Universidad de San Carlos de Guatemala
Curso: Practicas Inciales
Ingeniero: Herman Veliz
Actividad desarrollo Web

Manual Técnico

Samuel Alejandro Pérez Neal

Dominic Juan Pablo Ruano Perez

Joab Israel Ajsivinac Ajsivinac

Javier Andrés Monjes Solórzano

Samuel Isaí Muñoz Pereira

Tutores
Eduardo René Agustin Mendoza
Josue Arturo Robledo Duque

Backend

¿Que es NodeJs?

Node.js es un entorno de tiempo de ejecución de JavaScript que permite a los desarrolladores crear aplicaciones altamente escalables y de alto rendimiento. Es ampliamente utilizado en el desarrollo de aplicaciones web y se ha convertido en una opción popular para la creación de servidores y aplicaciones de red.

Node.js se basa en el motor de JavaScript V8 de Google, que lo hace extremadamente rápido y eficiente en el manejo de solicitudes y eventos de entrada/salida. Esto lo convierte en una elección sólida para aplicaciones en tiempo real, como chats en línea y juegos multijugador.

Una de las características más destacadas de Node.js es su capacidad para manejar solicitudes de manera asincrónica. Utiliza un modelo de E/S no bloqueante, lo que significa que puede gestionar múltiples tareas simultáneamente sin bloquear el flujo de ejecución. Esto lo hace especialmente adecuado para aplicaciones que requieren una gran cantidad de conexiones simultáneas, como servidores web.

Node.js también cuenta con un ecosistema de módulos y paquetes muy activo, gracias a la plataforma npm (Node Package Manager). Esto facilita la reutilización de código y acelera el proceso de desarrollo. Los desarrolladores pueden encontrar una amplia variedad de módulos y bibliotecas disponibles en el repositorio de npm para abordar diversas necesidades de programación.

Además, Node.js es compatible con una gran cantidad de marcos y bibliotecas que simplifican tareas específicas, como Express.js para la creación de servidores web, Socket.io para aplicaciones en tiempo real y Mongoose para interactuar con bases de datos MongoDB.

Instalación de NodeJs

Para la instalación de esta herramienta en sistemas Windows es necesario bajar el instalador desde la pagina oficial https://nodejs.org/es/download, existen diversas versiones, descargue la que mas se adapte a sus necesidades, una vez descargado necesita ejecutar el instalador y seguir los pasos comunes de cualquier programa común.

Para sistemas basados en unix dependiendo de el sistema de paquetes varia la instalacion, pero en versiones como Ubuntu el comando de instalación seria: sudo apt install nodejs npm

Documentación sobre el código backend

Requisitos Previos:

- Asegúrese de tener Node.js instalado en su sistema.
- Asegúrese de haber instalado las dependencias necesarias ejecutando npm install en el directorio donde se encuentra el código.

Configuración de la Base de Datos:

- El código utiliza una conexión a una base de datos MySQL. Asegúrese de que los datos de conexión en el objeto db estén configurados correctamente con la información de su propia base de datos.

Iniciar el Servidor:

- El servidor se inicia en el puerto 3000 o en el puerto definido en la variable de entorno process.env.PORT. Puede cambiar el puerto si es necesario.

Rutas Disponibles:

1. **Registro de Usuarios (/usuario - POST):** Permite registrar nuevos usuarios en la base de datos. Los datos se envían en formato JSON en el cuerpo de la solicitud. Los campos necesarios son carnet, nombres, apellidos, contraseña, y email.
2. **Actualización de Contraseña (/contrasena - PUT):** Permite a los usuarios actualizar su contraseña. Los datos se envían en formato JSON en el cuerpo de la solicitud. Los campos requeridos son carnet, email, y la nueva contraseña.
3. **Actualización de Datos de Usuario (/usuario - PUT):** Permite a los usuarios actualizar su información personal, como nombres, apellidos y correo electrónico. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requiere el idUsuario para identificar al usuario.
4. **Búsqueda de Usuario por Carnet (/buscaru - POST):** Permite buscar un usuario en la base de datos por su número de carné. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se debe proporcionar el campo carnet.
5. **Inicio de Sesión (/login - POST):** Permite a los usuarios iniciar sesión verificando sus credenciales en la base de datos. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requieren los campos carnet y contraseña.
6. **Creación de Publicación (/publicacion - POST):** Permite a los usuarios crear una nueva publicación. Los datos se envían en formato JSON en el cuerpo de la solicitud. Los campos requeridos son titulo, contenido, fecha_publicacion, sobre_quien, tipo, y id_usuario.
7. **Obtención de Todas las Publicaciones (/publicaciones - GET):** Recupera todas las publicaciones disponibles en la base de datos.

8. **Búsqueda de Publicación por ID (/buscarpub - POST):** Permite buscar una publicación específica por su ID. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se debe proporcionar el campo id.
9. **Filtrado de Publicaciones por Tipo (/filtro - POST):** Permite filtrar publicaciones por su tipo. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se debe proporcionar el campo tipo.
10. **Creación de Comentario (/comentario - POST):** Permite a los usuarios crear un comentario en una publicación. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requieren los campos contenido, fecha, idPub, idUsuarioPub, y idUsuarioActual.
11. **Búsqueda de Comentarios por ID de Publicación (/buscarc - POST):** Permite buscar comentarios relacionados con una publicación específica. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se debe proporcionar el campo id_pub.
12. **Agregar Curso (/agregarcurso - POST):** Permite agregar un nuevo curso en la base de datos. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requieren los campos nombre y creditos.
13. **Búsqueda de Curso por ID (/buscarcurso - POST):** Permite buscar un curso específico por su ID. Los datos se envían en el cuerpo de la solicitud como parámetros de consulta.
14. **Obtención de Todos los Cursos (/cursos - GET):** Recupera todos los cursos disponibles en la base de datos.
15. **Obtención de Todos los Catedráticos (/catedratico - GET):** Recupera todos los catedráticos disponibles en la base de datos.
16. **Agregar Catedrático (/agregarcatedratico - POST):** Permite agregar un nuevo catedrático en la base de datos. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requieren los campos nombre y apellido.
17. **Obtención de Cursos Ganados por un Usuario (/cursosg - POST):** Recupera los cursos ganados por un usuario específico. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requiere el campo id_usuario.
18. **Registro de Curso Ganado (/ganado - POST):** Permite registrar un curso ganado por un usuario. Los datos se envían en formato JSON en el cuerpo de la solicitud. Se requieren los campos id_usuario e id_curso.
19. **Eliminar Elementos (Usuarios, Comentarios, Publicaciones, Cursos, Catedráticos):** Se proporcionan rutas DELETE para eliminar elementos de la base de datos. Los datos se envían en formato JSON en el cuerpo de la solicitud, especificando el ID del elemento a eliminar.

Es importante tener en cuenta que este código se conecta a una base de datos MySQL y utiliza Express.js para gestionar las rutas y las solicitudes HTTP. También se pueden encontrar llamadas a la API externa usando Axios.

¿Que es MySQL?

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS, por sus siglas en inglés) de código abierto ampliamente utilizado en el desarrollo de aplicaciones web y de escritorio. Aquí tienes información básica sobre MySQL y cómo se utiliza en el código que proporcionaste:

- **Base de Datos Relacional**: MySQL es un sistema de gestión de bases de datos relacionales, lo que significa que almacena y organiza datos en tablas relacionadas entre sí. Cada tabla consta de filas y columnas, y se pueden establecer relaciones entre tablas para realizar consultas y recuperar datos de manera eficiente.
- **Código de Conexión a la Base de Datos**: En el código proporcionado, se utiliza la biblioteca mysql de Node.js para establecer una conexión a una base de datos MySQL alojada en la nube (posiblemente en Clever Cloud). Aquí hay una breve descripción de los parámetros de conexión:
    - host: Es la dirección del servidor de base de datos al que se va a conectar. En este caso, parece ser una dirección en Clever Cloud.
    - user: Es el nombre de usuario utilizado para autenticar la conexión a la base de datos.
    - password: Es la contraseña asociada al nombre de usuario para la autenticación.
    - database: Es el nombre de la base de datos a la que se desea acceder una vez que la conexión se haya establecido con éxito.
- **Acceso a Datos en la Base de Datos**: Una vez que se establece la conexión, puedes utilizar este objeto db para realizar consultas y operaciones en la base de datos. Por ejemplo, puedes realizar consultas SELECT para recuperar datos, consultas INSERT para agregar nuevos registros y otras operaciones de manipulación de datos.
- **Lenguaje SQL**: MySQL utiliza el lenguaje SQL (Structured Query Language) para interactuar con la base de datos. Con SQL, puedes realizar diversas operaciones, como consulta, inserción, actualización y eliminación de datos, así como la creación y modificación de estructuras de tabla.
- **Seguridad**: Es importante asegurarse de que las credenciales de conexión a la base de datos sean seguras y no estén expuestas públicamente, ya que proporcionar acceso no autorizado a la base de datos puede tener graves consecuencias en la seguridad de la aplicación.

- **Gestión de Datos**: MySQL también proporciona herramientas y utilidades para la administración y gestión de bases de datos, lo que incluye realizar copias de seguridad, optimizar el rendimiento y mantener la integridad de los datos.

Frontend

¿Que es React?
React es una popular biblioteca de JavaScript utilizada para construir interfaces de usuario interactivas y dinámicas en aplicaciones web y móviles. Fue desarrollada por Facebook y se ha convertido en una de las herramientas más utilizadas en el desarrollo web moderno.

React se centra en la creación de componentes reutilizables que representan partes específicas de la interfaz de usuario. Estos componentes pueden contener su propio estado y lógica, lo que facilita la construcción de aplicaciones complejas a partir de piezas más pequeñas y manejables.

Algunas características importantes de React incluyen:

1. **Virtual DOM**: React utiliza un Virtual DOM (DOM virtual) para optimizar la manipulación del DOM real. Esto significa que React realiza cambios en el DOM solo cuando es necesario, lo que mejora el rendimiento de la aplicación.
2. **Unidireccionalidad de datos**: React sigue el principio de unidireccionalidad de datos, lo que significa que los datos fluyen en una sola dirección, desde el componente principal hacia los componentes secundarios. Esto facilita el seguimiento y la depuración de los datos en la aplicación.
3. **JSX**: React utiliza JSX (JavaScript XML) para definir la estructura de la interfaz de usuario en un formato similar a HTML dentro del código JavaScript. Esto facilita la creación y la comprensión de la interfaz de usuario en el código.
4. **Reactividad**: React es altamente reactivo, lo que significa que los cambios en los datos se reflejan automáticamente en la interfaz de usuario. Cuando los datos cambian, React actualiza solo las partes relevantes de la interfaz de usuario sin necesidad de recargar la página completa.
5. **Amplia comunidad y ecosistema**: React cuenta con una gran comunidad de desarrolladores y una amplia variedad de bibliotecas y herramientas complementarias que simplifican tareas como la gestión del estado (Redux, MobX), enrutamiento (React Router), y la interacción con APIs (Axios).

En resumen, React es una poderosa biblioteca de JavaScript que facilita la creación de interfaces de usuario dinámicas y eficientes en aplicaciones web y móviles. Su enfoque en

componentes reutilizables, su Virtual DOM y su amplia comunidad de desarrolladores la convierten en una elección popular para el desarrollo de aplicaciones modernas.

Pasos iniciales con Reac

1. **Instala Node.js y npm** Antes de poder trabajar con React, debes asegurarte de tener Node.js y npm instalados en tu sistema. Puedes descargar e instalar Node.js desde el sitio web oficial: https://nodejs.org/. Una vez instalado Node.js, npm se instalará automáticamente junto con él.\
2. **Crea una nueva aplicación React** una vez que Node.js y npm estén instalados, puedes crear una nueva aplicación React utilizando la herramienta de línea de comandos de React llamada "Create React App". Para hacerlo, sigue estos pasos:

   Este comando iniciará un servidor de desarrollo y abrirá automáticamente tu nueva aplicación React en tu navegador web predeterminado. A partir de aquí, puedes comenzar a editar el código fuente de tu aplicación y ver los cambios en tiempo real en tu navegador.

   Abre tu terminal o línea de comandos.

   Ejecuta el siguiente comando para instalar Create React App de forma global en tu sistema:

   npm install -g create-react-app

   Este comando instala Create React App para que puedas usarlo en cualquier proyecto de React.

   Una vez que la instalación esté completa, puedes crear una nueva aplicación React ejecutando el siguiente comando:

   npx create-react-app my-react-app

   Sustituye "my-react-app" por el nombre que desees para tu proyecto.

   Create React App creará una estructura de directorios y archivos para tu nueva aplicación React. Una vez que se complete el proceso, cambia al directorio de tu proyecto con el siguiente comando:

   cd my-react-app

3. **Iniciar la aplicación React** después de cambiar al directorio de tu proyecto, puedes iniciar la aplicación React ejecutando el siguiente comando:

   npm start

Este comando iniciará un servidor de desarrollo y abrirá automáticamente tu nueva aplicación React en tu navegador web predeterminado. A partir de aquí, puedes comenzar a editar el código fuente de tu aplicación y ver los cambios en tiempo real en tu navegador.

Documentación del Frontend

Registro de Usuarios:

- **Importación de bibliotecas y estilos CSS**: La página comienza importando las bibliotecas y los estilos CSS necesarios. Utiliza React, React DOM y algunas hojas de estilo CSS externas.
- **Componente "RegistroUsuario"**: La página define un componente funcional llamado "RegistroUsuario". Este componente representa la vista de registro de usuarios en la aplicación.
- **Estado del formulario**: Utiliza el hook de estado de React (useState) para mantener el estado del formulario. Esto incluye campos como "carnet", "nombre", "apellidos", "contraseña" y "correo". Los valores de estos campos se almacenan en el estado y se actualizan cuando el usuario interactúa con los campos del formulario.
- **Manejo de eventos**: El componente define funciones como handleChange para manejar los cambios en los campos del formulario y handleSubmit para manejar la presentación del formulario.
- **Validación de contraseña**: Existe una variable de estado llamada contraseñaObligatoria que se utiliza para rastrear si la contraseña es obligatoria. Si el usuario intenta enviar el formulario sin proporcionar una contraseña, se establece esta variable a true.
- **Envío de datos**: Cuando el usuario envía el formulario, se realiza una solicitud POST a una URL de una API externa (https://api-taller4.onrender.com/usuario) para registrar al usuario. Los datos del formulario se envían en formato JSON en el cuerpo de la solicitud.
- **Manejo de respuestas de la API**: Se maneja la respuesta de la API para mostrar mensajes de éxito o error. Si la respuesta es exitosa, se muestra un mensaje de éxito y se redirige al usuario a la página de inicio de sesión. En caso de error, se muestra un mensaje de error.
- **Interfaz de usuario del formulario**: La página muestra un formulario con campos para el carnet, nombre, apellidos, contraseña y correo electrónico del usuario. También hay un botón de "Registrar" para enviar el formulario.

- **Enlace a la página de inicio de sesión**: Además del formulario de registro, hay un enlace que permite a los usuarios ir a la página de inicio de sesión (<InicioSesion />) haciendo clic en el botón "ir a login".

Inicio de Sesion:

- **Importación de bibliotecas y estilos CSS**: El componente importa React, ReactDOM y algunas hojas de estilo CSS. También importa otros componentes como PantallaPrincipal y ActualizarContraseña, así como la biblioteca js-cookie para gestionar cookies.
- **Estado del formulario**: Al igual que en el componente anterior, este componente utiliza el hook de estado de React (useState) para mantener el estado del formulario. Los campos del formulario incluyen "carnet" y "contraseña". Los valores de estos campos se almacenan en el estado y se actualizan cuando el usuario interactúa con los campos del formulario.
- **Manejo de eventos**: El componente define funciones como handleChange para manejar los cambios en los campos del formulario y handleSubmit para manejar la presentación del formulario.
- **Envío de datos de inicio de sesión**: Cuando el usuario envía el formulario, se realiza una solicitud POST a una URL de una API externa (https://api-taller4.onrender.com/login) para iniciar sesión. Los datos del formulario se envían en formato JSON en el cuerpo de la solicitud.
- **Manejo de respuestas de la API**: El componente maneja la respuesta de la API. Si la respuesta es exitosa, se muestra un mensaje de bienvenida y se establece una cookie llamada 'idUsuario' con el valor proporcionado por la API. Luego, el componente redirige al usuario a la página de inicio principal (<PantallaPrincipal />). En caso de error, se muestra un mensaje de error indicando que las credenciales son incorrectas.
- **Interfaz de usuario del formulario**: El componente muestra un formulario con campos para el "carnet" y la "contraseña" del usuario. También hay un botón "Iniciar Sesión" para enviar el formulario.
- **Recuperación de contraseña**: Además del formulario de inicio de sesión, hay un enlace que permite a los usuarios recuperar su contraseña. Al hacer clic en el botón "Recuperar contraseña", el componente renderiza el componente ActualizarContraseña en el elemento con el ID 'root', lo que permite a los usuarios iniciar el proceso de recuperación de contraseña.

Recuperación de Contraseña

- **Importación de bibliotecas y estilos CSS:** El componente importa React, ReactDOM y un archivo de estilos CSS llamado ActualizarContraseña.css. También importa el componente InicioSesion, que se utilizará para redirigir a los usuarios a la página de inicio de sesión.
- **Estado del formulario**: Al igual que en los componentes anteriores, este componente utiliza el hook useState para mantener el estado del formulario. Los campos del formulario incluyen "contraseña", "carnet" y "correo". Los valores de estos campos se almacenan en el estado y se actualizan cuando el usuario interactúa con los campos del formulario.
- **Manejo de eventos**: El componente define funciones como handleChange para manejar los cambios en los campos del formulario y handleSubmit para manejar la presentación del formulario.
- **Envío de datos de actualización de contraseña**: Cuando el usuario envía el formulario, se realiza una solicitud PUT a una URL de una API externa (https://api-taller4.onrender.com/contrasena) para actualizar la contraseña. Los datos del formulario se envían en formato JSON en el cuerpo de la solicitud.
- **Manejo de respuestas de la API**: El componente maneja la respuesta de la API. Si la respuesta es exitosa, se muestra un mensaje de éxito indicando que la contraseña se actualizó correctamente. Luego, el componente redirige al usuario a la página de inicio de sesión (<InicioSesion />). En caso de error, se muestra un mensaje de error indicando que los datos son incorrectos.
- **Interfaz de usuario del formulario**: El componente muestra un formulario con campos para el "carnet", el "correo electrónico" y la "nueva contraseña" del usuario. También hay un botón "Actualizar" para enviar el formulario.
- **Redirección a la página de inicio de sesión**: Además del formulario de actualización de contraseña, hay un botón que permite a los usuarios volver a la página de inicio de sesión haciendo clic en "ir a login".

Pantalla principal

- **Importación de bibliotecas y estilos CSS**: El componente importa React, ReactDOM, useState, y useEffect de React, así como otros componentes como Publicacion y CrearPublicacion. También importa dos archivos de estilos CSS: pantallaPrincipal.css y CrearPublicacion.css.
- **Estado y efecto de efecto**: El componente utiliza los hooks useState y useEffect. El estado publicaciones se utiliza para almacenar las publicaciones recuperadas de una API. Además, hay estados filtroTipo y filtroSobreQuien para filtrar las publicaciones.
- **Recuperación de publicaciones**: En el efecto de efecto (useEffect), el componente realiza una solicitud GET a una URL de una API externa (https://api-

taller4.onrender.com/publicaciones) para obtener las publicaciones. Una vez que se obtiene la respuesta, se almacenan las publicaciones en el estado publicaciones.

- **Función de filtrado de publicaciones**: La función filtrarPublicaciones se utiliza para aplicar filtros a las publicaciones en función de los valores seleccionados en los elementos de filtro (tipo y sobre quién). Filtra las publicaciones en función de los valores de estado filtroTipo y filtroSobreQuien.
- **Interfaz de usuario**: El componente muestra una interfaz de usuario que incluye un título "Publicaciones Usac Ingenieria". También presenta dos elementos de filtro: uno para seleccionar el tipo de publicación (curso, catedrático o todos) y otro para buscar por el nombre del curso o maestro. Además, hay un botón "Publicar" que permite a los usuarios crear una nueva publicación haciendo clic en él.
- **Visualización de publicaciones**: Las publicaciones se muestran en la sección de "publicaciones" utilizando el mapeo de las publicaciones filtradas. Cada publicación se muestra utilizando el componente Publicacion, que probablemente contenga detalles como el título, el autor y el contenido de la publicación.

Crear publicación

- **Importación de bibliotecas y estilos CSS**: El componente importa React, useState, useEffect, format de la biblioteca date-fns, ReactDOM y otros componentes, así como los estilos CSS necesarios desde CrearPublicacion.css.
- **Estado del formulario**: El componente utiliza el hook useState para mantener el estado del formulario. Los campos del formulario incluyen "titulo", "contenido", "sobre_quien", "tipo" e "id_usuario". Los valores de estos campos se almacenan en el estado y se actualizan cuando el usuario interactúa con los campos del formulario.
- **Manejo de mensajes de éxito y error**: El componente utiliza los estados mensajeError y mensajeExito para mostrar mensajes de error o éxito en la interfaz de usuario en función de la respuesta de la API o validaciones de formulario.
- **Obtención de cursos y catedráticos**: El componente utiliza el hook useEffect para realizar dos solicitudes GET a las API externas (https://api-taller4.onrender.com/cursos y https://api-taller4.onrender.com/catedratico) para obtener la lista de cursos y catedráticos. Estos datos se utilizan para llenar el campo "sobre_quien" en función del tipo de publicación seleccionado.
- **Manejo de eventos**: El componente define la función handleChange para manejar los cambios en los campos del formulario y la función handleSubmit para manejar la presentación del formulario.
- **Envío de datos de la publicación**: Cuando el usuario envía el formulario, se realiza una solicitud POST a una URL de la API externa

(https://api-taller4.onrender.com/Publicacion) para crear una nueva publicación. Los datos del formulario se envían en formato JSON en el cuerpo de la solicitud.

- **Interfaz de usuario del formulario**: El componente muestra un formulario que incluye campos para seleccionar el "tipo" de publicación (curso o catedrático), el "sobre_quien" (que se llena dinámicamente según el tipo seleccionado), el "título" y el "contenido" de la publicación. También hay un botón "Crear Publicación" para enviar el formulario.
- **Redirección y botón de regreso**: Después de una publicación exitosa, se muestra un mensaje de éxito y se redirige al usuario a la página de Publicacion. Además, hay un botón "Regresar" que permite al usuario volver a la página de PantallaPrincipal.

Apendices y Anexo

Conexión con la base de datos

```
var mysql = require("mysql");

var db = mysql.createConnection({
  host: "bxbuke117r4gmf7aqsgj-mysql.services.clever-cloud.com",
  user: "u4nabtaemgcrgwmo",
  password: "7aAMGKGBDpIvdo2fZTGe",
  database: "bxbuke117r4gmf7aqsgj",
});

db.connect((err) => {
  if (err) {
    //console.error("Error al conectar a la base de datos: ", err);
  } else {
    console.log("Conexión exitosa a la base de datos");
  }
});
```

Petición creación de usuario

```
// REGISTRAR USUARIOS
app.post("/usuario", (req, res) => {
  const { carnet, nombres, apellidos, contrasena, email } = req.body;

  // Realiza la inserción en la tabla de Usuarios
  db.query(
    "INSERT INTO usuario (carnet, nombres, apellidos, contrasena, email) VALUES (?, ?, ?,?,?)",
    [carnet, nombres, apellidos, contrasena, email],
    (err, result) => {
      if (err) {
        // console.error("Error al registrar al usuario: ", err);
        res.status(500).send({ error: err });
      } else {
        // console.log("Usuario registrado con éxito");
        res.status(200).send({ message: "Usuario registrado con éxito" });
      }
    }
  );
});
```

Consumir desde el Frontend

```
const url = 'https://api-taller4.onrender.com/usuario'; // URL de la API

try {
  const response = await fetch(url, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      carnet: formData.carnet,
      nombres: formData.nombre,
      apellidos: formData.apellidos,
      contrasena: formData.contraseña,
      email: formData.correo,
    }),
  });

  if (response.ok) {
    console.log('Registro exitoso');
    mostrarMensajeDeError('Usuario Registrado con exito, se redirecciona a la pagina de inicio de sesion.');
    ReactDOM.render(<InicioSesion />, document.getElementById('root'));
    // Puedes redirigir al usuario o mostrar un mensaje de éxito aquí.
  } else {
    const errorData = await response.json();
    console.error('Error:', errorData);
    mostrarMensajeDeError('El correo electrónico o  Carné ya está registrado.');
  }
```

Código HTML

```
return (
  <div className="registro-usuario">
    <h1>Registro de Usuario</h1>
    <form onSubmit={handleSubmit}>
      <div className="campo">
        <label htmlFor="carnet">Carné:</label>
        <input
          type="text"
          id="carnet"
          name="carnet"
          value={formData.carnet}
          onChange={handleChange}
        />
      </div>
      <div className="campo">
        <label htmlFor="nombre">Nombre:</label>
        <input
          type="text"
          id="nombre"
          name="nombre"
          value={formData.nombre}
          onChange={handleChange}
        />
      </div>
      <div className="campo">
        <label htmlFor="apellidos">Apellidos:</label>
```

Configuración CSS

```css
/* src/RegistroUsuario.css */
.registro-usuario {
    font-family: "Montserrat", sans-serif;
    background-color: #333;
    color: #fff;
    padding: 20px;
    border-radius: 5px;
    width: 350px;
    margin: 45px auto ;
  }

  h1 {
    font-size: 24px;
    text-align: center;
  }

  .campo {
    margin: 10px 0;
  }

  label {
    display: block;
    font-size: 16px;
    margin-bottom: 5px;
  }
```

Peticiones consumiendo el API