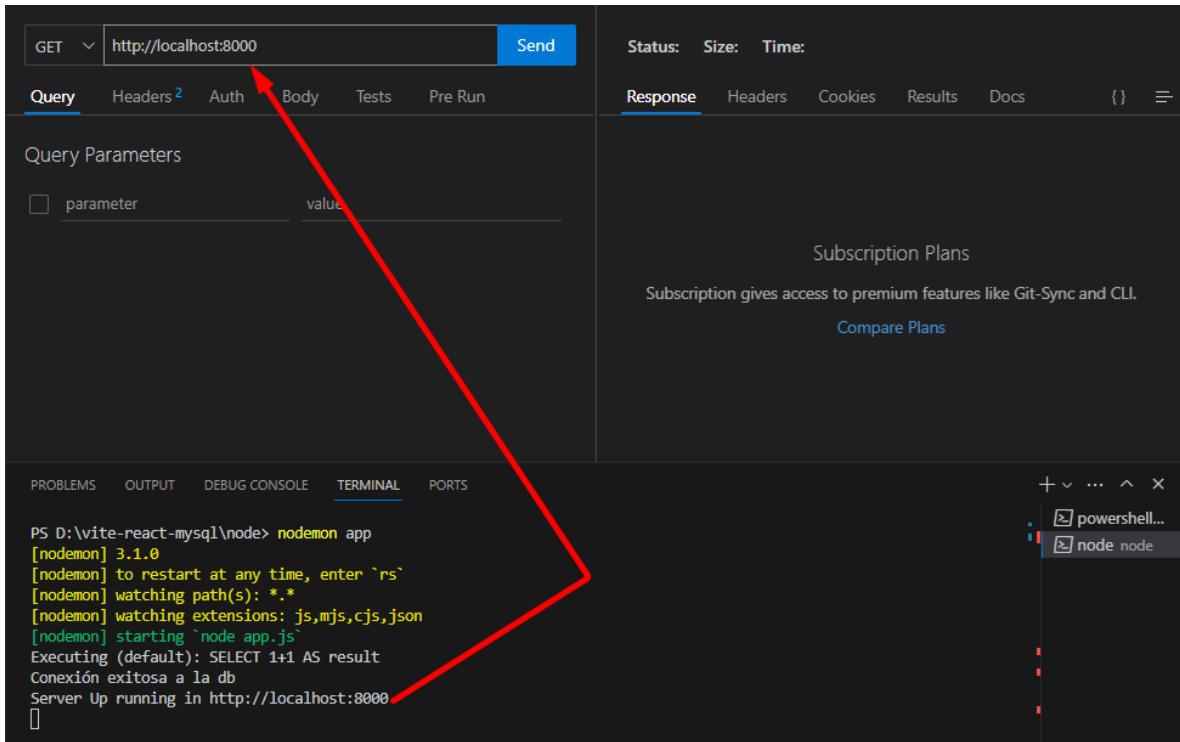


Ejecutar Thunder Client

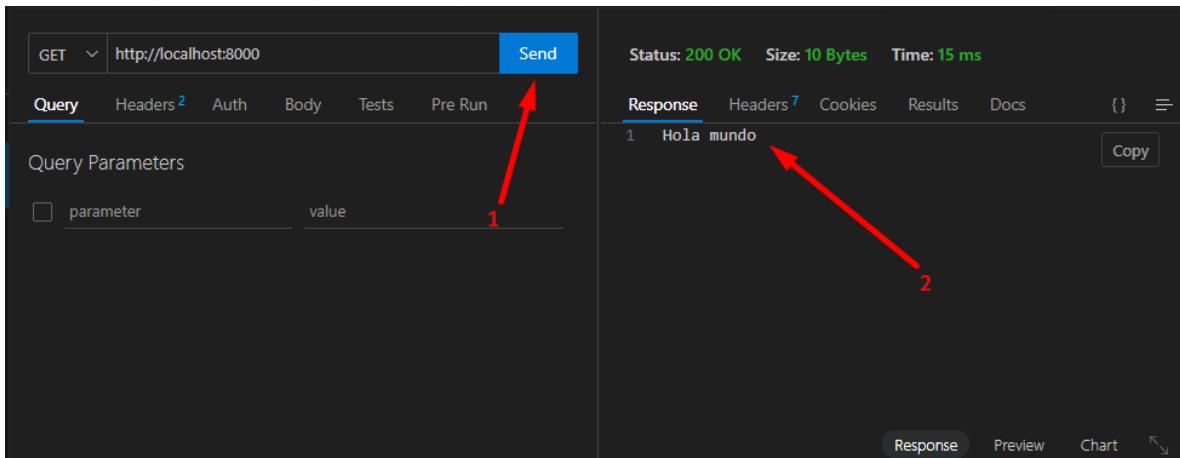


Copiar la dirección del servidor:

Instructor Jorge E. Andrade C.
jandrade@senra.edu.co
Centro Agropecuario La Granja
Regional Tolima



Realizar una prueba:



Probar los métodos del controlador **playerController**:

Recuerde que en el archivo `app.js` en la línea 11 se propuso que la ruta para el módulo de jugadores será '/players':

```

Welcome package.json app.js Extension: Thunder Client Release Notes New Request
node > js app.js > ...
1 import express from 'express'
2 import cors from 'cors'
3
4 import db from './database/db.js'
5 import playerRoutes from './routes/routes.js'
6
7 const app = express()
8
9 app.use(cors())
10 app.use(express.json())
11 app.use('/players', playerRoutes) ← Red arrow points here
12
13
14 try {
15   await db.authenticate()
16   console.log("Conexión exitosa a la db")
17 } catch (error) {
18   console.log("Error de conexión a la db: ${error}")
19 }

```

Agregando esa parte de la ruta y dando clic en **Send** la base de datos responde con los registros de la tabla (por el momento solo uno):

Status: 200 OK Size: 162 Bytes Time: 14 ms

```

1 [
2   {
3     "id": 1,
4     "documento": "12345678",
5     "nombres": "Jorge",
6     "apellidos": "Andrade",
7     "genero": "M",
8     "createdAt": "2024-03-19T15:46:21.000Z",
9     "updatedAt": "2024-03-19T15:46:21.000Z"
10   }
11 ]

```

Insertar otro registro:

Columna	Tipo	Función	Nulo	Valor
id	int(11)			
documento	int(11)			98745612
nombres	varchar(50)			Lina
apellidos	varchar(50)			Linares
genero	enum	--		M
createdat	timestamp			current_timestamp()
updatedat	timestamp			current_timestamp()

id	documento	nombres	apellidos	genero	createdat	updatedat
1	12345678	Jorge	Andrade	M	2024-03-19 10:46:21	2024-03-19 10:46:21
2	98745612	Lina	Linares	M	2024-03-19 11:06:58	2024-03-19 11:06:58

Ahora realizar una consulta por id (a la derecha el archivo de las rutas que se establecieron manualmente):

The screenshot shows a developer environment with two tabs open: `playerModel.js` and `playerController.js`. On the right, the `routes.js` file is displayed with the following code:

```
node > routes > JS routes.js > ...
1 import express from "express";
2 import { createPlayer, deletePlayer, getAllPlayers,
3   getPlayer, updatePlayer } from "../controllers/
4   playerController.js";
5 const router = express.Router();
6
7 router.get('/', getAllPlayers)
8 router.get('/:id', getPlayer)
9 router.post('/', createPlayer)
10 router.put('/:id', updatePlayer)
11 router.delete('/:id', deletePlayer)
```

On the left, a browser-like interface shows a `POST` request to `http://localhost:8000/players/2`. The response status is `200 OK`, size is `159 Bytes`, and time is `9 ms`. The response body is a JSON object:

```
{ "id": 2, "documento": 98745612, "nombres": "Lina", "apellidos": "Linares", "genero": "M", "createdAt": "2024-03-19T16:06:58.000Z", "updatedAt": "2024-03-19T16:06:58.000Z" }
```

A red arrow points from the `Send` button in the request interface to the `getPlayer` method in the `routes.js` code.

Recuerde el código usado en el controlador para consultar un registro:

The screenshot shows the VS Code Explorer on the left and the `playerController.js` file on the right. The `playerController.js` file contains the following code:

```
//Mostrar un registro
12 export const getPlayer = async (req, res) => {
13   try {
14     const player = await PlayerModel.findAll({
15       where: { id: req.params.id }
16     })
17     res.json(player[0])
18   } catch (error) {
19     res.json({ message: error.message })
20   }
21 }
```

Red arrows highlight the `node` folder in the Explorer and the `getPlayer` method in the `playerController.js` code.

Para agregar un registro nuevo, se cambia el método a **POST** y en la parte inferior se envían los datos como un objeto JSON:

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8000/players/
- Body:** JSON (highlighted by red arrow 2)
- Request Body Content:**

```
{
  "documento": 14785200,
  "nombres": "Juan",
  "apellidos": "Rodriguez",
  "genero": "M"
}
```
- Send Button:** A blue button labeled "Send" (highlighted by red arrow 5).
- Status:** 200 OK
- Response:**

```
{
  "message": "¡Registro creado exitosamente!"
}
```
- Code Snippet:** A snippet of routes.js code from a Node.js application (highlighted by red arrow 4).

```

node > routes > JS routes.js > ...
1  import express from "express";
2  import { createPlayer, deletePlayer, getAllPlayers, get
3  const router = express.Router()
4
5  router.get('/', getAllPlayers)
6  router.get('/:id', getPlayer)
7  router.post('/', createPlayer)
8  router.put('/:id', updatePlayer)
9  router.delete('/:id', deletePlayer)
10
11 export default router

```

Resultado en base de datos:

The screenshot shows the phpMyAdmin interface for the "torneo_react" database, specifically the "players" table.

- Table Structure:** The table has columns: id, documento, nombres, apellidos, genero, createdat, updatedat.
- Data Rows:**

	id	documento	nombres	apellidos	genero	createdat	updatedat
<input type="checkbox"/>	1	12345678	Jorge	Andrade	M	2024-03-19 10:46:21	2024-03-19 10:46:21
<input type="checkbox"/>	2	98745612	Lina	Linares	M	2024-03-19 11:06:58	2024-03-19 11:06:58
<input type="checkbox"/>	3	14785200	Juan	Rodriguez	M	2024-03-19 11:12:57	2024-03-19 11:12:57
- Operations:** Buttons for Editar, Copiar, and Borrar are available for each row.
- Left Sidebar:** Shows the database structure with "Nueva" and "players" highlighted (highlighted by red arrow 1).

Para actualizar un registro es similar al de agregar, la diferencia esta en el método a usar y en el identificador que debe ir en la barra de direcciones:

The screenshot shows a browser-based API testing interface. At the top, there are tabs for 'Welcome', 'app.js', 'New Request', 'play', and more. The 'New Request' tab is active, showing a 'PUT' method selected in the dropdown. The URL is set to 'http://localhost:8000/players/2'. Below the URL, there are tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Body' tab is selected, showing JSON input:

```
1 {
2   "nombres": "Lina Ximena",
3   "genero": "F"
4 }
```

A red box highlights the JSON input. On the right side of the interface, the file 'routes.js' is open in a code editor, showing the Express.js router configuration:

```
1 import express from "express";
2 import { createPlayer, deletePlayer, getAllPlayers, getOnePlayer, updatePlayer } from "../controllers/playerController";
3 const router = express.Router();
4
5 router.get('/', getAllPlayers);
6 router.get('/:id', getOnePlayer);
7 router.post('/', createPlayer);
8 router.put('/:id', updatePlayer); // This line is highlighted with a red box
9 router.delete('/:id', deletePlayer);
10
11 export default router;
```

A red arrow points from the highlighted 'updatePlayer' line in the routes.js code to the highlighted JSON input in the testing tool. At the bottom, the status bar shows 'Status: 200 OK', 'Size: 50 Bytes', and 'Time: 287 ms'. The response pane displays the JSON output:

```
1 {
2   "message": "¡Registro actualizado exitosamente!" // This line is highlighted with a red box
3 }
```

En la ejecución anterior se actualizó el nombre y el género del registro con id: 2:

Fíjese que también se actualiza el campo updateat con la fecha en que se realizó la actualización:

Para eliminar un registro se cambia el método por **DELETE** y se envía en la URL el identificador del registro:

The screenshot shows the Postman interface with a successful DELETE request to `http://localhost:8000/players/3`. The response body is:

```
1 {  
2   "message": "¡Registro borrado exitosamente!"  
3 }
```

Resultado: ya no se encuentra el registro con id: 3:

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0002 segundos.)

```
SELECT * FROM `players`
```

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

	Mostrar todo	Número de filas:	25	Filtrar filas:	Buscar en esta tabla	Sort by key:	Ninguna				
+ Opciones	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
	<input type="button" value="←"/>	<input type="button" value="→"/>	<input type="button" value="▼"/>	<input type="button" value="id"/>	<input type="button" value="documento"/>	<input type="button" value="nombres"/>	<input type="button" value="apellidos"/>	<input type="button" value="genero"/>	<input type="button" value="createdat"/>	<input type="button" value="updatedat"/>	
	<input type="checkbox"/>	<input type="button" value="Editar"/>	<input type="button" value="Copiar"/>	<input type="button" value="Borrar"/>	1	12345678	Jorge	Andrade	M	2024-03-19 10:46:21	2024-03-19 10:46:21
	<input type="checkbox"/>	<input type="button" value="Editar"/>	<input type="button" value="Copiar"/>	<input type="button" value="Borrar"/>	2	98745612	Lina Ximena	Linares	F	2024-03-19 11:06:58	2024-03-20 10:45:47

Seleccionar todo Para los elementos que están marcados:

Front-end de la aplicación con REACT

1. Abrir una nueva consola en VSC sin detener el servidor de node.
 2. Crear aplicación con el comando `npm create vite`.
 3. Asignar el nombre de la aplicación.
 4. Seleccionar el framework REACT.
 5. Seleccionar la variante JavaScript.
 6. Ingresar a la carpeta que se ha creado con el nombre del proyecto.

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4 import './index.css'
5
6 ReactDOM.createRoot(document.getElementById('root')).render(
7   <React.StrictMode>
8     | <App />
9   </React.StrictMode>,
10 )
11
```

¿Qué es Axios?

Es una librería de JavaScript que se utiliza para hacer solicitudes HTTP desde el navegador o desde un servidor usando Node.js. Está basada en la simplicidad y una de sus características principales es la facilidad de uso mientras ofrece funciones avanzadas para el manejo de solicitudes y respuestas en la API. En otras palabras, es una librería que permite la comunicación entre el frontend y el backend para gestionar los registros de una base de datos MySQL.

Instalación de Axios:

Ingresar a la carpeta del proyecto de REACT y digitar el comando `npm install axios`:

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4 import './index.css'
5
6 ReactDOM.createRoot(document.getElementById('root')).render(
7   <React.StrictMode>
8     | <App />
9   </React.StrictMode>,
10 )
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
found 0 vulnerabilities
PS D:\vite-react-mysql\torneo-sena-mysql> * History restored
● PS D:\vite-react-mysql> cd torneo-sena-mysql
● PS D:\vite-react-mysql\torneo-sena-mysql> npm install axios
added 9 packages, and audited 236 packages in 2s
101 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
PS D:\vite-react-mysql\torneo-sena-mysql>
```

Instalar React Router:

<https://reactrouter.com/en/main/start/tutorial>

The screenshot shows a browser window with the URL <https://reactrouter.com/en/main/start/tutorial>. The page content includes a sidebar with 'Getting Started' and 'Upgrading' sections, and a main area with a 'Tutorial' section. In the 'Tutorial' section, there is a code block for setting up a Vite project with React Router. A red arrow points to the command `npm install react-router-dom localforage match-sorter sort-by`. Below this, a terminal window is open in a dark mode interface. The terminal shows the command being run in a directory `D:\vite-react-mysql\torneo-sena-mysql>`. A red arrow points from the browser's code block to this terminal command. The terminal also displays the output of the command, including package installation details and audit results.

Instalar Bootstrap

<https://getbootstrap.com/>

The screenshot shows the official Bootstrap website at getbootstrap.com. The main heading is "Build fast, responsive sites with Bootstrap". Below it, a subtext reads: "Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins." A red arrow points to the "Read the docs" button, which is located next to a "Download" button containing the command "\$ npm i bootstrap@5.3.2". At the bottom, it says "Currently v5.3.2 · Download · All releases".

The screenshot shows a terminal window with the following text:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\vite-react-mysql\torneo-sena-mysql> npm i bootstrap@5.3.2
● added 2 packages, and audited 250 packages in 2s

103 packages are looking for funding
  run `npm fund` for details

2 vulnerabilities (1 moderate, 1 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\vite-react-mysql\torneo-sena-mysql>
  
```

A red arrow points to the command "npm i bootstrap@5.3.2".

The screenshot shows a code editor with the file "main.jsx" selected in the Explorer sidebar. The code in the editor is:

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App.jsx';
import './index.css';
import 'bootstrap/dist/css/bootstrap.min.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    | <App />
  </React.StrictMode>
)
  
```

A red arrow points to the import statement "import 'bootstrap/dist/css/bootstrap.min.css';".

Instalar sweetalert

<https://sweetalert2.github.io/>

1

2

3

INTEGRATIONS WITH MAJOR JS FRAMEWORKS

React Vue Angular

CONFIGURATION PARAMS

npm install --save sweetalert2 sweetalert2-react-content

Installation

Usage Example

```
import Swal from 'sweetalert2'
import withReactContent from 'sweetalert2-react-content'

const MySwal = withReactContent(Swal)

MySwal.fire({
  title: <p>Hello World</p>,
  didOpen: () => {
    // `MySwal` is a subclass of `Swal` with all the same instance & static methods
    MySwal.showLoading()
  },
}).then(() => {
  return MySwal.fire(<p>Shorthand works too</p>)
})
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run `npm audit` for details.
PS D:\vite-react-mysql\torneo-sena-mysql> npm install --save sweetalert2 sweetalert2-react-content
added 2 packages, and audited 252 packages in 3s
104 packages are looking for funding
  run `npm fund` for details
3 vulnerabilities (1 low, 1 moderate, 1 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\vite-react-mysql\torneo-sena-mysql>
```

Instalar fontawesome

Instructor Jorge E. Andrade C.
jandrade@senra.edu.co
Centro Agropecuario La Granja
Regional Tolima

<https://cdnjs.com/libraries/font-awesome>

The screenshot shows the CDNJS library page for the font-awesome package. The URL in the browser's address bar is https://cdnjs.com/libraries/font-awesome. The page displays the following information:

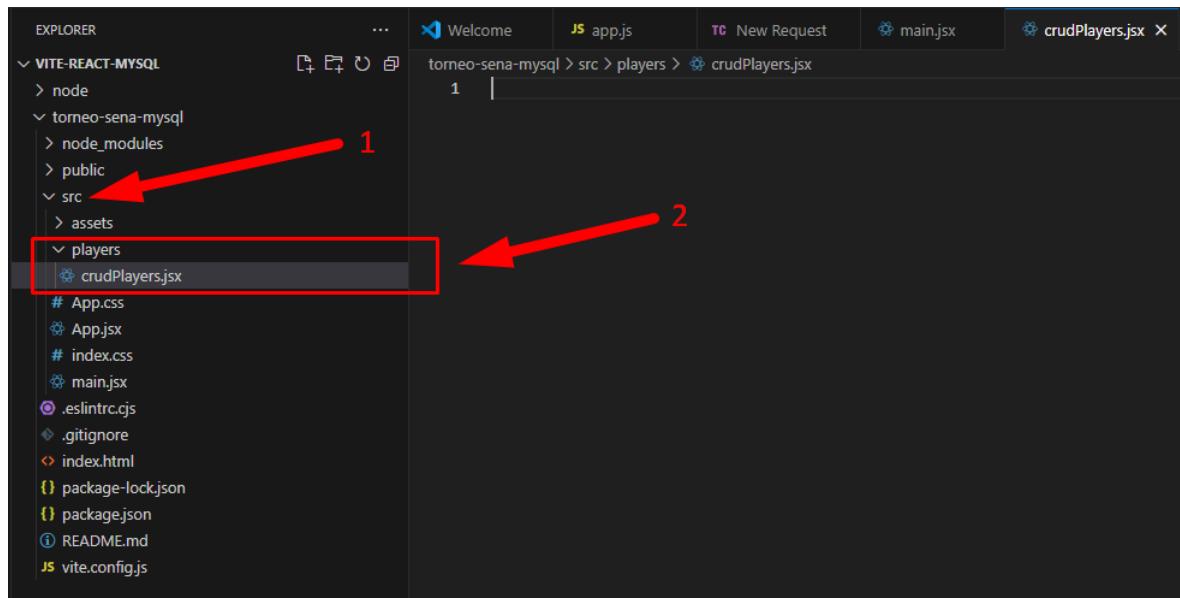
- Version: 6.5.1
- Asset Type: All
- Some files are hidden, click to show all files.
- Links to three CSS files:
 - https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css
 - https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/brands.min.css
 - https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/fontawesome.min.css

Below the browser screenshot is a code editor window titled "Welcome". The file "index.html" is open, showing the following code:

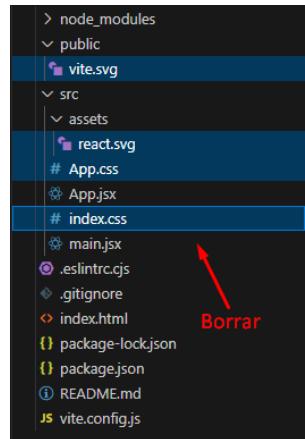
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<link rel="icon" type="image/svg+xml" href="/vite.svg" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Vite + React</title>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css" integrity="sha512-DTOQ9RWChppGcIaEAIBIZOG6xalWsw9C2Q0eAftl1VegovInee1c9QX4TctnWm13TzYe+giMm8e2lwA==" crossorigin="anonymous" referrerPolicy="no-referrer" />
</head>
<body>
<div id="root"></div>
<script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

A red arrow points from the "gitignore" file in the Explorer sidebar to the "index.html" file in the code editor. Another red arrow points from the "index.html" file in the code editor to the line of code that includes the Font Awesome CSS link.

En la carpeta SRC crear una carpeta para el módulo de players y un archivo llamado crudPlayers.jsx para el componente principal:



Borrar archivos que vienen predeterminados y que no se utilizarán:



```

1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5
6 function App() {
7   const [count, setCount] = useState(0)
8
9   return (
10     <>
11       <div>
12         <a href="https://vitejs.dev" target="_blank">
13           <img src={viteLogo} className="logo" alt="Vite logo" />
14         </a>
15         <a href="https://react.dev" target="_blank">
16           <img src={reactLogo} className="logo react" alt="React logo" />
17         </a>
18       </div>
19       <h1>Vite + React</h1>
20       <div className="card">
21         <button onClick={() => setCount((count) => count + 1)}>
22           count is {count}
23         </button>
24         <p>
25           Edit <code>src/App.jsx</code> and save to test HMR
26         </p>
27       </div>
28       <p className="read-the-docs">
29         Click on the Vite and React logos to learn more
30       </p>
31     </>
32   )
33 }
34
35 export default App
36

```

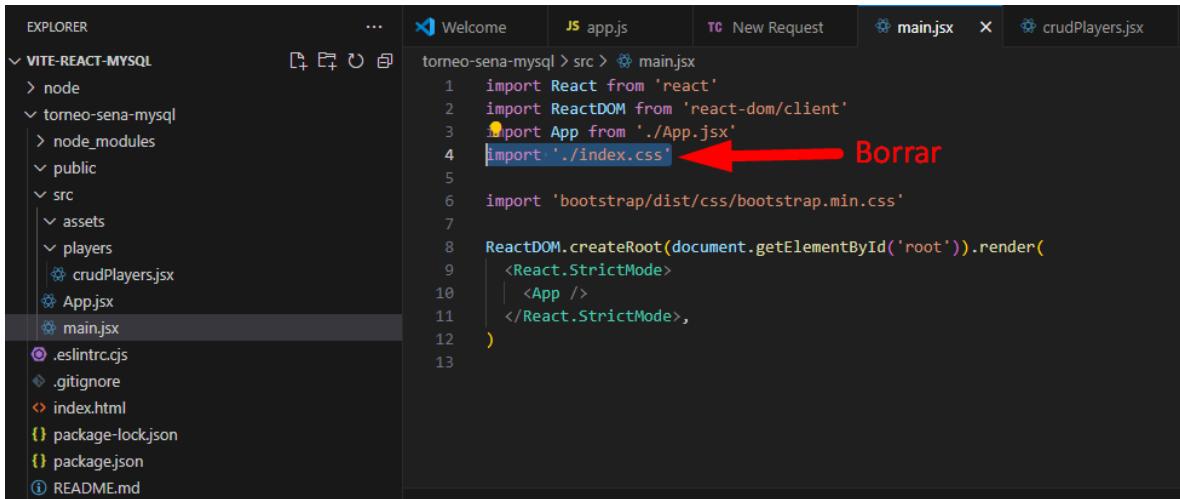
Debe quedar así:

```

1 import { useState } from 'react'
2
3
4 function App() {
5
6   return (
7     <>
8
9     </>
10   )
11 }
12
13
14 export default App
15

```

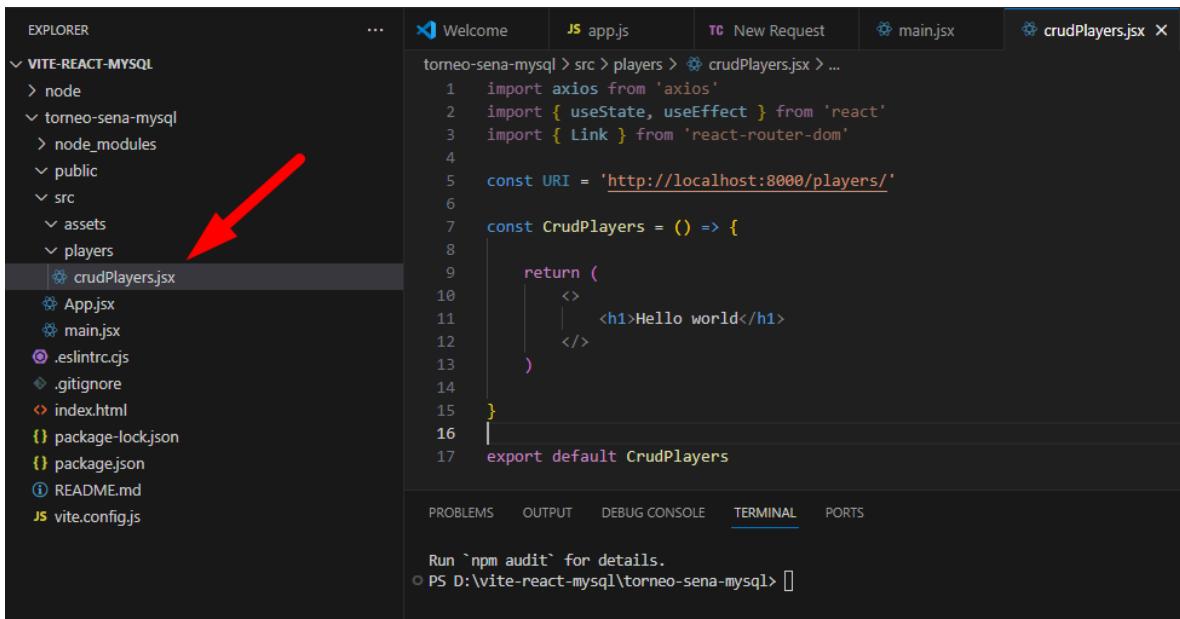
En el archivo main.jsx borrar la siguiente línea:



Code editor showing the `main.js` file:

```
torneo-sena-mysql > src > main.jsx
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4 import './index.css' Borrar
5
6 import 'bootstrap/dist/css/bootstrap.min.css'
7
8 ReactDOM.createRoot(document.getElementById('root')).render(
9   <React.StrictMode>
10   | <App />
11   </React.StrictMode>,
12 )
13
```

Código inicial en el componente crudPlayers



Code editor showing the `crudPlayers.jsx` file:

```
torneo-sena-mysql > src > players > crudPlayers.jsx ...
1 import axios from 'axios'
2 import { useState, useEffect } from 'react'
3 import { Link } from 'react-router-dom'
4
5 const URI = 'http://localhost:8000/players/'
6
7 const CrudPlayers = () => {
8
9   return (
10     <>
11       <h1>Hello world</h1>
12     </>
13   )
14 }
15
16 export default CrudPlayers
```

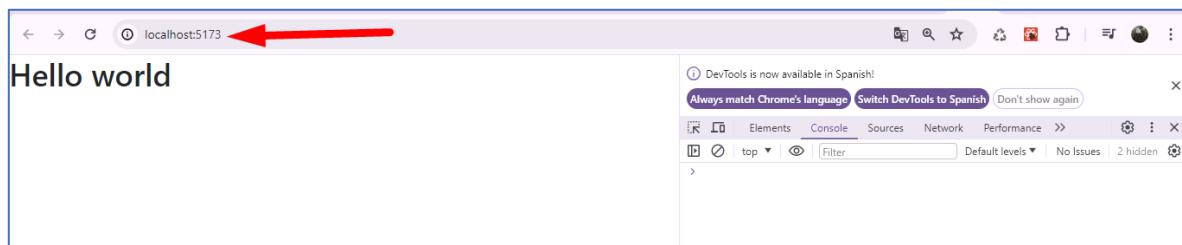
Agregar componente en el archivo main.jsx:

Ejecutar con el comando `npm run dev` y observar en el navegador:

VS Code Screenshot:

- EXPLORER:** Shows the project structure for 'VITE-REACT-MYSQL'. Files listed include .eslintrc.js, .gitignore, index.html, package-lock.json, package.json, README.md, and vite.config.js.
- CODE EDITOR:** The main.jsx file is open. Line 12 contains the code: `<React.StrictMode>` followed by a red arrow pointing to it. The code also includes imports for React, ReactDOM, and App.
- TERMINAL:** Shows the command `PS D:\vite-react-mysql\torneo-sena-mysql> npm run dev` followed by its output: `> torneo-sena-mysql@0.0.0 dev` and `> vite`. Below this, VITE v5.1.6 is shown as ready in 321 ms, followed by three arrows pointing to the local host URL: `→ Local: http://localhost:5173/` and `→ Network: use --host to expose`.

Resultado:

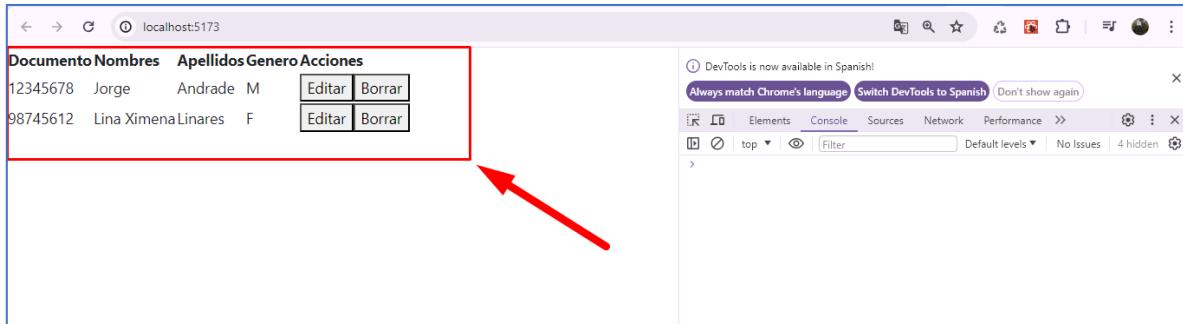


Código en crudPlayer para consultar todos los jugadores:

```
torneo-sena-mysql > src > players > crudPlayers.jsx > [o] CrudPlayers > [o] getAllPlayers
  1 import axios from 'axios'
  2 import { useState, useEffect } from 'react'
  3 import { Link } from 'react-router-dom'
  4
  5 const URI = 'http://localhost:8000/players/'
  6
  7 const CrudPlayers = () => {
  8
    // uso del Hook useState
  9    const [playerList, setPlayerList] = useState([])
 10   // inicializar la lista con un arreglo vacío
 11
 12   // Hook useEffect: es un hook que recibe como primer parámetro una función que se ejecutará cada vez que nuestro componente se renderice,
 13   // ya sea por un cambio de estado, por recibir props nuevas o, y esto es importante, porque es la primera vez que se MONTA.
 14   useEffect(() => {
 15     getAllPlayers() // Ejecuta la función getAllPlayers
 16   }, [])
 17
 18
 19   const getAllPlayers = async () => [
 20     const respuesta = await axios.get(URI)
 21     // A través de AXIOS se logra la conexión con el backend
 22     // Recuerde que la variable URI tiene la dirección del servidor NODE que creamos anteriormente
 23     // A través de esta dirección se realizan las peticiones CRUD
 24     setPlayerList(respuesta.data)
 25
 26   ]
 27
 28
 29   return (
 30     <>
 31       <table>
 32         <thead>
 33           <tr>
 34             <th>Documento</th>
 35             <th>Nombres</th>
 36             <th>Apellidos</th>
 37             <th>Genero</th>
 38             <th>Acciones</th>
 39           </tr>
 40         </thead>
 41         <tbody>
 42           /* En el contexto de React y firestore, se utiliza map en lugar de forEach porque genera un nuevo array de elementos que pueden ser renderizados en
 43           JSX. */
 44           {playerList.map((player) => (
 45             // con el paréntesis en lugar de llaves se ejecuta un return de manera explícita, o sea, no hay necesidad de digitar la palabra return
 46             <tr key={player.id}>
 47               <td>{player.documento}</td>
 48               <td>{player.nombres}</td>
 49               <td>{player.apellidos}</td>
 50               <td>{player.genero}</td>
 51               <td>
 52                 <button>Editar</button>
 53                 <button>Borrar</button>
 54               </td>
 55             </tr>
 56           ))}
 57         </tbody>
 58       </table>
 59     </>
 60   )
 61
 62
 63   export default CrudPlayers
```

```
torneo-sena-mysql > src > players > crudPlayers.jsx > [o] CrudPlayers
  7 const CrudPlayers = () => {
  27
  28
  29   return (
 30     <>
 31       <table>
 32         <thead>
 33           <tr>
 34             <th>Documento</th>
 35             <th>Nombres</th>
 36             <th>Apellidos</th>
 37             <th>Genero</th>
 38             <th>Acciones</th>
 39           </tr>
 40         </thead>
 41         <tbody>
 42           /* En el contexto de React y firestore, se utiliza map en lugar de forEach porque genera un nuevo array de elementos que pueden ser renderizados en
 43           JSX. */
 44           {playerList.map((player) => (
 45             // con el paréntesis en lugar de llaves se ejecuta un return de manera explícita, o sea, no hay necesidad de digitar la palabra return
 46             <tr key={player.id}>
 47               <td>{player.documento}</td>
 48               <td>{player.nombres}</td>
 49               <td>{player.apellidos}</td>
 50               <td>{player.genero}</td>
 51               <td>
 52                 <button>Editar</button>
 53                 <button>Borrar</button>
 54               </td>
 55             </tr>
 56           ))}
 57         </tbody>
 58       </table>
 59     </>
 60   )
 61
 62
 63   export default CrudPlayers
```

Resultado:



Componente FormPlayers (Formulario para agregar y actualizar):

Fase 1:

En el archivo crudPlayers.jsx

```

EXPLORER ... Welcome JS app.js JS routes.js main.js crudPlayers.jsx formPlayers.jsx App.jsx index.html ...
VITE-REACT-MYSQL ...
  node
  > database
  > models
  > node_modules
  > routes
    JS routes.js
  JS app.js
  (1) package-lock.json
  (1) package.json
  > torneo-sena-mysql
    > node_modules
    > public
    > src
      > assets
      > players
        crudPlayers.jsx
        formPlayers.jsx
        main.js
        .eslintrc.js
        .gitignore
        index.html
        (1) package-lock.json
        (1) package.json
        README.md
        JS vite.config.js
> OUTLINE

```

```

import axios from 'axios';
import { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import FormPlayers from './formPlayers';

const URI = 'http://localhost:8000/players/';

const CrudPlayers = () => {
  //uso del Hook useState
  const [playerList, setPlayerList] = useState([]);
  //inicializar la lista con un arreglo vacío
  //Definir el texto del botón del formulario el cual varía entre 'Enviar' y 'Actualizar'
  const [buttonForm, setButtonForm] = useState('Enviar');

  // Hook useEffect: es un hook que recibe como primer parámetro una función que se ejecutará cada vez que nuestro componente se renderice,
  // ya sea por un cambio de estado, por recibir props nuevas o, y esto es importante, porque es la primera vez que se MONTA.
  useEffect(() => {
    getAllPlayers() //Ejecuta la función getAllPlayers
  }, [ ]);

  const getAllPlayers = async () => {
    const respuesta = await axios.get(URI)
    //A través de AXIOS se logra la conexión con el backend
    //Recuerde que la variable URI tiene la dirección del servidor NODE que creamos anteriormente
    //A través de esta dirección se realizan las peticiones CRUD
    setPlayerList(respuesta.data)
  }
}

```

```

const CrudPlayers = () => {
  const playerList = [
    {documento: '1234567890', nombres: 'Juan', apellido: 'Pérez', genero: 'Masculino'},
    {documento: '9876543210', nombres: 'Ana', apellido: 'Gómez', genero: 'Femenino'}
  ];
  return (
    <table border="1">
      <thead>
        <tr>
          <th>Documento</th>
          <th>Nombre(s)</th>
          <th>Apellido(s)</th>
          <th>Genero</th>
        </tr>
      </thead>
      <tbody>
        {playerList.map((player) => (
          <tr>
            <td>{player.documento}</td>
            <td>{player.nombres}</td>
            <td>{player.apellido}</td>
            <td>{player.genero}</td>
            <td>
              <button>Editar</button>
              <button>Borrar</button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
    <br />
    {/* Incluir el componente FormPlayers y enviarle el botón como "parámetro" */}
    <FormPlayers buttonForm={buttonForm} />
  );
}

export default CrudPlayers;

```

Definir el componente FormPlayers (archivo formPlayers.jsx)

```

const FormPlayers = ({ buttonForm }) => { // Agregar como parámetro el botón que llega desde el componente crudPlayers
  const [documento, setDocumento] = useState('')
  const [nombres, setNombres] = useState('')
  const [apellidos, setApellidos] = useState('')
  const [genero, setGenero] = useState('')

  // Función que recibe los datos del formulario
  const sendForm = (e) => {
    e.preventDefault()

    if (buttonForm === 'Actualizar') {
      console.log('actualizando ando....')
      // Aquí va el código para actualizar
    } else if (buttonForm === 'Enviar') {
      console.log('guardando ando....')
      // Aquí va el código para guardar
    }
  }
}

return (
  <>
    <form id="playerForm" action="" onSubmit={sendForm}>
      <label htmlFor="documento">Documento</label>
      {/* Cuando se trabaja con formularios, los campos deben contar siempre con el evento onChange, el cual, ejecuta la función de gestión de dicho campo */}
      <input type="text" id="documento" value={documento} onChange={(e) => setDocumento(e.target.value)} />
      <br />
      <label htmlFor="nombres">Nombres</label>
      <input type="text" id="nombres" value={nombres} onChange={(e) => setNombres(e.target.value)} />
      <br />
      <label htmlFor="apellidos">Apellidos</label>
      <input type="text" id="apellidos" value={apellidos} onChange={(e) => setApellidos(e.target.value)} />
      <br />
      <label htmlFor="genero">Genero</label>
      <select name="" id="genero" value={genero} onChange={(e) => setGenero(e.target.value)}>
        <option value="">Selecciona uno...</option>
        <option value="F">Femenino</option>
        <option value="M">Masculino</option>
        <option value="O">Otro</option>
      </select>
      <br />
      <input type="submit" id="boton" value={buttonForm} className="btn btn-success" />
    </form>
  </>
)
}

export default FormPlayers;

```

Recuerde que la propiedad class de las etiquetas se reemplaza por className porque la palabra 'class' es una instrucción reservada de JavaScript para la creación de clases.

Documento	Nombres	Apellidos	Género	Acciones
12345678	Jorge	Andrade	M	<button>Editar</button> <button>Borrar</button>
98745612	Lina Ximena	Linares	F	<button>Editar</button> <button>Borrar</button>

Formulario para editar un jugador:

Documento:

Nombres:

Apellidos:

Género: Selecciona uno...

Enviar

Fase 2:

Programar el botón de editar:

En el archivo crudPlayers.jsx

```

import React, { useState } from 'react';
import axios from 'axios';
import FormPlayers from './FormPlayers';

const URI = 'http://localhost:8000/players/';

const CrudPlayers = () => {
  const [playerList, setPlayerList] = useState([]);
  //Inicializar la lista con un arreglo vacío

  //Definir el texto del botón del formulario el cual varía entre 'Enviar' y 'Actualizar'
  const [buttonForm, setButtonForm] = useState('Enviar');

  // Definir prop para el registro de un player
  const [player, setPlayer] = useState({
    id: '',
    documento: '',
    nombres: '',
    apellidos: '',
    genero: ''
  });

  // Hook useEffect: es un hook que recibe como primer parámetro una función que se ejecutará cada vez que nuestro componente se renderice,
  // ya sea por un cambio de estado, por recibir props nuevas o, y esto es importante, porque es la primera vez que se MONTA.
  useEffect(() => {
    getAllPlayers() //Ejecuta la función getAllPlayers
  }, []);

  const getAllPlayers = async () => {
    const respuesta = await axios.get(URI);
    //A través de AXIOS se logra la conexión con el backend
    //Recuerde que la variable URI tiene la dirección del servidor NODE que creamos anteriormente
    setPlayerList(respuesta.data)
  }
}

export default CrudPlayers;

```

```

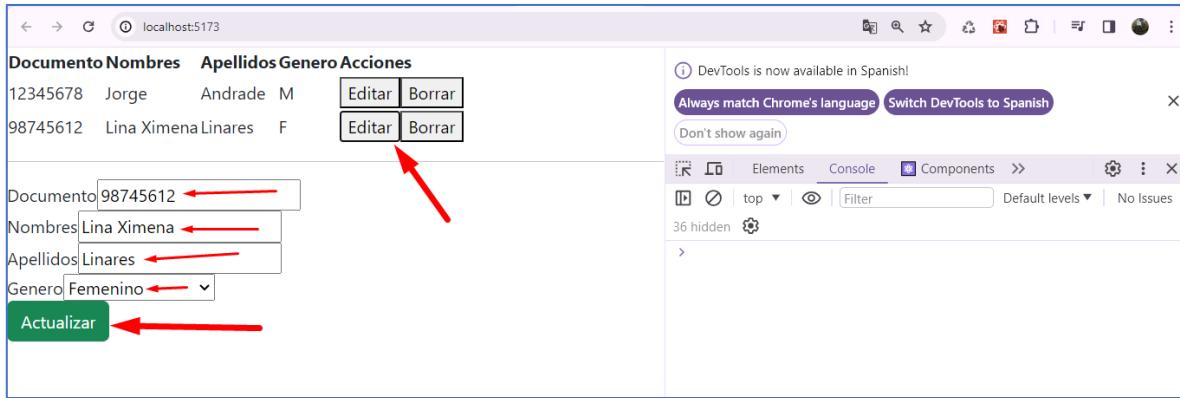
    crudPlayers.jsx
    41 const getPlayer = async (idPlayer) => { //Función para consultar un player
    42   setButtonForm('Actualizar') // Cambiar el texto del botón
    43
    44   const respuesta = await axios.get(URI + idPlayer) // concatenar a la ruta principal el / y el id
    45
    46   setPlayer({
    47     ...respuesta.data // llenar el objeto player con los datos consultados
    48   })
    49
    50 }
    51
    52
    53   return (
    54     <>
    55       <table>
    56         <thead>
    57           <tr>
    58             <th>Documento</th>
    59             <th>Nombres</th>
    60             <th>Apellidos</th>
    61             <th>Genero</th>
    62             <th>Acciones</th>
    63           </tr>
    64         </thead>
    65         <tbody>
    66           // En el contexto de React y firestore, se utiliza map en lugar de forEach porque genera un nuevo array de elementos que pueden ser renderizados
    67           {playerList.map((player) => (
    68             <tr key={player.id}>
    69               <td>{player.documento}</td>
    70               <td>{player.nombres}</td>
    71               <td>{player.apellidos}</td>
    72               <td>{player.genero}</td>
    73               <td>
    74                 <button onClick={() => getPlayer(player.id)}>Editar</button>
    75                 <button>Borrar</button>
    76               </td>
    77             </tr>
    78           ))}
    79         </tbody>
    80       </table>
    81       <hr />
    82       /* Incluir el componente FormPlayers y enviarle el botón como "parámetro" */
    83       <FormPlayers buttonForm={buttonForm} player={player}>
    84     </FormPlayers>
    85   </>
    86 )
    87
    88
    89
    90 export default CrudPlayers
  
```

En el archivo formPlayers.jsx

```

    formPlayers.jsx
    4 const FormPlayers = ({ buttonForm, player }) => { // Agregar como parámetro el botón que llega desde el componente crudPlayers
    5
    6   // Hooks para cada uno de los campos del formulario
    7   const [documento, setDocumento] = useState('')
    8   const [nombres, setNombres] = useState('')
    9   const [apellidos, setApellidos] = useState('')
   10   const [genero, setGenero] = useState('')
   11
   12   // Función que recibe los datos del formulario
   13   const sendForm = (e) => {
   14
   15     e.preventDefault()
   16
   17     if (buttonForm == 'Actualizar') {
   18       console.log('actualizando ando...')
   19       // Aquí va el código para actualizar
   20     } else if (buttonForm == 'Enviar') {
   21       console.log('guardando ando...')
   22       // Aquí va el código para guardar
   23     }
   24
   25   }
   26
   27   const setData = () => { // Función que establece los valores a los campos para que se muestren en el formulario cuando presionen el botón editar, los datos
   28     // llegan del objeto 'player'
   29     setDocumento(player.documento)
   30     setNombres(player.nombres)
   31     setApellidos(player.apellidos)
   32     setGenero(player.genero)
   33
   34   useEffect(() => { // useEffect escucha los cambios en el objeto 'player' y se ejecuta la función 'setData'
   35     setData()
   36   }, [player])
   37
   38   return (
   39     <>
   40       <form id="playerForm" action="" onSubmit={sendForm}>
   41         <label htmlFor="documento">Documento</label>
  
```

Al dar clic en el botón de editar, se deben mostrar los datos en el formulario:



Fase 3:

Actualizar registro

Cambios en el archivo crudPlayers.jsx

```

const CrudPlayers = () => {
  // Hook useEffect: es un hook que recibe como primer parámetro una función que se ejecutará cada vez que nuestro componente se renderice,
  // ya sea por un cambio de estado, por recibir props nuevas o, y esto es importante, porque es la primera vez que se MONTA.
  useEffect(() => {
    getAllPlayers() //Ejecuta la función getAllPlayers
  }, [playerList]) //Cada vez que haya un cambio en la lista de players se ejecuta la función getAllPlayers() ←
  // o sea, cuando se crea, se actualiza o elimina un registro

  const getAllPlayers = async () => {
    const respuesta = await axios.get(URI)
    //A través de AXIOS se logra la conexión con el backend
    //Recuerde que la variable URI tiene la dirección del servidor NODE que creamos anteriormente
    //A través de esta dirección se realizan las peticiones CRUD
    setPlayerList(respuesta.data)
  }

  const getPlayer = async (idPlayer) => { //Función para consultar un player
    setButtonForm('Actualizar') // Cambiar el texto del botón

    const respuesta = await axios.get(URI + idPlayer) // concatenar a la ruta principal el / y el id

    setPlayer({
      ...respuesta.data // Llenar el objeto player con los datos consultados
    })
  }

  const updateTextButton = (texto) => { // Función que actualiza el texto del botón del formulario
    setButtonForm(texto)
  }
}

```

```
const CrudPlayers = () => {
  <thead>
    <tr>
      <th>Acciones</th>
    </tr>
  </thead>
  <tbody>
    {/* En el contexto de React y firestore, se utiliza map en lugar de forEach porque genera un nuevo array de elementos que pueden ser renderizados en JSX. */}
    {playerList.map((player) => (
      <tr key={player.id}>
        <td>{player.documento}</td>
        <td>{player.nombres}</td>
        <td>{player.apellidos}</td>
        <td>{player.genero}</td>
        <td>
          <button onClick={() => getPlayer(player.id)}>Editar</button>
          <button>Borrar</button>
        </td>
      </tr>
    )));
  </tbody>
  <hr />
  {/* Incluir el componente FormPlayers y enviarle el botón como "parámetro" */}
  <FormPlayers buttonForm={buttonForm} player={player} URI={URI} updateTextButton={updateTextButton} />
</>
}
export default CrudPlayers
```

Cambios en el archivo formPlayers.jsx

- Importar axios

```

    import axios from "axios";
    import { useState } from "react";
    import { useEffect } from "react";

    const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
        // Hooks para cada uno de los campos del formulario
        const [documento, setDocumento] = useState("");
        const [nombres, setNombres] = useState("");
        const [apellidos, setApellidos] = useState("");
        const [genero, setGenero] = useState("");

        // Función que recibe los datos del formulario
        const sendForm = (e) => {
            e.preventDefault();

            if (buttonForm === 'Actualizar') {
                console.log('actualizando ando...')
                // Aquí va el código para actualizar
                axios.put(URI + player.id, {
                    documento: documento,
                    nombres: nombres,
                    apellidos: apellidos,
                    genero: genero
                })
                updateTextButton('Enviar') // Cambiar el texto del botón
                clearForm() // Limpiar el formulario
            } else if (buttonForm === 'Enviar') {
                console.log('guardando ando...')
                // Aquí va el código para guardar
            }
        }
    }

    const clearForm = () => {
        // Se vacía el elemento player
        setDocumento('')
        setNombres('')
        setApellidos('')
        setGenero('')
    }

    const setData = () => {
        // Función que establece los valores a los campos para que se muestren en el formulario cuando presionen el botón editar, los datos
        // llegan del objeto 'player'
        setDocumento(player.documento)
        setNombres(player.nombres)
        setApellidos(player.apellidos)
        setGenero(player.genero)
    }

    useEffect(() => {
        // useEffect escucha los cambios en el objeto 'player' y se ejecuta la función 'setData'
        setData()
    }, [player])
}

return (
    <>
)

```

Resultado:

Documento	Nombres	Apellidos	Genero	Acciones
12345678	Jorge	Andrade Cruz M		Editar Borrar
98745612	Lina Ximena Linares	F		Editar Borrar

localhost:5173

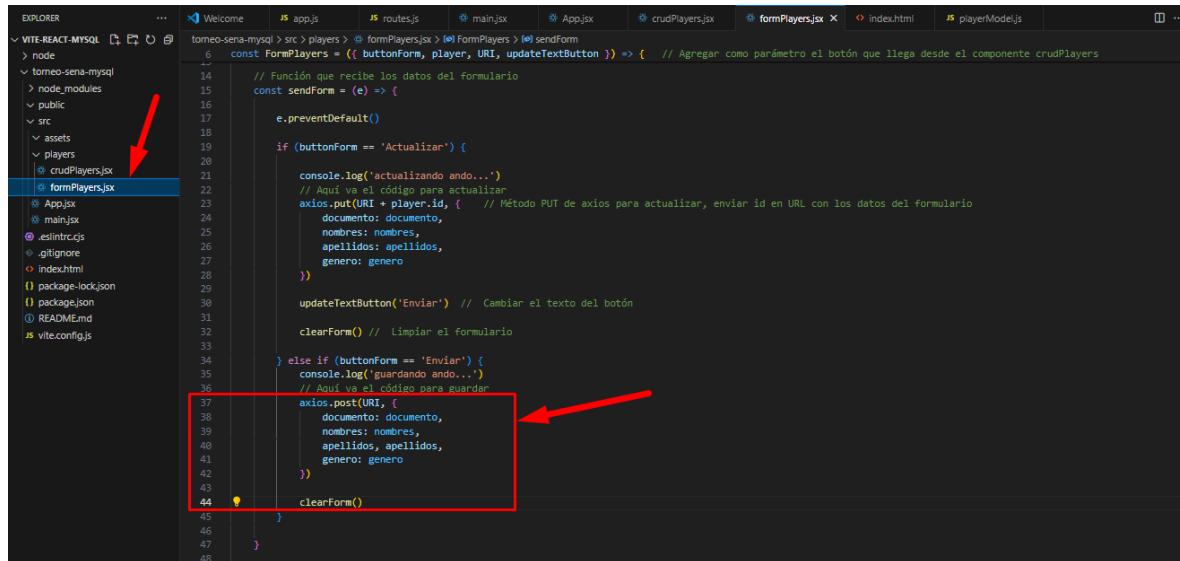
Documento	Nombres	Apellidos	Genero	Acciones
12345678	Jorge Eliecer Andrade Cruz	M		Editar Borrar
98745612	Lina Ximena Linares	F		Editar Borrar

Documento	<input type="text"/>
Nombres	<input type="text"/>
Apellidos	<input type="text"/>
Genero	<input type="text"/> Selecciona uno... ▾
Enviar	

Fase 4:

Guardar un registro

Cambios en archivo formPlayers.jsx



```

EXPLORER          Welcome      app.js      routes.js      main.jsx      App.jsx      crudPlayers.jsx      formPlayers.jsx      index.html      playerModel.js
VITE-REACT-MYSQL  node_modules
> node
< torneo-sena-mysql
  > src
    > players
      crudPlayers.jsx
      formPlayers.jsx
      App.jsx
      main.jsx
      .eslintrc.js
      .gitignore
      index.html
      package-lock.json
      package.json
      README.md
      vite.config.js

```

```

const FormPlayers = ((buttonForm, player, URI, updateTextButton)) => {
  // Función que recibe los datos del formulario
  const sendForm = (e) => {
    e.preventDefault()

    if (buttonForm === 'Actualizar') {
      console.log('actualizando ando...')
      // Aquí va el código para actualizar
      axios.put(URI + player.id, {
        documento: documento,
        nombres: nombres,
        apellidos: apellidos,
        genero: genero
      })
      updateTextButton('Enviar') // Cambiar el texto del botón
      clearForm() // Limpiar el formulario
    } else if (buttonForm === 'Enviar') {
      console.log('guardando ando...')
      // Aquí va el código para guardar
      axios.post(URI, {
        documento: documento,
        nombres: nombres,
        apellidos: apellidos,
        genero: genero
      })
      clearForm()
    }
  }
}

```

localhost:5173

Documento	Nombres	Apellidos	Genero	Acciones
12345678	Jorge	Andrade Cruz M		Editar Borrar
98745612	Lina Ximena Linares	F		Editar Borrar

Documento: 789321
 Nombres: Rubi
 Apellidos: Rosa
 Genero: Femenino

[Enviar](#)

DevTools is now available in Spanish!
[Always match Chrome's language](#) [Switch DevTools to Spanish](#)

Elements Console Sources Network Filter Default levels No Issues

localhost:5173

Documento	Nombres	Apellidos	Genero	Acciones
12345678	Jorge	Andrade Cruz M		Editar Borrar
98745612	Lina Ximena Linares	F		Editar Borrar
789321	Rubi	Rosa	F	Editar Borrar

Documento:
 Nombres:
 Apellidos:
 Genero: Selecciona uno...

[Enviar](#)

guardando ando...

DevTools is now available in Spanish!
[Always match Chrome's language](#) [Switch DevTools to Spanish](#)

Elements Console Sources Network Filter Default levels No Issues

localhost/phpmyadmin/index.php?route=/sql&db=torneo_react&table=players&pos=0

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: torneo_react » Tabla: players

	id	documento	nombres	apellidos	genero	createdat	updatedat
<input type="checkbox"/>	1	12345678	Jorge	Andrade Cruz	M	2024-03-19 10:46:21	2024-04-09 08:39:29
<input type="checkbox"/>	2	98745612	Lina Ximena	Linares	F	2024-03-19 11:06:58	2024-04-09 08:39:19
<input type="checkbox"/>	4	789321	Rubi	Rosa	F	2024-04-09 08:53:32	2024-04-09 08:53:32

Nueva
 #mysql50#campeonato_sena_da
 campeonato_sena
 h2o-lab
 information_schema
 milkcow-lps
 mysql
 performance_schema
 phpmyadmin
 sicgelam
 test
 tienda_ropa
 torneo_react
 Nueva
 players
 usuario

Mostrando filas 0 - 2 (total de 3, La consulta tardó 0,0002 segundos.)

SELECT * FROM `players`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Sort by key: Ninguna

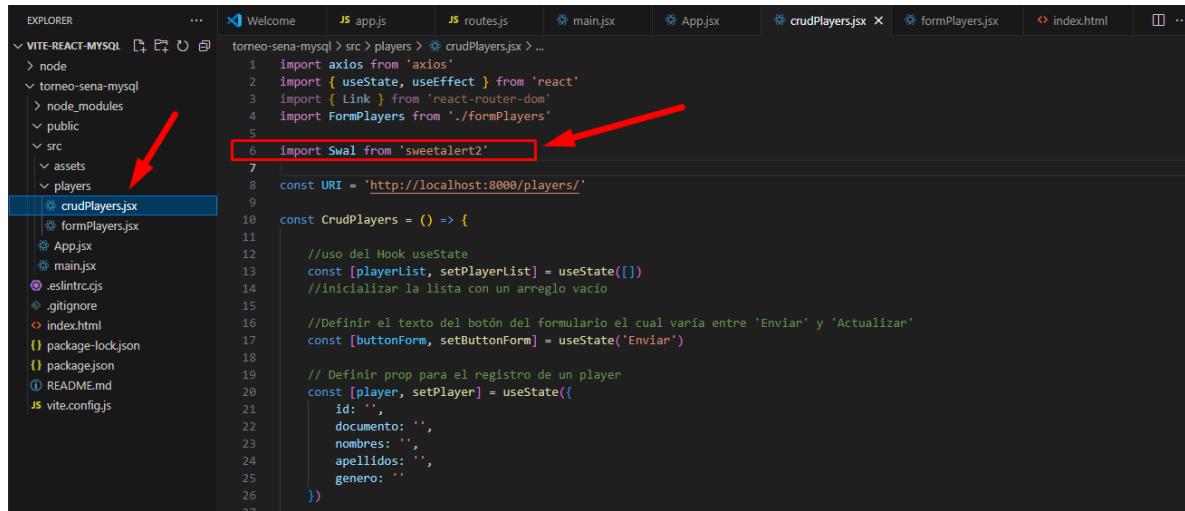
Opciones

Para los elementos que están marcados: [Editar](#) [Copiar](#) [Borrar](#) [Exportar](#)

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Sort by key: Ninguna

Fase 5:

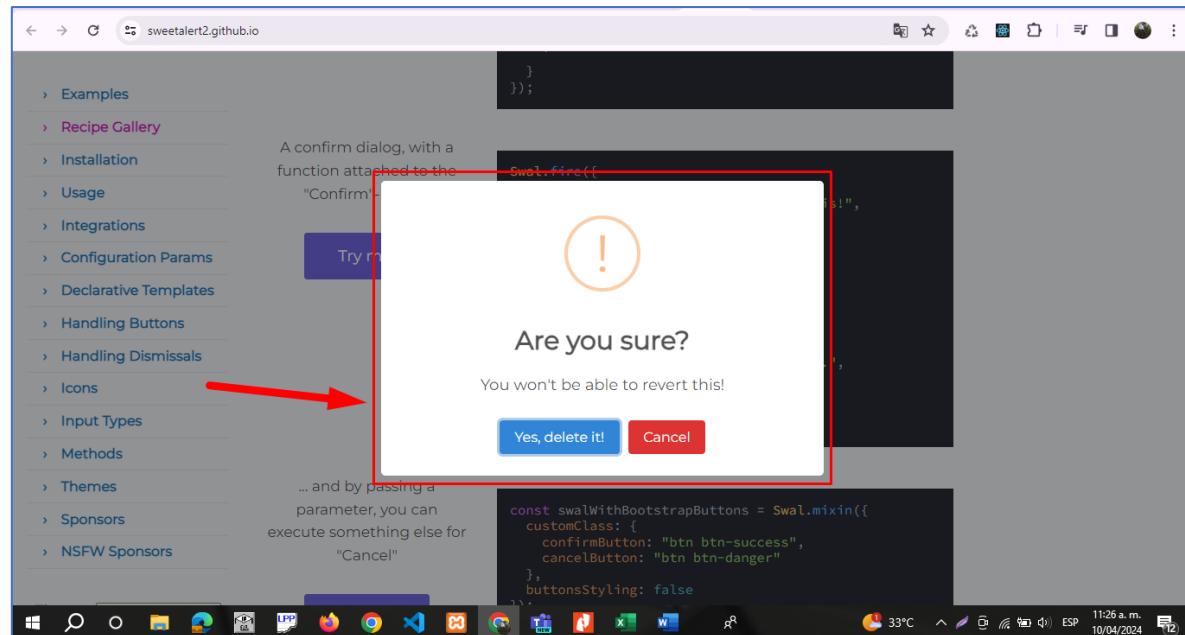
Borrar un registro

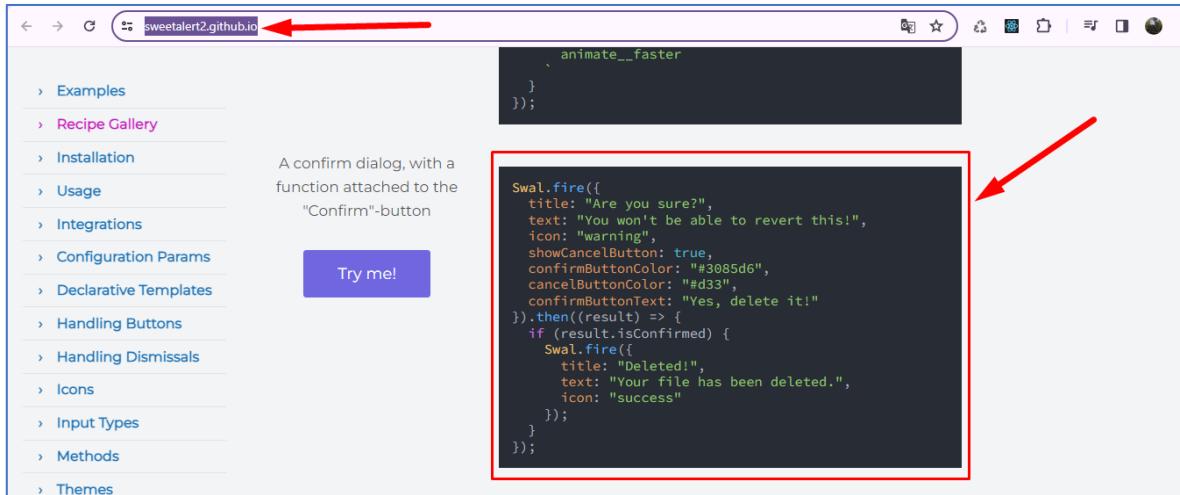


```
import axios from 'axios';
import { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import FormPlayers from './FormPlayers';
import Swal from 'weetalert2';

const URI = 'http://localhost:8000/players/'
```

En la página de sweetalert buscar el código de la alerta que se requiere para eliminar registros:





Copiar el código y ajustarlo, tenga en cuenta que en la línea 67 debe agregar el `async` y en la línea 69 se agrega la instrucción para borrar a través de axios.

The screenshot shows a code editor with the file "crudPlayers.jsx" open. The Explorer sidebar on the left shows the project structure, with "crudPlayers.jsx" selected. The code editor displays the following JavaScript code:

```

10 const CrudPlayers = () => {
11
12   const updateTextButton = (texto) => { // Función que actualiza el texto del botón del formulario
13     setButtonForm(texto)
14   }
15
16   const deletePlayer = (idPlayer) => {
17     Swal.fire({
18       title: "Estás seguro?",
19       text: "No podrás revertir esto!",
20       icon: "warning",
21       showCancelButton: true,
22       confirmButtonColor: "#3085d6",
23       cancelButtonColor: "#d33",
24       confirmButtonText: "Sí, borrar!"
25     }).then(async (result) => {
26       if (result.isConfirmed) {
27         await axios.delete(URI + idPlayer)
28         Swal.fire({
29           title: "Borrado!",
30           text: "El registro ha sido borrado.",
31           icon: "success"
32         });
33     }
34   }
35
36   return (
37     <div>
38       <h2>CRUD Players</h2>
39       <table>
40         <thead>
41           <tr>
42             <th>Nombre</th>
43             <th>Apellido</th>
44             <th>Acciones</th>
45           </tr>
46         </thead>
47         <tbody>
48           <tr>
49             <td>Juan</td>
50             <td>Pérez</td>
51             <td><button onClick={updateTextButton('Juan Pérez')}>Actualizar</button><br/><button onClick={deletePlayer('Juan Pérez')}>Borrar</button></td>
52           </tr>
53         </tbody>
54       </table>
55     </div>
56   )
57 }

```

A red box highlights the line `}.then(async (result) => {`. A red arrow points from this box to the word `async`. Another red arrow points from the word `await` in the line `await axios.delete(URI + idPlayer)`.

```

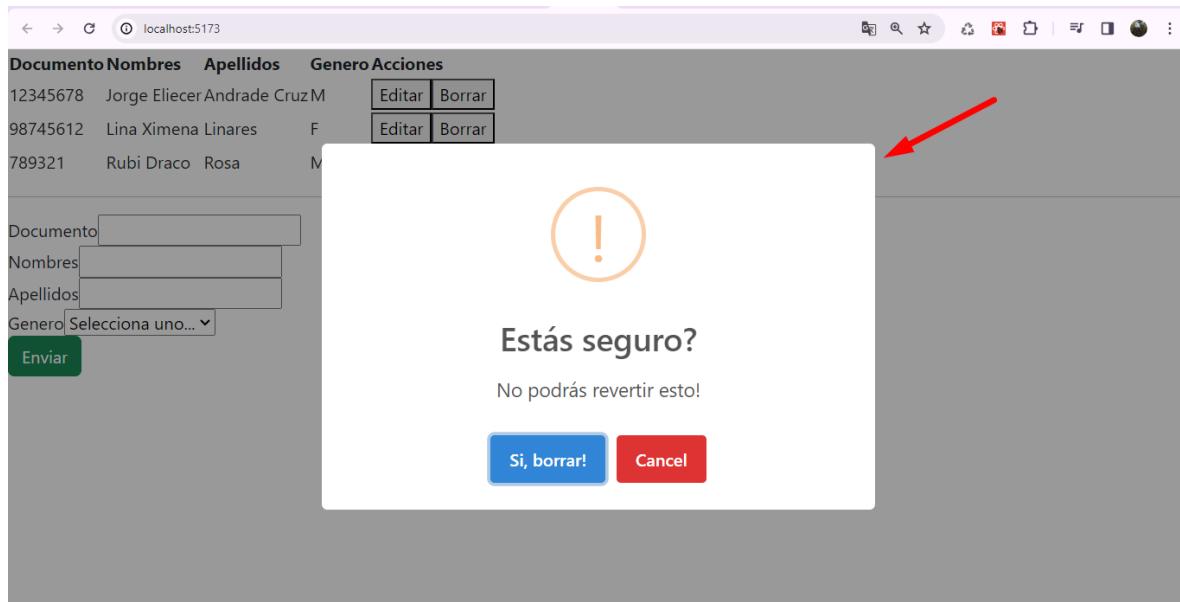
EXPLORER ... Welcome JS app.js JS routes.js main.jsx App.jsx crudPlayers.jsx formPlayers.jsx index.html ...
VITE-REACT-MYSQL node
  > node
    > torneo-sena-mysql
      > node_modules
        > public
          > src
            > assets
              > players
                crudPlayers.jsx
                formPlayers.jsx
                App.jsx
                main.jsx
                .eslintrc.js
                .gitignore
                index.html
                package-lock.json
                package.json
                README.md
                vite.config.js
  > OUTLINE

```

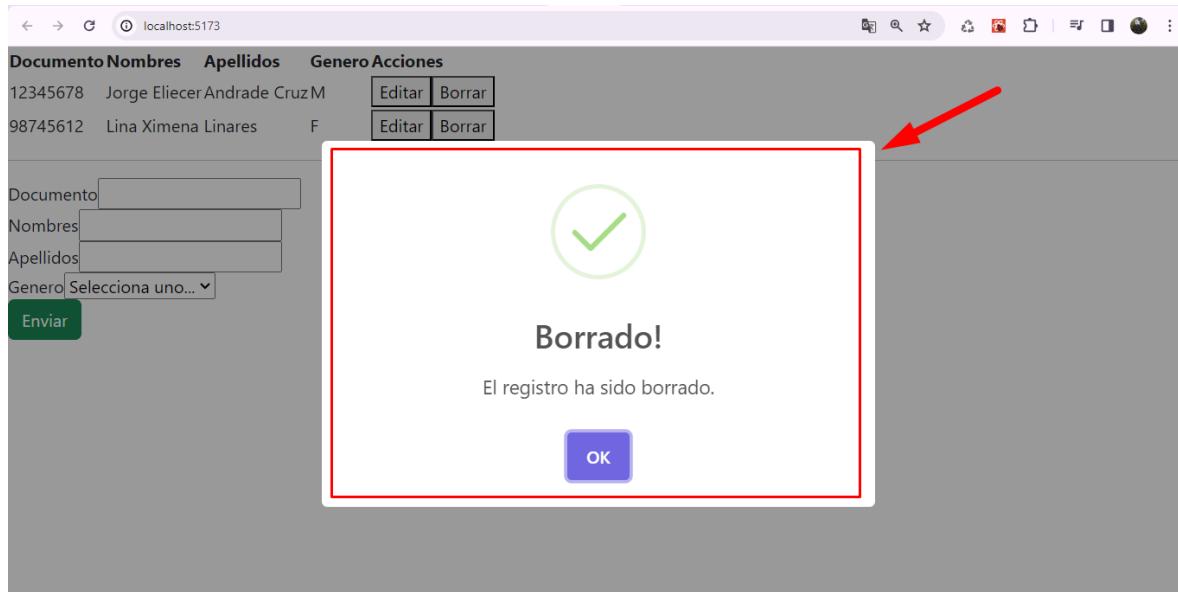
const CrudPlayers = () => {
 <table>
 <thead>
 <tr>
 <th>Documento</th>
 <th>Nombres</th>
 <th>Apellidos</th>
 <th>Genero</th>
 <th>Acciones</th>
 </tr>
 </thead>
 <tbody>
 /* En el contexto de React y firestore, se utiliza map en lugar de forEach porque genera un nuevo array de elementos */
 {playerList.map((player) => (
 <tr key={player.id}>
 <td>{player.documento}</td>
 <td>{player.nombres}</td>
 <td>{player.apellidos}</td>
 <td>{player.genero}</td>
 <td>
 <button onClick={() => getPlayer(player.id)}>Editar</button>
 <button onClick={() => deletePlayer(player.id)}>Borrar</button>
 </td>
 </tr>
))}
 </tbody>
 </table>

 /* Incluir el componente FormPlayers y enviarle el botón como "parámetro" */
 <FormPlayers buttonForm={buttonForm} player={player} URI={URI} updateTextButton={updateTextButton} />
}

Resultado:



Después de borrar:



Consultar registros por documento:

Fase 1:

En el archivo playerController.js crear la función getQueryplayer:

```

node > controllers > JS playerController.js > getAllPlayers
59 //Consultar player por documento
60 export const getQueryPlayer = async (req, res) => {
61   try {
62     const player = await PlayerModel.findAll({
63       where: {
64         documento: [
65           Sequelize.Op.like]: `%"${req.params.documento}"%` // Esta instrucción reemplaza al LIKE de una consulta en MySQL
66           // Ej: SELECT * FROM players WHERE documento LIKE '%1'
67           // Ej: SELECT * FROM players WHERE nombres LIKE '%u%'` // Esto es lo que queremos
68         }
69       }
70     })
71   }
72   res.json(player) // Obtener la respuesta
73 } catch (error) {
74   res.json({ message: error.message })
75 }
76 }
77 }
78 }
79 }
80 }

```

En el archivo routes.js crear la ruta para consultar por documento:

Ponga gran atención en la definición de la ruta como “/documento/:documento”:

```

VITE-RE...  rController.js  JS playerModel.js  crudPlayers.jsx  crudPlayers.jsx  routes.jsx  routes.jsx

node > routes > routes.js > ...
1 import express from "express";
2 import { createPlayer, deletePlayer, getAllPlayers, getPlayer, updatePlayer, getQueryPlayer } from "../controllers/playerController.js";
3 const router = express.Router();
4
5 router.get('/', getAllPlayers);
6 router.get('/:id', getPlayer);
7 router.post('/', createPlayer);
8 router.put('/:id', updatePlayer);
9 router.delete('/:id', deletePlayer);
10 //Consultar por documento de identidad
11 router.get('/documento/:documento', getQueryPlayer);
12
13 export default router;

```

Fase 2:

En el Front-end, crear el componente formQueryPlayer:

```

VITE-RE...  Welcome  JS app.js  JS playerController.js  JS playerModel.js  formQueryPlayer.jsx  crudPlayers.jsx  formPlayers.jsx  routes.jsx

torneo-sena-mysql > src > players > formQueryPlayer.jsx > FormQueryPlayer
1 import axios from "axios";
2 import { useEffect, useState } from "react";
3
4 const FormQueryPlayer = ({ URI, getPlayer, deletePlayer, buttonForm }) => {
5   //importar elementos del componente crudPlayers
6
7   const [playerQuery, setPlayerQuery] = useState([]);
8   const [documento, setDocumento] = useState("");
9
10  const sendFormQuery = async (documento) => {
11    if (documento) {
12      //Si el input está vacío no realizará la consulta a la db
13      //Incluir axios para la consulta
14      const respuesta = await axios.get(URI + 'documento/' + documento);
15      //console.log(respuesta.data)
16
17      setPlayerQuery([
18        ...playerQuery,
19        respuesta.data
20      ]);
21    } else {
22      //Si el input está vacío se establece de nuevo como vacío el arreglo playerQuery
23      setPlayerQuery([]);
24    }
25
26    useEffect(() => {
27      setPlayerQuery([]);
28      setDocumento("");
29    }, [buttonForm]);
30  };
31
32  useEffect(() => {
33    setPlayerQuery([]);
34    setDocumento("");
35  }, [buttonForm]);
36
37  return (
38    <>
39      <form action="" id="queryForm">
40        <label htmlFor="documentoQuery">Documento</label>
41        <input type="number" id="documentoQuery" value={documento} onChange={(e) => sendFormQuery(e.target.value)}>
42        </input>
43      </form>
44
45      {/**Uso del operador ternario: si hay datos en el arreglo de consulta se muestra la tabla, si no, no se muestra */
46      /* condicion ? ejecuto algo: ejecuto otra cosa */
47      playerQuery.length > 0 ? <table>
48        <thead>
49          <tr>
50            <th>Documento</th>
51            <th>Nombres</th>
52            <th>Apellidos</th>
53            <th>Genero</th>
54            <th>Acciones</th>
55          </tr>
56        </thead>
57        <tbody>
58        </tbody>
59      </table>
60    }
61  );
62}

```

```

VITE-RE...  Welcome  JS app.js  JS playerController.js  JS playerModel.js  formQueryPlayer.jsx  crudPlayers.jsx  formPlayers.jsx  routes.jsx

torneo-sena-mysql > src > players > formQueryPlayer.jsx > FormQueryPlayer
4 const FormQueryPlayer = ({ URI, getPlayer, deletePlayer, buttonForm }) => {
5   //importar elementos del componente crudPlayers
6   useEffect(() => {
7     // el useEffect se ejecuta cada vez que haya un cambio en el botón
8   }, [buttonForm]);
9
10  return (
11    <>
12      <form action="" id="queryForm">
13        <label htmlFor="documentoQuery">Documento</label>
14        <input type="number" id="documentoQuery" value={documento} onChange={(e) => sendFormQuery(e.target.value)}>
15        </input>
16      </form>
17
18      {/**Uso del operador ternario: si hay datos en el arreglo de consulta se muestra la tabla, si no, no se muestra */
19      /* condicion ? ejecuto algo: ejecuto otra cosa */
20      playerQuery.length > 0 ? <table>
21        <thead>
22          <tr>
23            <th>Documento</th>
24            <th>Nombres</th>
25            <th>Apellidos</th>
26            <th>Genero</th>
27            <th>Acciones</th>
28          </tr>
29        </thead>
30        <tbody>
31        </tbody>
32      </table>
33    }
34  );
35}

```

```

const FormQueryPlayer = ({ URI, getPlayer, deletePlayer, buttonForm }) => {
  // Importar elementos del componente crudPlayers
  <thead>
    <tr>
      <th>Acciones</th>
    </tr>
  </thead>
  <tbody>
    /* En el contexto de React, se utiliza map en lugar de forEach porque genera un nuevo array de elementos que pueden ser renderizados en JSX. */
    {playerQuery.map((player) => (
      <tr key={player.id}>
        <td>{player.documento}</td>
        <td>{player.nombres}</td>
        <td>{player.apellidos}</td>
        <td>{player.genero}</td>
        <td>
          <button onClick={() => getPlayer(player.id)}>Editar</button>
          <button onClick={() => deletePlayer(player.id)}>Borrar</button>
        </td>
      </tr>
    ))}
  </tbody>
</table> : ''
}

export default FormQueryPlayer

```

Fase 3:

En el archivo crudPlayers.js incluir los elementos a exportar para que los reciba el componente queryFormPlayer.jsx

```

const CrudPlayers = () => {
  playerList.map((player) => (
    <tr>
      <td>{player.genero}</td>
      <td>
        <button onClick={() => getPlayer(player.id)}>Editar</button>
        <button onClick={() => deletePlayer(player.id)}>Borrar</button>
      </td>
    </tr>
  ))
}

export default CrudPlayers

```

Resultado:

The screenshot shows a web browser window at `localhost:5173`. At the top, there is a search bar with placeholder text "Documento Nombres Apellidos Genero Acciones". Below it is a table with four rows of player data:

Documento	Nombres	Apellidos	Genero	Acciones
12345678	Jorge Felipe	Andrade Cruz	M	<button>Editar</button> <button>Borrar</button>
98745612	Lina Fernanda	Linares Mosquera	F	<button>Editar</button> <button>Borrar</button>
654387129	Tomas Emilio	Fernandez	M	<button>Editar</button> <button>Borrar</button>

Below the table is a search form with fields for "Documento" (containing "98745612"), "Nombres" (containing "Lina Fernanda"), "Apellidos" (containing "Linares Mosquera"), and "Genero" (containing "Femenino"). A green "Actualizar" button is present. The entire search form area is highlighted with a red box.

At the bottom of the page, another table is shown with the same structure and data as the one above, but it only contains two rows:

Documento	Nombres	Apellidos	Genero	Acciones
12345678	Jorge Felipe	Andrade Cruz	M	<button>Editar</button> <button>Borrar</button>
12345	Juan	Montaña Ruiz	M	<button>Editar</button> <button>Borrar</button>

RETO: Mejorar la programación para que se desaparezca la tabla inferior así no se haya dado clic en el botón de Actualizar.

Solución al reto:

The code editor shows the file `crudPlayers.jsx` with the following code snippet highlighted:

```

const CrudPlayers = () => {
  const getPlayer = async (idPlayer) => { //Función para consultar un player
    setButtonForm('Enviar')
    // Cambiar el nombre del botón a 'Enviar' y luego a 'actualizar', es un truco que permite mejorar la reacción del
    // componente formQueryPlayer (desaparecer la tabla del resultado de la búsqueda), esto ocurre porque se había programado
    // que se limpiara la tabla cuando el botón cambiara su valor
  }
}

```

A red box highlights the commented-out code. Two red arrows point from the left and right towards this box, indicating the specific code to be modified.

En esta parte mostrar resumen de cómo agregar un módulo al proyecto (en este caso, el módulo de centros de formación).

Implementación de React Router:

Rutas para el front-end gestionadas desde el componente App.jsx

```

EXPLORER ... main.jsx App.jsx Home.jsx centerController.js formQueryPlayer.jsx crudPlayers.jsx package.json package-lock.json
VITE-REACT-MYSQL > node > torneo-sena-mysql > src > App.jsx ...
1 import { useState } from 'react'
2
3 //Importar componentes necesarios para el sistema de rutas en REACT
4 import { Routes, Route, Link } from 'react-router-dom'
5 //importar componentes que van en las rutas
6 import Home from './home/Home'
7 import CrudPlayers from './players/crudPlayers'
8 import CrudCenters from './centers/crudCenters'
9
10 function App() {
11   return (
12     <>
13       <nav> /* Crear una barra sencilla de navegación temporalmente */
14         <ul>
15           <li>
16             <Link to="/">Inicio</Link> /* En lugar de usar etiqueta <a> se utiliza el componente <Link></Link> */
17           </li>
18           <li>
19             <Link to="/players">Players</Link>
20           </li>
21           <li>
22             <Link to="/centers">Centers</Link>
23           </li>
24         </ul>
25       </nav>
26     /*Aqui empieza la gestión de rutas */
27
28     <Routes>
29       /* en el componente Route van dos propiedades, la primera es path y la segunda es element, en path va la dirección donde queremos mostrar el componente, y en element va el componente que se quiere mostrar.*/
30       <Route path="/" element={Home} />
31       <Route path='/players' element={CrudPlayers} />
32       <Route path='/centers' element={CrudCenters} />
33     </Routes>
34   </>
35 }
36
37
38 export default App

```

Cambios en el archivo main.jsx

```

EXPLORER ... main.jsx App.jsx Home.jsx centerController.js formQueryPlayer.jsx crudPlayers.jsx package.json package-lock.json
VITE-REACT-MYSQL > node > torneo-sena-mysql > src > main.jsx ...
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4
5 import { BrowserRouter } from 'react-router-dom' //importar Componente BrowserRouter
6
7 import 'bootstrap/dist/css/bootstrap.min.css'
8 // import CrudPlayers from './players/crudPlayers' comentar o borrar la importación del componente que se estaba usando
9
10 ReactDOM.createRoot(document.getElementById('root')).render(
11   <React.StrictMode>
12     <BrowserRouter>
13       /* Envolver el punto de entrada de la aplicación */
14       <App />
15     </BrowserRouter>
16     /* <CrudPlayers /> comentar o borrar el componente que se estaba mostrando*/
17   </React.StrictMode>
18 )
19
20

```

Como se observa, se suspende la importación del componente CrudPlayers desde el archivo main.jsx porque se empiezan a gestionar las rutas desde el archivo App.jsx

Crear una carpeta para el componente Home.jsx, el cual hará de página inicial del sitio web.

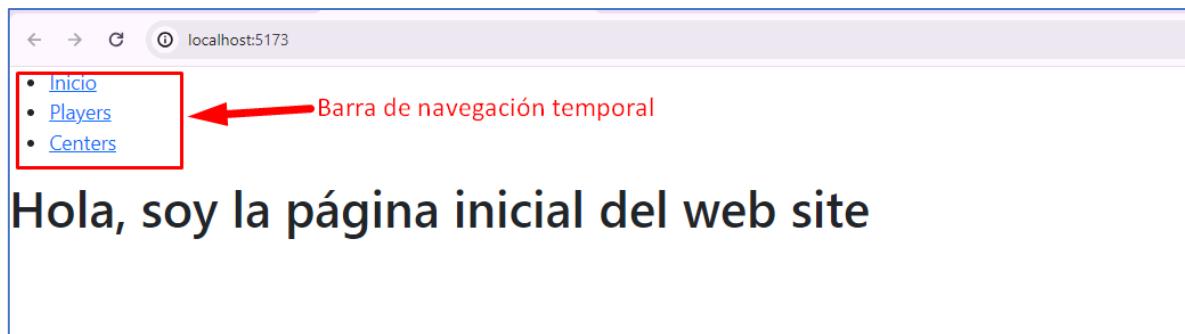
The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure under 'VITE-REACT-MYSQL'. A red arrow points from the sidebar to the 'Home.jsx' file in the center editor area. The 'Home.jsx' file contains the following code:

```

1 const Home = () => {
2
3     return (
4         <>
5             |   <h1>Hola, soy la página inicial del web site</h1>
6         </>
7     )
8 }
9
10
11
12 export default Home

```

Así se ve en el navegador:



Componente REACT para Centros de Formación (Listar y Crear)

Objetivo: exponer el uso de consultas JOIN (SQL) a través del ORM Sequelize para obtener registros que tienen relación con otras tablas, en este caso, la tabla centros de formación tiene como llaves foráneas el identificador del departamento y el del municipio.

Fase 1:

Se inicia el CRUD con la consulta de todos los registros de la tabla de centros de formación.

En el back-end (carpeta node): Definir archivos para el modelo, controlador y rutas para centros de formación:

```

node > models > JS centerModel.js > [?] default
1 import db from '../database/db.js'
2 import { DataTypes } from 'sequelize';
3
4 const CenterModel = db.define('centros_formacion', {
5   codigo_centro: { type: DataTypes.NUMBER },
6   idDepto: { type: DataTypes.NUMBER },
7   id_municipio: { type: DataTypes.NUMBER },
8   nombre_centro: { type: DataTypes.CHAR }
9 }, {
10   //evitar pluralización en la gestión de la tabla
11   freezeTableName: true
12 });
13
14
15 export default CenterModel

```

Fíjese que el archivo inicial de rutas para el módulo Players se llamaba 'routes.js', sin embargo, como se empiezan a integrar los demás módulos al sistema y cada uno tiene su archivo de rutas, es conveniente modificar su nombre a 'routerPlayers.js'.

Tener en cuenta que la estructura de la tabla de la base de datos es la siguiente:

#		Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	id	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
<input type="checkbox"/>	2	codigo_centro	int(11)			No	Ninguna		Cambiar Eliminar Más	
<input type="checkbox"/>	3	idDepto	int(11)			No	Ninguna		Cambiar Eliminar Más	
<input type="checkbox"/>	4	id_municipio	int(11)			No	Ninguna		Cambiar Eliminar Más	
<input type="checkbox"/>	5	nombre_centro	varchar(120)	utf8_general_ci		No	Ninguna		Cambiar Eliminar Más	
<input type="checkbox"/>	6	createdat	timestamp			No	current_timestamp()		Cambiar Eliminar Más	
<input type="checkbox"/>	7	updatedat	timestamp			No	current_timestamp()		Cambiar Eliminar Más	

En el controlador la consulta inicial para obtener todos los registros es la siguiente:

EXPLORER

VITE-REACT-MYSQL

- node
- controllers
 - JS centerController.js**
 - JS playerController.js
- database
- models
 - JS centerModel.js
 - JS playerModel.js
- node_modules
- routes
 - JS routerCenters.js
 - JS routerPlayers.js
- app.js
- package-lock.json
- package.json

centerController.js

```

node > controllers > JS centerController.js > getAllCenters
1 import { Sequelize } from "sequelize";
2 import CenterModel from "../models/centerModel.js";
3
4 //Mostrar todos los registros
5 export const getAllCenters = async (req, res) => {
6   try {
7     const centers = await CenterModel.findAll()
8     res.json(centers)
9   } catch (error) {
10    res.json({ message: error.message })
11  }
12}
13
14 //Mostrar un registro
15 export const getCenter = async (req, res) => {
16   try {
17     const center = await CenterModel.findAll({
18       where: { id: req.params.id }
19     })
20     res.json(center[0])
21   } catch (error) {
22     res.json({ message: error.message })
23   }
24 }

```

Definir archivo routerCenters.js para gestionar las rutas del módulo de Centros de formación:

EXPLORER

VITE-REACT-MYSQL

- node
- controllers
 - JS centerController.js
 - JS playerController.js
- database
- models
 - JS centerModel.js
 - JS playerModel.js
- node_modules
- routes
 - JS routerCenters.js**
 - JS routerPlayers.js

routerCenters.js

```

node > routes > JS routerCenters.js > ...
1 import express from "express";
2 import { createCenter, deleteCenter, getAllCenters, getCenter, getQueryCenter, updateCenter } from "../controllers/centerController.js";
3 const router = express.Router();
4
5 router.get('/', getAllCenters)
6 router.get('/:id', getCenter)
7 router.post('/', createCenter)
8 router.put('/:id', updateCenter)
9 router.delete('/:id', deleteCenter)
10
11 router.get('/nombre_centro/:nombre_centro', getQueryCenter)
12
13 export default router

```

Importar y usar las rutas back-end en el archivo app.js:

EXPLORER

VITE-REACT-MYSQL

- node
- controllers
 - JS centerController.js
 - JS playerController.js
- database
- models
 - JS centerModel.js
 - JS playerModel.js
- node_modules
- routes
 - JS routerCenters.js
 - JS routerPlayers.js
- app.js
- package-lock.json
- package.json

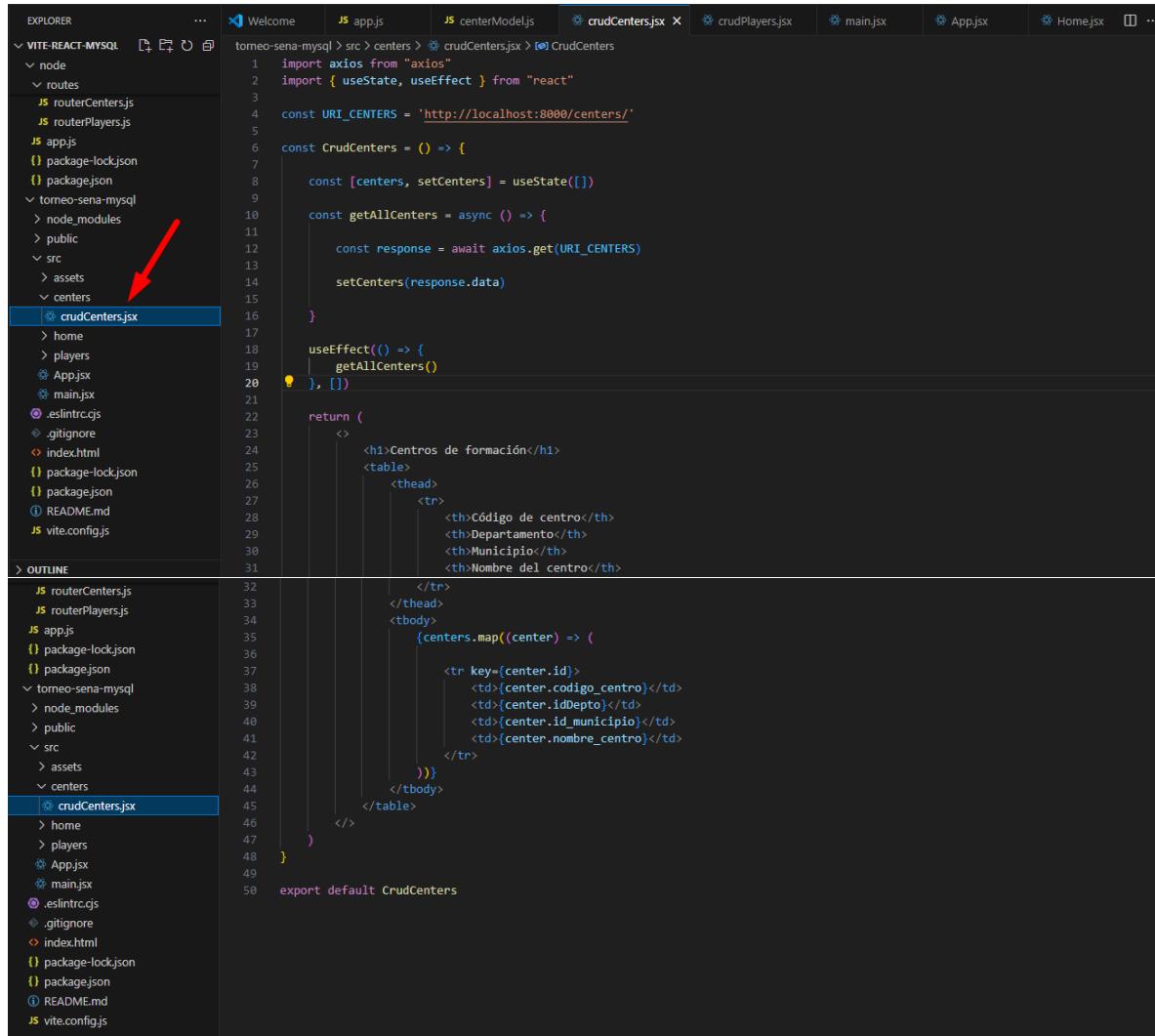
app.js

```

node > app.js > ...
1 import express from 'express';
2 import cors from 'cors';
3
4 import db from './database/db.js'
5 import playerRoutes from './routes/routerPlayers.js'
6 import centerRoutes from './routes/routerCenters.js' ←
7
8 const app = express()
9
10 app.use(cors())
11 app.use(express.json())
12 app.use('/players', playerRoutes)
13 app.use('/centers', centerRoutes) ←
14
15 try {
16   await db.authenticate()
17   console.log("Conexión exitosa a la db")
18 } catch (error) {
19   console.log("Error de conexión a la db: ${error}")
20 }
21
22
23
24 app.get('/', (req, res) => {
25   res.send('Hola mundo')
26 })
27
28 app.listen(8000, () => {
29   console.log('Server Up running in http://localhost:8000')
30 })
31

```

En el componente crudCenters.jsx se realiza la codificación para cargar la información en una tabla:



```

EXPLORER ... Welcome app.js centerModel.js crudCenters.jsx crudPlayers.jsx main.jsx App.jsx Home.jsx ...
VITE-REACT-MYSQL ...
  node
  routes
    routerCenters.js
    routerPlayers.js
  app.js
  package-lock.json
  package.json
  torneo-sena-mysql
    node_modules
    public
    src
      assets
      centers
        crudCenters.jsx
      home
      players
      App.jsx
      main.jsx
      .eslintrc.js
      .gitignore
      index.html
      package-lock.json
      package.json
      README.md
      vite.config.js
  OUTLINE ...
  routerCenters.js
  routerPlayers.js
  app.js
  package-lock.json
  package.json
  torneo-sena-mysql
    node_modules
    public
    src
      assets
      centers
        crudCenters.jsx
      home
      players
      App.jsx
      main.jsx
      .eslintrc.js
      .gitignore
      index.html
      package-lock.json
      package.json
      README.md
      vite.config.js

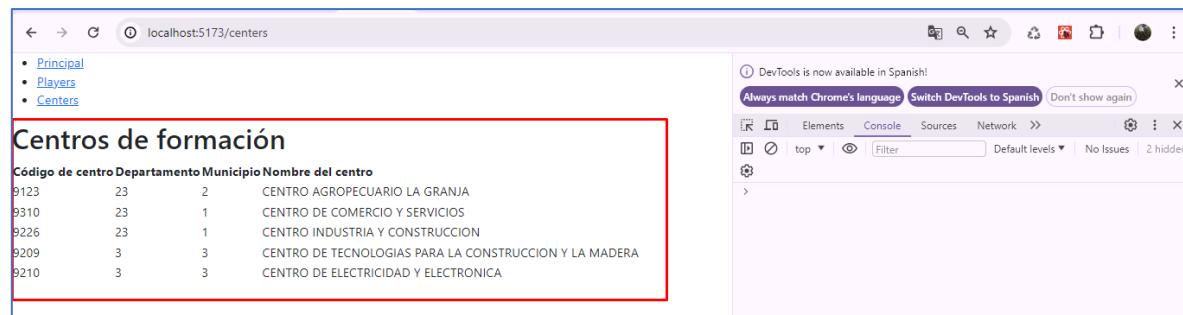
```

```

Welcome crudCenters.jsx ...
1 import axios from "axios"
2 import { useState, useEffect } from "react"
3
4 const URI_CENTERS = 'http://localhost:8000/centers/'
5
6 const CrudCenters = () => {
7
8   const [centers, setCenters] = useState([])
9
10  const getAllCenters = async () => {
11
12    const response = await axios.get(URI_CENTERS)
13
14    setCenters(response.data)
15
16  }
17
18  useEffect(() => {
19    getAllCenters()
20  }, [1])
21
22  return (
23    <>
24      <h1>Centros de formación</h1>
25      <table>
26        <thead>
27          <tr>
28            <th>Código de centro</th>
29            <th>Departamento</th>
30            <th>Municipio</th>
31            <th>Nombre del centro</th>
32          </tr>
33        </thead>
34        <tbody>
35          {centers.map((center) => (
36
37            <tr key={center.id}>
38              <td>{center.codigo_centro}</td>
39              <td>{center.idDepto}</td>
40              <td>{center.id_municipio}</td>
41              <td>{center.nombre_centro}</td>
42            </tr>
43          ))}
44        </tbody>
45      </table>
46    </>
47  )
48}
49
50 export default CrudCenters

```

El resultado es el siguiente:

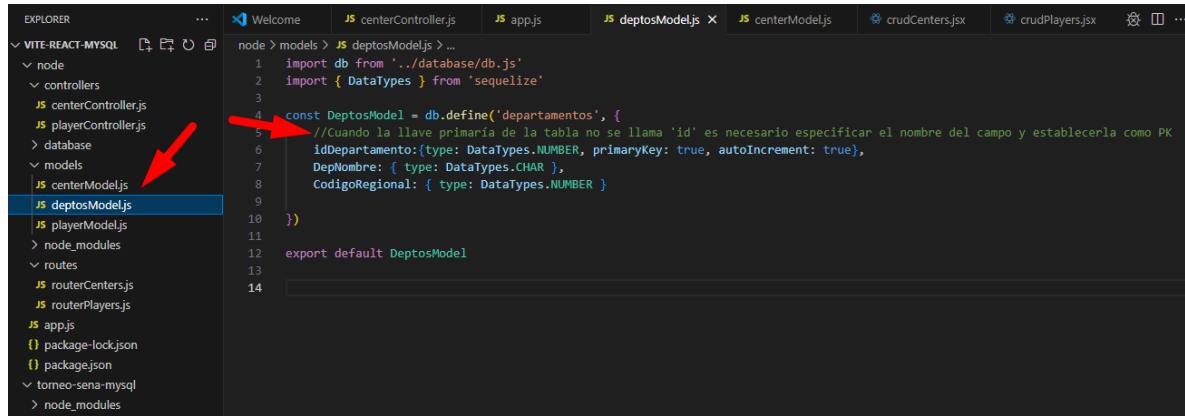


Código de centro	Departamento	Municipio	Nombre del centro
9123	23	2	CENTRO AGROPECUARIO LA GRANJA
9310	23	1	CENTRO DE COMERCIO Y SERVICIOS
9226	23	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	3	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	3	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Como se observa, en las filas de departamento y municipio se muestra un número que representa el identificador de cada uno, no es correcto dejarlo de esta manera, en su lugar hay que mostrar el nombre del departamento y el nombre del municipio.

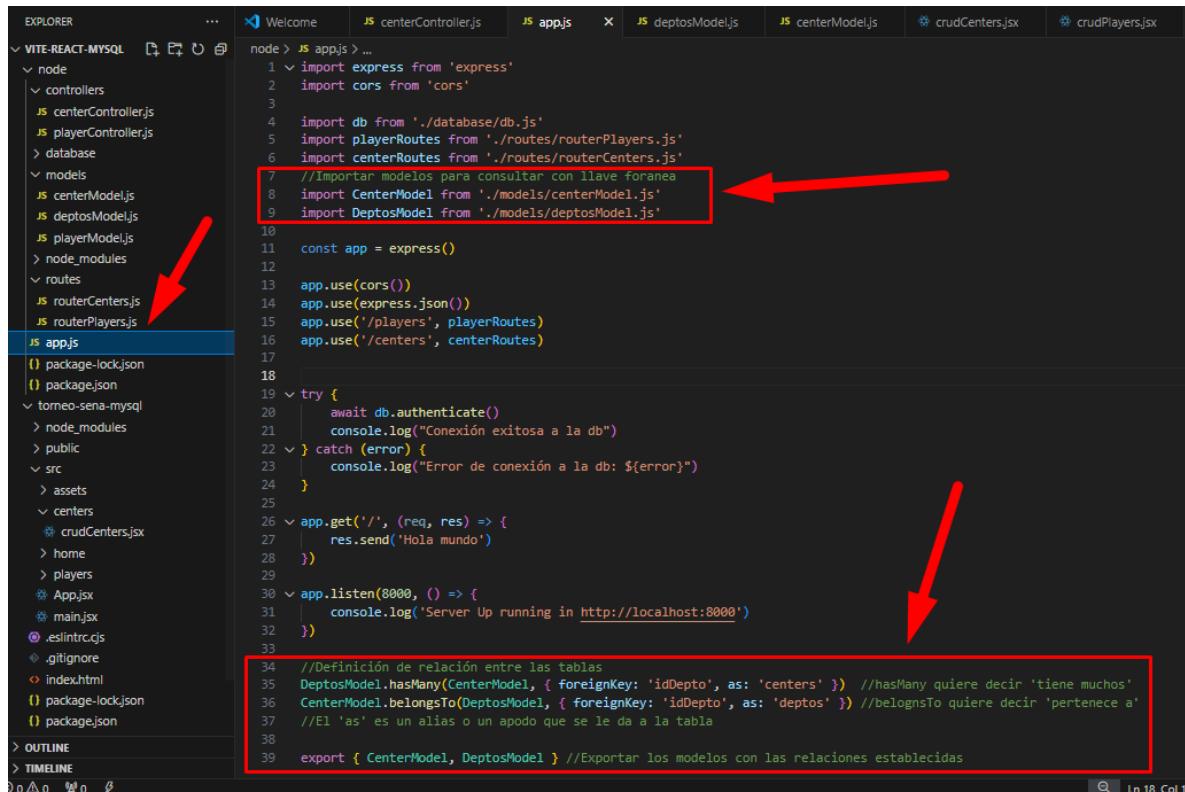
Fase 2:

Crear modelo de la tabla departamentos:



```
node > models > JS deptosModel.js > ...
1 import db from '../database/db.js'
2 import { DataTypes } from 'sequelize'
3
4 const DeptosModel = db.define('departamentos', {
5   //Cuando la llave primaria de la tabla no se llama 'id' es necesario especificar el nombre del campo y establecerla como PK
6   id: DataTypes.NUMBER, primaryKey: true, autoIncrement: true,
7   Nombre: { type: DataTypes.CHAR },
8   CodigoRegional: { type: DataTypes.NUMBER }
9 })
10
11 export default DeptosModel
12
13
14
```

Modificaciones en el archivo **app.js** del back-end, en este archivo es donde se establece la relación entre las tablas:



```
node > JS app.js > ...
1 // Import express from 'express'
2 import cors from 'cors'
3
4 import db from './database/db.js'
5 import playerRoutes from './routes/routerPlayers.js'
6 import centerRoutes from './routes/routerCenters.js'
7 //Importar modelos para consultar con llave foranea
8 import CenterModel from './models/centerModel.js'
9 import DeptosModel from './models/deptosModel.js'
10
11 const app = express()
12
13 app.use(cors())
14 app.use(express.json())
15 app.use('/players', playerRoutes)
16 app.use('/centers', centerRoutes)
17
18
19 try {
20   await db.authenticate()
21   console.log("Conexión exitosa a la db")
22 } catch (error) {
23   console.log("Error de conexión a la db: ${error}")
24 }
25
26 app.get('/', (req, res) => {
27   res.send('Hola mundo')
28 })
29
30 app.listen(8000, () => {
31   console.log('Server Up running in http://localhost:8000')
32 })
33
34 //Definición de relación entre las tablas
35 DeptosModel.hasMany(CenterModel, { foreignKey: 'idDepto', as: 'centers' }) //hasMany quiere decir 'tiene muchos'
36 CenterModel.belongsTo(DeptosModel, { foreignKey: 'idDepto', as: 'deptos' }) //belongsTo quiere decir 'pertenece a'
37 //El 'as' es un alias o un apodo que se le da a la tabla
38
39 export { CenterModel, DeptosModel } //Exportar los modelos con las relaciones establecidas
40
```

Modificación en el controlador del centro de formación (centerController.js):

```

    node > controllers > JS centerController.js > [o] getCenter
    1 import { Sequelize } from "sequelize";
    2 import CenterModel from "../models/centerModel.js";
    3 import { DeptosModel } from "../app.js"; // Importar modelo modificado en el archivo app.js
    4
    5 //Mostrar todos los registros
    6 export const getAllCenters = async (req, res) => {
    7   try {
    8     const centers = await CenterModel.findAll({
    9       include: [
    10         {
    11           model: DeptosModel,
    12           as: 'deptos'
    13         }
    14       ]
    15     })
    16     res.json(centers)
    17   } catch (error) {
    18     res.json({ message: error.message })
    19   }
    20 }
    21
  
```

Al realizar la consulta en el Thunder Client se observa que a la respuesta se le agregan los datos del departamento con el que hay relación a través del identificador del departamento:

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

GET localhost:8000 17 hours ago

GET localhost:8000/players/1 28 days ago

GET localhost:8000/players/ 1 month ago

DEL localhost:8000/players/2 2 months ago

1

2

3

Send

Body

4

5

Status: 200 OK Size: 1.49 KB Time: 189 ms

Response Headers Cookies Results Docs

JSON Content

```

1 {
2   "documento": 379154268,
3   "nombres": "Tomas",
4   "apellidos": "Perez",
5   "genero": "F"
6 }

```

```

1 [
2   {
3     "id": 1,
4     "codigo_centro": 9123,
5     "idDepto": 23,
6     "id_municipio": 2,
7     "nombre_centro": "CENTRO AGROPECUARIO LA GRANJA",
8     "createdAt": "2021-01-10T15:53:11.000Z",
9     "updatedAt": "2024-05-10T16:07:13.000Z",
10    "deptos": [
11      {
12        "idDepartamento": 23,
13        "DepNombre": "Tolima",
14        "CodigoRegional": 73,
15        "createdAt": null,
16        "updatedAt": null
17      },
18      {
19        "id": 2,
20        "codigo_centro": 9310,
21        "idDepto": 23,
22        "id_municipio": 1,
23        "nombre_centro": "CENTRO DE COMERCIO Y SERVICIOS",
24        "createdAt": "2021-01-10T15:55:17.000Z",
25        "updatedAt": "2024-05-10T16:07:13.000Z",
26        "deptos": [
27          {
28            "idDepartamento": 23
29          }
30        ]
31      }
32    ]
33  }
34 ]

```

El departamento tiene como llave primaria el campo idDepartamento, que a su vez es la llave foránea de la tabla de centros de formación, pero allí aparece con el nombre de idDepto:

```

GET http://localhost:8000/centers/
Send
Query Headers2 Auth Body1 Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content Format
1 [
2   "documento": 379154268,
3   "nombres": "Tomasa",
4   "apellidos": "Perez",
5   "genero": "F"
6 ]
Status: 200 OK Size: 1.49 KB Time: 189 ms
Response Headers7 Cookies Results Docs
1 [
2   {
3     "id": 1,
4     "codigo_centro": 9123,
5     "idDepto": 23, // clave foranea
6     "id_municipio": 2,
7     "nombre_centro": "CENTRO AGROPECUARIO LA GRANJA",
8     "createdAt": "2021-01-10T15:53:11.000Z",
9     "updatedAt": "2024-05-10T16:07:13.000Z",
10    "deptos": [
11      {
12        "idDepartamento": 23,
13        "DepNombre": "Tolima",
14        "CodigoRegional": 73,
15        "createdAt": null,
16        "updatedAt": null
17      }
18    ],
19    "id": 2,
20    "codigo_centro": 9310,
21    "idDepto": 23, // clave foranea
22    "id_municipio": 1,
23    "nombre_centro": "CENTRO DE COMERCIO Y SERVICIOS",
24    "createdAt": "2021-01-10T15:55:17.000Z",
25    "updatedAt": "2024-05-10T16:07:13.000Z",
26    "deptos": [
27      {
28        "idDepartamento": 23,
29        "DepNombre": "Tolima",
30        "CodigoRegional": 73,
31        "createdAt": null,
32        "updatedAt": null
33      }
34    ]
35  ]

```

Ajustes en el componente crudCenters.jsx para mostrar datos en la vista:

```

EXPLORER ... Welcome centerController.jsx app.jsx deptosModel.js centerModel.js crudCenters.jsx
VITE-REACT-MYSQL ...
node
  database
    models
      centerModel.js
      deptosModel.js
      playerModel.js
    node_modules
  routes
    routerCenters.js
    routerPlayers.js
  app.js
  package-lock.json
  package.json
tomeo-sena-mysql
  node_modules
  public
  src
    assets
    centers
      crudCenters.jsx
    home
    players
    App.jsx
    main.jsx
    .eslintrc.js
    .gitignore
    index.html
    package-lock.json
    package.json
    README.md
    vite.config.js
  OUTLINE

```

```

6 const CrudCenters = () => {
10   const getAllCenters = async () => {
16   }
17
18   useEffect(() => {
19     getAllCenters()
20   }, [])
21
22   return (
23     <>
24       <h1>Centros de formación</h1>
25       <table>
26         <thead>
27           <tr>
28             <th>Código de centro</th>
29             <th>Departamento</th>
30             <th>Municipio</th>
31             <th>Nombre del centro</th>
32           </tr>
33         </thead>
34         <tbody>
35           {centers.map((center) => (
36             <tr key={center.id}>
37               <td>{center.codigo_centro}</td>
38               <td>{center.deptos.DepNombre}</td> // here
39               <td>{center.id_municipio}</td>
40               <td>{center.nombre_centro}</td>
41             </tr>
42           ))
43         </tbody>
44       </table>
45     </>
46   )
47 }
48
49 export default CrudCenters

```

El resultado es el siguiente:

localhost:5173/centers

- Principal
- Players
- Centers

Centros de formación

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Fase 3:

Aquí la sección de mostrar el nombre del municipio en el listado de registros. **Queda como tarea para los aprendices.**

Fase 4:

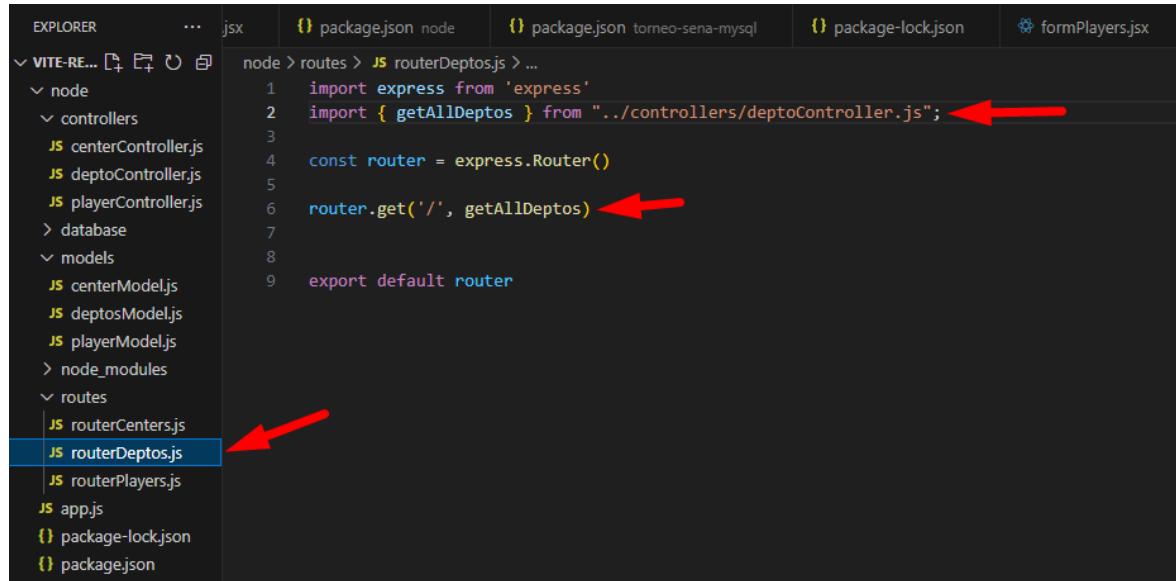
Crear la parte del back-end que está pendiente para gestionar los departamentos (el controlador y el archivo de rutas).

Archivo **deptoController.js** con función `getAllDeptos` para consultar todos los departamentos:

```
EXPLORER ... JS deptosModel.js JS deptoController.js X JS playerController.js JS playerModel.js
VITE-RE... node controllers > JS deptoController.js > [e] getAllDeptos
  > database
  > models
    JS centerModel.js
    JS deptosModel.js
    JS playerModel.js
  > node_modules
  > routes
    JS routerCenters.js
    JS routerDeptos.js
    JS routerPlayers.js
JS app.js
{} package-lock.json
{} package.json
```

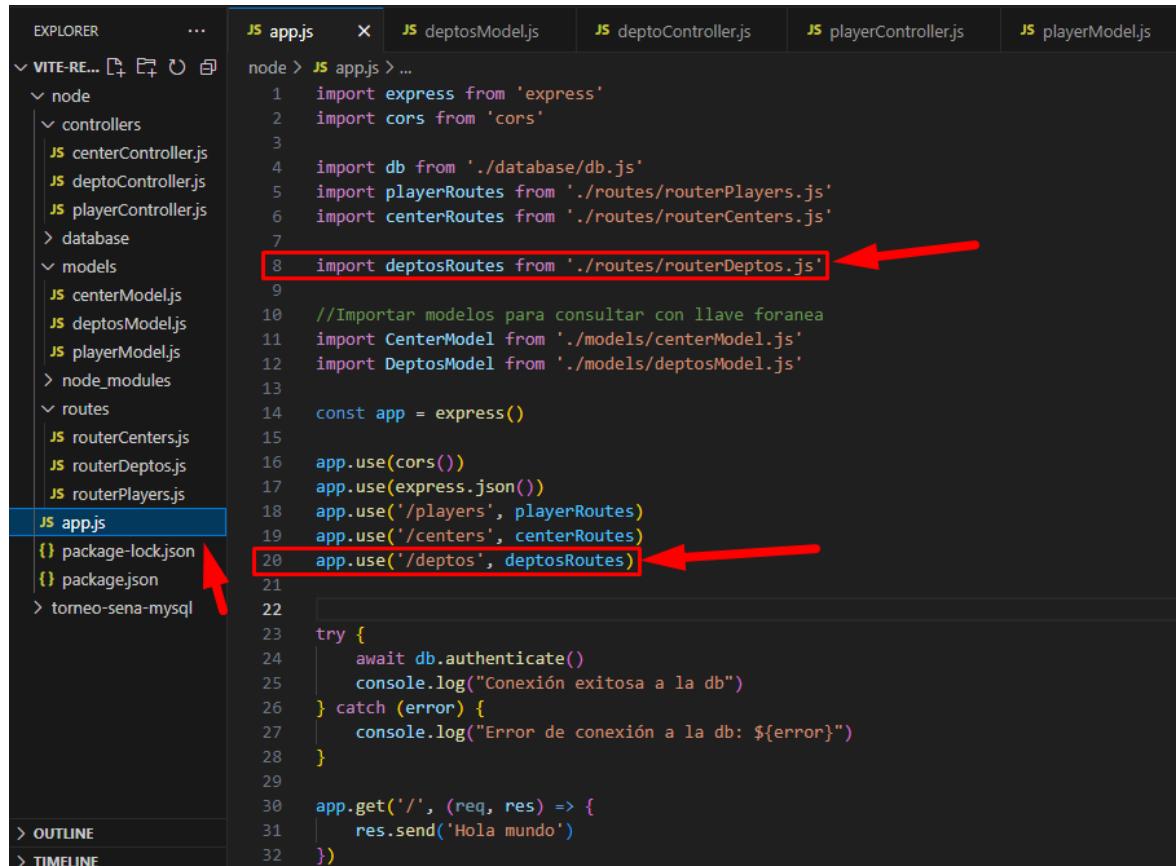
```
node > controllers > JS deptoController.js > [e] getAllDeptos
1 import { Sequelize } from "sequelize";
2 import DeptosModel from "../models/deptosModel.js"
3
4 export const getAllDeptos = async (req, res) => {
5
6   try {
7
8     const deptos = await DeptosModel.findAll()
9     res.json(deptos)
10
11   } catch (error) {
12
13     res.json({ message: error.message })
14   }
15
16 }
17 }
```

Archivo **routerDeptos.js**, por el momento definir solamente ruta para traer todos los departamentos:



```
node > routes > JS routerDeptos.js ...
1 import express from 'express'
2 import { getAllDeptos } from "../controllers/deptoController.js";
3
4 const router = express.Router()
5
6 router.get('/', getAllDeptos)
7
8
9 export default router
```

Ajustes en archivo **app.js** para establecer rutas de departamentos:



```
node > JS app.js > ...
1 import express from 'express'
2 import cors from 'cors'
3
4 import db from './database/db.js'
5 import playerRoutes from './routes/routerPlayers.js'
6 import centerRoutes from './routes/routerCenters.js'
7
8 import deptosRoutes from './routes/routerDeptos.js'
9
10 //Importar modelos para consultar con llave foranea
11 import CenterModel from './models/centerModel.js'
12 import DeptosModel from './models/deptosModel.js'
13
14 const app = express()
15
16 app.use(cors())
17 app.use(express.json())
18 app.use('/players', playerRoutes)
19 app.use('/centers', centerRoutes)
20 app.use('/deptos', deptosRoutes)
21
22
23 try {
24     await db.authenticate()
25     console.log("Conexión exitosa a la db")
26 } catch (error) {
27     console.log("Error de conexión a la db: ${error}")
28 }
29
30 app.get('/', (req, res) => {
31     res.send('Hola mundo')
32 })
```

Crear el componente `formCenters.jsx` para el formulario que servirá para guardar y actualizar registros.

Componente `formCenters.jsx`

The screenshot shows a code editor with the following details:

- EXPLORER:** Shows the project structure:
 - VITE-RE... (with a red arrow pointing to it)
 - node
 - torneo-sena-mysql
 - node_modules
 - public
 - src
 - assets
 - centers
 - crudCenters.jsx
 - formCenters.jsx (highlighted with a blue background)
 - home
 - players
 - App.jsx
 - main.jsx
 - .eslintrc.js
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md
 - vite.config.js
- CODE:** The content of `formCenters.jsx`:

```
1 import axios from "axios"
2 import { useState, useEffect } from "react"
3
4 const URI_DEPTOS = 'http://localhost:8000/deptos/'
5 //esta uri ya debe estar definida en el backend (routerDeptos.js, deptosModel.js y deptoController)
6
7 const FormCenters = () => {
8     //props para los campos del formulario
9     const [codigoCentro, setCodigoCentro] = useState('')
10    const [departamento, setDepartamento] = useState('')
11    const [municipio, setMunicipio] = useState('')
12    const [nombreCentro, setNombreCentro] = useState('')
13
14    //prop para departamentos
15    const [datosDepartamentos, setDatosDepartamentos] = useState([])
16
17    //función para obtener los departamentos, con el fin de cargarlos en el select de departamentos
18    const getDeptos = async () => {
19        const deptos = await axios.get(URI_DEPTOS)
20        // console.log(deptos.data)
21        setDatosDepartamentos(deptos.data)
22    }
23
24    useEffect(() => {
25        getDeptos() //Ejecutar al cargar la página para obtener los departamentos
26    }, [])
27
28    return (
29        <>
30
31        <form>
32            <label htmlFor="codigoCentro">Codigo: </label>
33            <input type="number" value={codigoCentro} onChange={(e) => e.target.value} id="codigoCentro" />
34            <br />
35            <label htmlFor="departamento">Departamento: </label>
36            <select type="number" value={departamento} onChange={(e) => setDepartamento(e.target.value)} id="departamento">
37                <option value="">selecciona uno..</option>
38                {datosDepartamentos.map(depto =>
39                    <option key={depto.idDepartamento} value={depto.idDepartamento}>{depto.DepNombre}</option>
40                )}
41            </select>
42        </form>
43    )
44}
45
46
47
48
49 export default FormCenters
```

Agregar el componente `formCenters.jsx` al componente `crudCenters.jsx`

Recuerde importarlo en la parte superior:

```

EXPLORER ... JS playerModel.js JS centerModel.js crudCenters.jsx X formCenters.jsx App.jsx Home.jsx
VITE-RE... [+] ⚡ ⚡ ...
  > node
  > torneo-sena-mysql
    > node_modules
    > public
  > src
    > assets
    > centers
      crudCenters.jsx (highlighted)
      formCenters.jsx
    > home
    > players
    App.jsx
    main.jsx
    .eslintrc.js
    .gitignore
    index.html
    package-lock.json
    package.json
    README.md
    vite.config.js
  > OUTLINE
  > TIMELINE

```

7 const CrudCenters = () => {
23 return (
24 <>
25 <h1>Centros de formación</h1>
26 <table>
27 <thead>
28 <tr>
29 <th>Código de centro</th>
30 <th>Departamento</th>
31 <th>Municipio</th>
32 <th>Nombre del centro</th>
33 </tr>
34 </thead>
35 <tbody>
36 {centers.map((center) => (
37 <tr key={center.id}>
38 <td>{center.codigo_centro}</td>
39 <td>{center.deptos.DepNombre}</td>
40 <td>{center.id_municipio}</td>
41 <td>{center.nombre_centro}</td>
42 </tr>
43))}
44 </tbody>
45 </table>
46 <hr />
47 <FormCenters /> (highlighted)
48)</>
49 }
50
51 }
52
53 export default CrudCenters

Resultado:

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Codigo:
 Departamento:

Al dar clic se despliegan los departamentos:

Código de centro

	Nombre del centro
9123	CENTRO AGROPECUARIO LA GRANJA
9310	CENTRO DE COMERCIO Y SERVICIOS
9226	CENTRO INDUSTRIA Y CONSTRUCCION
9209	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Codigo:

Departamento: Selecciona uno..

Antioquia
Atlantico
D. C. Santa Fe de Bogotá
Bolivar
Boyaca
Caldas
Caqueta
Cauca
Cesar
Cordova
Cundinamarca
Choco
Huila
La Guajira
Magdalena
Meta
Nariño
Norte de Santander
Quindio

Selecciona uno..

Si se inspeccionan los elementos en el navegador, se observan todas las etiquetas option con su respectivo value:

The screenshot shows a browser window with the URL `localhost:5173/centers`. The page displays a table titled "Centros de formación" with columns: Código de centro, Departamento, Municipio, and Nombre del centro. Below the table is a form with two input fields: "Codigo:" and "Departamento". The "Departamento" field is a dropdown menu labeled "Selecciona uno..". A red box highlights the dropdown menu in the browser's developer tools (Elements tab). Another red box highlights the list of options within the dropdown menu, which includes: Antioquia, Atlántico, C. Santa Fe de Bogotá, Bogotá, Caldas, Cesar, Córdoba, Cundinamarca, Cúcuta, Guajira, Magdalena, Meta, Nariño, and Norte de Santander.

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Codigo:

Departamento: Selecciona uno..

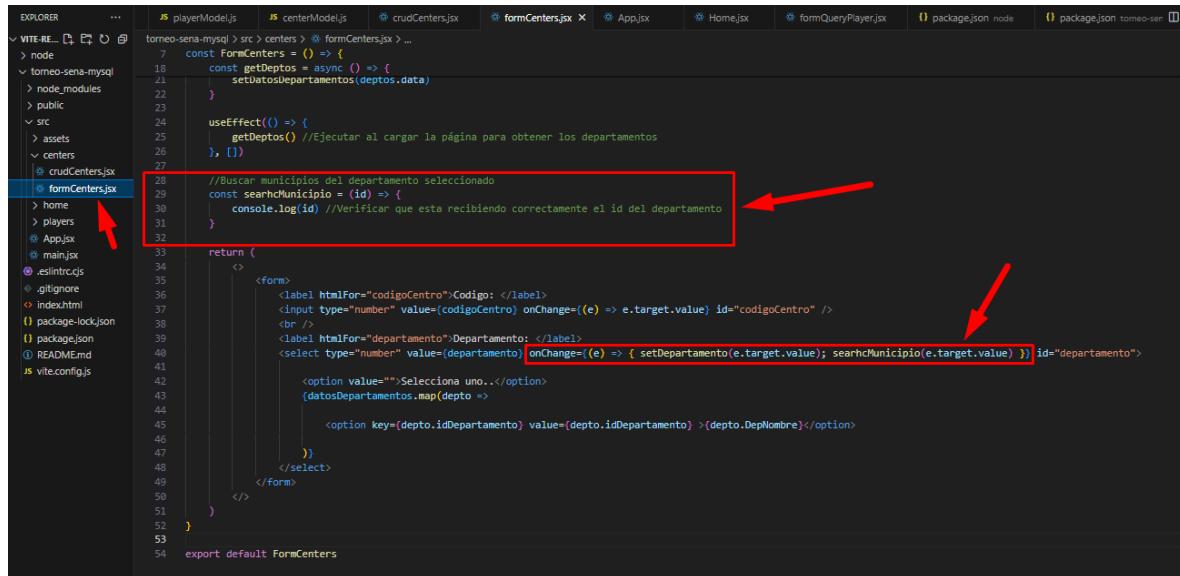
DevTools is now available in Spanish! Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network >

```
html body div#root form select#departamento
```

Styles Computed Layout Event Listeners DOM Breakpoints Properties >

Ahora hay que definir una función que reciba el identificador del departamento seleccionado por el usuario, una vez se reciba este número hay que buscar los municipios de dicho departamento y cargarlos en el select de municipios:



```

    const FormCenters = () => {
      const getDeptos = async () => {
        setDatosDepartamentos(deptos.data)
      }
      useEffect(() => {
        getDeptos() //Ejecutar al cargar la página para obtener los departamentos
      }, [])
    }

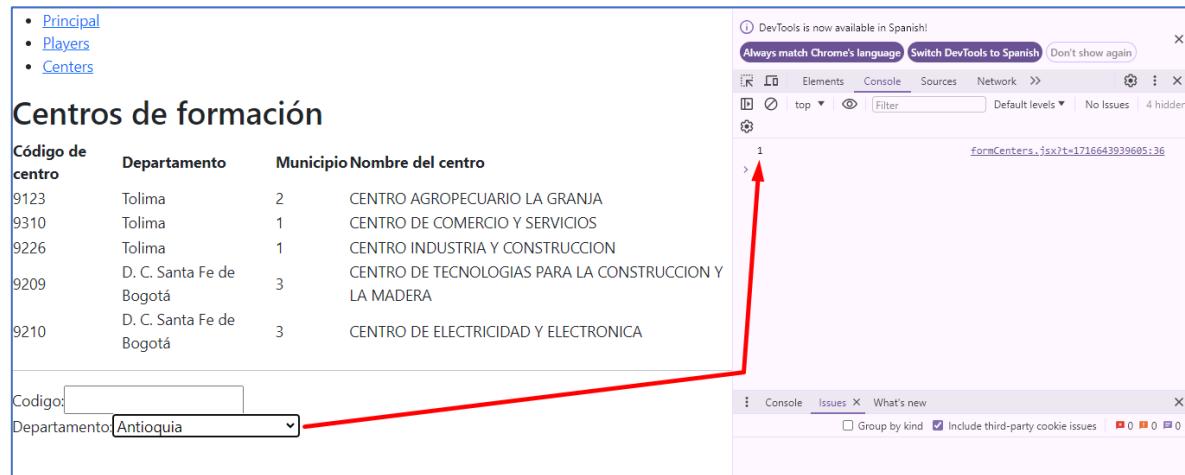
    //Buscar municipios del departamento seleccionado
    const searchMunicipio = (id) => {
      console.log(id) //Verificar que esta recibiendo correctamente el id del departamento
    }

    return (
      <form>
        <label htmlFor="codigoCentro">Codigo:</label>
        <input type="number" value={codigoCentro} onChange={(e) => e.target.value} id="codigoCentro" />
        <br />
        <label htmlFor="departamento">Departamento:</label>
        <select type="number" value={departamento} onChange={(e) => { setDepartamento(e.target.value); searchMunicipio(e.target.value) }} id="departamento">
          <option value="">Selecciona uno..</option>
          {datosDepartamentos.map(depto =>
            <option key={depto.idDepartamento} value={depto.idDepartamento}>{depto.DepNombre}</option>
          )}
        </select>
      </form>
    )
  }

  export default FormCenters

```

Resultado:



• Principal
• Players
• Centers

Centros de formación

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Codigo:

Departamento:

Una vez verificado que se imprimen correctamente los identificadores de los departamentos, es necesario hacer la consulta en la base de datos para traer los municipios de dicho departamento, para ello es necesario definir el back-end de municipios (modelo, controlador y rutas):

Estructura de la tabla municipios:

Servidor: 127.0.0.1 » Base de datos: torneo_react » Tabla: municipios

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar
2	mcipioCodigo	int(11)			No	Ninguna			Cambiar Eliminar
3	mcipioNombre	varchar(150)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar
4	idDept	int(11)			No	Ninguna			Cambiar Eliminar
5	createdat	timestamp			No	current_timestamp()			Cambiar Eliminar
6	updatedat	timestamp			No	current_timestamp()			Cambiar Eliminar

Modelo de la tabla municipios:

```

EXPLORER ... centerModel.js JS municipioModel.js X JS playerController.js JS municipioController.js
VITE-REACT... node > models > JS municipioModel.js ...
  > controllers
  > database
  > models
    JS centerModel.js
    JS deptosModel.js
    JS municipioModel.js ✓
    JS playerModel.js
  > node_modules
  > routes
  JS app.js
  {} package-lock.json
  {} package.json

```

```

node > models > JS municipioModel.js ...
1 import { DataTypes } from "sequelize"
2 import db from "../database/db.js"
3
4 const MunicipioModel = db.define('municipios', {
5   mcipioCodigo: { type: DataTypes.NUMBER },
6   mcipioNombre: { type: DataTypes.CHAR },
7   idDept: { type: DataTypes.NUMBER }
8 }, {
9   freezeTableName: true
10 })
11
12 export default MunicipioModel

```

Controlador:

```

EXPLORER ... erController.js JS app.js TC Release Notes TC localhost:8000 JS municipioModel.js JS municipioController.js X JS routerMunicipios.js
VITE-REACT... node > controllers > JS municipioController.js > getMunicipiosPorDept
  > controllers
    JS centerController.js
    JS deptoController.js
    JS municipioController.js ✓
    JS playerController.js
  > database
  > models
  > node_modules
  > routes
    JS routerCenters.js
    JS routerDeptos.js
    JS routerMunicipios.js
    JS routerPlayers.js
  JS app.js
  {} package-lock.json
  {} package.json
  > torneo-sena-mysql
    > node_modules
    > public
    > src
      > assets
      > centers
        crudCenters.jsx
        formCenters.jsx

```

```

node > controllers > JS municipioController.js > getMunicipiosPorDept
1 import { Sequelize } from "sequelize";
2 import MunicipioModel from "../models/municipioModel.js"
3
4 //obtener todos los municipios
5 export const getAllMunicipios = async (req, res) => {
6   try {
7     const municipios = await MunicipioModel.findAll()
8     res.json(municipios)
9   } catch (error) {
10     res.json({ message: error.message })
11   }
12 }
13
14 //obtener los municipios de un departamento
15 export const getMunicipiosPorDept = async (req, res) => {
16   try {
17     const municipios = await MunicipioModel.findAll({
18       where: { idDept: req.params.idDept } //Hacer la búsqueda con el id del departamento
19     })
20     console.log(municipios)
21     res.json(municipios)
22   } catch (error) {
23     res.json({ message: error.message })
24   }
25 }
26
27
28 }

```

Archivo de rutas:

Instructor Jorge E. Andrade C.
jandrade@sena.edu.co
Centro Agropecuario La Granja
Regional Tolima

```

EXPLORER ... centerModel.js JS municipioModel.js JS playerController.js JS municipioController.js JS routerMunicipios.js X ⚡ ⓘ ...
VITE-REACT... node > routes > JS routerMunicipios.js > ...
  1 import express from 'express'
  2 import { getAllMunicipios, getMunicipiosPorDepto } from '../controllers/municipioController.js'
  3
  4 const router = express.Router()
  5
  6 router.get('/', getAllMunicipios)
  7 router.get('/depto/:idDepto', getMunicipiosPorDepto) //consultar municipios por id de departamento
  8
  9 export default router

```

Ajustes en archivo app.js

```

EXPLORER ... JS app.js X JS deptosModel.js JS deptoController.js JS playerModel.js JS centerModel.js JS municip ...
VITE-REACT... node > JS app.js > ...
  1 import express from 'express'
  2 import cors from 'cors'
  3
  4 import db from './database/db.js'
  5 import playerRoutes from './routes/routerPlayers.js'
  6 import centerRoutes from './routes/routerCenters.js'
  7 import deptosRoutes from './routes/routerDeptos.js'
  8
  9 import municipioRoutes from './routes/routerMunicipios.js' ←
 10
 11 //Importar modelos para consultar con llave foranea
 12 import CenterModel from './models/centerModel.js'
 13 import DeptosModel from './models/deptosModel.js'
 14
 15 const app = express()
 16
 17 app.use(cors())
 18 app.use(express.json())
 19 app.use('/players', playerRoutes)
 20 app.use('/centers', centerRoutes)
 21 app.use('/deptos', deptosRoutes)
 22 app.use('/municipios', municipioRoutes) ←
 23
 24
 25 try {
 26   await db.authenticate()

```

Prueba en **Thunder Client**, fíjese en la ruta utilizada para hacer la consulta, en este caso se realiza la búsqueda de los municipios con idDepto = 23:

THUNDER CLIENT

New Request

Activity Collections Env

Query Headers 2 Auth Body 1 Tests Pre Run

Query Parameters

parameter value

Status: 200 OK Size: 293 Bytes Time: 39 ms

Response Headers 7 Cookies Results Docs

```

1 [
2   {
3     "id": 1,
4     "municipioCodigo": "73001",
5     "municipioNombre": "IBAGUE",
6     "idDepto": 23, ←
7     "createdAt": "2024-05-10T14:52:57.000Z",
8     "updatedAt": "2024-05-10T14:52:57.000Z"
9   },
10  {
11    "id": 2,
12    "municipioCodigo": "73268",
13    "municipioNombre": "ESPINAL",
14    "idDepto": 23, ←
15    "createdAt": "2024-05-10T14:52:57.000Z",
16    "updatedAt": "2024-05-10T14:52:57.000Z"
17  }
18 ]

```

Ajustes en el componente formCenters.jsx para verificar el funcionamiento de la consulta:

```

EXPLORER ... Welcome JS centerController.js JS app.js TO Release Notes TO localhost8000 JS municipioModel.js JS municipioController.js JS routerMunicipios.js
VITE-REACT-... + ⏪ ⏴ ⏵
node controllers
JS centerController.js
JS deptoController.js
JS municipioController.js
JS playerController.js
> database
> models
> node_modules
routes
JS routerCenters.js
JS routerDepts.js
JS routerMunicipios.js
JS routerPlayers.js
JS app.js
(I) package-lock.json
(I) package.json
tomeo-sena-mysql
> node_modules
public
src
> assets
> centers
crudCenters.jsx
JS formCenters.jsx
home
players
App.jsx
main.jsx
.eslintrc.js
.gitignore
index.html

```

```

1 import axios from "axios"
2 import { useState, useEffect } from "react"
3
4 const URI_DEPTOS = 'http://localhost:8000/depts/'
5 //esta url ya debe estar definida en el backend (routerDepts.js, deptosModel.js y deptoController)
6 const URI_MUNICIPIOS_POR_DEPTO = 'http://localhost:8000/municipios/depto/' ↑
7
8 const FormCenters = () => {
9   //props para los campos del formulario
10  const [codigoCentro, setCodigoCentro] = useState('')
11  const [departamento, setDepartamento] = useState('')
12  const [municipio, setMunicipio] = useState('')
13  const [nombreCentro, setNombreCentro] = useState('')
14
15  //prop para departamentos
16  const [datosDepartamentos, setDatosDepartamentos] = useState([])
17
18  //función para obtener los departamentos, con el fin de cargarlos en el select de departamentos
19  const getDeptos = async () => {
20    const deptos = await axios.get(URI_DEPTOS)
21    // console.log(deptos.data)
22    setDatosDepartamentos(deptos.data)
23  }
24
25  useEffect(() => {
26    getDeptos() //Ejecutar al cargar la página para obtener los departamentos
27  }, [])
28
29  //Buscar municipios del departamento seleccionado
30  const searchMunicipio = async (id) => {
31    //console.log(id) //Verificar que esta recibiendo correctamente el id del departamento
32    const municipios = await axios.get(URI_MUNICIPIOS_POR_DEPTO + id)
33    console.log(municipios.data)
34  }
35
36  return (
37    return (

```

Al seleccionar el departamento se imprimen los municipios correspondientes:

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Codigo:

Departamento: Tolima ↓

```

(2) [Object, Object]
  ▷ 0: {id: 1, mcipioCodigo: 73001, mcipioNombre: 'IBAGUÉ', idDepto: 23}
  ▷ 1: {id: 2, mcipioCodigo: 73268, mcipioNombre: 'ESPINAL', idDepto: 23}
  length: 2
  [[Prototype]]: Object

```

Ahora hay que definir una prop para los municipios y gestionarla para cargarlos en el select de municipios:

```

 6  const FormCenters = () => {
 7    //prop para departamentos
 8    const [datosDepartamentos, setDatosDepartamentos] = useState([ ])
 9    const [municipiosPorDepto, setMunicipiosPorDepto] = useState([ ])
10
11   //función para obtener los departamentos, con el fin de cargarlos en el select de departamentos
12   const getDeptos = async () => {
13     const deptos = await axios.get(URI_DEPTOS)
14     // console.log(deptos.data)
15     setDatosDepartamentos(deptos.data)
16   }
17
18   useEffect(() => {
19     getDeptos() //Ejecutar al cargar la página para obtener los departamentos
20   }, [])
21
22   //Buscar municipios del departamento seleccionado
23   const searchMunicipio = async (id) => {
24
25     //console.log(id) //Verificar que esta recibiendo correctamente el id del departamento
26     const mcipios = await axios.get(URI_MUNICIPIOS POR DEPTO + id)
27     // console.log(mcipios.data)
28     setMunicipiosPorDepto(mcipios.data) //Cargar los municipios en la prop
29   }
30
31
32   //Función para seleccionar un departamento
33   const setDepartamento = (e) => {
34     searchMunicipio(e.target.value)
35   }
36
37   //Función para seleccionar un municipio
38   const setMunicipio = (e) => {
39     console.log(e.target.value)
40   }
41
42   return (
43     <>
44       <form>
45         <label htmlFor="codigoCentro">Codigo: </label>
46         <input type="number" value={codigoCentro} onChange={(e) => e.target.value} id="codigoCentro" />
47         <br />
48         <label htmlFor="departamento">Departamento: </label>
49         <select value={departamento} onChange={(e) => { setDepartamento(e.target.value); searchMunicipio(e.target.value) }} id="departamento">
50           <option value="">Selecciona uno...</option>
51           {datosDepartamentos.map(depto =>
52             <option key={depto.idDepartamento} value={depto.idDepartamento}>{depto.DepNombre}</option>
53           )}
54         </select>
55         <br />
56         <label htmlFor="municipio">Municipio: </label>
57         <select value={municipio} onChange={(e) => setMunicipio(e.target.value)} id="municipio">
58           <option value="">Selecciona uno...</option>
59           {municipiosPorDepto.map(mcipio =>
60             <option key={mcipio.id} value={mcipio.id}>{mcipio.mcipioNombre}</option>
61           )}
62         </select>
63       </form>
64     </>
65   )
66 }
67
68
69
70
71
72
73
74
75 export default FormCenters

```

Resultado:

• [Principal](#)
• [Players](#)
• [Centers](#)

Centros de formación

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Código:

Departamento: Tolima

Municipio: Seleccion uno...

IBAGUÉ
ESPINAL

Municipios del Tolima

Al inspeccionar los elementos en el navegador allí aparecen los municipios cargados:

The screenshot shows a web application interface. On the left, a sidebar lists navigation items: 'Principal', 'Players', and 'Centers'. The main content area has a title 'Centros de formación' and displays a table with columns: 'Código de centro', 'Departamento', 'Municipio', and 'Nombre del centro'. The table data is as follows:

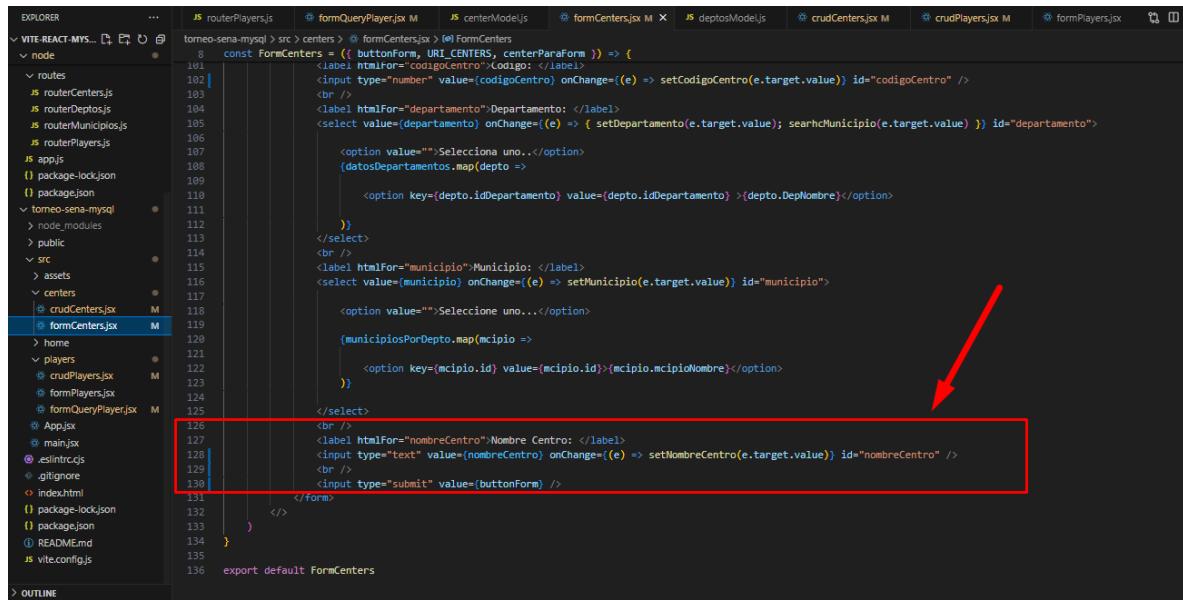
Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION
9209	D. C. Santa Fe de Bogotá	3	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA
9210	D. C. Santa Fe de Bogotá	3	CENTRO DE ELECTRICIDAD Y ELECTRONICA

Below the table are three input fields: 'Codigo:' with a text input, 'Departamento:' with a dropdown menu containing 'Tolima' and 'Bogotá', and 'Municipio:' with a dropdown menu containing 'IBAGUÉ' and 'SPINAL'. A red arrow points from the 'Municipio:' dropdown to the DevTools sidebar.

The DevTools sidebar is open, showing the page's source code. The code snippet for the 'Municipio' dropdown is highlighted with a red box and a red arrow pointing to it. The code is as follows:

```
<label for="municipio">Municipio:</label>
<select id="municipio">
  <option value="Selecione uno...></option>
  <option value="1">IBAGUÉ</option>
  <option value="2">SPINAL</option>
</select>
```

Completar los campos del formulario:



```
const FormCenters = ({ buttonForm, URL_CENTERS, centerParaForm }) => {
  const [codigoCentro, setCodigoCentro] = useState("");
  const [departamento, setDepartamento] = useState("");
  const [municipio, setMunicipio] = useState("");
  const [nombreCentro, setNombreCentro] = useState("");

  const handleCodigoCentroChange = (e) => setCodigoCentro(e.target.value);
  const handleDepartamentoChange = (e) => setDepartamento(e.target.value);
  const handleMunicipioChange = (e) => setMunicipio(e.target.value);
  const handleNombreCentroChange = (e) => setNombreCentro(e.target.value);

  const handleSubmit = (e) => {
    e.preventDefault();
    const newCenter = {
      id: URL_CENTERS,
      codigoCentro,
      departamento,
      municipio,
      nombreCentro
    };
    centerParaForm(newCenter);
  };

  return (
    <Form>
      <div>
        <label htmlFor="codigoCentro">Centro:</label>
        <input type="number" value={codigoCentro} onChange={handleCodigoCentroChange} id="codigoCentro" />
      </div>
      <div>
        <label htmlFor="departamento">Departamento:</label>
        <select value={departamento} onChange={handleDepartamentoChange}>
          <option value="">Selecciona uno...</option>
          {datosDepartamentos.map(depto =>
            <option key={depto.idDepartamento} value={depto.idDepartamento}>{depto.DepNombre}</option>
          )}
        </select>
        <br />
        <label htmlFor="municipio">Municipio:</label>
        <select value={municipio} onChange={handleMunicipioChange}>
          <option value="">Selecciona uno...</option>
          {municipiosPorDeptos.map(mcipio =>
            <option key={mcipio.id} value={mcipio.id}>{mcipio.mcipioNombre}</option>
          )}
        </select>
        <br />
        <label htmlFor="nombreCentro">Nombre Centro:</label>
        <input type="text" value={nombreCentro} onChange={handleNombreCentroChange} id="nombreCentro" />
        <br />
        <input type="submit" value={buttonForm} />
      </div>
    </Form>
  );
};

export default FormCenters;
```

Fase 5:

Completar código del componente `formCenters.jsx` para habilitar la funcionalidad de crear registros:

Desde `<CrudCenters />` enviar las propiedades que se utilizarán en `<FormCenters />`

```

    const CrudCenters = () => {
      const [centers, setCenters] = useState([]);
      const [buttonForm, setButtonForm] = useState(null);
      const [URI_CENTERS, setURI_CENTERS] = useState('');

      const handleNuevoCentro = async () => {
        const newCenter = {
          id_municipio: '',
          nombre_centro: '',
          codigo_centro: '',
          deptos: {
            DepNombre: ''
          }
        };
        const response = await axios.post(URI_CENTERS, newCenter);
        setCenters([...centers, response.data]);
        setButtonForm(null);
        setURI_CENTERS('');
      };

      const handleActualizarCentro = async (center) => {
        const updatedCenter = {
          ...center,
          id_municipio: '',
          nombre_centro: '',
          codigo_centro: '',
          deptos: {
            DepNombre: ''
          }
        };
        const response = await axios.put(`http://localhost:8000/centros/${center.id}`, updatedCenter);
        setCenters(centers.map((c) => c.id === center.id ? response.data : c));
        setButtonForm(null);
        setURI_CENTERS('');
      };

      const handleBorrarCentro = async (center) => {
        const response = await axios.delete(`http://localhost:8000/centros/${center.id}`);
        setCenters(centers.filter((c) => c.id !== center.id));
        setButtonForm(null);
        setURI_CENTERS('');
      };

      const handleNuevoDepartamento = async (depto) => {
        const newDept = {
          id_municipio: '',
          nombre_departamento: ''
        };
        const response = await axios.post(`http://localhost:8000/deptos`, newDept);
        const updatedDept = { ...depto, DeptNombre: response.data.DepNombre };
        setDatosDepartamentos([updatedDept, ...datosDepartamentos]);
        setDeptosModel(updatedDept);
        setButtonForm(null);
        setURI_CENTERS('');
      };

      const handleNuevoMunicipio = async (municipio) => {
        const newMun = {
          id_departamento: '',
          nombre_municipio: ''
        };
        const response = await axios.post(`http://localhost:8000/municipios`, newMun);
        const updatedMun = { ...municipio, MunicipioNombre: response.data.MunicipioNombre };
        setDatosMunicipios([updatedMun, ...datosMunicipios]);
        setDeptosModel(updatedMun);
        setButtonForm(null);
        setURI_CENTERS('');
      };

      const handleNuevoCentroForm = (e) => {
        e.preventDefault();
        handleNuevoCentro();
      };

      const handleActualizarCentroForm = (e) => {
        e.preventDefault();
        handleActualizarCentro(buttonForm);
      };

      const handleBorrarCentroForm = (e) => {
        e.preventDefault();
        handleBorrarCentro(buttonForm);
      };

      const handleNuevoDepartamentoForm = (e) => {
        e.preventDefault();
        handleNuevoDepartamento(buttonForm);
      };

      const handleNuevoMunicipioForm = (e) => {
        e.preventDefault();
        handleNuevoMunicipio(buttonForm);
      };

      return (
        <Table>
          <thead>
            <tr>
              <th>Nombre del centro</th>
              <th>Acciones</th>
            </tr>
          </thead>
          <tbody>
            {centers.map((center) => (
              <tr key={center.id}>
                <td>{center.codigo_centro}</td>
                <td>{center.deptos.DepNombre}</td>
                <td>{center.id_municipio}</td>
                <td>{center.nombre_centro}</td>
                <td>
                  <span className="btn btn-primary" onClick={() => handleActualizarCentroForm(e)}>Actualizar</span>
                  <span className="btn btn-danger m-1" onClick={() => handleBorrarCentroForm(e)}>Borrar</span>
                </td>
              </tr>
            ))}
          </tbody>
        </Table>
        <hr />
        <FormCenters buttonForm={buttonForm} URI_CENTERS={URI_CENTERS} />
      );
    }

    export default CrudCenters;
  
```

Recibir dichos argumentos en <FormCenters /> y utilizarlos:

```

    const FormCenters = ({ buttonForm, URI_CENTERS }) => {
      //props para los campos del formulario
      const [codigoCentro, setCodigoCentro] = useState('');
      const [departamento, setDepartamento] = useState('');
      const [municipio, setMunicipio] = useState('');
      const [nombreCentro, setNombreCentro] = useState('');

      //prop para departamentos
      const [datosDepartamentos, setDatosDepartamentos] = useState([]);

      //prop para municipios
      const [municipiosPorDepto, setMunicipiosPorDepto] = useState([]);

      //función para obtener los departamentos, con el fin de cargarlos en el select de departamentos
      const getDeptos = async () => {
        const deptos = await axios.get(URI_DEPTOS);
        //console.log(deptos.data)
        setDatosDepartamentos(deptos.data);
      }

      return (
        <Form>
          <div>
            <label>Nombre del centro:</label>
            <input type="text" value={nombreCentro} onChange={(e) => setNombreCentro(e.target.value)} />
          </div>
          <div>
            <label>Departamento:</label>
            <select>
              <option value=""></option>
              {datosDepartamentos.map((depto) => (
                <option value={depto.id}>{depto.DepNombre}</option>
              ))}
            </select>
          </div>
          <div>
            <label>Municipio:</label>
            <select>
              <option value=""></option>
              {municipiosPorDepto.map((municipio) => (
                <option value={municipio.id}>{municipio.MunicipioNombre}</option>
              ))}
            </select>
          </div>
          <div>
            <label>Codigo centro:</label>
            <input type="text" value={codigoCentro} onChange={(e) => setCodigoCentro(e.target.value)} />
          </div>
          <div>
            <label>Nombre departamento:</label>
            <input type="text" value={departamento} onChange={(e) => setDepartamento(e.target.value)} />
          </div>
          <div>
            <label>Nombre municipio:</label>
            <input type="text" value={municipio} onChange={(e) => setMunicipio(e.target.value)} />
          </div>
          <div>
            <button type="button" onClick={buttonForm}>Guardar</button>
          </div>
        </Form>
      );
    }

    export default FormCenters;
  
```

Agregar funciones para guardar y limpiar formulario en el archivo formCenters.jsx:

```

const FormCenters = ({ buttonForm, URI_CENTERS, centerParaForm }) => {
  ...
  // Función que recibe los datos del formulario
  const sendForm = (e) => {
    e.preventDefault()

    if (buttonForm === 'Actualizar') {
      ...
    } else if (buttonForm === 'Enviar') {
      ...
    }
  }

  const clearForm = () => {
    ...
  }
}

```

Resultado:

Código de centro	Departamento	Municipio	Nombre del centro	Acciones
9123	Tolima	2	CENTRO AGROPECUARIO LA GRANJA	
9310	Tolima	1	CENTRO DE COMERCIO Y SERVICIOS	
9226	Tolima	1	CENTRO INDUSTRIA Y CONSTRUCCION	
9209	D. C. Santa Fe de Bogotá 3		CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA	
9210	D. C. Santa Fe de Bogotá 3		CENTRO DE ELECTRICIDAD Y ELECTRONICA	
9127	Antioquia	45	Centro de Formación Profesional Minero Ambiental	

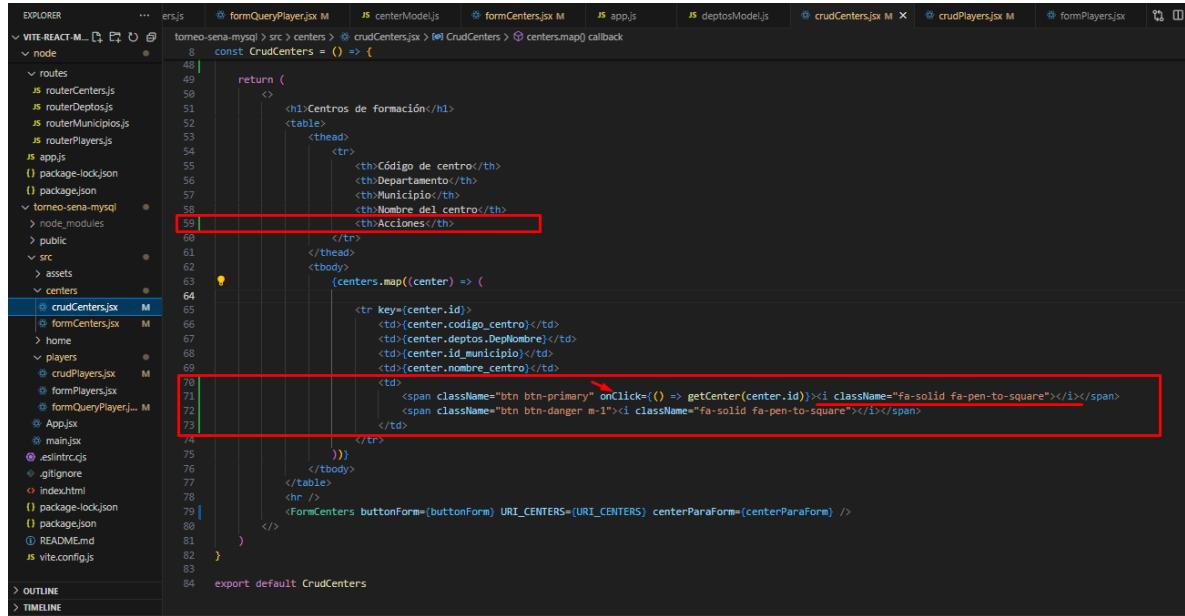
Código:
 Departamento:
 Municipio:
 Nombre Centro:

Componente REACT para Centros de Formación (Editar y Borrar)

Fase 1: Agregar columna de acciones con botones de editar y borrar.

Instructor Jorge E. Andrade C.
 jandrade@senia.edu.co
 Centro Agropecuario La Granja
 Regional Tolima

En el onClick se ejecuta la función getCenter() para buscar el centro de formación y enviar los datos al formulario. Fíjese que no se incluye una etiqueta <button> sino una y se agrega un ícono de Font-awasome:



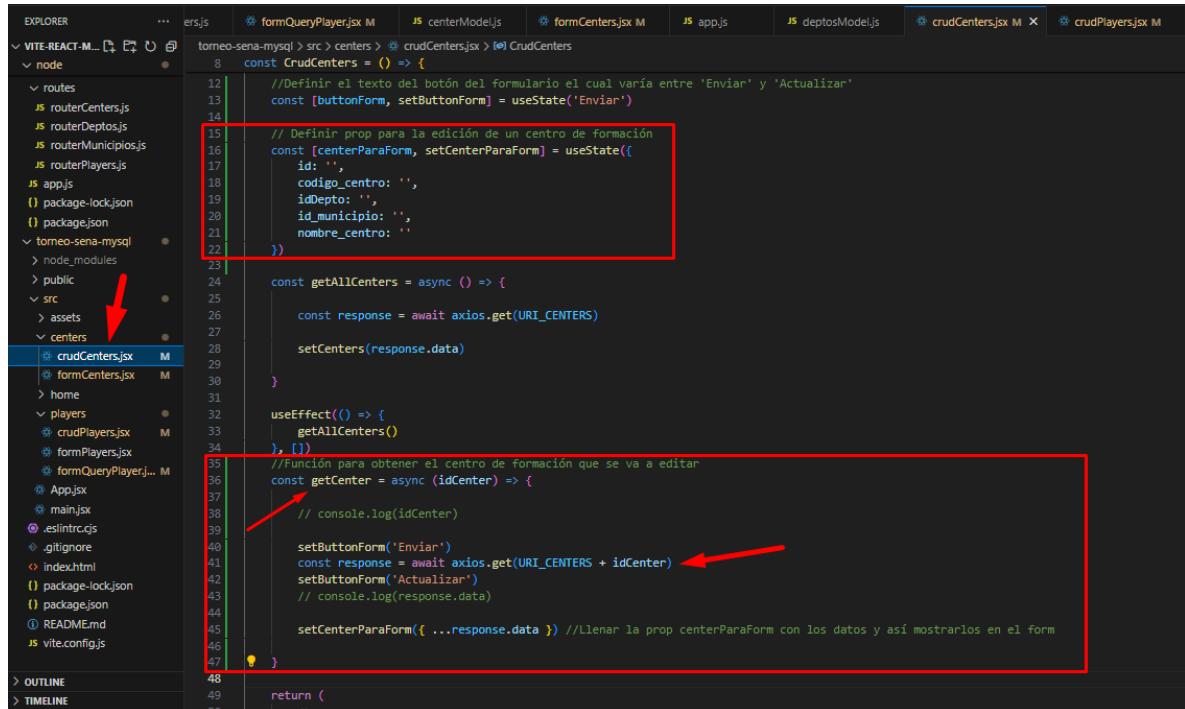
```

const CrudCenters = () => {
  return (
    <>
      <h1>Centros de formación</h1>
      <table>
        <thead>
          <tr>
            <th>Código de centro</th>
            <th>Departamento</th>
            <th>Municipio</th>
            <th>Nombre del centro</th>
            <th>Acciones</th>
          </tr>
        </thead>
        <tbody>
          {centers.map((center) => (
            <tr key={center.id}>
              <td>{center.codigo_centro}</td>
              <td>{center.deptos.DepLombra}</td>
              <td>{center.id_municipio}</td>
              <td>{center.nombre_centro}</td>
              <td>
                <span className="btn btn-primary" onClick={() => getCenter(center.id)}>i className="fa-solid fa-pen-to-square"</i></span>
                <span className="btn btn-danger m-1">i className="fa-solid fa-pen-to-square"</i></span>
              </td>
            </tr>
          ))}
        </tbody>
        <hr />
        <FormCenters buttonForm={buttonForm} URI_CENTERS={URI_CENTERS} centerParaForm={centerParaForm} />
      </table>
    </>
  )
}

export default CrudCenters

```

Elementos para búsqueda del centro de formación y envío de datos al form:



```

const CrudCenters = () => {
  //Definir el texto del botón del formulario el cual varía entre 'Enviar' y 'Actualizar'
  const [buttonForm, setButtonForm] = useState('Enviar')

  // Definir prop para la edición de un centro de formación
  const [centerParaForm, setCenterParaForm] = useState({
    id: '',
    codigo_centro: '',
    idDepto: '',
    id_municipio: '',
    nombre_centro: ''
  })

  const getAllCenters = async () => {
    const response = await axios.get(URI_CENTERS)
    setCenters(response.data)
  }

  useEffect(() => {
    getAllCenters()
  }, [])

  //Función para obtener el centro de formación que se va a editar
  const getCenter = async (idCenter) => {
    // console.log(idCenter)

    setButtonForm('Enviar')
    const response = await axios.get(URI_CENTERS + idCenter)
    setButtonForm('Actualizar')
    // console.log(response.data)

    setCenterParaForm({ ...response.data }) //Llenar la prop centerParaForm con los datos y así mostrarlos en el form
  }
}

return (

```

Enviar elemento 'centerParaForm' al componente formCenters.jsx:

```

EXPLORER      ... ers.js  formQueryPlayer.jsx M  JS centerModel.js  formCenters.jsx M  app.js  deptosModel.js  crudCenters.jsx M  crudCenters.jsx M  OUTLINE
VITE-REACT-M...  node  package-lock.json  package.json  tomeo-sena-mysql  node_modules  public  assets  centers  crudCenters.jsx M  formCenters.jsx M  home  players  crudPlayers.jsx M  formPlayers.jsx  formQueryPlayer.j... M  App.jsx  main.jsx  .eslintrc.js  .gitignore  index.html  package-lock.json  package.json  README.md  vite.config.js

crudCenters.jsx M
  const CrudCenters = () => {
    <thead>
      <tr>
        <th>Nombre del centro</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody>
      {centers.map((center) => (
        <tr key={center.id}>
          <td>{center.codigo_centro}</td>
          <td>{center.deptos.DepNombre}</td>
          <td>{center.id_municipio}</td>
          <td>{center.nombre_centro}</td>
          <td>
            <span className="btn btn-primary" onClick={() => getCenter(center.id)}><i className="fa-solid fa-pen-to-square">/</i></span>
            <span className="btn btn-danger m-1"><i className="fa-solid fa-pen-to-square">/</i></span>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
  <br />
  <FormCenters buttonForm={buttonForm} URI_CENTERS={URI_CENTERS} centerParaForm={centerParaForm} />
}
export default CrudCenters

```

Recibirlo en formCenters.jsx

```

EXPLORER      ... ers.js  formQueryPlayer.jsx M  JS centerModel.js  formCenters.jsx M  app.js  deptosModel.js  crudCenters.jsx M
VITE-REACT-M...  node  package-lock.json  package.json  tomeo-sena-mysql  node_modules  public  assets  centers  crudCenters.jsx M  formCenters.jsx M  home  players  crudPlayers.jsx M  formPlayers.jsx  formQueryPlayer.j... M  App.jsx  main.jsx  .eslintrc.js  .gitignore  index.html  package-lock.json  package.json  README.md  vite.config.js

formCenters.jsx M
  import axios from "axios"
  import { useState, useEffect } from "react"

  const URI_DEPTOS = 'http://localhost:8000/deptos'
  //esta uri ya debe estar definida en el backend (routerDeptos.js, deptosModel.js y deptoController)
  const URI_MUNICIPIOS_POR_DEPTO = 'http://localhost:8000/municipios/depto/'

  const FormCenters = ({ buttonForm, URI_CENTERS, centerParaForm }) => {
    //props para los campos del formulario
    const [codigoCentro, setCodigoCentro] = useState('')
    const [departamento, setDepartamento] = useState('')
    const [municipio, setMunicipio] = useState('')
    const [nombreCentro, setNombreCentro] = useState('')

    //prop para departamentos
    const [datosDepartamentos, setDatosDepartamentos] = useState([])

    //prop para municipios
    const [municipiosPorDepto, setMunicipiosPorDepto] = useState([])

    //función para obtener los departamentos, con el fin de cargarlos en el select de departamentos
    const getDeptos = async () => {
      const deptos = await axios.get(URI_DEPTOS)
      // console.log(deptos.data)
      setDatosDepartamentos(deptos.data)
    }
  }

```

```

VITE-REACT-M...
  |- routes
    |- routerCenters.js
    |- routerDeptos.js
    |- routerMunicipios.js
    |- routerPlayers.js
    |- app.js
    |- package-lock.json
    |- package.json
  |- node_modules
  |- public
  |- src
    |- assets
    |- centers
      |- crudCenters.jsx M
      |- formCenters.jsx M
      |- home
      |- players
      |- crudPlayers.jsx M
      |- formPlayers.jsx
      |- formQueryPlayer.jsx M
    |- App.jsx
    |- main.jsx
    |- .eslintrc.js
    |- .gitignore
    |- index.html
    |- package-lock.json
    |- package.json
    |- README.md
    |- vite.config.js

```

The code editor shows the `formCenters.jsx` file with several sections of code highlighted by red boxes:

- Section 1 (Line 28-34):** A function `setDataCenterParaForm` that sets department data to the form.
- Section 2 (Line 35-45):** A `useEffect` hook that executes `getDeptos` on mount and updates the form state when `centerParaForm` changes.
- Section 3 (Line 46-56):** A function `searchMunicipio` that fetches municipalities by department ID.

Cuando se da clic en el botón editar se cargan los datos en el form y se actualiza el texto del botón:

The screenshot shows a web application at `localhost:5173/centers`. The page title is "Centros de formación". It displays a table of center data with columns: Código de centro, Departamento, Municipio, Nombre del centro, and Acciones (with edit and delete icons). Below the table, there is a form to edit a center:

Código de centro	Departamento	Municipio	Nombre del centro	Acciones
9123	Tolima	977	CENTRO AGROPECUARIO LA GRANJA	
9310	Tolima	962	CENTRO DE COMERCIO Y SERVICIOS	
9226	Tolima	962	CENTRO INDUSTRIA Y CONSTRUCCION	
9209	D. C. Santa Fe de Bogotá	149	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA	
9210	D. C. Santa Fe de Bogotá	149	CENTRO DE ELECTRICIDAD Y ELECTRONICA	
9127	Antioquia	45	Centro de Formación Profesional Minero Ambiental	

Below the table, there is a form to edit a center:

Código:

Departamento:

Municipio:

Nombre Centro:

Fase 2: Funcionalidad al botón de actualizar.

Definir función para cambiar el texto del botón después de actualizar:

```

const CrudCenters = () => {
  const updateTextButton = (texto) => { // Función que actualiza el texto del botón del formulario
    setButtonForm(texto)
  }

  return (
    <>
      <h1>Centros de formación</h1>
      <table>
        <thead>
          <tr>
            <th>Código de centro</th>
            <th>Departamento</th>
            <th>Municipio</th>
            <th>Nombre del centro</th>
            <th>Acciones</th>
          </tr>
        </thead>
        <tbody>
          <tr key={center.id}>
            <td>{center.codigo_centro}</td>
            <td>{center.deptos.DepNombre}</td>
            <td>{center.id_municipio}</td>
            <td>{center.nombre_centro}</td>
            <td>
              <span className="btn btn-primary" onClick={()=>getCenter(center.id)}><i className="fa-solid fa-pen-to-square"></i></span>
              <span className="btn btn-danger m-1"><i className="fa-solid fa-pen-to-square"></i></span>
            </td>
          </tr>
        </tbody>
      </table>
      <hr />
      <FormCenters buttonForm={buttonForm} URI_CENTERS={URI_CENTERS} centerParaForm={centerParaForm} updateTextButton={updateTextButton} />
    </>
  )
}

```

Recibir elemento en componente FormCenters:

```

const FormCenters = ({ buttonForm, URI_CENTERS, centerParaForm, updateTextButton }) => {
  //props para los campos del formulario
  const [codigoCentro, setCodigoCentro] = useState('')
  const [departamento, setDepartamento] = useState('')
  const [municipio, setMunicipio] = useState('')
  const [nombreCentro, setNombreCentro] = useState('')

  //prop para departamentos
  const [datosDepartamentos, setDatosDepartamentos] = useState([])

  //prop para municipios
  const [municipiosPorDepto, setMunicipiosPorDepto] = useState([])

  //función para obtener los departamentos, con el fin de cargarlos en el select de departamentos
  const getDeptos = async () => {
    const deptos = await axios.get(URI_DEPTOS)
    // console.log(deptos.data)
    setDatosDepartamentos(deptos.data)
  }
}

```

Editar función que escucha el envío del formulario:

```

const FormCenters = ({ buttonForm, URI_CENTERS, centerParaForm, updateTextButton }) => {
  ...
  const sendForm = (e) => {
    e.preventDefault()
    if (buttonForm === 'Actualizar') {
      console.log('actualizando ando...')
      // Aquí va el código para actualizar
      axios.put(URI_CENTERS + centerParaForm.id, {
        codigo_centro: codigoCentro,
        idDepto: departamento,
        id_municipio: municipio,
        nombre_centro: nombreCentro
      })
    } else if (buttonForm === 'Enviar') {
      console.log('guardando ando...')
      // Aquí va el código para guardar
      axios.post(URI_CENTERS, {
        codigo_centro: codigoCentro,
        idDepto: departamento,
        id_municipio: municipio,
        nombre_centro: nombreCentro
      })
    }
    clearForm() // Limpiar el formulario
  }
  updateTextButton('Enviar') // Cambiar el texto del botón
}

```

Sin embargo, al realizar la actualización los datos no se refrescan automáticamente, para verlos es necesario actualizar la página:

Antes de actualizar:

Código de centro	Departamento	Municipio	Nombre del centro	Acciones
9123	Tolima	977	CENTRO AGROPECUARIO LA GRANJA	
9310	Tolima	962	CENTRO DE COMERCIO Y SERVICIOS	
9226	Tolima	962	CENTRO INDUSTRIA Y CONSTRUCCION	
9209	D. C. Santa Fe de Bogotá 149		CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA	
9210	D. C. Santa Fe de Bogotá 149		CENTRO DE ELECTRICIDAD Y ELECTRONICA	
9127	Antioquia	45	CENTRO DE Formación Profesional Minero Ambiental	

Código:
 Departamento:
 Municipio:
 Nombre Centro:

Después de actualizar y refrescar la página:

Código de centro	Departamento	Municipio	Nombre del centro	Acciones
9123	Tolima	977	CENTRO AGROPECUARIO LA GRANJA	
9310	Tolima	962	CENTRO DE COMERCIO Y SERVICIOS	
9226	Tolima	962	CENTRO INDUSTRIA Y CONSTRUCCION	
9209	D. C. Santa Fe de Bogotá	149	CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA	
9210	D. C. Santa Fe de Bogotá	149	CENTRO DE ELECTRICIDAD Y ELECTRONICA	
9127	Antioquia	45	CENTRO DE FORMACIÓN Profesional Minero Ambiental	

Código:
 Departamento: Selección uno..
 Municipio: Selección uno...
 Nombre Centro:

↑
Cambio de minúsculas a mayúsculas

Para que la información se muestre apenas se realiza la actualización, solamente se agrega la dependencia en el useEffect (centers es la variable que contiene el listado de los centros de formación).

```

EXPLORER          ...  Players.js  formQueryPlayer.jsx M  JS centerModel.js  formCenters.jsx M  crudCenters.jsx M  JS app.js  JS deptosModel.js  ⌂ ...
VITE-REACT-M...  crudCenters.jsx  torneo-sena-mysql > src > centers > crudCenters.jsx > CrudCenters
  node
  routes
    JS app.js
    package-lock.json
    package.json
  torneo-sena-mysql
    node_modules
    public
  src
    assets
    centers
      crudCenters.jsx M
      formCenters.jsx M
    players
      crudPlayers.jsx M
      formPlayers.jsx
      formQueryPlayer.j... M
    App.jsx
    main.jsx
    .eslintrc.js
    .gitignore
    index.html
  8 const CrudCenters = () => {
  23
  24
  25
  26
  27
  28
  29
  30
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  41
  42
  43
  44
  45
  46
  47
  const getAllCenters = async () => {
    const response = await axios.get(URI_CENTERS)
    setCenters(response.data)
  }
  useEffect(() => {
    getAllCenters()
  }, [centers]) //Agregar elemento al useEffect para que se ejecute la función getAllCenters después de actualizar un registro.
  //Función para obtener el centro de formación que se va a editar
  const getCenter = async (idCenter) => {
    // console.log(idCenter)
    setButtonForm('Enviar')
    const response = await axios.get(URI_CENTERS + idCenter)
    setButtonForm('Actualizar')
    // console.log(response.data)
    setCenterParaForm({ ...response.data }) //Llenar la prop centerParaForm con los datos y así mostrarlos en el form
}
  
```

Fase 3: Funcionalidad al botón de borrar.

Importar sweet alert en crudCenters.jsx:

```

EXPLORER ... JS centerModel.js ❁ formCenters.jsx M crudCenters.jsx M X JS app.js JS deptosModel.js crudPlayers.jsx M
VITE-REACT-M... [+] ⏪ ⏴ ⏵
  node
    routes
      JS app.js
      {} package-lock.json
      {} package.json
  torneo-sena-mysql
    node_modules
    public
    src
      assets
      centers
        crudCenters.jsx M
        formCenters.jsx M
        home
        players
          crudPlayers.jsx M
          formPlayers.jsx
          formQueryPlayer.j... M
          App.jsx
          main.jsx
        .eslintrc.cjs
        .gitignore
        index.html
        {} package-lock.json
        {} package.json
        README.md
      > OUTLINE
      > TIMELINE

```

```

1 import axios from "axios"
2 import { useState, useEffect } from "react"
3
4 import FormCenters from "./formCenters"
5
6 import Swal from "weetalert2" ← Red arrow
7
8 const URI_CENTERS = 'http://localhost:8000/centers/'
9
10 const CrudCenters = () => {
11
12   const [centers, setCenters] = useState([])
13
14   //Definir el texto del botón del formulario el cual varía entre 'Enviar' y 'Actualizar'
15   const [buttonForm, setButtonForm] = useState('Enviar')
16
17   // Definir prop para la edición de un centro de formación
18   const [centerParaForm, setCenterParaForm] = useState({
19     id: '',
20     codigo_centro: '',
21     idDepto: '',
22     id_municipio: '',
23     nombre_centro: ''
24   })
25
26   const getAllCenters = async () => {
27
28     const response = await axios.get(URI_CENTERS)
29
30     setCenters(response.data)
31
32   }

```

Crear función para borrar el centro de formación (incluye alerta para confirmar la eliminación del registro):

```

EXPLORER ... JS centerModel.js ❁ formCenters.jsx M crudCenters.jsx M X JS app.js JS deptosModel.js crudPlayers.jsx M
VITE-REACT-M... [+] ⏪ ⏴ ⏵
  node
    routes
      JS app.js
      {} package-lock.json
      {} package.json
  torneo-sena-mysql
    node_modules
    public
    src
      assets
      centers
        crudCenters.jsx M
        formCenters.jsx M
        home
        players
          crudPlayers.jsx M
          formPlayers.jsx
          formQueryPlayer.j... M
          App.jsx
          main.jsx
        .eslintrc.cjs
        .gitignore
        index.html
        {} package-lock.json
        {} package.json
        README.md
      > OUTLINE
      > TIMELINE

```

```

10 const CrudCenters = () => {
11   const getCenter = async (idCenter) => {
12
13     const updateTextButton = (texto) => { // Función que actualiza el texto del botón del formulario
14       setButtonForm(texto)
15     }
16
17     //Uso de sweet alert
18     const deleteCenter = (idCenter) => {
19       Swal.fire({
20         title: "Estás seguro?",
21         text: "No podrás revertir esto!",
22         icon: "warning",
23         showCancelButton: true,
24         confirmButtonColor: "#3085d6",
25         cancelButtonColor: "#d33",
26         confirmButtonText: "Sí, borrar!"
27       }).then(async (result) => {
28         if (result.isConfirmed) {
29           await axios.delete(URI_CENTERS + idCenter) ← Red box
30           Swal.fire({
31             title: "Borrado!",
32             text: "El registro ha sido borrado.",
33             icon: "success"
34           });
35         }
36       });
37
38     return (
39       <>
40     );
41   }
42
43   const getAllCenters = async () => {
44     const response = await axios.get(URI_CENTERS)
45     setCenters(response.data)
46   }
47
48   const updateCenter = (center) => {
49     setCenterParaForm(center)
50   }
51
52   const handleDelete = (idCenter) => {
53     deleteCenter(idCenter);
54   }
55
56   const handleEdit = (center) => {
57     updateCenter(center);
58   }
59
60   const handleUpdate = (buttonText) => {
61     updateTextButton(buttonText);
62   }
63
64   const handleGetAllCenters = () => {
65     getAllCenters();
66   }
67
68   const handleDeleteCenter = (idCenter) => {
69     deleteCenter(idCenter);
70   }
71
72   const handleEditCenter = (center) => {
73     updateCenter(center);
74   }
75
76   const handleUpdateTextButton = (buttonText) => {
77     updateTextButton(buttonText);
78   }
79
80   const handleGetAllCenters = () => {
81     getAllCenters();
82   }
83
84   const handleDeleteCenter = (idCenter) => {
85     deleteCenter(idCenter);
86   }
87
88   const handleEditCenter = (center) => {
89     updateCenter(center);
90   }
91
92   const handleUpdateTextButton = (buttonText) => {
93     updateTextButton(buttonText);
94   }
95
96   const handleGetAllCenters = () => {
97     getAllCenters();
98   }
99
100  const handleDeleteCenter = (idCenter) => {
101    deleteCenter(idCenter);
102  }
103
104  const handleEditCenter = (center) => {
105    updateCenter(center);
106  }
107
108  const handleUpdateTextButton = (buttonText) => {
109    updateTextButton(buttonText);
110  }
111
112  const handleGetAllCenters = () => {
113    getAllCenters();
114  }
115
116  const handleDeleteCenter = (idCenter) => {
117    deleteCenter(idCenter);
118  }
119
120  const handleEditCenter = (center) => {
121    updateCenter(center);
122  }
123
124  const handleUpdateTextButton = (buttonText) => {
125    updateTextButton(buttonText);
126  }
127
128  const handleGetAllCenters = () => {
129    getAllCenters();
130  }
131
132  const handleDeleteCenter = (idCenter) => {
133    deleteCenter(idCenter);
134  }
135
136  const handleEditCenter = (center) => {
137    updateCenter(center);
138  }
139
140  const handleUpdateTextButton = (buttonText) => {
141    updateTextButton(buttonText);
142  }
143
144  const handleGetAllCenters = () => {
145    getAllCenters();
146  }
147
148  const handleDeleteCenter = (idCenter) => {
149    deleteCenter(idCenter);
150  }
151
152  const handleEditCenter = (center) => {
153    updateCenter(center);
154  }
155
156  const handleUpdateTextButton = (buttonText) => {
157    updateTextButton(buttonText);
158  }
159
160  const handleGetAllCenters = () => {
161    getAllCenters();
162  }
163
164  const handleDeleteCenter = (idCenter) => {
165    deleteCenter(idCenter);
166  }
167
168  const handleEditCenter = (center) => {
169    updateCenter(center);
170  }
171
172  const handleUpdateTextButton = (buttonText) => {
173    updateTextButton(buttonText);
174  }
175
176  const handleGetAllCenters = () => {
177    getAllCenters();
178  }
179
180  const handleDeleteCenter = (idCenter) => {
181    deleteCenter(idCenter);
182  }
183
184  const handleEditCenter = (center) => {
185    updateCenter(center);
186  }
187
188  const handleUpdateTextButton = (buttonText) => {
189    updateTextButton(buttonText);
190  }
191
192  const handleGetAllCenters = () => {
193    getAllCenters();
194  }
195
196  const handleDeleteCenter = (idCenter) => {
197    deleteCenter(idCenter);
198  }
199
200  const handleEditCenter = (center) => {
201    updateCenter(center);
202  }
203
204  const handleUpdateTextButton = (buttonText) => {
205    updateTextButton(buttonText);
206  }
207
208  const handleGetAllCenters = () => {
209    getAllCenters();
210  }
211
212  const handleDeleteCenter = (idCenter) => {
213    deleteCenter(idCenter);
214  }
215
216  const handleEditCenter = (center) => {
217    updateCenter(center);
218  }
219
220  const handleUpdateTextButton = (buttonText) => {
221    updateTextButton(buttonText);
222  }
223
224  const handleGetAllCenters = () => {
225    getAllCenters();
226  }
227
228  const handleDeleteCenter = (idCenter) => {
229    deleteCenter(idCenter);
230  }
231
232  const handleEditCenter = (center) => {
233    updateCenter(center);
234  }
235
236  const handleUpdateTextButton = (buttonText) => {
237    updateTextButton(buttonText);
238  }
239
240  const handleGetAllCenters = () => {
241    getAllCenters();
242  }
243
244  const handleDeleteCenter = (idCenter) => {
245    deleteCenter(idCenter);
246  }
247
248  const handleEditCenter = (center) => {
249    updateCenter(center);
250  }
251
252  const handleUpdateTextButton = (buttonText) => {
253    updateTextButton(buttonText);
254  }
255
256  const handleGetAllCenters = () => {
257    getAllCenters();
258  }
259
260  const handleDeleteCenter = (idCenter) => {
261    deleteCenter(idCenter);
262  }
263
264  const handleEditCenter = (center) => {
265    updateCenter(center);
266  }
267
268  const handleUpdateTextButton = (buttonText) => {
269    updateTextButton(buttonText);
270  }
271
272  const handleGetAllCenters = () => {
273    getAllCenters();
274  }
275
276  const handleDeleteCenter = (idCenter) => {
277    deleteCenter(idCenter);
278  }
279
280  const handleEditCenter = (center) => {
281    updateCenter(center);
282  }
283
284  const handleUpdateTextButton = (buttonText) => {
285    updateTextButton(buttonText);
286  }
287
288  const handleGetAllCenters = () => {
289    getAllCenters();
290  }
291
292  const handleDeleteCenter = (idCenter) => {
293    deleteCenter(idCenter);
294  }
295
296  const handleEditCenter = (center) => {
297    updateCenter(center);
298  }
299
300  const handleUpdateTextButton = (buttonText) => {
301    updateTextButton(buttonText);
302  }
303
304  const handleGetAllCenters = () => {
305    getAllCenters();
306  }
307
308  const handleDeleteCenter = (idCenter) => {
309    deleteCenter(idCenter);
310  }
311
312  const handleEditCenter = (center) => {
313    updateCenter(center);
314  }
315
316  const handleUpdateTextButton = (buttonText) => {
317    updateTextButton(buttonText);
318  }
319
320  const handleGetAllCenters = () => {
321    getAllCenters();
322  }
323
324  const handleDeleteCenter = (idCenter) => {
325    deleteCenter(idCenter);
326  }
327
328  const handleEditCenter = (center) => {
329    updateCenter(center);
330  }
331
332  const handleUpdateTextButton = (buttonText) => {
333    updateTextButton(buttonText);
334  }
335
336  const handleGetAllCenters = () => {
337    getAllCenters();
338  }
339
340  const handleDeleteCenter = (idCenter) => {
341    deleteCenter(idCenter);
342  }
343
344  const handleEditCenter = (center) => {
345    updateCenter(center);
346  }
347
348  const handleUpdateTextButton = (buttonText) => {
349    updateTextButton(buttonText);
350  }
351
352  const handleGetAllCenters = () => {
353    getAllCenters();
354  }
355
356  const handleDeleteCenter = (idCenter) => {
357    deleteCenter(idCenter);
358  }
359
360  const handleEditCenter = (center) => {
361    updateCenter(center);
362  }
363
364  const handleUpdateTextButton = (buttonText) => {
365    updateTextButton(buttonText);
366  }
367
368  const handleGetAllCenters = () => {
369    getAllCenters();
370  }
371
372  const handleDeleteCenter = (idCenter) => {
373    deleteCenter(idCenter);
374  }
375
376  const handleEditCenter = (center) => {
377    updateCenter(center);
378  }
379
380  const handleUpdateTextButton = (buttonText) => {
381    updateTextButton(buttonText);
382  }
383
384  const handleGetAllCenters = () => {
385    getAllCenters();
386  }
387
388  const handleDeleteCenter = (idCenter) => {
389    deleteCenter(idCenter);
390  }
391
392  const handleEditCenter = (center) => {
393    updateCenter(center);
394  }
395
396  const handleUpdateTextButton = (buttonText) => {
397    updateTextButton(buttonText);
398  }
399
400  const handleGetAllCenters = () => {
401    getAllCenters();
402  }
403
404  const handleDeleteCenter = (idCenter) => {
405    deleteCenter(idCenter);
406  }
407
408  const handleEditCenter = (center) => {
409    updateCenter(center);
410  }
411
412  const handleUpdateTextButton = (buttonText) => {
413    updateTextButton(buttonText);
414  }
415
416  const handleGetAllCenters = () => {
417    getAllCenters();
418  }
419
420  const handleDeleteCenter = (idCenter) => {
421    deleteCenter(idCenter);
422  }
423
424  const handleEditCenter = (center) => {
425    updateCenter(center);
426  }
427
428  const handleUpdateTextButton = (buttonText) => {
429    updateTextButton(buttonText);
430  }
431
432  const handleGetAllCenters = () => {
433    getAllCenters();
434  }
435
436  const handleDeleteCenter = (idCenter) => {
437    deleteCenter(idCenter);
438  }
439
440  const handleEditCenter = (center) => {
441    updateCenter(center);
442  }
443
444  const handleUpdateTextButton = (buttonText) => {
445    updateTextButton(buttonText);
446  }
447
448  const handleGetAllCenters = () => {
449    getAllCenters();
450  }
451
452  const handleDeleteCenter = (idCenter) => {
453    deleteCenter(idCenter);
454  }
455
456  const handleEditCenter = (center) => {
457    updateCenter(center);
458  }
459
460  const handleUpdateTextButton = (buttonText) => {
461    updateTextButton(buttonText);
462  }
463
464  const handleGetAllCenters = () => {
465    getAllCenters();
466  }
467
468  const handleDeleteCenter = (idCenter) => {
469    deleteCenter(idCenter);
470  }
471
472  const handleEditCenter = (center) => {
473    updateCenter(center);
474  }
475
476  const handleUpdateTextButton = (buttonText) => {
477    updateTextButton(buttonText);
478  }
479
480  const handleGetAllCenters = () => {
481    getAllCenters();
482  }
483
484  const handleDeleteCenter = (idCenter) => {
485    deleteCenter(idCenter);
486  }
487
488  const handleEditCenter = (center) => {
489    updateCenter(center);
490  }
491
492  const handleUpdateTextButton = (buttonText) => {
493    updateTextButton(buttonText);
494  }
495
496  const handleGetAllCenters = () => {
497    getAllCenters();
498  }
499
500  const handleDeleteCenter = (idCenter) => {
501    deleteCenter(idCenter);
502  }
503
504  const handleEditCenter = (center) => {
505    updateCenter(center);
506  }
507
508  const handleUpdateTextButton = (buttonText) => {
509    updateTextButton(buttonText);
510  }
511
512  const handleGetAllCenters = () => {
513    getAllCenters();
514  }
515
516  const handleDeleteCenter = (idCenter) => {
517    deleteCenter(idCenter);
518  }
519
520  const handleEditCenter = (center) => {
521    updateCenter(center);
522  }
523
524  const handleUpdateTextButton = (buttonText) => {
525    updateTextButton(buttonText);
526  }
527
528  const handleGetAllCenters = () => {
529    getAllCenters();
530  }
531
532  const handleDeleteCenter = (idCenter) => {
533    deleteCenter(idCenter);
534  }
535
536  const handleEditCenter = (center) => {
537    updateCenter(center);
538  }
539
540  const handleUpdateTextButton = (buttonText) => {
541    updateTextButton(buttonText);
542  }
543
544  const handleGetAllCenters = () => {
545    getAllCenters();
546  }
547
548  const handleDeleteCenter = (idCenter) => {
549    deleteCenter(idCenter);
550  }
551
552  const handleEditCenter = (center) => {
553    updateCenter(center);
554  }
555
556  const handleUpdateTextButton = (buttonText) => {
557    updateTextButton(buttonText);
558  }
559
560  const handleGetAllCenters = () => {
561    getAllCenters();
562  }
563
564  const handleDeleteCenter = (idCenter) => {
565    deleteCenter(idCenter);
566  }
567
568  const handleEditCenter = (center) => {
569    updateCenter(center);
570  }
571
572  const handleUpdateTextButton = (buttonText) => {
573    updateTextButton(buttonText);
574  }
575
576  const handleGetAllCenters = () => {
577    getAllCenters();
578  }
579
580  const handleDeleteCenter = (idCenter) => {
581    deleteCenter(idCenter);
582  }
583
584  const handleEditCenter = (center) => {
585    updateCenter(center);
586  }
587
588  const handleUpdateTextButton = (buttonText) => {
589    updateTextButton(buttonText);
590  }
591
592  const handleGetAllCenters = () => {
593    getAllCenters();
594  }
595
596  const handleDeleteCenter = (idCenter) => {
597    deleteCenter(idCenter);
598  }
599
600  const handleEditCenter = (center) => {
601    updateCenter(center);
602  }
603
604  const handleUpdateTextButton = (buttonText) => {
605    updateTextButton(buttonText);
606  }
607
608  const handleGetAllCenters = () => {
609    getAllCenters();
610  }
611
612  const handleDeleteCenter = (idCenter) => {
613    deleteCenter(idCenter);
614  }
615
616  const handleEditCenter = (center) => {
617    updateCenter(center);
618  }
619
620  const handleUpdateTextButton = (buttonText) => {
621    updateTextButton(buttonText);
622  }
623
624  const handleGetAllCenters = () => {
625    getAllCenters();
626  }
627
628  const handleDeleteCenter = (idCenter) => {
629    deleteCenter(idCenter);
630  }
631
632  const handleEditCenter = (center) => {
633    updateCenter(center);
634  }
635
636  const handleUpdateTextButton = (buttonText) => {
637    updateTextButton(buttonText);
638  }
639
640  const handleGetAllCenters = () => {
641    getAllCenters();
642  }
643
644  const handleDeleteCenter = (idCenter) => {
645    deleteCenter(idCenter);
646  }
647
648  const handleEditCenter = (center) => {
649    updateCenter(center);
650  }
651
652  const handleUpdateTextButton = (buttonText) => {
653    updateTextButton(buttonText);
654  }
655
656  const handleGetAllCenters = () => {
657    getAllCenters();
658  }
659
660  const handleDeleteCenter = (idCenter) => {
661    deleteCenter(idCenter);
662  }
663
664  const handleEditCenter = (center) => {
665    updateCenter(center);
666  }
667
668  const handleUpdateTextButton = (buttonText) => {
669    updateTextButton(buttonText);
670  }
671
672  const handleGetAllCenters = () => {
673    getAllCenters();
674  }
675
676  const handleDeleteCenter = (idCenter) => {
677    deleteCenter(idCenter);
678  }
679
680  const handleEditCenter = (center) => {
681    updateCenter(center);
682  }
683
684  const handleUpdateTextButton = (buttonText) => {
685    updateTextButton(buttonText);
686  }
687
688  const handleGetAllCenters = () => {
689    getAllCenters();
690  }
691
692  const handleDeleteCenter = (idCenter) => {
693    deleteCenter(idCenter);
694  }
695
696  const handleEditCenter = (center) => {
697    updateCenter(center);
698  }
699
700  const handleUpdateTextButton = (buttonText) => {
701    updateTextButton(buttonText);
702  }
703
704  const handleGetAllCenters = () => {
705    getAllCenters();
706  }
707
708  const handleDeleteCenter = (idCenter) => {
709    deleteCenter(idCenter);
710  }
711
712  const handleEditCenter = (center) => {
713    updateCenter(center);
714  }
715
716  const handleUpdateTextButton = (buttonText) => {
717    updateTextButton(buttonText);
718  }
719
720  const handleGetAllCenters = () => {
721    getAllCenters();
722  }
723
724  const handleDeleteCenter = (idCenter) => {
725    deleteCenter(idCenter);
726  }
727
728  const handleEditCenter = (center) => {
729    updateCenter(center);
730  }
731
732  const handleUpdateTextButton = (buttonText) => {
733    updateTextButton(buttonText);
734  }
735
736  const handleGetAllCenters = () => {
737    getAllCenters();
738  }
739
740  const handleDeleteCenter = (idCenter) => {
741    deleteCenter(idCenter);
742  }
743
744  const handleEditCenter = (center) => {
745    updateCenter(center);
746  }
747
748  const handleUpdateTextButton = (buttonText) => {
749    updateTextButton(buttonText);
750  }
751
752  const handleGetAllCenters = () => {
753    getAllCenters();
754  }
755
756  const handleDeleteCenter = (idCenter) => {
757    deleteCenter(idCenter);
758  }
759
760  const handleEditCenter = (center) => {
761    updateCenter(center);
762  }
763
764  const handleUpdateTextButton = (buttonText) => {
765    updateTextButton(buttonText);
766  }
767
768  const handleGetAllCenters = () => {
769    getAllCenters();
770  }
771
772  const handleDeleteCenter = (idCenter) => {
773    deleteCenter(idCenter);
774  }
775
776  const handleEditCenter = (center) => {
777    updateCenter(center);
778  }
779
780  const handleUpdateTextButton = (buttonText) => {
781    updateTextButton(buttonText);
782  }
783
784  const handleGetAllCenters = () => {
785    getAllCenters();
786  }
787
788  const handleDeleteCenter = (idCenter) => {
789    deleteCenter(idCenter);
790  }
791
792  const handleEditCenter = (center) => {
793    updateCenter(center);
794  }
795
796  const handleUpdateTextButton = (buttonText) => {
797    updateTextButton(buttonText);
798  }
799
800  const handleGetAllCenters = () => {
801    getAllCenters();
802  }
803
804  const handleDeleteCenter = (idCenter) => {
805    deleteCenter(idCenter);
806  }
807
808  const handleEditCenter = (center) => {
809    updateCenter(center);
810  }
811
812  const handleUpdateTextButton = (buttonText) => {
813    updateTextButton(buttonText);
814  }
815
816  const handleGetAllCenters = () => {
817    getAllCenters();
818  }
819
820  const handleDeleteCenter = (idCenter) => {
821    deleteCenter(idCenter);
822  }
823
824  const handleEditCenter = (center) => {
825    updateCenter(center);
826  }
827
828  const handleUpdateTextButton = (buttonText) => {
829    updateTextButton(buttonText);
830  }
831
832  const handleGetAllCenters = () => {
833    getAllCenters();
834  }
835
836  const handleDeleteCenter = (idCenter) => {
837    deleteCenter(idCenter);
838  }
839
840  const handleEditCenter = (center) => {
841    updateCenter(center);
842  }
843
844  const handleUpdateTextButton = (buttonText) => {
845    updateTextButton(buttonText);
846  }
847
848  const handleGetAllCenters = () => {
849    getAllCenters();
850  }
851
852  const handleDeleteCenter = (idCenter) => {
853    deleteCenter(idCenter);
854  }
855
856  const handleEditCenter = (center) => {
857    updateCenter(center);
858  }
859
860  const handleUpdateTextButton = (buttonText) => {
861    updateTextButton(buttonText);
862  }
863
864  const handleGetAllCenters = () => {
865    getAllCenters();
866  }
867
868  const handleDeleteCenter = (idCenter) => {
869    deleteCenter(idCenter);
870  }
871
872  const handleEditCenter = (center) => {
873    updateCenter(center);
874  }
875
876  const handleUpdateTextButton = (buttonText) => {
877    updateTextButton(buttonText);
878  }
879
880  const handleGetAllCenters = () => {
881    getAllCenters();
882  }
883
884  const handleDeleteCenter = (idCenter) => {
885    deleteCenter(idCenter);
886  }
887
888  const handleEditCenter = (center) => {
889    updateCenter(center);
890  }
891
892  const handleUpdateTextButton = (buttonText) => {
893    updateTextButton(buttonText);
894  }
895
896  const handleGetAllCenters = () => {
897    getAllCenters();
898  }
899
900  const handleDeleteCenter = (idCenter) => {
901    deleteCenter(idCenter);
902  }
903
904  const handleEditCenter = (center) => {
905    updateCenter(center);
906  }
907
908  const handleUpdateTextButton = (buttonText) => {
909    updateTextButton(buttonText);
910  }
911
912  const handleGetAllCenters = () => {
913    getAllCenters();
914  }
915
916  const handleDeleteCenter = (idCenter) => {
917    deleteCenter(idCenter);
918  }
919
920  const handleEditCenter = (center) => {
921    updateCenter(center);
922  }
923
924  const handleUpdateTextButton = (buttonText) => {
925    updateTextButton(buttonText);
926  }
927
928  const handleGetAllCenters = () => {
929    getAllCenters();
930  }
931
932  const handleDeleteCenter = (idCenter) => {
933    deleteCenter(idCenter);
934  }
935
936  const handleEditCenter = (center) => {
937    updateCenter(center);
938  }
939
940  const handleUpdateTextButton = (buttonText) => {
941    updateTextButton(buttonText);
942  }
943
944  const handleGetAllCenters = () => {
945    getAllCenters();
946  }
947
948  const handleDeleteCenter = (idCenter) => {
949    deleteCenter(idCenter);
950  }
951
952  const handleEditCenter = (center) => {
953    updateCenter(center);
954  }
955
956  const handleUpdateTextButton = (buttonText) => {
957    updateTextButton(buttonText);
958  }
959
960  const handleGetAllCenters = () => {
961    getAllCenters();
962  }
963
964  const handleDeleteCenter = (idCenter) => {
965    deleteCenter(idCenter);
966  }
967
968  const handleEditCenter = (center) => {
969    updateCenter(center);
970  }
971
972  const handleUpdateTextButton = (buttonText) => {
973    updateTextButton(buttonText);
974  }
975
976  const handleGetAllCenters = () => {
977    getAllCenters();
978  }
979
980  const handleDeleteCenter = (idCenter) => {
981    deleteCenter(idCenter);
982  }
983
984  const handleEditCenter = (center) => {
985    updateCenter(center);
986  }
987
988  const handleUpdateTextButton = (buttonText) => {
989    updateTextButton(buttonText);
990  }
991
992  const handleGetAllCenters = () => {
993    getAllCenters();
994  }
995
996  const handleDeleteCenter = (idCenter) => {
997    deleteCenter(idCenter);
998  }
999
1000 const handleEditCenter = (center) => {
1001   updateCenter(center);
1002 }
1003
1004 const handleUpdateTextButton = (buttonText) => {
1005   updateTextButton(buttonText);
1006 }
1007
1008 const handleGetAllCenters = () => {
1009   getAllCenters();
1010 }
1011
1012 const handleDeleteCenter = (idCenter) => {
1013   deleteCenter(idCenter);
1014 }
1015
1016 const handleEditCenter = (center) => {
1017   updateCenter(center);
1018 }
1019
1020 const handleUpdateTextButton = (buttonText) => {
1021   updateTextButton(buttonText);
1022 }
1023
1024 const handleGetAllCenters = () => {
1025   getAllCenters();
1026 }
1027
1028 const handleDeleteCenter = (idCenter) => {
1029   deleteCenter(idCenter);
1030 }
1031
1032 const handleEditCenter = (center) => {
1033   updateCenter(center);
1034 }
1035
1036 const handleUpdateTextButton = (buttonText) => {
1037   updateTextButton(buttonText);
1038 }
1039
1040 const handleGetAllCenters = () => {
1041   getAllCenters();
1042 }
1043
1044 const handleDeleteCenter = (idCenter) => {
1045   deleteCenter(idCenter);
1046 }
1047
1048 const handleEditCenter = (center) => {
1049   updateCenter(center);
1050 }
1051
1052 const handleUpdateTextButton = (buttonText) => {
1053   updateTextButton(buttonText);
1054 }
1055
1056 const handleGetAllCenters = () => {
1057   getAllCenters();
1058 }
1059
1060 const handleDeleteCenter = (idCenter) => {
1061   deleteCenter(idCenter);
1062 }
1063
1064 const handleEditCenter = (center) => {
1065   updateCenter(center);
1066 }
1067
1068 const handleUpdateTextButton = (buttonText) => {
1069   updateTextButton(buttonText);
1070 }
1071
1072 const handleGetAllCenters = () => {
1073   getAllCenters();
1074 }
1075
1076 const handleDeleteCenter = (idCenter) => {
1077   deleteCenter(idCenter);
1078 }
1079
1080 const handleEditCenter = (center) => {
1081   updateCenter(center);
1082 }
1083
1084 const handleUpdateTextButton = (buttonText) => {
1085   updateTextButton(buttonText);
1086 }
1087
1088 const handleGetAllCenters = () => {
1089   getAllCenters();
1090 }
1091
1092 const handleDeleteCenter = (idCenter) => {
1093   deleteCenter(idCenter);
1094 }
1095
1096 const handleEditCenter = (center) => {
1097   updateCenter(center);
1098 }
1099
1100 const handleUpdateTextButton = (buttonText) => {
1101   updateTextButton(buttonText);
1102 }
1103
1104 const handleGetAllCenters = () => {
1105   getAllCenters();
1106 }
1107
1108 const handleDeleteCenter = (idCenter) => {
1109   deleteCenter(idCenter);
1110 }
1111
1112 const handleEditCenter = (center) => {
1113   updateCenter(center);
1114 }
1115
1116 const handleUpdateTextButton = (buttonText) => {
1117   updateTextButton(buttonText);
1118 }
1119
1120 const handleGetAllCenters = () => {
1121   getAllCenters();
1122 }
1123
1124 const handleDeleteCenter = (idCenter) => {
1125   deleteCenter(idCenter);
1126 }
1127
1128 const handleEditCenter = (center) => {
1129   updateCenter(center);
1130 }
1131
1132 const handleUpdateTextButton = (buttonText) => {
1133   updateTextButton(buttonText);
1134 }
1135
1136 const handleGetAllCenters = () => {
1137   getAllCenters();
1138 }
1139
1140 const handleDeleteCenter = (idCenter) => {
1141   deleteCenter(idCenter);
1142 }
1143
1144 const handleEditCenter = (center) => {
1145   updateCenter(center);
1146 }
1147
1148 const handleUpdateTextButton = (buttonText) => {
1149   updateTextButton(buttonText);
1150 }
1151
1152 const handleGetAllCenters = () => {
1153   getAllCenters();
1154 }
1155
1156 const handleDeleteCenter = (idCenter) => {
1157   deleteCenter(idCenter);
1158 }
1159
1160 const handleEditCenter = (center) => {
1161   updateCenter(center);
1162 }
1163
1164 const handleUpdateTextButton = (buttonText) => {
1165   updateTextButton(buttonText);
1166 }
1167
1168 const handleGetAllCenters = () => {
1169   getAllCenters();
1170 }
1171
1172 const handleDeleteCenter = (idCenter) => {
1173   deleteCenter(idCenter);
1174 }
1175
1176 const handleEditCenter = (center) => {
1177   updateCenter(center);
1178 }
1179
1180 const handleUpdateTextButton = (buttonText) => {
1181   updateTextButton(buttonText);
1182 }
1183
1184 const handleGetAllCenters = () => {
1185   getAllCenters();
1186 }
1187
1188 const handleDeleteCenter = (idCenter) => {
1189   deleteCenter(idCenter);
1190 }
1191
1192 const handleEditCenter = (center) => {
1193   updateCenter(center);
1194 }
1195
1196 const handleUpdateTextButton = (buttonText) => {
1197   updateTextButton(buttonText);
1198 }
1199
1200 const handleGetAllCenters = () => {
1201   getAllCenters();
1202 }
1203
1204 const handleDeleteCenter = (idCenter) => {
1205   deleteCenter(idCenter);
1206 }
1207
1208 const handleEditCenter = (center) => {
1209   updateCenter(center);
1210 }
1211
1212 const handleUpdateTextButton = (buttonText) => {
1213   updateTextButton(buttonText);
121
```

```
centerModel.js formCenters.jsx crudCenters.jsx app.js deptosModel.js crudPlayers.jsx formPlayers.jsx
VITE-REACT-M... node routes
  app.jsx
  package-lock.json
  package.json
  node_modules
  public
  assets
  centers
    crudCenters.jsx M
    formCenters.jsx M
    home
    players
    crudPlayers.jsx M
    formPlayers.jsx
    formQueryPlayer.j... M
    App.jsx
    main.jsx
    .eslintrc.js
    .gitignore
    index.html
    package-lock.json
    package.json
    README.md

  OUTLINE
  TIMELINE
```

centerModel.js formCenters.jsx crudCenters.jsx app.js deptosModel.js crudPlayers.jsx formPlayers.jsx

VITE-REACT-M... node routes

app.jsx
 package-lock.json
 package.json
 node_modules
 public
 assets
 centers

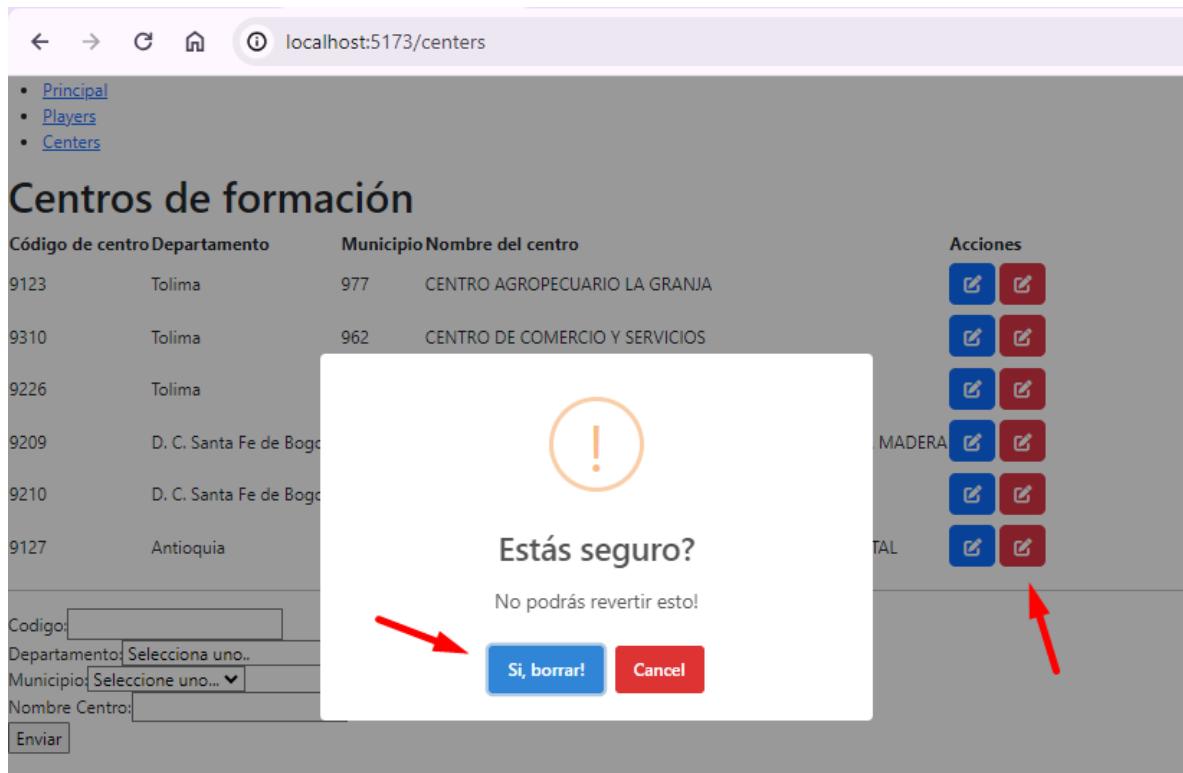
crudCenters.jsx M
 formCenters.jsx M
 home
 players
 crudPlayers.jsx M
 formPlayers.jsx
 formQueryPlayer.j... M
 App.jsx
 main.jsx
 .eslintrc.js
 .gitignore
 index.html
 package-lock.json
 package.json
 README.md

OUTLINE
 TIMELINE

```
10  const CrudCenters = () => {
11    <thead>
12      <th>Nombre del centro</th>
13      <th>Acciones</th>
14    </thead>
15    <tbody>
16      {centers.map((center) => (
17        <tr key={center.id}>
18          <td>{center.codigo_centro}</td>
19          <td>{center.deptos.DepNombre}</td>
20          <td>{center.id_municipio}</td>
21          <td>{center.nombre_centro}</td>
22          <td>
23            <span className="btn btn-primary" onClick={() => getCenter(center.id)}><i className="fa-solid fa-pen-to-square"></i></span>
24            <span className="btn btn-danger m-1" onClick={() => deleteCenter(center.id)}><i className="fa-solid fa-pen-to-square"></i></span>
25          </td>
26        </tr>
27      ))
28    </tbody>
29  </table>
30  <br />
31  <FormCenters buttonForm={buttonForm} URI_CENTERS={URI_CENTERS} centerParaForm={centerParaForm} updateTextButton={updateTextButton} />
32
33  )
34}
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
```

export default CrudCenters

Resultado:



The screenshot shows a list of centers of formation on the left and a modal dialog on the right. The modal contains a green checkmark icon, the text 'Borrado!', and the message 'El registro ha sido borrado.' A red arrow points from the top of the modal towards the center of the page.

Código de centro	Departamento	Municipio	Nombre del centro
9123	Tolima	977	CENTRO AGROPECUARIO LA GRANJA
9310	Tolima	962	CENTRO DE COMERCIO Y SERVICIOS
9226	Tolima		
9209	D. C. Santa Fe de Bogotá		
9210	D. C. Santa Fe de Bogotá		

Acciones

MADERA	

Código:
Departamento: Selección uno..
Municipio: Selección uno...
Nombre Centro:
Enviar

Borrado!
El registro ha sido borrado.
OK

Cargar archivos desde un formulario (Agregar campo ‘foto’ al crud de players)

Fase 1: En archivo formPlayers.jsx agregar campo en los lugares donde se gestionan los datos del player, fíjese que en las funciones para actualizar y agregar player hay un nuevo elemento en la petición del AXIOS (**headers**), recuerde leer los comentarios.

```

    const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
      // Agregar como parámetro el botón que llega desde el componente crudPlayers
      // Hooks para cada uno de los campos del formulario
      const [documento, setDocumento] = useState('')
      const [nombres, setNombres] = useState('')
      const [apellidos, setApellidos] = useState('')
      const [genero, setGenero] = useState('')
      const [estado, setEstado] = useState('')
      const [foto, setFoto] = useState(null) ←

      // Función que recibe los datos del formulario
      const sendForm = (e) => {
        e.preventDefault()

        if (buttonForm === 'Actualizar') {
          console.log('actualizando ando...')
          // Aquí va el código para actualizar
          axios.put(URI + player.id, {
            documento: documento,
            nombres: nombres,
            apellidos: apellidos,
            genero: genero,
            estado: estado,
            foto: foto
          })
          //Esta es una propiedad para el form que permite el envío de archivos ←
          headers: { "Content-Type": "multipart/form-data" }
        })
      }

      updateTextButton('Enviar') // Cambiar el texto del botón
      clearForm() // Limpiar el formulario
    }
  
```

```

    const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
      const sendForm = (e) => {
        if (buttonForm === 'Actualizar') {
          console.log('actualizando ando... + foto')
          // Aquí va el código para actualizar
          axios.put(URI, {
            documento: documento,
            nombres: nombres,
            apellidos: apellidos,
            genero: genero,
            estado: estado,
            foto: foto
          })
          //Esta es una propiedad para el form que permite el envío de archivos ←
          headers: { "Content-Type": "multipart/form-data" }
        })
      }

      clearForm()
    }

    const clearForm = () => {
      // Se vacía el elemento player
      setDocumento('')
      setNombres('')
      setApellidos('')
      setGenero('')
      setEstado('')
      setFoto(null) ←
    }

    const setData = () => { // Función que establece los valores a los campos para que se muestren en el formulario cuando presionen el botón editar, los datos llegan de
      setDocumento(player.documento)
      setNombres(player.nombres)
      setApellidos(player.apellidos)
      setGenero(player.genero)
      setEstado(player.estado)
      setFoto(player.foto) ←
    }
  
```

```

VITE-REACT-M... js formQueryPlayer.jsx M JS centerModel.js formCenters.jsx M crudCenters.jsx M crudPlayers.jsx M formPlayers.jsx X JS app.js JS deptosModel.js
node
routes
  routerCenters.js
  routerDeptos.js
  routerMunicipios.js
  routerPlayers.js
  app.js
  package-lock.json
  package.json
tomeo-sena-mysql
  node_modules
  public
src
  assets
  centers
    crudCenters.jsx M
    formCenters.jsx M
  home
  players
    crudPlayers.jsx M
    formPlayers.jsx M
    formQueryPlayer... M
  App.jsx
  main.jsx
  .eslintrc.js
  .gitignore
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js

```

```

const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
  // Agregar como parámetro el botón que llega desde el componente crudPlayers
  return (
    <>
      <form id="playerForm" action="" onSubmit={sendForm}>
        <label htmlFor="documento">Documento</label>
        /* Cuando se trabaja con formularios, los campos deben contar siempre con el evento onChange,
         | el cual, ejecuta la función de gestión de dicho campo */
        <input type="text" id="documento" value={documento} onChange={(e) => setDocumento(e.target.value)} />
        <br />
        <label htmlFor="nombres">Nombres</label>
        <input type="text" id="nombres" value={nombres} onChange={(e) => setNombres(e.target.value)} />
        <br />
        <label htmlFor="apellidos">Apellidos</label>
        <input type="text" id="apellidos" value={apellidos} onChange={(e) => setApellidos(e.target.value)} />
        <br />
        <label htmlFor="genero">Genero</label>
        <select name="" id="genero" value={genero} onChange={(e) => setGenero(e.target.value)}>
          <option value="">>Selecciona uno...</option>
          <option value="F">Femenino</option>
          <option value="M">Masculino</option>
          <option value="O">Otro</option>
        </select>
        <br />
        <label htmlFor="estado">Estado</label>
        <select name="" id="estado" value={estado} onChange={(e) => setEstado(e.target.value)}>
          <option value="">>Selecciona uno...</option>
          <option value="habilitado">Habilitado</option>
          <option value="desabilitado">Desabilitado</option>
        </select>
        <br />
        <label htmlFor="foto">Foto</label>
        <input type="file" id="foto" onChange={(e) => setFoto(e.target.files[0])}></input>
        <input type="submit" id="boton" value={buttonForm} className="btn btn-success" />
      </form>
    </>
  )
}

```

Fase 2: Instalar multer en el back-end (carpeta node) y configurarlo:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\vite-react-mysql> cd node
PS D:\vite-react-mysql\node> npm install multer
added 18 packages, and audited 119 packages in 3s

14 packages are looking for funding
  run `npm fund` for details

2 vulnerabilities (1 moderate, 1 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS D:\vite-react-mysql\node>

```

En el archivo de rutas de los players (routerPlayers.js) importar multer y path:

```

VITE-REACT-M...
  node > routes > routerPlayers.js ...
    1 import express from "express";
    2 import { createPlayer, deletePlayer, getAllPlayers, getPlayer, updatePlayer, getQueryPlayer } from "../controllers/playerController.js";
    3 import multer from 'multer';
    4 import path from 'path'
    5
    6 const router = express.Router();
    7
    8
    9
  10 //Configuración de multer
  11 const storage = multer.diskStorage({
  12   //Asginar la carpeta donde quedarán los archivos
  13   destination: (req, file, cb) => {
  14     //cb es una función que se utiliza para indicar el destino del archivo.
  15     cb(null, 'public/uploads/')
  16   },
  17   //Renombrar el archivo con la marca del tiempo actual en milisegundos conservando la extensión original del archivo
  18   filename: (req, file, cb) => {
  19     cb(null, Date.now() + path.extname(file.originalname))
  20   }
  21 })
  22
  23 const upload = multer({ storage })
  24
  25 router.get('/', getAllPlayers)
  26 router.get('/:id', getPlayer)
  27 router.post('/', upload.single('foto'), createPlayer) //En las rutas de post y put hay que agregar el elemento que se espera recibir
  28 router.put('/:id', upload.single('foto'), updatePlayer)
  29 router.delete('/:id', deletePlayer)
  30
  31 router.get('/documento/:documento', getQueryPlayer)
  32
  33 export default router

```

En el archivo app.js:

Se crea la carpeta 'public/uploads' automática

```

VITE-REACT-M...
  node > app.js ...
    1 import express from 'express'
    2 import cors from 'cors'
    3
    4 import db from './database/db.js'
    5 import playerRoutes from './routes/routerPlayers.js'
    6 import centerRoutes from './routes/routerCenters.js'
    7 import deptosRoutes from './routes/routerDeptos.js'
    8
    9 import municipioRoutes from './routes/routerMunicipios.js'
    10
    11 //Importar modelos para consultar con llave foranea
    12 import CenterModel from './models/centerModel.js'
    13 import DeptosModel from './models/deptosModel.js'
    14
    15 const app = express()
    16
    17 app.use(cors())
    18 app.use(express.json())
    19 app.use('/players', playerRoutes)
    20 app.use('/centers', centerRoutes)
    21 app.use('/deptos', deptosRoutes)
    22 app.use('/municipios', municipioRoutes)
    23
    24 //Configurar express para que pueda servir los archivos estaticos desde el directorio de cargas
    25 app.use('/public/uploads', express.static('public/uploads/'))
    26
    27 try {
    28   await db.authenticate()
    29   console.log("Conexión exitosa a la db")
    30 } catch (error) {
    31   console.log("Error de conexión a la db: ${error}")
    32 }
    33
    34 app.get('/', (req, res) => {
    35   res.send('Hola mundo')
    36 })
    37
    38 app.listen(8000, () => {
    39   console.log('Server Up running in http://localhost:8000')

```

Agregar el campo en playerModel.js:

EXPLORER

```

VITE-REACT-M... ller.js JS municipioController.js JS playerController.js M JS playerModel.js M X
  node > models > JS playerModel.js ...
    1 import db from "../database/db.js";
    2 import { DataTypes } from "sequelize";
    3
    4
    5 const PlayerModel = db.define('players', {
    6   documento: { type: DataTypes.NUMBER },
    7   nombres: { type: DataTypes.STRING },
    8   apellidos: { type: DataTypes.STRING },
    9   genero: { type: DataTypes.CHAR },
   10  estado: { type: DataTypes.CHAR },
   11  foto: { type: DataTypes.STRING } ←
   12 })
   13
   14 export default PlayerModel

```

node
controllers
JS centerController.js
JS deptoController.js
JS municipioController.js
JS playerController.js M
database
models
JS centerModel.js
JS deptosModel.js
JS municipioModel.js
JS playerModel.js M
node_modules
public
routes

En el controlador playerController.js modificar la función createPlayer:

EXPLORER

```

VITE-REACT-M... ller.js JS municipioController.js JS playerController.js M X JS playerModel.js M JS routerPlayers.js M JS app.js M
  node > controllers > JS playerController.js ...
    26 //Crear un player
    27 export const createPlayer = async (req, res) => {
    28   try {
    29     //Cambia la manera de leer los datos que llegan del formulario
    30     const { documento, nombres, apellidos, genero } = req.body
    31     //Operador ternario, si el archivo existe se toma su nombre, si no, se establece como null
    32     const foto = req.file ? req.file.filename : null
    33
    34
    35     await PlayerModel.create({
    36       documento,
    37       nombres,
    38       apellidos,
    39       genero,
    40       foto
    41     })
    42     res.json({ "message": "¡Registro creado exitosamente!" })
    43
    44   } catch (error) {
    45     res.json({ message: error.message })
    46   }
    47 }
    48

```

node
controllers
JS centerController.js
JS deptoController.js
JS municipioController.js
JS playerController.js M
database
models
JS centerModel.js
JS deptosModel.js
JS municipioModel.js
JS playerModel.js M
node_modules
public
routes
JS routerCenters.js
JS routerDeptos.js
JS routerMunicipios.js
JS routerPlayers.js M
app.js M

Fase 3: Agregar el campo en la base de datos y probar con el Thunder Client:

Servidor: 127.0.0.1 > Base de datos: torneo_react > Tabla: players

Estructura de tabla

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
2	documento	int(11)			No	Ninguna		Cambiar Eliminar Más	
3	nombres	varchar(50)	utf8mb4_general_ci		No	Ninguna		Cambiar Eliminar Más	
4	apellidos	varchar(50)	utf8mb4_general_ci		No	Ninguna		Cambiar Eliminar Más	
5	genero	enum('F', 'M', 'O')	utf8mb4_general_ci		No	Ninguna		Cambiar Eliminar Más	
6	id_ficha	int(11)			No	Ninguna		Cambiar Eliminar Más	
7	foto	varchar(255)	utf8mb4_general_ci	Sí	NULL			Cambiar Eliminar Más	
8	estado	enum('habilitado', 'deshabilitado')	utf8mb4_general_ci		No	habilitado		Cambiar Eliminar Más	
9	createdat	timestamp			No	current_timestamp()		Cambiar Eliminar Más	
10	updatedat	timestamp			No	current_timestamp()		Cambiar Eliminar Más	

↑ Seleccionar todo Para los elementos que están marcados: Examinar Cambiar Eliminar Primaria Único Índice Espacial

Texto completo

Hacer la prueba con el Thunder Client, y en lugar de seleccionar JSON, seleccionar Form para obtener la opción de cargar un archivo:

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

POST localhost:8000/player... 17 hours ago

1. POST 2. Body 3. Form 4. foto 5. Send

Status: Size: Time: Response Headers Cookies Results Docs

Free Version Terms

The free version is for individual use only. Please refer to our terms.

Learn more

Shortcuts

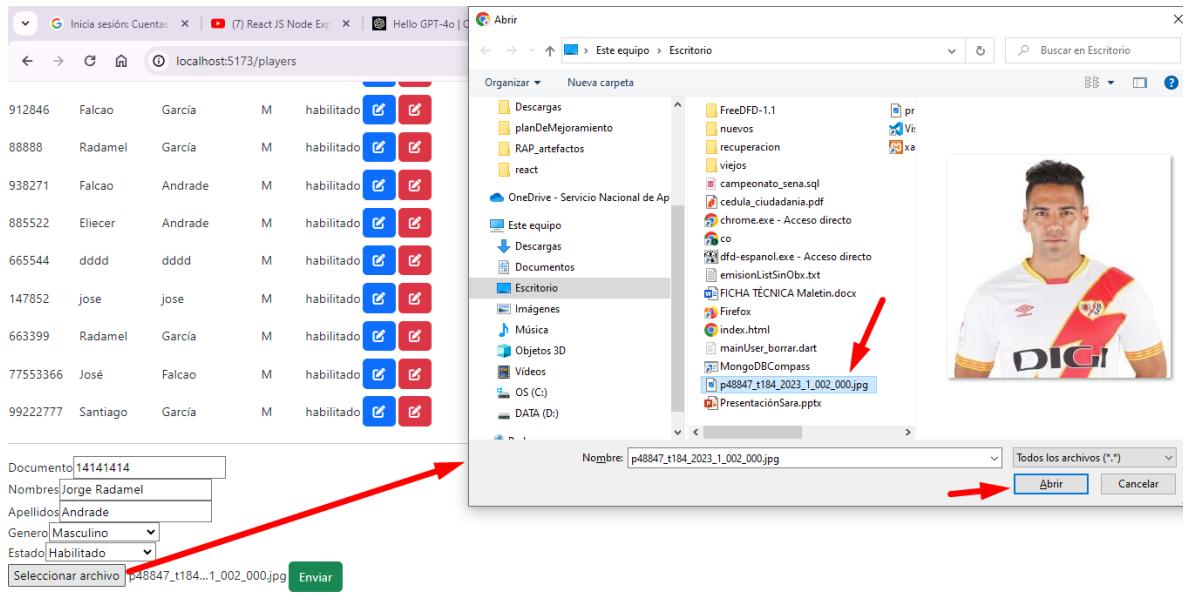
Send Request Ctrl + Enter Import cURL Ctrl + U Change Environment Ctrl + E

Resultado en la base de datos:

Servidor: 127.0.0.1 > Base de datos: torneo_react > Tabla: players

	Examinar	Estructura	SQL	Buscar	Insertar	Exportar	Importar	Privilegios	Operaciones	Disparadores		
Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]												
<input type="checkbox"/> Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Sort by key: Ninguna												
+ Opciones												
		id	documento	nombr es	apellidos	genero	id_ficha	foto	estado	createdat	updatedat	
<input type="checkbox"/>	Editar	Copiar	Borrar	1	12345678	Jorge E	Andrade Cruz	M	1	NULL	habilitado 2024-03-19 10:46:21	2024-06-01 12:19:59
<input type="checkbox"/>	Editar	Copiar	Borrar	2	98745612	Lina	Linares Mosquera	F	1	NULL	habilitado 2024-03-19 11:06:58	2024-05-03 12:12:28
<input type="checkbox"/>	Editar	Copiar	Borrar	3	654387129	Tomas	Fernandez	M	1	NULL	habilitado 2024-05-02 09:09:41	2024-05-03 12:14:05
<input type="checkbox"/>	Editar	Copiar	Borrar	7	12345	Juan Fernando	Montaña Ruiz	M	1	NULL	habilitado 2024-04-26 09:35:45	2024-05-03 07:21:12
<input type="checkbox"/>	Editar	Copiar	Borrar	12	379154268	Tomasa	Perez	F	0	NULL	habilitado 2024-05-03 07:51:18	2024-05-03 07:51:18
<input type="checkbox"/>	Editar	Copiar	Borrar	13	95759	gogog	gogogo	M	0	NULL	habilitado 2024-05-16 12:08:12	2024-05-16 12:08:12
<input type="checkbox"/>	Editar	Copiar	Borrar	14	77777	oooooo	oooooo	F	0	NULL	habilitado 2024-05-16 12:14:09	2024-05-16 12:14:09
<input type="checkbox"/>	Editar	Copiar	Borrar	15	912846	Falcao	Garcia	M	0	NULL	habilitado 2024-06-01 14:41:36	2024-06-01 14:41:36
<input type="checkbox"/>	Editar	Copiar	Borrar	16	888888	Radamel	Garcia	M	0	NULL	habilitado 2024-06-01 14:45:17	2024-06-01 14:45:17
<input type="checkbox"/>	Editar	Copiar	Borrar	17	938271	Falcao	Andrade	M	0	NULL	habilitado 2024-06-01 14:58:47	2024-06-01 14:58:47
<input type="checkbox"/>	Editar	Copiar	Borrar	18	885522	Eliecer	Andrade	M	0	NULL	habilitado 2024-06-01 15:21:47	2024-06-01 15:21:47
<input type="checkbox"/>	Editar	Copiar	Borrar	19	665544	ddd	ddd	M	0	NULL	habilitado 2024-06-01 15:27:35	2024-06-01 15:27:35
<input type="checkbox"/>	Editar	Copiar	Borrar	20	147852	jose	jose	M	0	1717274510820.jpg	habilitado 2024-06-01 15:41:50	2024-06-01 15:41:50
<input type="checkbox"/>	Editar	Copiar	Borrar	21	663399	Radamel	Garcia	M	0	NULL	habilitado 2024-06-01 15:43:54	2024-06-01 15:43:54
<input type="checkbox"/>	Editar	Copiar	Borrar	22	77553366	José	Falcao	M	0	1717334831089.jpg	habilitado 2024-06-02 08:27:11	2024-06-02 08:27:11
<input type="checkbox"/>	Editar	Copiar	Borrar	23	99222777	Santiago	Garcia	M	0	1717335024984.jpg	habilitado 2024-06-02 08:30:24	2024-06-02 08:30:24

Ahora debe realizar la prueba desde el formulario en la vista de REACT:



Resultado:

Instructor Jorge E. Andrade C.
jandradec@sena.edu.co
Centro Agropecuario La Granja
Regional Tolima

Server: 127.0.0.1 » Base de datos: torneo_react » Tabla: players

	<input type="checkbox"/>	Editar	Copiar	Borrar	id	documento	nombres	apellidos	genero	id_ficha	foto	estado	createdat	updatedat
	<input type="checkbox"/>				7	12345	Juan Fernando	Montaña Ruiz	M	1	NULL	habilitado	2024-04-26 09:35:45	2024-05-03 07:21:12
	<input type="checkbox"/>				12	379154268	Tomas	Perez	F	0	NULL	habilitado	2024-05-03 07:51:18	2024-05-03 07:51:18
	<input type="checkbox"/>				13	95759	gogog	gogogo	M	0	NULL	habilitado	2024-05-16 12:08:12	2024-05-16 12:08:12
	<input type="checkbox"/>				14	77777	ooooo	ooooo	F	0	NULL	habilitado	2024-05-16 12:14:09	2024-05-16 12:14:09
	<input type="checkbox"/>				15	912846	Falcao	Garcia	M	0	NULL	habilitado	2024-06-01 14:41:36	2024-06-01 14:41:36
	<input type="checkbox"/>				16	88888	Radamel	Garcia	M	0	NULL	habilitado	2024-06-01 14:45:17	2024-06-01 14:45:17
	<input type="checkbox"/>				17	938271	Falcao	Andrade	M	0	NULL	habilitado	2024-06-01 14:58:47	2024-06-01 14:58:47
	<input type="checkbox"/>				18	885522	Ellecer	Andrade	M	0	NULL	habilitado	2024-06-01 15:21:47	2024-06-01 15:21:47
	<input type="checkbox"/>				19	665544	dddd	dddd	M	0	NULL	habilitado	2024-06-01 15:27:35	2024-06-01 15:27:35
	<input type="checkbox"/>				20	147852	jose	jose	M	0	1717274510820.jpg	habilitado	2024-06-01 15:41:50	2024-06-01 15:41:50
	<input type="checkbox"/>				21	663399	Radamel	Garcia	M	0	NULL	habilitado	2024-06-01 15:43:54	2024-06-01 15:43:54
	<input type="checkbox"/>				22	77553366	José	Falcao	M	0	1717334831089.jpg	habilitado	2024-06-02 08:27:11	2024-06-02 08:27:11
	<input type="checkbox"/>				23	99222777	Santiago	Garcia	M	0	1717335024984.jpg	habilitado	2024-06-02 08:30:24	2024-06-02 08:30:24
	<input type="checkbox"/>				25	141414	Jorge Radamel	Garcia	M	0	1717337782716.jpg	habilitado	2024-06-02 09:16:22	2024-06-02 09:16:22

Como se observa, después de enviada la imagen sigue apareciendo el nombre original de la misma al lado del input.

Documento
Nombres
Apellidos
Genero Selecciona uno...
Estado Selecciona uno...
Seleccionar archivo p48847_t184...1_002_000.jpg **Enviar**

No es suficiente limpiar la prop del campo con setFoto(null), para ello es necesario crear una referencia del input con el HOOK **useRef** (ver el video a continuación para comprenderlo)

Video: useRef en React - <https://www.youtube.com/watch?v=raJJjm3rhhU>

Ajustes en archivo formPlayers.jsx:

```

EXPLORER ... roller.js M JS playerModel.js M JS routerPlayers.js M JS app.js M ⚡ formQueryPlayer.jsx crudPlayers.jsx formPlayers.jsx M ...
VITE-REACT-M... node routes
  -> node
  -> routes
  JS app.js M
    -> package-lock.json
    -> package.json
  <-> torneo-sena-mysql
    -> node_modules
    -> public
      -> src
        -> assets
        -> centers
        -> home
        -> players
          -> crudPlayers.jsx
          -> formPlayers.jsx M
          -> formQueryPlayer.jsx
          -> App.jsx
          -> main.jsx
          -> .eslintrc.js
          -> .gitignore
          -> index.html
          -> package-lock.json
          -> package.json
          -> README.md
          JS vite.config.js
        > OUTLINE
        > TIMELINE

```

```

roller.js M JS playerModel.js M JS routerPlayers.js M JS app.js M ⚡ formQueryPlayer.jsx crudPlayers.jsx formPlayers.jsx M ...
torneo-sena-mysql > src > players > ⚡ formPlayers.jsx > FormPlayers
1 import axios from "axios";
2 import { useRef, useState } from "react";
3 import { useEffect } from "react";
4
5 const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
  // Agregar como parámetro el botón que llega desde el componente
  // Hooks para cada uno de los campos del formulario
  const [documento, setDocumento] = useState('')
  const [nombres, setNombres] = useState('')
  const [apellidos, setApellidos] = useState('')
  const [genero, setGenero] = useState('')
  const [estado, setEstado] = useState('')
  const [foto, setFoto] = useState(null)
15 const inputFoto = useRef(null)
16
17 // Función que recibe los datos del formulario
18 const sendForm = (e) => {
19   e.preventDefault()
20
21   if (buttonForm === 'Actualizar') {
22     console.log('actualizando ando...')
23     // Aquí va el código para actualizar
24     axios.put(URI + player.id, {
25       documento: documento,
26       nombres: nombres,
27       apellidos: apellidos,
28       genero: genero,
29       estado: estado,
30       foto: foto
31     })
32   }

```

```

EXPLORER ... roller.js M JS playerModel.js M JS routerPlayers.js M JS app.js M ⚡ formQueryPlayer.jsx crudPlayers.jsx formPlayers.jsx M ...
VITE-REACT-M... node routes
  -> node
  -> routes
  JS app.js M
    -> package-lock.json
    -> package.json
  <-> torneo-sena-mysql
    -> node_modules
    -> public
      -> src
        -> assets
        -> centers
        -> home
        -> players
          -> crudPlayers.jsx
          -> formPlayers.jsx M
          -> formQueryPlayer.jsx
          -> App.jsx
          -> main.jsx
          -> .eslintrc.js
          -> .gitignore
          -> index.html
          -> package-lock.json
          -> package.json
          -> README.md
          JS vite.config.js
        > OUTLINE
        > TIMELINE

```

```

roller.js M JS playerModel.js M JS routerPlayers.js M JS app.js M ⚡ formQueryPlayer.jsx crudPlayers.jsx formPlayers.jsx M ...
torneo-sena-mysql > src > players > ⚡ formPlayers.jsx > FormPlayers
1 const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
  // Agregar como parámetro el botón que llega desde el componente
  const clearForm = () => {
    // Se vacía el elemento player
    setDocumento('')
    setNombres('')
    setApellidos('')
    setGenero('')
    setEstado('')
    setFoto(null)
29   inputFoto.current.value = '' //Restablecer el valor del input de la foto
  }
20
21 const setData = () => {
  // Función que establece los valores a los campos para que se muestren en el formulario cuando presionen el botón 'Actualizar'
  setDocumento(player.documento)
  setNombres(player.nombres)
  setApellidos(player.apellidos)
  setGenero(player.genero)
  setEstado(player.estado)
  setFoto(player.foto)
}
25
30 useEffect(() => {
  // useEffect escucha los cambios en el objeto 'player' y se ejecuta la función 'setData'
  setData()
}, [player])
35
36 return (
37   <>
38     <form id="playerForm" action="" onSubmit={sendForm}>
39       <label htmlFor="documento">Documento</label>

```

```

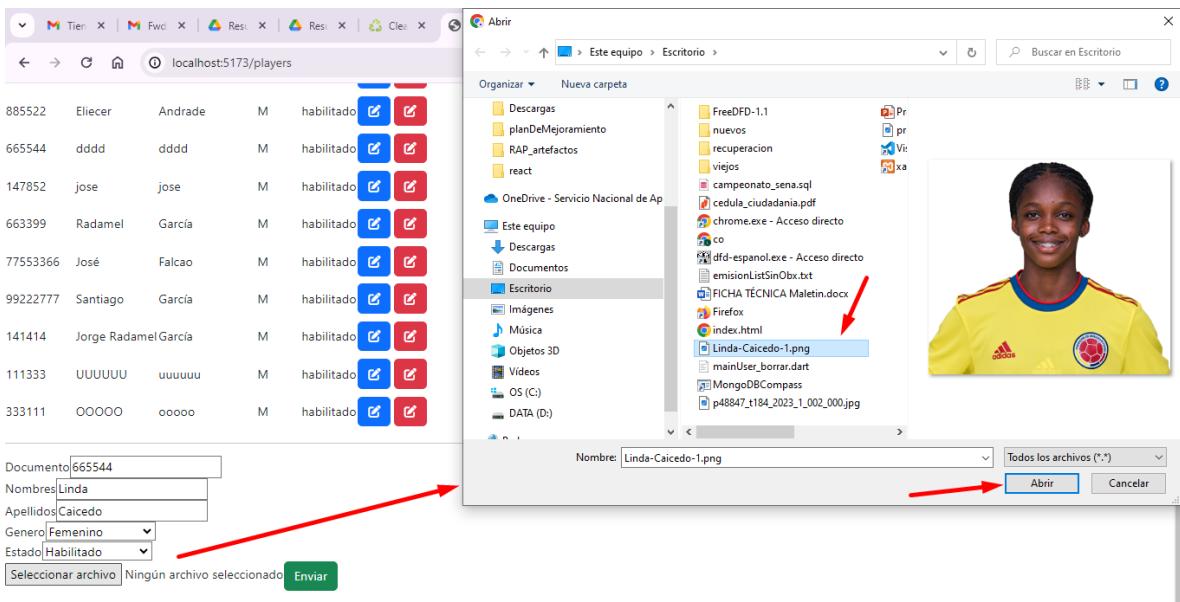
const FormPlayers = ({ buttonForm, player, URI, updateTextButton }) => {
  // Agregar como parámetro el botón que llega desde el componente padre
  <input type="text" id="nombres" value={nombres} onChange={(e) => setNombres(e.target.value)} />
  <br />
  <label htmlFor="apellidos">Apellidos</label>
  <input type="text" id="apellidos" value={apellidos} onChange={(e) => setApellidos(e.target.value)} />
  <br />
  <label htmlFor="genero">Genero</label>
  <select name="" id="genero" value={genero} onChange={(e) => setGenero(e.target.value)}>
    <option value="">Selecciona uno...</option>
    <option value="F">Femenino</option>
    <option value="M">Masculino</option>
    <option value="O">Otro</option>
  </select>
  <br />
  <label htmlFor="estado">Estado</label>
  <select name="" id="estado" value={estado} onChange={(e) => setEstado(e.target.value)}>
    <option value="">Selecciona uno...</option>
    <option value="H">Habilitado</option>
    <option value="D">Deshabilitado</option>
  </select>
  <br />
  <label htmlFor="foto"></label>
  <input type="file" id="foto" onChange={(e) => setFoto(e.target.files[0])} ref={inputFoto}></input>
  <input type="submit" id="boton" value={buttonForm} className="btn btn-success" />
</form>
}

```

> OUTLINE

126 export default FormPlayers

Resultado:



Después del envío queda limpio el input y se podrá registrar un nuevo jugador sin percances:

665544 Linda Caicedo F habilitado

Documento
Nombres
Apellidos
Genero Selecciona uno...
Estado Selecciona uno...
Seleccionar archivo Ningún archivo seleccionado Enviar

Fase 4: Mostrar las fotos de los players en el listado en el archivo crudPlayers.jsx:

EXPLORER

```

VITE-REACT-M... roller.js M JS playerModel.js M JS routerPlayers.js M JS app.js M formQueryPlayer.js crudPlayers.jsx X formPlayers.jsx M
  node
    routes
      routerCentros.js
      routerDeptos.js
      routerMunicipios.js
      routerPlayers.js M
    app.js M
    package-lock.json
    package.json
  torneo-sena-mysql
    node_modules
    public
      assets
      centers
      home
      players
        crudPlayers.jsx M
        formPlayers.jsx M
        formQueryPlayer.jsx
        App.jsx
        main.jsx
        .eslintrc.cjs
        .gitignore
        index.html
        package-lock.json
        package.json
  OUTLINE

```

roller.js M playerModel.js M routerPlayers.js M app.js M formQueryPlayer.js crudPlayers.jsx X formPlayers.jsx M

```

torneo-sena-mysql > src > players > crudPlayers.jsx > [x] CrudPlayers
1 import axios from 'axios'
2 import { useState, useEffect } from 'react'
3 import { Link } from 'react-router-dom'
4 import FormPlayers from './formPlayers'
5
6 import Swal from 'sweetalert2'
7 import FormQueryPlayer from './formQueryPlayer'
8
9 const URI = 'http://localhost:8000/players/'
10 const PATH_FOTOS = 'http://localhost:8000/public/uploads/' ←
11
12 const CrudPlayers = () => {
13
  //uso del Hook useState
  const [playerList, setPlayerList] = useState([])
  //inicializar la lista con un arreglo vacío
14
15
  //Definir el texto del botón del formulario el cual varía entre 'Enviar' y 'Actualizar'
  const [buttonForm, setButtonForm] = useState('Enviar')
16
17
  // Definir prop para el registro de un player
  const [player, setPlayer] = useState({
18
    id: '',
    documento: '',
    nombres: '',
    apellidos: '',
    genero: '',
    foto: null, ←
19
    estado: ''
  })
20
21
22
23
24
25
26
27
28
29
30
31

```

Code editor showing the `crudPlayers.jsx` file. A red arrow points from the file list to the code area. A red box highlights the `<th>Foto</th>` line. Another red box highlights the `src={PATH_FOTOS + player.foto}` line with the annotation: "Insertar etiqueta para imagen y en el src concatenar la ruta con el nombre de la imagen".

```

const CrudPlayers = () => {
  return (
    <table>
      <thead>
        <tr>
          <th>Documento</th>
          <th>Nombres</th>
          <th>Apellidos</th>
          <th>Genero</th>
          <th>Foto</th>
          <th>Estado</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        {playerList.map(player) =>
          <tr key={player.id}>
            <td>{player.documento}</td>
            <td>{player.nombres}</td>
            <td>{player.apellidos}</td>
            <td>{player.genero}</td>
            <td><img width="80px" src={PATH_FOTOS + player.foto}></td>
            <td>{player.estado}</td>
            <td>
              <span className="btn btn-primary" onClick={() => getPlayer(player.id)}>i class="fa-solid fa-pen-to-square"</span>
              <span className="btn btn-danger m-1" onClick={() => deletePlayer(player.id)}>i class="fa-solid fa-trash"</span>
            </td>
          </tr>
        })
      </tbody>
    </table>
  )
}

```

Resultado:

Screenshot of a browser showing the players list at `localhost:5173/players`. A red box highlights the image of the player with ID 147852. The DevTools panel on the right shows the element `` selected, with its style properties displayed.

ID	Nombre	Apellido	Género	Foto	Estado	Opciones
379154268	Tomas	Perez	F		Habilitado	Actualizar Eliminar
147852	jose	jose	M		Habilitado	Actualizar Eliminar
77553366	José	Falcao	M		Habilitado	Actualizar Eliminar
99222777	Santiago	Garcia	M		Habilitado	Actualizar Eliminar
141414	Jorge Radamel	Garcia	M		Habilitado	Actualizar Eliminar
111333	UUUUU	uuuuu	M		Habilitado	Actualizar Eliminar
665544	Linda	Caicedo	F		Habilitado	Actualizar Eliminar

Formulario para agregar un nuevo jugador:

Documento:

Nombres:

Apellidos:

Género:

Estado:

Seleccionar archivo:

Enviar

Fase 5: Opción de 'Actualizar' en `playerController.js`:

EXPLORER

```

VITE-REACT-M...  lier.js JS authController.js U crudPlayers.jsx M formPlayers.jsx M formQueryPlayer.jsx M playerController.js M X localhost8000/players/1 routerPlayers.js M
  node
    controllers
      JS authController.js U
      JS centerController.js
      JS deptoController.js
      JS municipioController.js
      JS playerController.js M
    > database
      > models
        JS centerModel.js
        JS deptosModel.js
        JS municipioModel.js
        JS playerModel.js M
        JS userModel.js U
    > node_modules
    > public
    > routes
      JS routerCenters.js
      JS routerDeptos.js
      JS routerMunicipios.js
      JS routerPlayers.js M
    JS app.js M
    () package-lock.json M
    () package.json M
  > node_modules
  > public
  > src
    > assets
    > centers
    > home
    > players

```

```

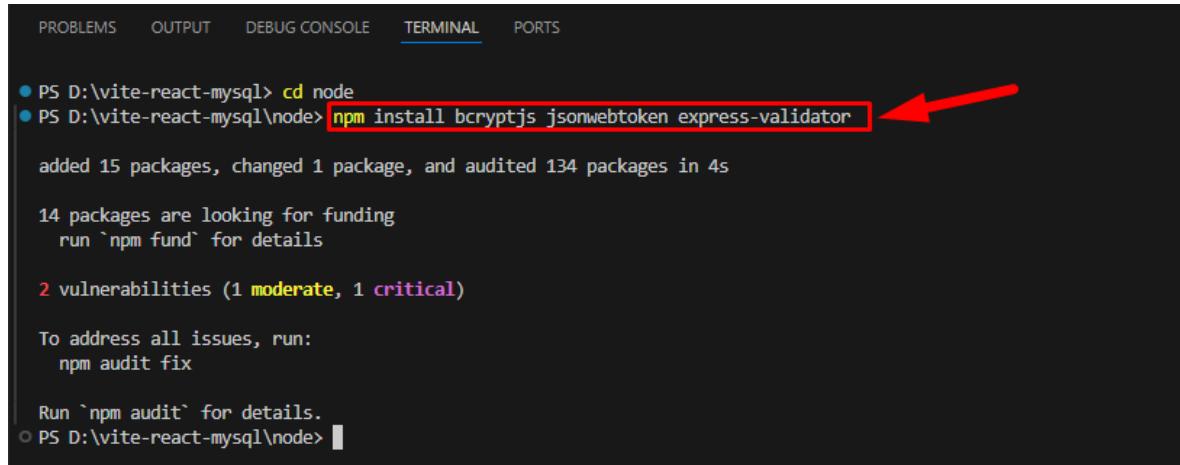
playerController.js M
  49 //Actualizar un registro
  50 export const updatePlayer = async (req, res) => {
  51
  52   try {
  53     //Cambia la manera de leer los datos que llegan del formulario
  54     const { documento, nombres, apellidos, genero } = req.body
  55
  56     //Operador ternario, si el archivo existe se toma su nombre, si no, se establece como null
  57     const foto = req.file ? req.file.filename : null
  58
  59     if (foto != null) { //Si viene una foto actualizar teniendo en cuenta el campo foto
  60
  61       await PlayerModel.update({
  62         documento,
  63         nombres,
  64         apellidos,
  65         genero,
  66         foto // Aquí se tiene en cuenta el campo foto
  67       }, { where: { id: req.params.id } })
  68
  69     } else { //Si no viene la foto no se tiene en cuenta el campo foto porque de lo contrario borraría el nombre de la foto en la base de datos
  70
  71       await PlayerModel.update({
  72         documento,
  73         nombres,
  74         apellidos,
  75         genero
  76       }, { where: { id: req.params.id } })
  77
  78     res.json({ "message": "¡Registro actualizado exitosamente!" })
  79
  80   } catch (error) {
  81     res.json({ message: error.message })
  82   }
  83 }
  84
  85

```

Resultado:

Documento	Nombres	Apellidos	Genero	Foto	Estado	Acciones
14137	Jorge Eliecer	Andrade Cruz	M		habilitado	 
147852	jose	Guzman	M		habilitado	 
77553366	José	Falcao	M		habilitado	 
99222777	Santiago	Garcia	M		habilitado	 
141414	Jorge Radamel	Garcia	M		habilitado	 
111333	UUUUUU	uuuuuu	M		habilitado	 
665544	Linda	Caicedo	F		habilitado	 
88888	Linda	Torres	F		habilitado	 

Autenticación



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS D:\vite-react-mysql> cd node
● PS D:\vite-react-mysql\node> npm install bcryptjs jsonwebtoken express-validator
added 15 packages, changed 1 package, and audited 134 packages in 4s

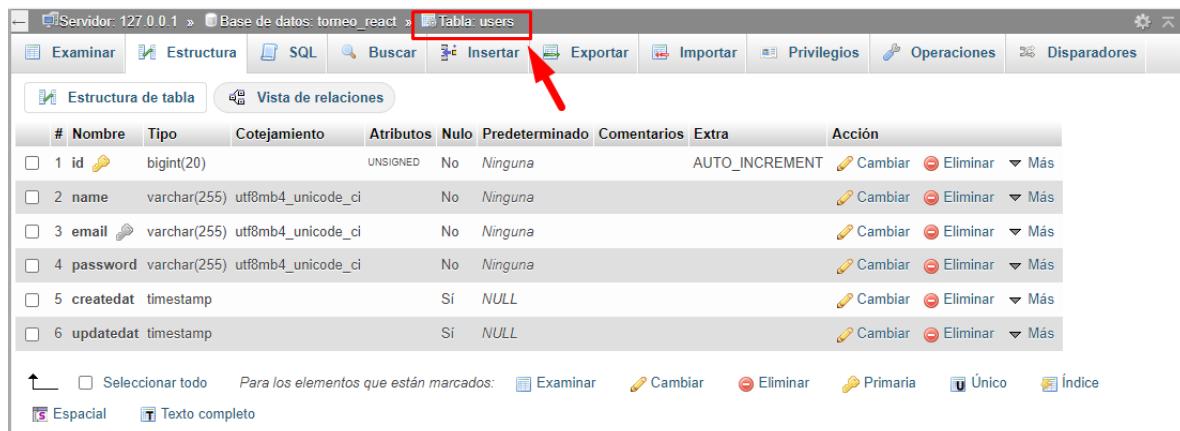
14 packages are looking for funding
  run `npm fund` for details

2 vulnerabilities (1 moderate, 1 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
○ PS D:\vite-react-mysql\node>
```

Creación de tabla para usuarios:



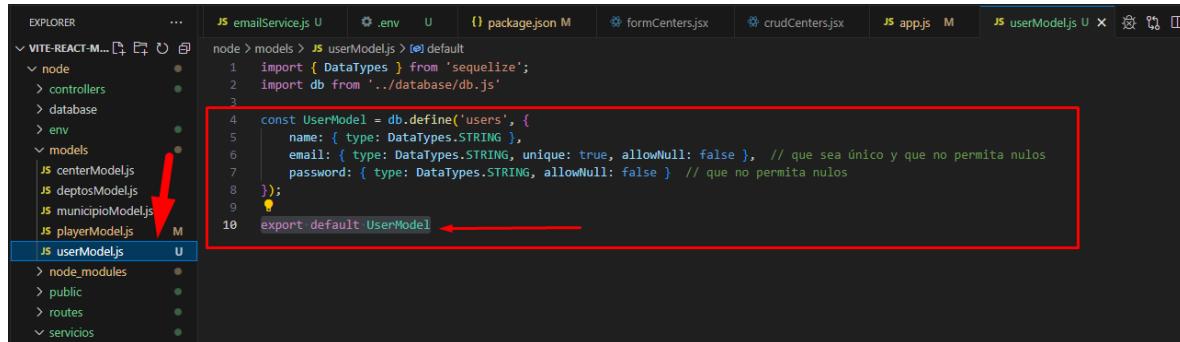
The screenshot shows the MySQL Workbench interface with the following details:

- Path: Servidor: 127.0.0.1 > Base de datos: torneo_react > Tabla: users
- Toolbar buttons: Examinar, Estructura, SQL, Buscar, Insertar (highlighted with a red arrow), Exportar, Importar, Privilegios, Operaciones, Disparadores.
- Table structure:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	bigint(20)	UNSIGNED	No	Ninguna			AUTO_INCREMENT	Cambiar Eliminar Más
2	name	varchar(255)	utf8mb4_unicode_ci	No	Ninguna				Cambiar Eliminar Más
3	email	varchar(255)	utf8mb4_unicode_ci	No	Ninguna				Cambiar Eliminar Más
4	password	varchar(255)	utf8mb4_unicode_ci	No	Ninguna				Cambiar Eliminar Más
5	createdat	timestamp		Sí	NULL				Cambiar Eliminar Más
6	updatedat	timestamp		Sí	NULL				Cambiar Eliminar Más

- Bottom toolbar buttons: Seleccionar todo, Examinar, Cambiar, Eliminar, Primaria, Único, Índice, Espacial, Texto completo.

Configuración del modelo authModel.js



The screenshot shows the VS Code interface with the following details:

- Explorer sidebar: VITE-REACT-M..., node, controllers, database, env, models (centerModel.js, deptosModel.js, municipioModel.js, playerModel.js, userModel.js - highlighted with a red arrow).
- File tab: emailService.js, .env, package.json, formCenters.jsx, crudCenters.jsx, app.js, userModel.js (highlighted with a red box).
- Code editor:

```
node > models > userModel.js > [default]
1 import { DataTypes } from 'sequelize';
2 import db from '../database/db.js';
3
4 const UserModel = db.define('users', {
5   name: { type: DataTypes.STRING },
6   email: { type: DataTypes.STRING, unique: true, allowNull: false }, // que sea único y que no permita nulos
7   password: { type: DataTypes.STRING, allowNull: false } // que no permita nulos
8 });
9
10 export default UserModel
```

Configuración del controlador authController.js

EXPLORER

```

node > controllers > JS authController.js ...
1 import bcryptjs from 'bcryptjs' ←
2 import UserModel from '../models/userModel.js'
3
4 export const createUser = async (req, res) => { //Función para crear usuarios ←
5
6   try {
7     //Cambia la manera de leer los datos que llegan del formulario ←
8     const { name, email, password } = req.body
9
10    console.log(password)
11
12    //encriptar el password ←
13    let passHash = await bcryptjs.hash(password, 8) ←
14
15    //enviar datos a base de datos
16    await UserModel.create({
17      "name": name,
18      "email": email,
19      "password": passHash ←
20    })
21
22    res.json({ "message": "Usuario creado de manera exitosa." })
23
24  } catch (error) {
25
26    res.json({ "message": error })
27
28  }
29
30

```

Configuración de archivo de rutas (routerAuth.js)

EXPLORER

```

node > routes > JS routerAuth.js ...
1 import express from 'express'
2 import { createUser } from '../controllers/authController.js' ←
3
4 const router = express.Router()
5
6 router.post('/', createUser) ←
7
8
9 export default router
10

```

Ajuste en el archivo app.js

```

node > JS app.js > ...
1 import express from 'express'
2 import cors from 'cors'
3
4 import db from './database/db.js'
5 import playerRoutes from './routes/routerPlayers.js'
6 import centerRoutes from './routes/routerCenters.js'
7 import deptosRoutes from './routes/routerDeptos.js'
8 import authRoutes from './routes/routerAuth.js' ← Red box and arrow
9
10 import municipioRoutes from './routes/routerMunicipios.js'
11
12 //Importar modelos para consultar con llave foranea
13 import CenterModel from './models/centerModel.js'
14 import DeptosModel from './models/deptosModel.js'
15
16 const app = express()
17
18 app.use(cors())
19 app.use(express.json())
20 app.use('/players', playerRoutes)
21 app.use('/centers', centerRoutes)
22 app.use('/deptos', deptosRoutes)
23 app.use('/municipios', municipioRoutes)
24 app.use('/auth', authRoutes) ← Red box and arrow
25
26 //Configurar express para que pueda servir los archivos estaticos desde el directorio de cargas
27 app.use('/public/uploads', express.static('public/uploads/'))
28
29 try {
30   await db.authenticate()
31   console.log("Conexión exitosa a la db")
32 } catch (error) {

```

Prueba en Thunder Client:

Activity Collections Env

POST <http://localhost:8000/auth/>

Query	Headers 2	Auth	Body 1	Tests	Pre Run
			JSON	XML	Form

JSON Content

```

1 {
2   "name": "jorge",
3   "email": "jandradec@sena.edu.co",
4   "password": "12345678"
5 }

```

Status: 200 OK Size: 47 Bytes Time: 199 ms

Response Headers 7 Cookies Results Docs

```

1 {
2   "message": "Usuario creado de manera exitosa."
3 }

```

Resultado en la base de datos:

Mostrando filas 0 - 3 (total de 4. La consulta tardó 0,00003 segundos.)

```
SELECT * FROM `users`
```

Permitido [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

	<input type="checkbox"/> Mostrar todo	Número de filas:	25	Filtrar filas:	Buscar en esta tabla	Sort by key:	Ninguna
+ Opciones	<input type="checkbox"/>						
	<input type="checkbox"/>	<input type="checkbox"/> id	<input type="checkbox"/> name	<input type="checkbox"/> email	<input type="checkbox"/> password	<input type="checkbox"/> createdat	<input type="checkbox"/> updatedat
	<input type="checkbox"/>	Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	1 JORGE ELEICER jorge@senra.edu.co	\$2y\$10\$Go9E4e/8Ix85gErzEO4eLvtYNsO3wK0hgErPEZ...	2021-08-03 01:43:05 2021-08-03 01:43:05
	<input type="checkbox"/>	Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	2 NICOLAS CASTRO ncastro402@misena.edu.co	\$2y\$10\$CizGzkLmpsGjiH4EFLoHfeVDVcZFUGZyKM2WmVbr...	2021-08-16 16:09:18 2021-08-16 16:09:18
	<input type="checkbox"/>	Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	3 jorge jorge@hotmail.com	\$2y\$10\$JuGxsA8phxtobXQMIk9CSeoURUxCNGzxB7T36oVGeK...	2021-12-23 17:11:28 2021-12-23 17:19:30
	<input type="checkbox"/>	Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	4 jorge jandrade@senra.edu.co	\$2a\$08\$e9MzK1.KkwqqPx/CFNwTO3V/bY5ksc5lYepeJtUH...	2024-08-10 11:02:02 2024-08-10 11:02:02

↑ Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Sort by key: Ninguna

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

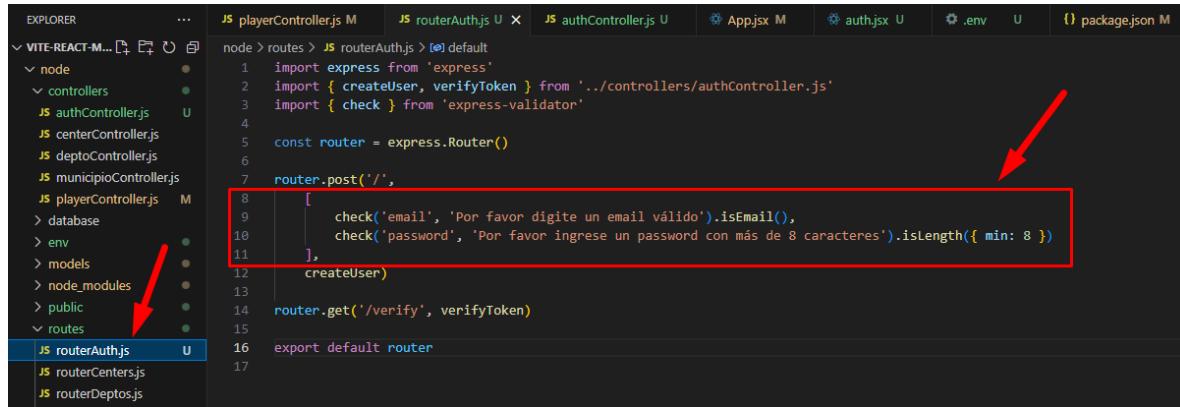
Validar si el usuario ya existe:

```

EXPLORER ... JS playerController.js M JS centerController.js JS routerAuth.js U JS authController.js U X JS app.js M TO localhost:8000/players/1
VITE-REACT-M... node controllers authController.js
node controllers authController.js
JS authController.js U
JS centerController.js
JS deptoController.js
JS municipioController.js
JS playerController.js M
database
models
JS centerModel.js
JS deptosModel.js
JS municipioModel.js
JS playerModel.js M
JS userModel.js U
node_modules
public
routes
JS routerAuth.js U
JS routerCenters.js
JS routerDeptos.js
JS routerMunicipios.js
JS routerPlayers.js M
app.js M
package-lock.json M
package.json M
torneo-sena-mysql
node_modules
public
src
assets
centers
home
OUTLINE
TIMELINE
node > controllers > authController.js > ...
1 import bcryptjs from 'bcryptjs'
2 import UserModel from '../models/userModel.js'
3
4 export const createUser = async (req, res) => { //Función para crear usuarios
5
6     try {
7         //Cambia la manera de leer los datos que llegan del formulario
8         const { name, email, password } = req.body
9
10        //Validar si ya existe un usuario con el mismo correo
11        let user = await UserModel.findOne({ where: { email: email } })
12
13        if (user) { //Si el usuario existe, no permite la creación del usuario
14            res.json({ "message": "El usuario ya existe" })
15
16        } else { //Si el usuario no existe, permite la creación
17            //encriptar el password
18            let passHash = await bcryptjs.hash(password, 8)
19
20            //enviar datos a base de datos
21            await UserModel.create({
22                "name": name,
23                "email": email,
24                "password": passHash
25            })
26
27            res.json({ "message": "Usuario creado de manera exitosa." })
28
29        }
30
31    }
32
33    } catch (error) {
34        res.json({ "message": error })
35
36    }
37
38}
39

```

Crear validación en la ruta para permitir solo direcciones de correo validas y que la contraseña tenga mínimo 8 caracteres:



```
EXPLORER ... JS playerController.js M JS routerAuth.js U X JS authController.js U ⚡ App.jsx M ⚡ auth.jsx U ⚡ .env U ⚡ package.json M
VITE-REACT-M...
node controllers authController.js centerController.js deptoController.js municipioController.js playerController.js database env models node_modules public routes routerAuth.js routerCentersjs routerDeptosjs

JS routerAuth.js U
node > routes > JS routerAuth.js > default
1 import express from 'express'
2 import { createUser, verifyToken } from '../controllers/authController.js'
3 import { check } from 'express-validator'
4
5 const router = express.Router()
6
7 router.post('/', [
8   [
9     check('email', 'Por favor digite un email válido').isEmail(),
10    check('password', 'Por favor ingrese un password con más de 8 caracteres').isLength({ min: 8 })
11  ],
12  createUser
13]
14 router.get('/verify', verifyToken)
15
16 export default router
17
```

JWT

¿Qué es un JWT?

Un JSON Web Token (JWT) es un estándar abierto (RFC 7519) que define una manera compacta y autocontenido de transmitir información de manera segura entre partes como un objeto JSON. Este token se usa comúnmente para la autenticación y la autorización en aplicaciones web.

¿Cómo se usa un JWT?

Autenticación: El usuario inicia sesión con sus credenciales. El servidor verifica la autenticidad de estas credenciales y, si son válidas, genera un JWT y lo devuelve al cliente.

Autorización: El cliente envía el JWT en el encabezado de autorización de las solicitudes posteriores. El servidor verifica la firma del JWT y, si es válida, permite el acceso a los recursos solicitados.

Para generar un JWT se necesita una clave secreta, esta clave es una cadena de texto compleja y debe estar almacenada en una variable de entorno.

Variables de entorno y archivo '.env'

Las variables de entorno son variables dinámicas que pueden afectar el comportamiento de procesos que se ejecutan en un sistema operativo. Se utilizan para almacenar información de configuración que puede cambiar entre diferentes entornos (desarrollo, prueba, producción) sin necesidad de modificar el código fuente de una aplicación.

Ejemplos de uso:

Configuración de base de datos: URL, nombre de usuario y contraseña.

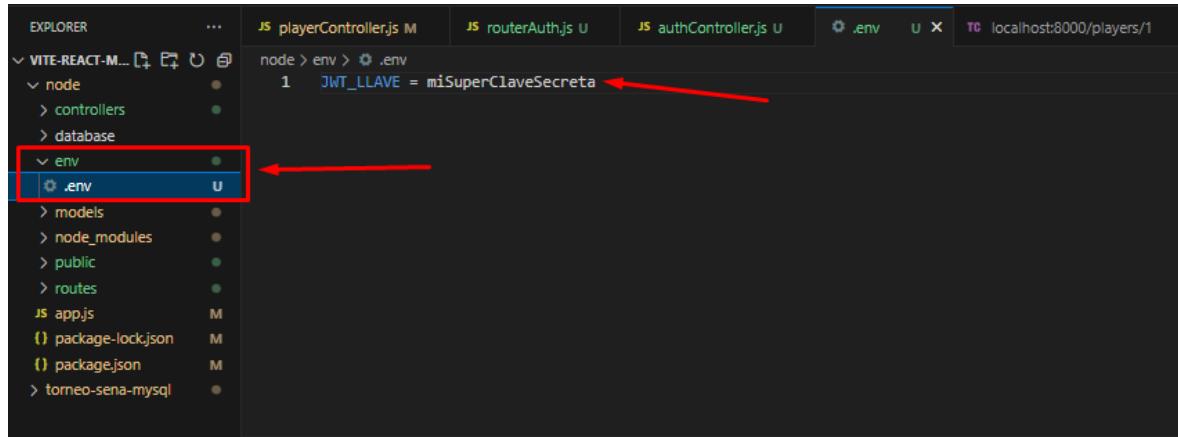
Claves secretas: Para firmar tokens JWT o integrarse con APIs externas.

Configuración del servidor: Puertos, modos de depuración, URLs de servicios externos.

El Archivo .env

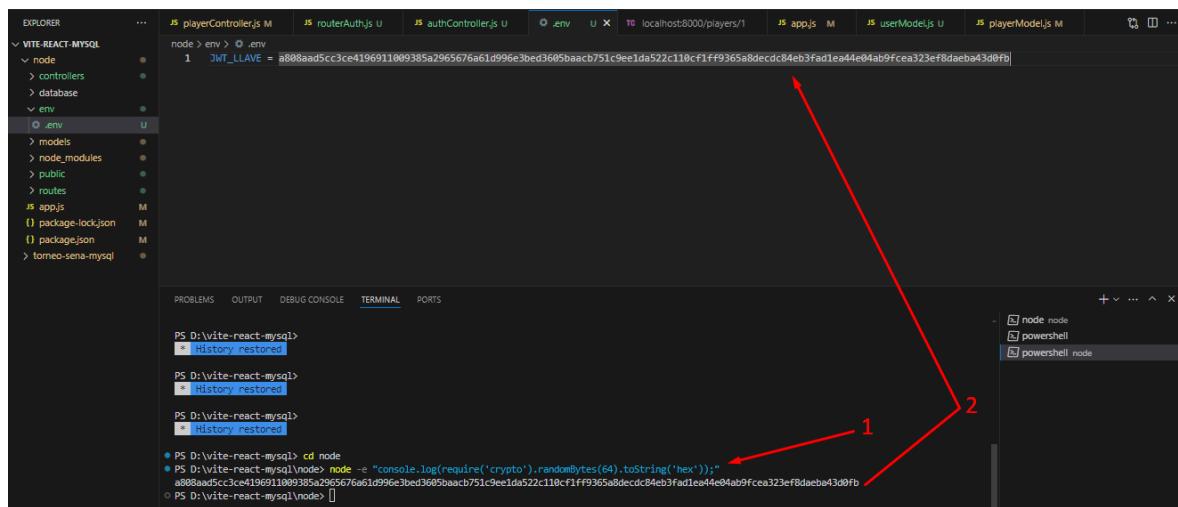
Un archivo .env es un archivo de texto que contiene pares clave-valor, utilizados para definir variables de entorno de manera local. Este archivo no debe ser incluido en el control de versiones (por ejemplo, en Git), porque puede contener información sensible como claves secretas.

Crear carpeta **env** y dentro el archivo **.env**



Generar una clave aleatoria más compleja con el comando

```
node -e "console.log(require('crypto').randomBytes(64).toString('hex'));"
```



Instalar la dependencia **dotenv** para cargar y leer las variables de entorno:

```
npm install dotenv
```

VS Code Explorer and Terminal Screenshot:

```

    "scripts": {
      "test": "echo \\\"Error: no test specified\\\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "dependencies": {
      "bcryptjs": "^2.4.3",
      "cors": "^2.8.5",
      "dotenv": "^16.4.5", highlighted
      "express": "^4.18.3",
      "express-validator": "^7.1.0",
      "jsonwebtoken": "^9.0.2",
      "multer": "^1.4.5-lts.1",
      "mysql2": "^3.9.2",
      "sequelize": "^6.37.1"
    }
  }

  PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
  ● PS D:\vite-react-mysql\node> npm install dotenv
  added 1 package, and audited 135 packages in 1s
  15 packages are looking for funding
  run `npm fund` for details
  2 vulnerabilities (1 moderate, 1 critical)

  To address all issues, run:
  npm audit fix

  Run `npm audit` for details.
  ○ PS D:\vite-react-mysql\node>
  
```

VS Code Explorer and Editor Screenshot:

```

    import express from 'express'
    import cors from 'cors'
    import dotenv from 'dotenv' highlighted
    import db from './database/db.js'
    import playerRoutes from './routes/routerPlayers.js'
    import centerRoutes from './routes/routerCenters.js'
    import deptosRoutes from './routes/routerDeptos.js'
    import authRoutes from './routes/routerAuth.js'
    import municipioRoutes from './routes/routerMunicipios.js'

    // Importar modelos para consultar con llave foranea
    import CenterModel from './models/centerModel.js'
    import DeptosModel from './models/deptosModel.js'

    const app = express()
    app.use(cors())
    app.use(express.json())
    app.use('/players', playerRoutes)
    app.use('/centers', centerRoutes)
    app.use('/deptos', deptosRoutes)
    app.use('/municipios', municipioRoutes)
    app.use('/auth', authRoutes)

    //Configurar express para que pueda servir los archivos estaticos desde el directorio de cargas
    app.use('/public/uploads', express.static('public/uploads/'))

    //Establecer carpeta para variables de entorno con dotenv
    dotenv.config({ path: './env/.env' }) highlighted
  
```

Ajustes en controlador authController.js para generar el JWT:

```

VITE-REACT-M...
node
  controllers
    authController.js
      centerController.js
      deptoController.js
      municipioController.js
      playerController.js
    database
    env
      .env
    models
    node_modules
    public
    routes
      app.js
    package-lock.json
    package.json
  torneo-sena-mysql

```

```

JS authController.js M JS routerAuth.js U JS authController.js U package.json M localhost:8000/players/1 JS app.js M JS userModel.js U JS ...
1 import bcryptjs from 'bcryptjs'
2 import UserModel from '../models/userModel.js'
3
4 import jwt from 'jsonwebtoken'
5
6 export const createUser = async (req, res) => {
7   try {
8     //Cambia la manera de leer los datos que llegan del formulario
9     const { name, email, password } = req.body
10    //Validar si ya existe un usuario con el mismo correo
11    let user = await UserModel.findOne( { where: { email: email } } )
12    if (user) { //Si el usuario existe, no permite la creación del usuario
13      res.json({ message: "El usuario ya existe" })
14    } else { //Si el usuario no existe, permite la creación
15      //Encriptar el password
16      let passhash = await bcryptjs.hash(password, 8) //el número representa la cantidad de veces que se ejecuta el hash para encriptar la clave
17      //Enviar datos a base de datos
18      const userOk = await UserModel.create({
19        name,
20        email,
21        password: passhash
22      })
23      //Para generar el token se asignan 3 elementos en este caso: 1ro los datos que se obtienen del usuario, 2do la llave del archivo .env y 3ro la duración de la sesión
24      const tokenUser = jwt.sign({ user: { email: userOk.email } }, process.env.JWT_LIVE, { expiresIn: '4h' })
25
26      console.log("TOKEN: " + tokenUser)
27
28      res.json({ tokenUser })
29    }
30  }
31
32
33 } catch (error) {
34
35   res.json({ message: error })
36
37 }
38

```

Prueba en ThunderClient donde la respuesta es el token:

Front-end para autenticación

Crear componente para registro de usuarios **auth.jsx** en carpeta auth:

```

VITE-REACT-M_ ...
  > node
  > node_modules
  > public
  > src
    > assets
    > auth
      auth.jsx
    > centers
      crudCenters.jsx
      formCenters.jsx
    > home
    > players
    > App.jsx M
    > main.jsx
    > .eslintrc.js
    > .gitignore
    > index.html
    > package-lock.json
    > package.json
    > README.md
    > vite.config.js

JS playerController.js M | JS routerAuth.js U | JS authController.js U | .env U | package.json M | auth.jsx U X | formCenters.jsx | crudCenters.jsx

tomeo-sena-mysql > src > auth > auth.jsx > ...
1 import axios from "axios"
2 import { useState } from "react"
3
4 const URI_AUTH = 'http://localhost:8000/auth/' ←
5
6 const Auth = () => {
7
8   const [name, setName] = useState('')
9   const [email, setEmail] = useState('')
10  const [password, setPassword] = useState('')
11
12  const [buttonForm, setButtonForm] = useState('Registrar')
13
14  const sendForm = async (e) => {
15
16    e.preventDefault()
17
18    if (buttonForm == 'Registrar') {
19
20      console.log('Registrando ando ...')
21
22      await axios.post(URI_AUTH, {
23        name: name,
24        email: email,
25        password: password
26      }).then(response => {
27
28        if (response.data.tokenUser) {
29          localStorage.setItem('userTorneo', JSON.stringify(response.data)) //Recibir el token y registrarlo en el localStorage
30
31        }
32      })
33
34    }
35
36  }
37
38  return (
39    <>
40    <form onSubmit={sendForm}>
41      <label htmlFor="name">Nombre completo:</label>
42      <input type="text" id="name" value={name} onChange={(e) => setName(e.target.value)} />
43      <br />
44      <label htmlFor="email">Correo electrónico:</label>
45      <input type="email" id="email" value={email} onChange={(e) => setEmail(e.target.value)} />
46      <br />
47      <label htmlFor="password">Password:</label>
48      <input type="password" id="password" value={password} onChange={(e) => setPassword(e.target.value)} />
49      <br />
50      <input type="submit" value={buttonForm} />
51
52    </form>
53  )
54
55  )
56
57  export default Auth

```

Crear enlace en componente principal

```

VITE-REACT-M_ ...
  > node
  > node_modules
  > public
  > src
    > assets
    > auth
      auth.jsx
    > centers
      crudCenters.jsx
      formCenters.jsx
    > home
    > players
    > App.jsx M
    > main.jsx
    > .eslintrc.js
    > .gitignore
    > index.html
    > package-lock.json
    > package.json
    > README.md
    > vite.config.js

JS playerController.js M | JS routerAuth.js U | JS authController.js U | .env U | package.json M | auth.jsx U X | formCenters.jsx | crudCenters.jsx | App.jsx M X | vite.config.js

tomeo-sena-mysql > src > App.jsx > App.jsx ...
1 import React from 'react'
2 import CrudPlayers from './players/crudPlayers'
3 import CrudCenters from './centers/crudCenters'
4 import Auth from './auth/auth' ←
5
6 function App() {
7
8  return (
9    <>
10    <nav> /* Crear una barra sencilla de navegación temporalmente */
11    <ul>
12      <li>
13        <Link to="/">Principal</Link> /* En lugar de usar etiquetas <a> se utiliza el componente <Link></Link> */
14      </li>
15      <li>
16        <Link to="/players">Players</Link>
17      </li>
18      <li>
19        <Link to="/centers">Centers</Link>
20      </li>
21      <li>
22        <Link to="/auth">Sesión</Link>
23      </li>
24    </ul>
25
26    <Routes>
27      /* en el componente Route van dos propiedades, la primera es path y la segunda es element, en path va la dirección donde queremos mostrar el componente, y en
28      element va el componente que se quiere mostrar.*/
29      <Route path="/" element={<Home />} />
30      <Route path="/players" element={<CrudPlayers />} />
31      <Route path="/centers" element={<CrudCenters />} />
32      <Route path="/auth" element={<Auth />} />
33
34    </Routes>
35
36  )
37
38  </nav>
39
40  /*Aqui empieza la gestión de rutas */
41
42  <Routes>
43
44  </Routes>

```

Resultado: Al enviar un registro de usuario se genera el token y queda registrado en las variables de entorno.

The screenshot illustrates the user registration process and its resulting environment variable. On the left, a browser window shows a registration form with fields for Nombre completo (Jenifer cruz), Correo electrónico (jencruz@gmail.com), and Password (*****). A red arrow labeled '1' points to the 'Sesion' checkbox. Another red arrow labeled '2' points to the 'Registrar' button. On the right, the Chrome DevTools Application tab is open, showing the Local storage for the origin http://localhost:5173. A red box labeled '3' highlights the 'Application' tab. A red box labeled '4' highlights the 'LocalStorage' section. A red arrow labeled '5' points to the 'userTorneo' key, which has a value of '("tokenUser": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJlc2VvI...')'. This demonstrates that the generated token is stored in the browser's local storage.

Proteger una vista para que solo pueda acceder un usuario autenticado:

En el archivo `authController.js` se establece un método para verificar que el token existe y que corresponde a la firma que se estableció con anterioridad en el archivo `.env`:

The screenshot shows the `authController.js` file in the VS Code editor. A red box highlights the `verifyToken` function. Three red arrows point to specific lines of code within this function: one to the `const token = req.header('Authorization').replace('Bearer ', '')` line, another to the `if (!token) { res.status(401).json({ message: 'Acceso denegado' }) }` line, and a third to the `const decoded = jwt.verify(token, process.env.JWT_LLAVE)` line. These annotations illustrate the logic for token extraction, verification, and decoding.

```

    export const verifyToken = (req, res) => {
      const token = req.header('Authorization').replace('Bearer ', '') //Borrar cabecera y dejar limpio el token para compararlo
      despues con la llave definida en el .env
      if (!token) {
        res.status(401).json({ message: 'Acceso denegado' })
      }
      try {
        const decoded = jwt.verify(token, process.env.JWT_LLAVE) //Verificar que el token no ha sido manipulado, se revisa
        con la llave creada en .env
        req.user = decoded
        res.status(200).json({ message: 'Token valido' })
      } catch (error) {
        res.status(400).json({ message: 'Token invalido' })
      }
    }
  
```

En el archivo de rutas se agrega el método de verificación:

EXPLORER

```

    VITE-REACT-M...
        node
            controllers
                authController.js
                centerController.js
                deptoController.js
                municipioController.js
                playerController.js
            database
            env
            models
            node_modules
            public
            routes
                routerAuth.js
                routerCenters.js

```

playerController.js M JS routerAuth.js U JS authController.js U App.jsx M auth.jsx U .env U package.json M

```

1 import express from 'express'
2 import { createUser, verifyToken } from '../controllers/authController.js'
3 import { check } from 'express-validator'
4
5 const router = express.Router()
6
7 router.post('/', [
8     check('email', 'Por favor digite un email válido').isEmail(),
9     check('password', 'Por favor ingrese un password con más de 8 caracteres').isLength({ min: 8 })
10 ],
11 createUser)
12
13 router.get('/verify', verifyToken)
14
15
16 export default router
17

```

En el archivo app.jsx

EXPLORER

```

    VITE-REACT-M...
        node
            routes
                routerAuth.js
                routerCenters.js
                routerDeptos.js
                routerMunicipios.js
                routerPlayers.js
            app.js
            package-lock.json
            package.json
        tomo-sena-mysql
            node_modules
            public
            src
                assets
                auth
                    auth.jsx
                centers
                    crudCenters.jsx
                    formCenters.jsx
                home
                players
                    App.jsx M
                    main.jsx
                    .eslintrc.js
                    .gitignore
                    index.html
            package-lock.json
            package.json
            README.md
            vite.config.js

```

routerAuth.js U JS authController.js U App.jsx M localhost:8000/players/1 auth.jsx U .env U package.json M

```

1 import { useEffect, useState } from 'react'
2
3 //Importar componentes necesarios para el sistema de rutas en REACT
4 import { Routes, Route, Link, Navigate } from 'react-router-dom'
5 //importar componentes que van en las rutas
6 import Home from './Home/Home'
7 import CrudPlayers from './players/crudPlayers'
8 import CrudCenters from './centers/crudCenters'
9 import Auth from './auth/auth'
10 import axios from 'axios'
11
12 const URI_AUTH = 'http://localhost:8000/auth/' ←
13
14 function App() {
15     const [isAuth, setIsAuth] = useState(false) //Prop que muestra si hay usuario autenticado o no ←
16
17     useEffect(() => {
18
19         //Verificar que hay usuario autenticado
20         const user = JSON.parse(localStorage.getItem('userTorneo')) //Obtener variable del localStorage ←
21
22         if (!user) {
23
24             setIsAuth(false) //Si no existe la variable se establece en falso la autenticación ←
25
26         } else {
27
28             //Si existe la variable en localStorage se verifica la autenticidad del token
29             axios.get(URI_AUTH + 'verify', {
30                 headers: { Authorization: `Bearer ${user.tokenUser}` }
31             }).then(response => {
32                 if (response.status === 200) {
33                     setIsAuth(true) // Si todo es correcto, se establece en TRUE la autenticación ←
34                 }
35             }).catch(() => {
36                 setIsAuth(false) // Si hay un error se establece en FALSE la autenticación ←
37             })
38         }
39     }, [])

```

```

routes
  JS routerAuth.js
  JS routerCenters.js
  JS routerDeptos.js
  JS routerMunicipios.js
  JS routerPlayers.js
  JS app.js
  () package-lock.json
  () package.json
  torneo-sena-mysql
    node_modules
    public
  src
    assets
    auth
      auth.jsx
    centers
      crudCenters.jsx
      formCenters.jsx
    home
    players
      App.jsx
      main.jsx
      .eslintrc.js
      .gitignore
      index.html
      package-lock.json
      package.json
      README.md
      vite.config.js

```

```

40  return (
41    <>
42      <nav>{/* Crear una barra sencilla de navegación temporalmente */
43        <ul>
44          <li>
45            <Link to="/">Principal</Link> /* En lugar de usar etiqueta <a> se utiliza el componente <Link></Link> */
46          </li>
47          <li>
48            <Link to="/players">Players</Link>
49          </li>
50          <li>
51            <Link to="/centers">Centers</Link>
52          </li>
53          <li>
54            <Link to="/auth">Sesión</Link>
55          </li>
56        </ul>
57      </nav>
58      {/*Aqui empieza la gestión de rutas */}
59
60      <Routes>
61        {/* en el componente Route van dos propiedades, la primera es path y la segunda es element, en path va la dirección donde
62        element va el componente que se quiere mostrar.*/
63        <Route path='/' element={<Home />} />
64
65        {isAuth ? //Operador ternario para preguntar si hay sesión iniciada o no
66          <Route path='/players' element={<CrudPlayers />} />
67          <Route path='/centers' element={<CrudCenters />} />
68        :}
69        :
70        <Route path='*' element={<Navigate to="/" />} />
71      }
72
73      <Route path='/auth' element={<Auth />} />
74    </Routes>
75  </>
76)

```

Resultado en navegador con sesión iniciada:

localhost:5173/centers

Centros de formación

Código de centro	Departamento	Municipio	Nombre del centro	Acciones
9123	Tolima	977	CENTRO AGROPECUARIO LA GRANJA	
9310	Tolima	962	CENTRO DE COMERCIO Y SERVICIOS	
9226	Tolima	962	CENTRO INDUSTRIA Y CONSTRUCCION	
9209	D. C. Santa Fe de Bogotá 149		CENTRO DE TECNOLOGIAS PARA LA CONSTRUCCION Y LA MADERA	
9210	D. C. Santa Fe de Bogotá 149		CENTRO DE ELECTRICIDAD Y ELECTRONICA	

Código:
 Departamento:
 Municipio:
 Nombre Centro:
 Enviar:

DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show again

Application

- Manifest
- Service workers
- Storage

Storage

- Local storage
- Session storage
- IndexedDB
- Cookies
- Private state tokens
- Interest groups
- Shared storage
- Cache storage

Background services

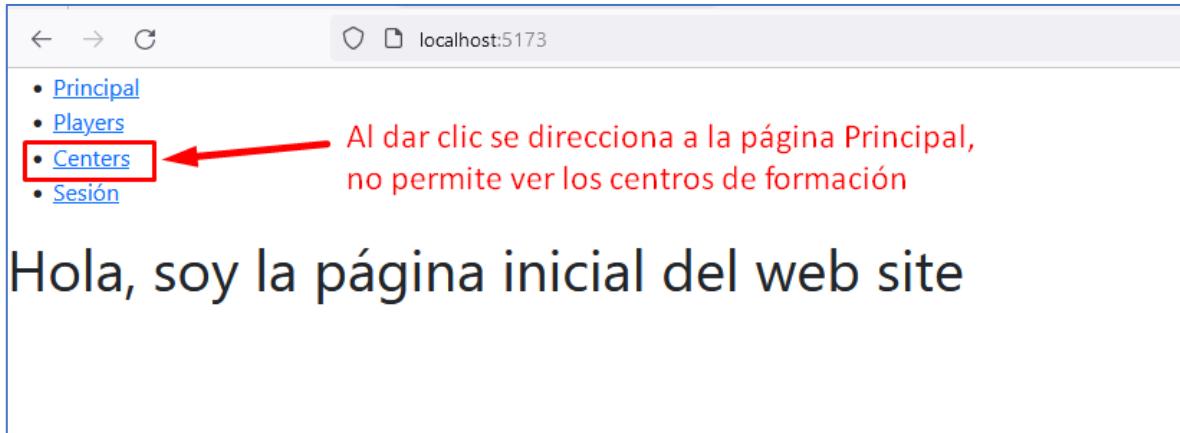
- Back/forward cache
- Background fetch
- Background sync
- Bounce tracking mitigation
- Notifications

Key Value

userTorneo {"tokenUser": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eYJic2VyIj}

tokenUser: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eYJic2VyIj

Resultado en navegador SIN el Token:



Organizar formulario para que tenga las dos funciones (registrar e iniciar sesión):

En el componente de autenticación auth.jsx:

```

const [buttonForm, setButtonForm] = useState('Registrar')
const [signInOrLogIn, setSignInOrLogIn] = useState('signIn')

const sendForm = async (e) => {
  e.preventDefault()

  if (buttonForm == 'Registrar') {
    console.log('Registrando ando ...')

    await axios.post(URI_AUTH, {
      name: name,
      email: email,
      password: password
    }).then(response => {
      if (response.data.tokenUser) {
        localStorage.setItem('userTorneo', JSON.stringify(response.data)) //Recibir el token y registrarlo en el localStorage
      }
    })
  }
}

const switchForm = (opcion) => {
  setSignInOrLogIn(opcion)
}

```

Agregar botón para cambiar la función del formulario:

```

VITE-REACT-M...  EXPLORER JS playerController.js M JS routerAuth.js U JS authController.js U ⚡ App.jsx M auth.jsx U .env U package.json M formCenters.jsx crudCenters.jsx
node
  routes
    JS routerAuth.js
    JS routerCenters.js
    JS routerDepts.js
    JS routerMunicipios.js
    JS routerPlayers.js M
    app.js M
    package.json M
    package-lock.json M
    package-lock.json
    torneo-sena-mysql M
      node_modules
        > public
          > assets
            > auth
              auth.jsx U
      src
        > assets
          > auth
            auth.jsx
        > centers
          crudCenters.jsx
          formCenters.jsx
        > home
        > players
        App.jsx
        main.jsx
        .eslintrc.js
        .gitignore
        index.html
        package-lock.json
        package.json
        README.md
        vite.config.js
  OUTLINE

```

```

JS auth.jsx
46 const Auth = () => {
47   return (
48     <>
49       {
50         signInOrLogin == 'signIn'
51           ? //Operador ternario para mostrar u ocultar botones para registro o inicio de sesión
52             <button className="btn btn-primary" onClick={() => { switchForm('logIn'); setButtonForm('Iniciar Sesión') }}>Iniciar sesión</button>
53             :
54           <span className="btn btn-primary" onClick={() => { switchForm('signIn'); setButtonForm('Registrarse') }}>Registrarse</span>
55       }
56     </>
57   )
58   <form onSubmit={sendForm}>
59     {
60       signInOrLogin == 'signIn'
61         ? //Operador ternario para ocultar el input para el nombre, solo se muestra si se quiere registrar
62           <>
63             <label htmlFor="name">Nombre completo:</label>
64             <input type="text" id="name" value={name} onChange={(e) => setName(e.target.value)} />
65           </>
66         :
67           <br />
68           <label htmlFor="email">Correo electrónico:</label>
69           <input type="email" id="email" value={email} onChange={(e) => setEmail(e.target.value)} />
70           <br />
71           <label htmlFor="password">Password:</label>
72           <input type="password" id="password" value={password} onChange={(e) => setPassword(e.target.value)} />
73           <br />
74           <input type="submit" value={buttonForm} className="btn btn-success" />
75         </form>
76       )
77     </>
78   }
79   </>
80 }
81
82 export default Auth

```

Resultado:

Formulario listo para registro	Formulario listo para inicio de sesión
<p>localhost:5173/auth</p> <ul style="list-style-type: none"> Principal Players Centers Sesión <p>Iniciar sesión</p> <p>Nombre completo: <input type="text"/></p> <p>Correo electrónico: <input type="email"/></p> <p>Password: <input type="password"/></p> <p>Registrar</p>	<p>localhost:5173/auth</p> <ul style="list-style-type: none"> Principal Players Centers Sesión <p>Registrarse</p> <p>Correo electrónico: <input type="email"/></p> <p>Password: <input type="password"/></p> <p>Iniciar Sesión</p>

Funcionalidad para formulario de inicio de sesión:

En el controlador `authController.js` definir método para inicio de sesión:

```
node > controllers > JS authController.js > verifyToken
41 export const verifyToken = (req, res) => {
42   res.json({ message: 'Verificación de token exitosa' })
43 }
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58 export const logInUser = async (req, res) => {
59   const { email, password } = req.body //Recibir los datos del form
60
61   try {
62     const userOk = await UserModel.findOne({ where: { email: email } }) //Buscar el usuario con ese correo
63
64     if (!userOk || !bcryptjs.compareSync(password, userOk.password)) { //preguntar si no existe o si el password es incorrecto
65       res.status(401).json({ message: 'Usuario o clave inválidos' })
66     } else {
67       //Si el usuario y el password son correctos, se crea la variable token
68       const tokenUser = jwt.sign({ user: { email: userOk.email } }, process.env.JWT_LLAVE, { expiresIn: '4h' })
69
70       res.json({ tokenUser }) // se retorna el token
71     }
72   } catch (error) {
73     res.status(500).json({ message: error.message })
74   }
75
76
77
78
79
80
81
82 }
```

En archivo de rutas agregar el método para login:

The screenshot shows a code editor with a sidebar containing project files and a main editor area displaying the `routerAuth.js` file. The file contains code for an Express router. Two specific lines of code are highlighted with red boxes and arrows pointing to them:

- Line 16: `router.post('/login', logInUser)`
- Line 2: `import { createUser, verifyToken, logInUser } from '../controllers/authController.js'`

Red arrows also point from the sidebar's `routes` folder to the `routerAuth.js` file in the main area, and from the `routes` folder to the bottom of the sidebar, indicating the relationship between the routes and the controllers.

```
EXPLORER          ...
JS playerController.js M   main.jsx   JS routerAuth.js U   JS authController.js U   App.jsx M   auth.jsx U   JS db.jsx

VITE-REACT-M... D+ ⏪ ⏴ ⏵

node
  controllers
    authController.js U
    centerController.js
    deptoController.js
    municipioController.js
    playerController.js M
  database
    db.js
  env
  models
    centerModel.js
    deptosModel.js
    municipioModel.js
    playerModel.js M
    userModel.js U
  node_modules
  public
  routes

JS routerAuth.js U
JS routerCenters.js
JS routerDeptos.js
JS routerMunicipios.js
JS routerPlayers.js M
JS app.js M
{} package-lock.json M

node > routes > JS routerAuth.js > ...
1 import express from 'express'
2 import { createUser, verifyToken, logInUser } from '../controllers/authController.js'
3 import { check } from 'express-validator'
4
5 const router = express.Router()
6
7 router.post('/', [
8   check('email', 'Por favor digite un email válido').isEmail(),
9   check('password', 'Por favor ingrese un password con más de 8 caracteres').isLength({ min: 8 })
10 ],
11   createUser)
12
13 router.get('/verify', verifyToken)
14
15 router.post('/login', logInUser)
16
17
18 export default router
19
```

En componente auth.jsx configurar la parte de la función sendForm para iniciar sesión:

```

    const Auth = () => {
      const sendForm = async (e) => {
        e.preventDefault()

        if (buttonForm == 'Registrar') {
          console.log('Registrando ando ...')

          await axios.post(URI_AUTH, {
            name: name,
            email: email,
            password: password
          }).then(response => {
            if (response.data.tokenUser) {
              localStorage.setItem('userTorneo', JSON.stringify(response.data)) //Recibir el token y registrarlo en el localStorage
            }
          })
        } else if (buttonForm == 'Iniciar Sesión') {
          console.log('Iniciando ando ...')

          await axios.post(URI_AUTH + 'login', {
            email: email,
            password: password
          }).then(response => {
            if (response.data.tokenUser) {
              localStorage.setItem('userTorneo', JSON.stringify(response.data)) //Recibir el token y registrarlo en el localStorage
              let miHost = window.location.host //Trae la dirección del servidor junto con el puerto, en este caso http://localhost:5173/
              console.log(miHost)
              window.location.href = miHost.toString() //Convertir en texto la dirección del host y cargarla
            }
          })
        }
      }
    }
  
```

En el archivo App.jsx se harán 3 cosas:

- Agregar useNavigate para redireccionar páginas.
- Crear función para cerrar sesión.
- Definir botón de cerrar sesión.
- Ocultar o mostrar el botón de cierre de sesión.

```

import { useEffect, useState } from 'react'
//Importar componentes necesarios para el sistema de rutas en REACT
//Importar { Routes, Route, Link, Navigate, useNavigate } from 'react-router-dom'
//Importar componentes que van en las rutas
import Home from './home/Home'
import CrudPlayers from './players/crudPlayers'
import CrudCenters from './centers/crudCenters'
import Auth from './auth/auth'
import axios from 'axios'

const URI_AUTH = 'http://localhost:8000/auth/'

function App() {
  const [isAuth, setIsAuth] = useState(false) //Prop que muestra si hay usuario autenticado o no

  const navigate = useNavigate(); // Usa useNavigate para redirección

  useEffect(() => {
    //Verificar que hay usuario autenticado
    const user = JSON.parse(localStorage.getItem('userTorneo')) //Obtener variable del localStorage

    if (user) {
      setIsAuth(true) //Si no existe la variable se establece en falso la autenticación
    } else {
      //Si existe la variable en localStorage se verifica la autenticidad del token
      axios.get(URI_AUTH + 'verify', {
        headers: { Authorization: `Bearer ${user.tokenUser}` }
      }).then(response => {
        if (response.status === 200) {
          setIsAuth(true) // Si todo es correcto, se establece en TRUE la autenticación
        }
      }).catch(() => {
        setIsAuth(false) // Si hay un error se establece en FALSE la autenticación
      })
    }
  }, [])
}

```

```

    const logOutUser = () => {
      localStorage.removeItem('userTorneo') //Borrar variable del localStorage
      setIsAuth(false) //Establecer en falso el inicio de sesión
      navigate("/auth") //Redireccionar a la página de inicio de sesión
    }
  }

  return (
    <>
      <nav>{/* Crear una barra sencilla de navegación temporalmente */
        <ul>
          <li>
            <Link to="/">Principal</Link> {/* En lugar de usar etiqueta <a> se utiliza el componente <Link>/<Link> */}
          </li>
          <li>
            <Link to="/players">Players</Link>
          </li>
          <li>
            <Link to="/centers">Centers</Link>
          </li>
          <li>
            <Link to="/auth">Sesión</Link>
          </li>
        </ul>
      </nav>
      {isAuth ? //operador ternario
        <li>
          <button onClick={() => logOutUser()} className='btn btn-secondary'><i className="fa-solid fa-door-closed"></i>Cerrar sesión</button>
        </li>
      : ''}
    </>
  )
}

```

Ajustes para mostrar y ocultar ruta de autenticación:

```

    !isAuth // Si no está autenticado muestra el botón de sesión
    ?
    <li>
      <button onClick={() => logOutUser()} className='btn btn-secondary'><i className="fa-solid fa-door-closed"></i>Cerrar sesión</button>
    </li>
    :
    ..
  }

  {isAuth ? //operador ternario
    <li>
      <button onClick={() => logOutUser()} className='btn btn-secondary'><i className="fa-solid fa-door-closed"></i>Cerrar sesión</button>
    </li>
  : ''}

```

```

VITE-REACT-M...  ...
EXPLORER ... JS playerController.js M main.jsx JS routerAuth.js U JS authController.js U JS App.jsx M auth.jsx U crudPlayers.jsx M
node package-lock.json package.json
torneo-sena-mysql node_modules public
src assets auth.jsx centers home
players crudPlayers.jsx M formPlayers.jsx M formQueryPlayer.j... M
App.jsx M main.jsx .eslintrc.js .gitignore index.html package-lock.json package.json README.md vite.config.js
OUTLINE
TIMELINE

```

```

JS playerController.js M main.jsx JS routerAuth.js U JS authController.js U JS App.jsx M auth.jsx U crudPlayers.jsx M
14 function App() {
15   /* Aquí comienza la descripción de rutas */
16
17   <Routes>
18     {/* en el componente Route van dos propiedades, la primera es path y la segunda es element, en path va la dirección donde queremos mostrar el componente, y en element va el componente que se quiere mostrar*/}
19     <Route path='/' element={<Home />} />
20
21     {isAuth ?//Operador ternario para preguntar si hay sesión iniciada o no
22      <>
23        <Route path='/players' element={<CrudPlayers />} />
24        <Route path='/centers' element={<CrudCenters />} />
25      </>
26      :
27        <Route path='*' element={<Navigate to="/" />} />
28      }
29
30    {
31      !isAuth //Si no está autenticado se deja acceso a la ruta de sesión
32      ?
33        <Route path='/auth' element={<Auth />} />
34      :
35        ...
36    }
37
38  </Routes>
39
40}
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119

```

Recuperación de contraseña

Fase 1: Instalar nodemailer

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\vite-react-mysql\node> npm install nodemailer
added 1 package, and audited 136 packages in 3s

15 packages are looking for funding
  run `npm fund` for details

2 vulnerabilities (1 moderate, 1 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS D:\vite-react-mysql\node>

```

Fase 2: Definir archivo para el envío del correo con el enlace que restablece la contraseña. El objetivo es enviarle un correo al usuario, y para eso se necesita habilitar una cuenta de correo electrónico que envíe el mensaje, en este caso se hará el ejemplo con la cuenta personal, sin embargo, usted debe usar su propia cuenta de correo.

Debe crear una carpeta llamada 'servicios' y dentro el archivo 'emailServices.js'

```
EXPLORER JS centerController.js main.jsx JS routerAuth.jsx JS authController.jsx App.jsx M auth.jsx U JS emailService.jsx X JS app.js M .env  
VITE-RE... node controllers authController.jsx U deptoController.jsx municipioController.jsx playerController.jsx database env .env models node_modules public routes servicios emailService.jsx U app.js M package-lock.json M package.json M torneo-sena-my... node_modules public src assets auth  
centerController.js  
1 import nodemailer from 'nodemailer'  
2  
3 export const transportador = nodemailer.createTransport({  
4   host: 'smtp.gmail.com',  
5   port: 587,  
6   secure: false,  
7   auth: {  
8     user: 'jorgejcp@gmail.com',  
9     pass: 'aqui va la contraseña sin espacios'  
10   }  
11 })  
12  
13 export const sendPasswordResetEmail = async (email, tokenForPassword) => {  
14  
15   const RESET_URL = `http://localhost:5173/reset-password?llave=${tokenForPassword}` //El host es donde se ejecuta el FRONT-END no el BACK-END  
16   const mailOptions = {  
17     from: 'jorgejcp@gmail.com',  
18     to: email,  
19     subject: 'Restablecer contraseña',  
20     text: 'Por favor use el siguiente enlace para restablecer su contraseña: ${RESET_URL}'  
21   }  
22  
23   await transportador.sendMail(mailOptions)  
24  
25 }  
26
```

Para que esto funcione debe **Habilitar la verificación en dos pasos** en la cuenta de Google. Se debe crear una clave para usar con aplicaciones, es diferente a la del correo, sin embargo, no son temas que se expliquen en este tutorial. Para la configuración debe seguir el enlace: <https://support.google.com/mail/answer/185833?hl=es-419>

Fase 3: En el controlador crear dos funciones:

1. Para enviar el correo con el enlace de restablecimiento de contraseña.
 2. Para recibir la nueva contraseña y actualizarla en el registro del usuario.

```
EXPLORER JS playerController.js M JS centerController.js main.jsx JS routerAuthjs U JS authController.js U X JS Appjsx M auth.jsx U ⚙️ ... VITE-RE... 🌐 🎯 🔍 🔍 node controllers JS authController.js U JS centerController.js JS deptoController.js JS municipioController... JS playerController.js M > database env .env U > models node_modules public routes servicios JS emailService.js U JS app.jsx M (1) package-lock.json M (1) package.json M torneo-sena-m... node_modules public src assets auth centers OUTLINE TIMELINE
```

```
JS authController.js U
node > controllers > JS authController.js ...
1 import bcryptjs from 'bcryptjs'
2 import UserModel from '../models/userModel.js'
3
4 import jwt from 'jsonwebtoken'
5 import { sendPasswordResetEmail } from './servicios/emailService.js' ← Red arrow points here
6 import { where } from 'sequelize'
7
8 export const createUser = async (req, res) => { //Función para crear usuarios
9
10    try {
11        //Cambia la manera de leer los datos que llegan del formulario
12        const { name, email, password } = req.body
13        //Validar si ya existe un usuario con el mismo correo
14        let user = await UserModel.findOne({ where: { email: email } })
15        if (user) { //Si el usuario existe, no permite la creación del usuario
16            res.json({ "message": "El usuario ya existe" })
17        } else { //Si el usuario no existe, permite la creación
18            //Encriptan el password
19            let passHash = await bcryptjs.hash(password, 6) //el número representa la cantidad de veces que se ejecuta el hash para
20            //encriptar la clave
21            //Enviar datos a base de datos
22            const userOk = await UserModel.create({
23                "name": name,
24                "email": email,
25                "password": passHash
26            })
27            //Para generar el token se asignan 3 elementos en este caso: 1ro los datos que se obtienen del usuario, 2do la llave del
28            //archivo .env y 3ro la duración de la sesión
29            const tokenUser = jwt.sign({ user: { email: userOk.email } }, process.env.JWT_LLAVE, { expiresIn: '4h' })
30
31            // console.log("TOKEN: " + tokenUser)
32        }
33    }
34}
```

```
JS authController.js U node > controllers > JS authController.js > getResetPassword
86 export const getResetPassword = async (req, res) => { //función para
87     const { email } = req.body //recibir el correo
88
89     const user = UserModel.findOne({ where: { email: email } }) //consultar al usuario con ese correo
90
91     if (!user) {
92
93         res.status(404).json({ message: 'usuario no encontrado' }) //si el usuario no existe se envía este mensaje
94
95     } else {
96
97         const tokenForPassword = jwt.sign({ user: { id: user.id, name: user.name, email: user.email } }, process.env.JWT_LLAVE, {
98             expiresIn: '30m'
99         })
100         //en este token, a manera de ejemplo, se traen más datos del usuario (id, name, email)
101         await sendPasswordResetEmail(email, tokenForPassword)
102
103         res.status(200).json({ message: 'El mensaje para restablecer contraseña fue enviado correctamente' })
104
105     }
106
107     export const setNewPassword = async (req, res) => {
108
109         const { tokenForPassword, newPassword } = req.body
110
111         try {
112             // extrae los datos que se enviaron en el token, estos datos reciben el nombre de payload
113             const decodificado = jwt.verify(tokenForPassword, process.env.JWT_LLAVE)
114             const user = await UserModel.findByPk(decodificado.id) //la función findByPk del modelo, buscar un registro por su Primary Key
115
116             if (user) {
117                 user.password = newPassword
118                 await user.save()
119
120                 res.status(200).json({ message: 'Contraseña actualizada exitosamente' })
121             } else {
122                 res.status(404).json({ message: 'Usuario no encontrado' })
123             }
124
125         } catch (error) {
126             res.status(500).json({ message: 'Error al intentar cambiar la contraseña' })
127         }
128     }
129 }
```

The screenshot shows a code editor with a sidebar containing a project tree. The main area displays a file named `authController.js`. Several red arrows point to specific lines of code, highlighting parts of the logic related to password reset and user authentication.

```
JS authController.js U JS centerController.js main.jsx JS routerAuth.js U JS authController.js X crudCenters.jsx formCenters.jsx App.jsx M auth.jsx U emailService.js U

node > controllers > JS authController.js ...
86 export const getResetPassword = async (req, res) => { //Función para
87 }
88
89 export const setNewPassword = async (req, res) => { //Función que recibe la nueva contraseña
90 }
91
92 const { tokenForPassword, newPassword } = req.body
93
94 try {
95     // extrae los datos que se enviarion en el token, estos datos reciben el nombre de payload
96     const decodificado = jwt.verify(tokenForPassword, process.env.JWT_LIVE)
97
98     const user = await UserModel.findById(decodificado.user.id) //la función findById del modelo, buscar un registro por su Primary Key
99     //recuerde que cuando se creo el token se puso una variable llamada user y dentro los campos, por ese motivo la instrucción es 'decodificado.user.id'
100
101     if (!user) {
102
103         res.status(404).json({ message: 'Usuario no encontrado' })
104
105     } else {
106         //si el usuario existe, se hace el proceso para encriptar el nuevo password
107         let passHash = await bcryptjs.hash(newPassword, 6) //el número representa la cantidad de veces que se ejecuta el hash para encriptar la clave
108         //se actualiza el password del usuario
109         await UserModel.update({
110             password: passHash
111         }, { where: { id: decodificado.user.id } })
112
113         res.status(200).json({ message: 'Contraseña actualizada correctamente' })
114
115     } catch (error) {
116
117         res.status(400).json({ message: 'Información inválida o el tiempo ha expirado' })
118     }
119
120 }
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
```

Fase 4: Definir rutas para las dos funciones del controlador en el archivo routerAuth.js

```

EXPLORER ... JS centerController.js main.jsx JS routerAuth.js U JS authController.js U App.jsx M auth.jsx U JS emailService.js U

VITE-RE... routes > JS routerAuth.js ...
  node
    controllers
      JS authController.js
      JS centerController.js
      JS deptoController.js
      JS municipioController.js
      JS playerController.js
    database
    env
      .env
    models
    node_modules
    public
    routes
      JS routerAuth.js U
        JS routerCenters.js
        JS routerDeptos.js
        JS routerMunicipios.js
        JS routerPlayers.js M
      servicios
        JS emailService.js U

```

```

1 import express from 'express'
2 import { createUser, verifyToken, logInUser, getResetPassword, setNewPassword } from '../controllers/authController.js'
3 import { check } from 'express-validator'
4
5 const router = express.Router()
6
7 router.post('/', [
8   [check('email', 'Por favor digite un email válido').isEmail(),
9    check('password', 'Por favor ingrese un password con más de 8 caracteres').isLength({ min: 8 })],
10   createUser
11 ],
12
13 router.get('/verify', verifyToken)
14 router.post('/login', logInUser)
15
16
17 router.post('/request-password-reset', getResetPassword)
18 router.post('/reset-password', setNewPassword)
19
20
21 export default router

```

Fase 5: En el FRONT-END agregar al componente `auth.jsx` un formulario para restablecer la contraseña, también se agrega un enlace para activar dicho formulario y ocultar lo relacionado con el inicio de sesión y el registro de usuarios:

```

EXPLORER ... JS centerController.js main.jsx JS routerAuth.js U JS authController.js U App.jsx M auth.jsx U JS emailService.js U JS api ...
VITE-RE... src > auth > auth.jsx > Auth
  node
    routes
      JS routerPlayers.js M
    servicios
      JS emailService.js U
    app.js M
    package-lock.json M
    package.json M
  torneo-sena-m...
    node_modules
      public
      src
        assets
        auth
          auth.jsx U
            JS authController.js U
            JS centerController.js
            JS main.jsx
            JS routerAuth.js U
            JS authController.js U
            JS App.jsx M
            JS auth.jsx U X JS emailService.js U
            JS api ...

```

```

1 import axios from "axios"
2 import { useState } from "react"
3 import { Link } from "react-router-dom"
4
5 const URI_AUTH = 'http://localhost:8000/auth/'
6
7 const Auth = () => {
8
9   const [name, setName] = useState('')
10  const [email, setEmail] = useState('')
11  const [password, setPassword] = useState('')
12
13  const [buttonForm, setButtonForm] = useState('Registrar')
14
15  const [signInOrLogIn, setSignInOrLogIn] = useState('signin')
16
17  const [resetPass, setResetPass] = useState(false) ←
18
19  const sendForm = async (e) => {
20
21    e.preventDefault()
22
23    if (buttonForm == 'Registrar') {
24
25      console.log('Registrando ando ...')
26
27      await axios.post(URI_AUTH, {
28        name: name,
29        email: email,
30        password: password
31      }).then(response => {

```

The screenshot shows a code editor with several red arrows and annotations pointing to specific parts of the code. The annotations highlight sections related to user authentication and registration logic.

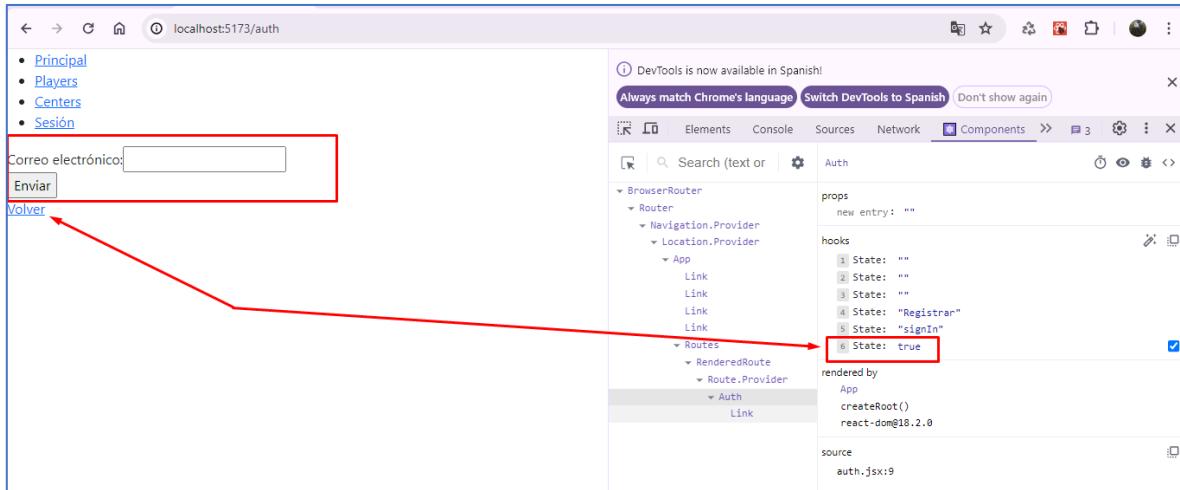
```
const Auth = () => {
  return (
    <>
      <div>
        <div>
          <button onClick={() => { setResetPass(!resetPass) }}>Restablecer contraseña</button>
          <button onClick={() => { setResetPass(!resetPass) }}>Volver</button>
        </div>
        <form onSubmit={sendForm}>
          <label htmlFor="name">Nombre completo:</label>
          <input type="text" id="name" value={name} onChange={(e) => setName(e.target.value)} />
        </form>
        <label htmlFor="email">Correo electrónico:</label>
        <input type="email" id="email" value={email} onChange={(e) => setEmail(e.target.value)} />
        <br />
        <label htmlFor="password">Password:</label>
        <input type="password" id="password" value={password} onChange={(e) => setPassword(e.target.value)} />
        <br />
        <input type="submit" value="buttonForm" className="btn btn-success" />
      </div>
    </div>
  )
}

Auth.propTypes = {
  name: PropTypes.string,
  email: PropTypes.string,
  password: PropTypes.string,
  sendForm: PropTypes.func,
}
```

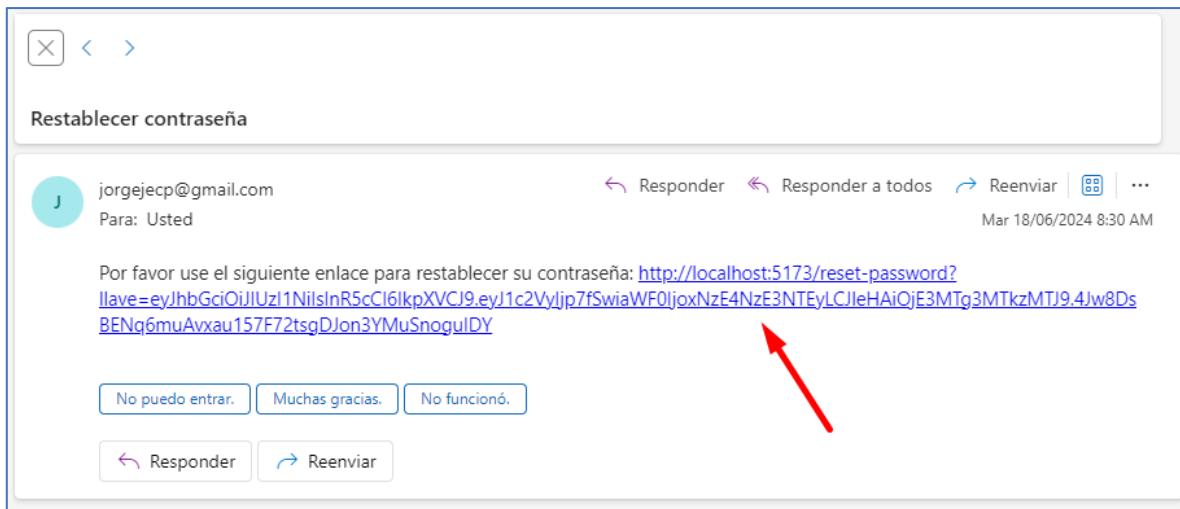
Resultado:

The screenshot shows a browser window with the URL `localhost:5173/auth`. On the left, there is a sidebar with navigation links: **Principal**, **Players**, **Centers**, and **Sesión**. Below this is a form titled "Iniciar sesión" with fields for "Nombre completo", "Correo electrónico", and "Password". There are two buttons: "Registrar" (green background) and "Restablecer contraseña" (blue background). The "Restablecer contraseña" button has a red arrow pointing to it from the bottom-left. On the right, the `Components` tab of the DevTools is open, showing the component tree for the `Auth` component. The tree includes `BrowserRouter`, `Router`, `Navigation.Provider`, `Location.Provider`, `App`, `Link`, `Link`, `Link`, `Link`, `Routes`, `RenderedRoute`, `Route.Provider`, and `Auth`. The `Auth` component has a prop `resetPass` set to `false`, which is highlighted with a red box and has a red arrow pointing to it from the bottom-right. The DevTools status bar at the top indicates "Always match Chrome's language" and "Switch DevTools to Spanish".

Al dar clic se oculta un formulario y se muestra el otro:



Cuando se realice la prueba solicitando el cambio de contraseña, llegará un correo así:



Fase 6: Crear el componente para que el usuario digite la nueva contraseña:

```
import { useState } from "react";
import axios from "axios";

const URI_AUTH = 'http://localhost:8000/auth/'

const ResetPassword = () => {
  const [newPassword, setNewPassword] = useState('')
  const [message, setMessage] = useState('')

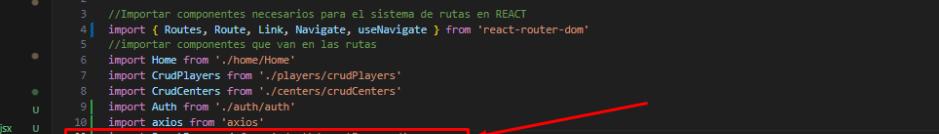
  //recibir el token que viene desde la URL, en este caso la variable se llama 'llave'
  const tokenForPassword = new URLSearchParams(location.search).get('llave'); //tokenForPassword es el nombre con el que la función del controlador espera el token
  const updatePassword = async (e) => { //Función que se activa con el envío del formulario
    e.preventDefault();
    try {

      const response = await axios.post(`${URI_AUTH}/reset-password`, { tokenForPassword, newPassword })
      setMessage(response.data.message)
      setNewPassword('')
    } catch (error) {
      setMessage(error.data.message)
    }
  }

  return (
    <>
      <form onSubmit={updatePassword}>
        <label>Nueva contraseña:</label>
        <input type="password" value={newPassword} onChange={(e) => setNewPassword(e.target.value)} required />
        <button type="submit">Reset Password</button>
        {message && <p className="bg-info">{message}</p>}
      </form>
    </>
  )
}

export default ResetPassword
```

Fase 7: Habilitar ruta de componente en App.jsx



```
EXPLORER          ... SX JS routerAuth.jsx U JS authController.jsx U @ crudCenters.jsx @ formCenters.jsx @ App.jsx M X @ auth.jsx U JS emailService.jsx U @ resetPassword.jsx U

VITE-REACT-M... D C O S
> node
+ tomeo-sena-mysql
  > node_modules
  > public
  > src
    > assets
    > auth
      @ auth.jsx U
      @ resetPassword.jsx U
    > centers
      @ crudCenters.jsx
      @ formCenters.jsx
    > home
    > players
      @ App.jsx M
        @ main.jsx
        @ .eslintrc.js
        @ .gitignore
        < index.html
        package-lock.json
        package.json
        README.md
        vite.config.js

JS routerAuth.jsx U JS authController.jsx U @ crudCenters.jsx @ formCenters.jsx @ App.jsx M X @ auth.jsx U JS emailService.jsx U @ resetPassword.jsx U

tomeo-sena-mysql > src > App.jsx ...
1 | import { useEffect, useState } from 'react'
2 |
3 | //Importar componentes necesarios para el sistema de rutas en REACT
4 | import { Routes, Route, Link, Navigate, useNavigate } from 'react-router-dom'
5 | //importar componentes que van en las rutas
6 | import Home from './home/Home'
7 | import CrudPlayers from './players/crudPlayers'
8 | import CrudCenters from './centers/crudCenters'
9 | import Auth from './auth/auth'
10| import axios from 'axios'
11| import ResetPassword from './auth/resetPassword' -----
12|
13| const URI_AUTH = 'http://localhost:8000/auth/'
14|
15| function App() {
16|   const [isAuth, setIsAuth] = useState(false) //Prop que muestra si hay usuario autenticado o no
17|
18|   const navigate = useNavigate(); // Usa useNavigate para redirección
19|
20|   useEffect(() => {
21|
22|     //Verificar que hay usuario autenticado
23|     const user = JSON.parse(localStorage.getItem('userTorneo')) //Obtener variable del localStorage
24|
25|     if (!user) {
26|
27|       setIsAuth(false) //Si no existe la variable se establece en falso la autenticación
28|     }
29|   })
30| }

const URI_AUTH = 'http://localhost:8000/auth/'

function App() {
  const [isAuth, setIsAuth] = useState(false) //Prop que muestra si hay usuario autenticado o no

  const navigate = useNavigate(); // Usa useNavigate para redirección

  useEffect(() => {

    //Verificar que hay usuario autenticado
    const user = JSON.parse(localStorage.getItem('userTorneo')) //Obtener variable del localStorage

    if (!user) {

      setIsAuth(false) //Si no existe la variable se establece en falso la autenticación
    }
  })
}
```

```

VITE-REACT-M...
sx routerAuth.jsx U authController.jsx crudCenters.jsx formCenters.jsx App.jsx M ...
tomeo-sena-mysql > src > App.jsx ...
node
torneo-sena-mysql
public
src
assets
auth
auth.jsx
resetPassword.jsx
centers
crudCenters.jsx
formCenters.jsx
home
players
App.jsx M
main.jsx
.eslintrc.js
.gitignore
index.html
package-lock.json
package.json
README.md
vite.config.js

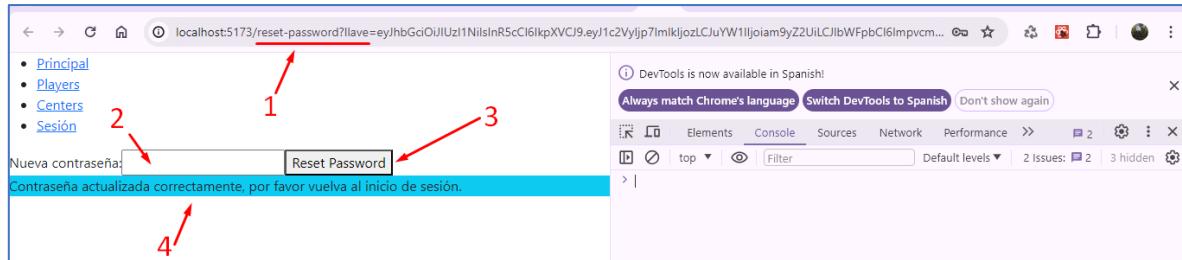
```

```

15  function App() {
92
93    <Routes>
94      /* en el componente Route van dos propiedades, la primera es path y la segunda es element, en path va la dirección donde queremos mostrar el com
95      element va el componente que se quiere mostrar.*/
96      <Route path='/' element={<Home />} />
97
98      {isAuth ?//Operador ternario para preguntar si hay sesión iniciada o no
99        <>
100          <Route path='/players' element={<CrudPlayers />} />
101          <Route path='/centers' element={<CrudCenters />} />
102        </>
103        :
104        <Route path='*' element={<Navigate to="/" />} />
105      }
106
107      {!isAuth //Si no está autenticado se deja acceso a la ruta de sesión
108        ?
109          <Route path='/auth' element={<Auth />} />
110        :
111      }
112
113      <Route path='/reset-password' element={<ResetPassword />} />
114
115    </Routes>
116  }
117
118
119
120  export default App
121

```

El resultado es el siguiente:



Paginación

Fase 1: Crear componente **pagination.jsx** directamente en la carpeta SRC, antes de digitar el código, haga una lectura de este teniendo en cuenta todos los comentarios:

Explorador de archivos:

```

VITE-REACT-M...
  node
    > public
    > routes
      JS routerAuth.js
      JS routerCenters.js
      JS routerDepts.js
      JS routerFuncipos.js
      JS routerPlayers.js
      JS services
      JS app.js
      ( package-lock.json
      ( package.json
    torneo-sena-mysql
      > node_modules
      > public
        > src
          > assets
          > auth
          > centers
          > home
          > players
            App.jsx
            main.jsx
          pagination.jsx
          ( eslintrc.js
          ( gitignore
          index.html
          ( package-lock.json
          ( package.json
          README.md
          vite.config.js

```

Contenido del archivo pagination.jsx:

```

playerController.js M main.jsx App.jsx M pagination.jsx U crudPlayers.jsx M JS routerPlayers.jsx M JS app.jsx M
  1 import axios from "axios"
  2 import { useState, useEffect } from "react"
  3
  4 const Pagination = ({ URI, setDesde, setHasta }) => {
  5   // Recibir variables desde el componente padre
  6
  7   const [numRegistros, setNumRegistros] = useState(0) // Prop para establecer el nro de registros que tiene la tabla en la base de datos
  8   const [registrosPorPagina, setRegistrosPorPagina] = useState(5) // Nro de registros a mostrar por pagina
  9   const [paginaActual, setpaginaActual] = useState(1) // Prop para saber la pagina actual, inicia siempre en 1
 10  const [paginas, setPaginas] = useState(0) // Prop para calcular la cantidad de paginas que se deben crear, por ejemplo, si hay 15 registros y se quieren mostrar 5
 11  const [ocultarMostrarAnterior, setOcultarMostrarAnterior] = useState("") // Prop para mostrar y ocultar, aqui se pondrá el nombre de una clase de bootstrap para DESHABILITAR el botón de anterior
 12  const [ocultarMostrarSiguiiente, setOcultarMostrarSiguiiente] = useState("") // Prop para mostrar y ocultar, aqui se pondrá el nombre de una clase de bootstrap para HABILITAR el botón de siguiente
 13  const [botones, setBotones] = useState([])
 14
 15  const getAllPlayers = async () => {
 16
 17    const respuesta = await axios.get(URI) // Usar la URI que llega del componente PADRE
 18
 19    let cantidadRegistros = respuesta.data.length
 20    setNumRegistros(cantidadRegistros) // Obtener la cantidad de registros de la tabla
 21
 22    // Calcular la cantidad de páginas (botones)
 23    let pages = Math.ceil(cantidadRegistros / registrosPorPagina) // Supongamos que hay 8 registros y se han asignado 5 registros por página, al hacer la división el resultado es 8 / 5 = 1,6. La función ceil de JS lo que hace es redondear el resultado a la cifra siguiente, o sea a 2 páginas
 24    setPaginas(pages)
 25
 26    // Aquí pasa algo interesante, tal vez se intente realizar el cálculo de páginas con la siguiente instrucción, la cual no funcionaría:
 27
 28    let pages = Math.ceil(numRegistros / registrosPorPagina)
 29
 30    // Por qué no funciona si se usa directamente la prop 'numRegistros'?
 31
 32    NO se puede usar en el cálculo directamente la prop 'numRegistros' porque 'numRegistros' es una variable de estado (state) que se actualiza de manera asíncrona. Cuando se intenta usar 'numRegistros' para calcular pages, el estado no se ha actualizado todavía.
 33
 34  }
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67

```

```

useEffect(() => {
  getAllPlayers()
  pagina(paginaActual) // Inicializar con la página 1
}, [pagina, paginaActual]) // El useEffect se ejecutará al cargar la página y cuando las prop pagina y página actual se actualicen
const pagina = (pagina) => {
  // Recibe el número de la página actual, esta función se encarga de mostrar el contenido de la nueva página
  setpaginaActual(pagina) // Establecer la página actual
  let desdePagina = ((paginaActual - 1) * registrosPorPagina) // Supongamos que la página actual es 2, 2 - 1 = 1, y 1 * 5 = 5 (quiere decir que la página 2 inicia desde el registro número 5)
  setDesde(desdePagina)
  let hastaPagina = paginaActual * registrosPorPagina // Supongamos que la página actual es 2, 2 * 5 = 10 (quiere decir que la página 2 termina en el registro 10)
  setHasta(hastaPagina)
  // Si la página actual es la primera, entonces se deshabilita el botón de anterior
  if (paginaActual === 1) {
    setOcultarMostrarAnterior("page-item disabled")
  } else {
    setOcultarMostrarAnterior("page-item")
  }
  // Si la página actual es la última, entonces se deshabilita el botón de siguiente
  if (paginaActual === paginas) {
    setOcultarMostrarSiguiiente("page-item disabled")
  } else {
    setOcultarMostrarSiguiiente("page-item")
  }
}

```

```

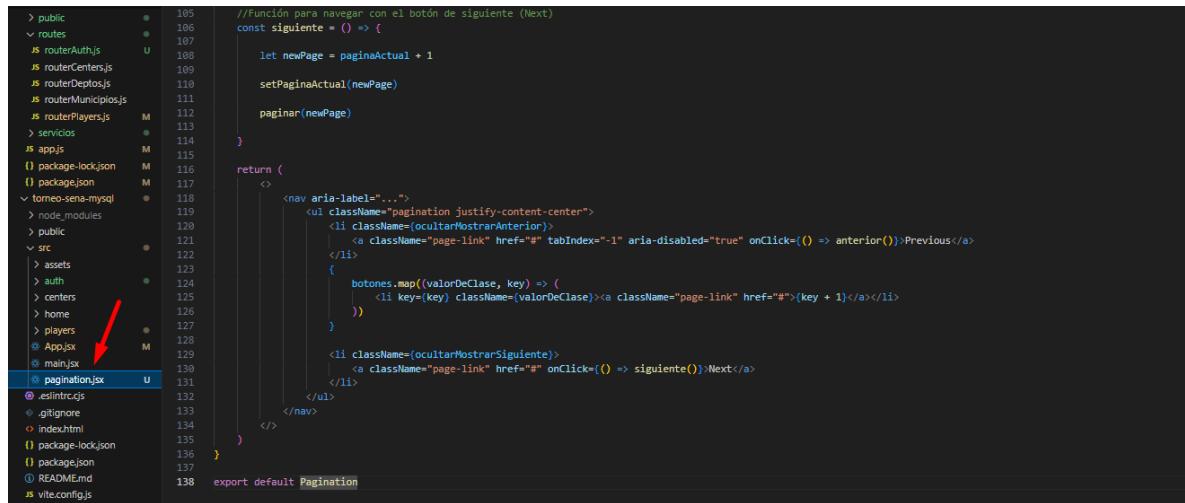
let arregloAuxiliar = []
// Pintar de color azul el botón actual (página actual)
// Crear un ciclo para que vaya desde cero hasta que sea menor a la cantidad de páginas existentes
for (var i = 0; i < paginas; i++) {
  // Objetivo: si la página actual es igual a la posición del arreglo de los botones, entonces se cambia la clase del <li> a active
  if ((i + 1) === paginaActual) { // Se le suma 1 a la i porque i empieza en cero (0) y no hay página cero, por eso se iguala a 1 para poder comparar
    arregloAuxiliar[i] = "page-item active"
  } else {
    arregloAuxiliar[i] = "page-item"
  }
}
setBotones(arregloAuxiliar)
// Fin función pagina
// Función para navegar con el botón de anterior (Previous)
const anterior = () => {
  let nuevaPage = paginaActual - 1
  setpaginaActual(nuevaPage)
  pagina(nuevaPage)
}

```

En la sección del return, se copia uno de los ejemplos de paginación de Bootstrap y se modifican las palabras class por className, de igual modo, se modifican las className de las etiquetas de los botones anterior y siguiente, esto se hace para activarlos o desactivarlos.

Se utiliza el arreglo **botones.map** en la línea 124 para dibujar un botón por cada página, en el className se pone el estilo para pintar el botón de la página actual.

La variable **key** sirve para mostrar el número en el botón, como esta empieza en cero (0), se le suma 1 para que no muestre la página cero (lo cuál sería un error).

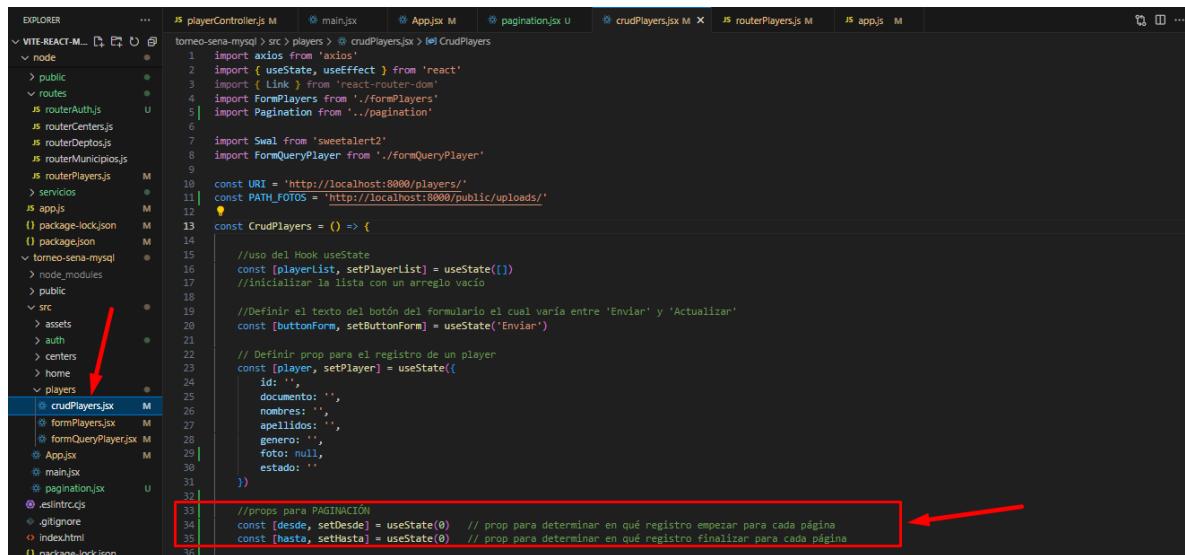


```

> public          • 105   //Función para navegar con el botón de siguiente (Next)
  < routes         • 106
    JS routerAuth.js U 107
    JS routerCenters.js 108
    JS routerDeptos.js 109
    JS routerMunicipios.js 110
    JS routerPlayers.js M 111
    > servicios      • 112
      JS app.js       M 113
        (1) package-lock.json M 114
        (1) package.json M 115
    < torneo-sena-mysql • 116
      > node_modules
        > public
          < src
            > assets
              > auth
              > centers
              > home
              > players
                App.jsx
                main.jsx
                pagination.jsx U
                (1) eslintrc.js
                (1) .gitignore
                index.html
                (1) package-lock.json
                (1) package.json
                (1) README.md
                vite.config.js
  138 export default Pagination

```

Fase 2: En el componente padre (en este caso crudPlayers.jsx) definir propiedades para definir los registros que se mostrarán en cada página, en cada página hay que saber **desde** y **hasta** donde mostrar los registros, por ejemplo, si se establece que muestre de a 5 registros por página, la página uno (1) mostrará desde el registro 1 hasta el registro 5, la página dos (2) mostrará desde el registro 6 hasta el registro 10, la página tres (3) mostrará desde el registro 11 hasta el registro 15, etc.

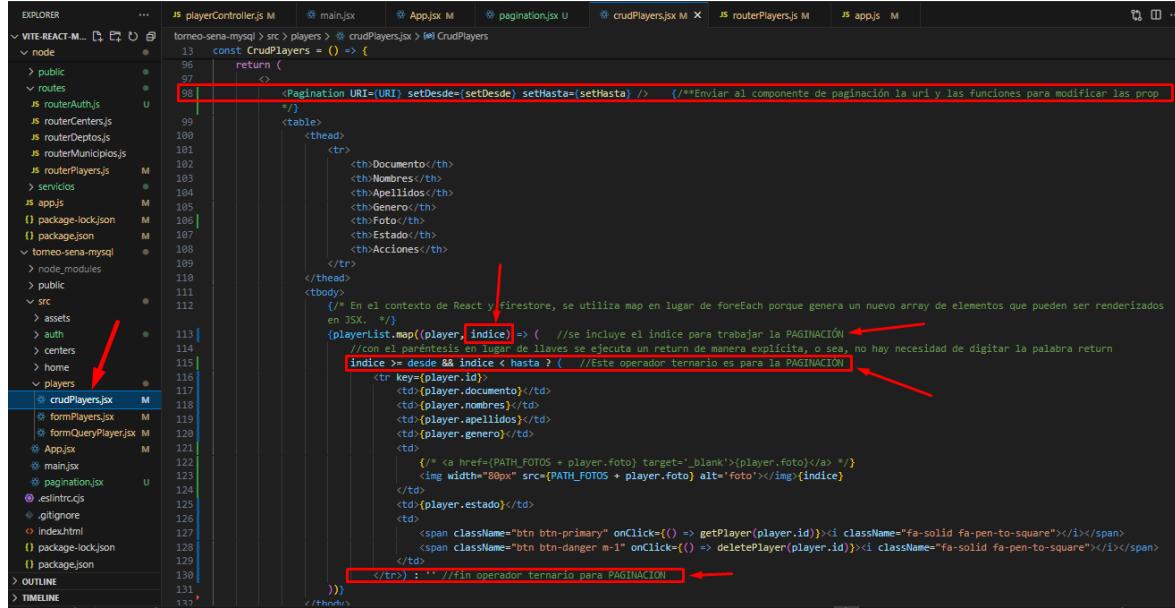


```

EXPLORER ... # playerController.js M main.jsx App.jsx pagination.jsx U crudPlayers.jsx M routerPlayers.js M app.js M
  < node
    < routes
      JS routerAuth.js U
      JS routerCenters.js
      JS routerDeptos.js
      JS routerMunicipios.js
      JS routerPlayers.js M
      > servicios
      JS app.js M
      (1) package-lock.json
      (1) package.json
    < torneo-sena-mysql
      > node_modules
        > public
          < src
            > assets
              > auth
              > centers
              > home
              > players
                crudPlayers.jsx M
                formPlayers.jsx M
                formQueryPlayer.jsx M
                App.jsx M
                main.jsx
                pagination.jsx U
                (1) eslintrc.js
                (1) .gitignore
                index.html
                (1) package-lock.json
  13 const CrudPlayers = () => {
  14
  15   //uso del Hook useState
  16   const [playerList, setPlayerList] = useState([])
  17   //Inicializar la lista con un arreglo vacío
  18
  19   //Definir el texto del botón del formulario el cual varía entre 'Enviar' y 'Actualizar'
  20   const [buttonForm, setButtonForm] = useState('Enviar')
  21
  22   //Definir prop para el registro de un player
  23   const [player, setPlayer] = useState({
  24     id: '',
  25     documentos: '',
  26     nombres: '',
  27     apellidos: '',
  28     genero: '',
  29     foto: null,
  30     estado: ''
  31   })
  32
  33   //props para PAGINACIÓN
  34   const [desde, setDesde] = useState(0) // prop para determinar en qué registro empezar para cada página
  35   const [hasta, setHasta] = useState(0) // prop para determinar en qué registro finalizar para cada página

```

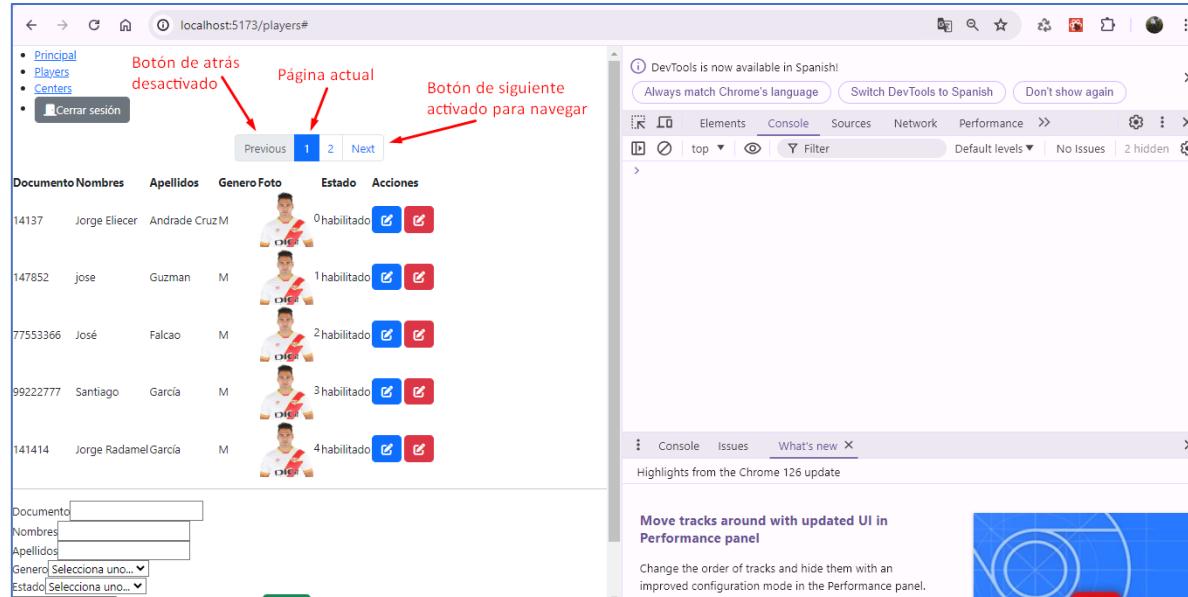
En el return del componente padre se debe agregar el componente de paginación y enviar a él las variables necesarias. Al momento de dibujar las filas de la tabla que muestra los registros, se debe poner un operador ternario que controle lo que se va a mostrar, para eso se agrega un índice a la iteración del map.



```

    const CrudPlayers = () => {
      return (
        <Pagination URI={URI} setDesde={setDesde} setHasta={setHasta} /> //Enviar al componente de paginación la url y las funciones para modificar las prop
      </table>
      <thead>
        <tr>
          <th>Documento</th>
          <th>Nombres</th>
          <th>Apellidos</th>
          <th>Genero</th>
          <th>Foto</th>
          <th>Estado</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        /* En el contexto de React y firestore, se utiliza map en lugar de forEach porque genera un nuevo array de elementos que pueden ser renderizados
        en JSX. */
        <playerList.map((player, indice) => ( //se incluye el indice para trabajar la PAGINACION
          //con el parentesis en lugar de llaves se ejecuta un return de manera explicita, o sea, no hay necesidad de digitar la palabra return
          indice > desde && indice < hasta ? ( //Este operador ternario es para la PAGINACION
            <tr key={player.id}>
              <td>(player.documento)</td>
              <td>(player.nombres)</td>
              <td>(player.apellidos)</td>
              <td>(player.genero)</td>
              <td>
                /* <a href=(PATH_FOTOS + player.foto) target='_blank'>(player.foto)</a> */
                <img width="80px" src={PATH_FOTOS + player.foto} alt="Foto"><img>{indice}
              </td>
              <td>(player.estado)</td>
              <td>
                <span className="btn btn-primary" onClick={() => getPlayer(player.id)}><i className="fa-solid fa-pen-to-square"></i></span>
                <span className="btn btn-danger m-1" onClick={() => deletePlayer(player.id)}><i className="fa-solid fa-pen-to-square"></i></span>
              </td>
            </tr> : '' //fin operador ternario para PAGINACION
          )
        ))
      </tbody>
    )
  }
  
```

El resultado es el siguiente:



The screenshot shows a table of player data with the following columns: Documento, Nombres, Apellidos, Genero, Foto, Estado, and Acciones. The table contains five rows of data. Above the table, there are navigation buttons labeled "Previous", "1", "2", and "Next". The number "1" is highlighted in blue, indicating it is the current page. Red arrows point from the text labels to these buttons: "Botón de atrás desactivado" points to the "Previous" button, "Página actual" points to the "1" button, and "Botón de siguiente activado para navegar" points to the "Next" button. Below the table, there is a search form with fields for Documento, Nombres, Apellidos, Genero (with a dropdown menu), and Estado (with a dropdown menu). The browser's developer tools are open on the right side, showing the "Elements" tab.

Documento	Nombres	Apellidos	Genero	Foto	Estado	Acciones
14137	Jorge Elicer	Andrade Cruz	M		0 habilitado	
147852	jose	Guzman	M		1 habilitado	
77553366	José	Falcao	M		2 habilitado	
99222777	Santiago	Garcia	M		3 habilitado	
141414	Jorge Radamel Garcia		M		4 habilitado	

Al dar clic en siguiente el botón 2 se alumbra, el botón de atrás se activa y el botón de siguiente se desactiva.

Se muestran los demás registros.

Ahora hay que agregar funcionalidad a cada botón de página:

```

EXPLORER
VITE-REACT-M...
node
routes
  routerDeptos.jsx
  routerMunicipios.jsx
  routerPlayers.jsx M
  servicios
    app.js
    package-lock.json
    package.json
torneo-sena-mysql
  node_modules
  public
  assets
  auth
  centers
  home
  players
    crudPlayers.jsx M
    formPlayers.jsx M
    formQueryPlayer.jsx M
    App.jsx M
    main.jsx
  pagination.jsx U
  .eslintrc.js
  .gitignore
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js

main.jsx
App.jsx M
crudPlayers.jsx M
pagination.jsx U
routerPlayers.jsx M
routerPlayers.js M
app.js M

playerController.js M
main.jsx
App.jsx M
pagination.jsx U
crudPlayers.jsx M
routerPlayers.js M
app.js M

const Pagination = ({ URI, setDesde, setHasta }) => {
  const siguiente = () => {
    let nuevaPage = paginaActual + 1
    setpaginaActual(nuevaPage)
    paginar(nuevaPage)
  }
  const goToPage = (numPage) => {
    setpaginaActual(numPage)
    paginar(numPage)
  }
  return (
    <nav aria-label="...">
      <ul className="pagination justify-content-center">
        <li className={ocultarMostrarAnterior}>
          <a className="page-link" href="#" tabIndex="-1" aria-disabled="true" onClick={() => anterior()}>Previous</a>
        </li>
        {
          botones.map((valorDeClase, key) => (
            <li key={key} className={valorDeClase}><a className="page-link" href="#" onClick={() => goToPage(key + 1)}>{key + 1}</a></li>
          ))
        }
        <li className={ocultarMostrarSiguiente}>
          <a className="page-link" href="#" onClick={() => siguiente()}>Next</a>
        </li>
      </ul>
    </nav>
  )
}

export default Pagination

```

localhost:5173/players#

- Principal
- Players
- Centers

Cerrar sesión

Documento Nombres Apellidos Genero Foto Estado Acciones

Documento	Nombres	Apellidos	Genero	Foto	Estado	Acciones
111333	UUUUU	uuuuu	M		5 habilidad	
665544	Linda	Caicedo	F		6 habilidad	
88888	Linda	Torres	F		7 habilidad	

Ya quedan funcionando los botones de página

Documento
Nombres
Apellidos
Genero **Selecciona uno...**
Estado **Selecciona uno...**
Seleccionar archivo Ningún archivo seleccionado **Enviar**

Documento

Previous 1 2 Next

[https://www.youtube.com/watch?v=o2QxRiybNZw&list=PLrAw40DbN0l0VBd23JYdpi8ALJtA2qBth
&index=4](https://www.youtube.com/watch?v=o2QxRiybNZw&list=PLrAw40DbN0l0VBd23JYdpi8ALJtA2qBth&index=4)

<https://codigofacilito.com/articulos/mvc-model-view-controller-explicado>

<https://www.youtube.com/watch?v=0Nz-3Q7HKb0&t=768s>

<https://www.youtube.com/watch?v=raJjjm3rhhU>