

Desarrollo de Software

Proyecto Final

Jesús Alberto Aréchiga Carrillo
22310439 4N

Profesor
José Luis García Cerpas

Junio 2024

Guadalajara, Jalisco

Introducción

Express.js es un framework minimalista y flexible para Node.js, diseñado para construir aplicaciones web y APIs robustas. Ofrece un conjunto de características fundamentales que facilitan el desarrollo de aplicaciones del lado del servidor de manera rápida y eficiente. Su simplicidad y modularidad lo hacen una elección popular para desarrolladores que buscan una solución ágil y escalable.

React, por otro lado, es una biblioteca de JavaScript mantenida por Facebook, utilizada para construir interfaces de usuario interactivas y dinámicas. Basado en componentes reutilizables, React permite a los desarrolladores crear aplicaciones frontales de manera declarativa y eficiente, con un manejo óptimo del estado y la renderización.

El desarrollo de este proyecto se divide en dos partes principales:

- La configuración e implementación del servidor backend utilizando Express.js, donde se establecen las rutas para las páginas principales.
- La creación de la interfaz de usuario con React, incluyendo la configuración de los componentes y la navegación entre páginas.

Desarrollo

Primero se inicializa un nuevo proyecto de Node.js

```
D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9>npm init -y
Wrote to D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9\package.json:

{
  "name": "practica-3.9",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Ahora se instala Express.js

```
D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9>npm install express

added 64 packages, and audited 65 packages in 5s

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

found 0 vulnerabilities
```

Se crea un archivo llamado index.js que será el servidor que tenga la página:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Página Principal');
});

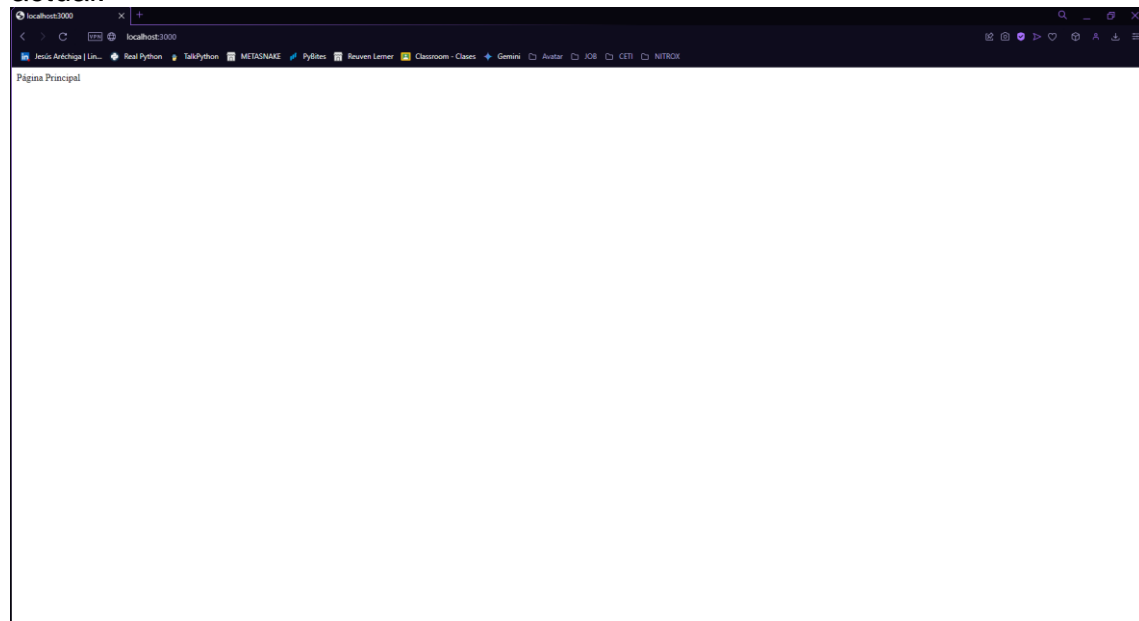
app.get('/nosotros', (req, res) => {
  res.send('Página Nosotros');
});

app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

Se hace una prueba del servidor para asegurarse de que esté funcionando correctamente:

```
D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9>node index.js
Servidor escuchando en http://localhost:3000
```

Y ahora si buscamos en localhost con el puerto 3000, encontraremos la página actual:



Ahora de hace la configuración del front-end con React, primero se empieza creando el proyecto de React:

```
D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9>npx create-react-app cliente
Creating a new React app in D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9\cliente.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1489 packages in 3m
258 packages are looking for funding
  run 'npm fund' for details
Installing template dependencies using npm...
added 67 packages, and changed 1 package in 11s
262 packages are looking for funding
  run 'npm fund' for details
Removing template package using npm...

removed 1 package, and audited 1556 packages in 2s
262 packages are looking for funding
  run 'npm fund' for details
8 vulnerabilities (2 moderate, 6 high)
To address all issues (including breaking changes), run:
  npm audit fix --force
Run 'npm audit' for details.
Success! Created cliente at D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9\cliente
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files

We suggest that you begin by typing:

  cd cliente
  npm start
```

Ahora se inicia la aplicación React llamada “cliente” que se acaba de crear:

```
Compiled successfully!

You can now view cliente in the browser.

Local:            http://localhost:3000
On Your Network:  http://192.168.1.92:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Edit src/App.js and save to reload.

[Learn React](#)

Ahora se crean los componentes para la página de inicio y la página “Sobre Nosotros”. Para esto se crea la carpeta “components” y los archivos “Home.js” y “About.js”.

Home.js:

```
import React from 'react';

const Home = () => {
  return (
    <div>
      <h1>Página Principal</h1>
      <p>Bienvenido a nuestra aplicación.</p>
    </div>
  );
};

export default Home;
```

About.js:

```
import React from 'react';

const About = () => {
  return (
    <div>
      <h1>Página Nosotros</h1>
      <p>Información sobre nosotros.</p>
    </div>
  );
};

export default About;
```

Ahora se configuran las rutas en App.js para poder ver las páginas en la aplicación, esto se hace utilizando “react-router-dom”:

```
D:\Otros\Documentos\CETI\4to-semester\Desarrollo Web I\Practica 3.9>npm install react-router-dom

added 8 packages, and audited 73 packages in 2s

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

found 0 vulnerabilities
```

```
import './App.css';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Header from './components/Header';
import Footer from './components/Footer';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

const App = () => {
  return (
    <Router>
      <div>
        <Header title="Proyecto" />
        <div className="content-container">
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
            <Route path="/contact" element={<Contact />} />
          </Routes>
        </div>
        <Footer />
      </div>
    </Router>
  );
};
```

```
export default App;
```

Utilizando Router, Routes y Route, podemos crear los vínculos que se utilizarán para las demás páginas.

Ahora se pueden crear el resto de las páginas y los componentes en archivos de JavaScript independientes.

Estilos (CSS):

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f5f5f5;
}

.header-logo {
  pointer-events: none;
  margin-top: 2px;
  flex: 1.06;
}

.imagen-fondo {
  height: 700px;
  background-size: cover;
  background-image: url('../public/imagenes/image.jpg');
  position: relative;
  display: flex;
  justify-content: center;
  align-items: center;
  color: #ffffff;
}

.text-container {
  position: absolute;
  top: 10%;
  left: 60%;
  text-align: center;
}

.text-container h2, .text-container h3 {
  font-family: "Lexend-Tera", sans-serif;
  font-optical-sizing: auto;
  font-weight: 300;
  font-style: normal;
  margin: 0;
```

```

padding: 0;
}

.header {
background-color: #FBE3AB;
display: flex;
align-items: center;
justify-content: space-between; /* Asegura que los elementos estén
espaciados correctamente */
padding: 10px;
width: 100%;
box-sizing: border-box;
}

.menu-container {
flex: 1;
display: flex;
justify-content: center;
}

.menu {
display: flex;
list-style-type: none;
margin: 0;
padding: 0;
justify-content: center;
}

.menu-item {
margin: 0 20px;
font-size: 1.5em;
transition: transform 0.3s ease, background-color 0.3s;
padding: 10px 15px;
border: 1px solid transparent;
border-radius: 5px;
}

.menu-item a {
color: rgb(0, 0, 0);
text-decoration: none;
}

.menu-item:hover {
transform: scale(1.1);
background-color: #ffce5d;
border-color: #c58a00;
}

.auth-container {

```



```

    flex: 1;
    display: flex;
    justify-content: flex-end; /* Alinea el botón de autenticación a la
derecha */
}

.auth-container a, .auth-container button {
    font-size: 1.5em;
    padding: 10px 15px;
    border: 1px solid transparent;
    border-radius: 5px;
    color: black;
    text-decoration: none;
    cursor: pointer;
    transition: transform 0.3s ease, background-color 0.3s;
}

.auth-container a:hover, .auth-container button:hover {
    transform: scale(1.1);
    background-color: #ffce5d;
    border-color: #c58a00;
}

.auth-container .signout-button {
    background-color: #FBE3AB;
    color: black;
    border: 1px solid transparent;
    padding: 10px 15px;
    border-radius: 5px;
    cursor: pointer;
    transition: transform 0.3s ease, background-color 0.3s;
}

.auth-container .signout-button:hover {
    transform: scale(1.1);
    background-color: #ffce5d;
    border-color: #c58a00;
}

.auth-button {
    display: flex;
    justify-content: center;
    margin: 5px 20px;
    font-size: 2em;
    transition: transform 0.3s ease, background-color 0.3s;
    padding: 10px 15px;
    border: 1px solid transparent;
    border-radius: 5px;
    width: 12%;

```

```

}

.auth-button:hover {
  transform: scale(1.1);
  background-color: #ffce5d;
  border-color: #c58a00;
}

.content {
  width: 60%;
  padding: 20px;
  background: rgb(255, 255, 255);
  box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
  margin: auto;
  margin-top: 30px;
  margin-bottom: 30px;
  border-radius: 10px;
  text-align: justify;
}

.content h1 {
  text-align: center;
}

.footer {
  background-color: #A4DDEA;
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
  width: 100%;
}

.footer-content {
  display: flex;
  align-items: center;
  justify-content: center;
  width: 100%;
}

.footer-logo {
  padding: 5px;
  width: 100px;
  align-items: center;
}

.footer-text {
  padding: 20px;
  padding-top: 30px;
}

```

```

}

.footer-menu {
  list-style-type: none;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
}

.footer-menu-item {
  margin: 0 10px;
}

.footer-menu-item a {
  color: rgb(0, 0, 0);
  text-decoration: none;
}

.footer-menu-item a:hover {
  text-decoration: underline;
}

.footer p {
  margin-top: 10px;
}

.map-container {
  display: flex;
  justify-content: center;
  margin-top: 20px;
}

.text-container {
  position: absolute;
  top: 10%;
  left: 60%;
  text-align: center;
}

.text-container h2, .text-container h3 {
  font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
  font-optical-sizing: auto;
  font-weight: 300;
  font-style: normal;
  margin: 0;
  padding: 0;
}

```

```

/* Form styles */
.form-container {
  width: 100%;
  max-width: 1000px;
  margin: auto;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 20px;
  background: #fff;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
}

.form-group {
  margin-bottom: 15px;
}

.form-group label {
  display: block;
  margin-bottom: 5px;
}

.form-group input {
  width: 95%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.submit-button {
  width: 100%;
  padding: 10px;
  background-color: #0033ff;
  border: none;
  border-radius: 5px;
  color: white;
  font-size: 1em;
  cursor: pointer;
}

.submit-button:hover {
  background-color: #0024b4;
  transition: transform 0.3s ease, background-color 0.3s;
}

/* Styles for PagAdmin */
.admin-container {
  width: 80%;

```

```

margin: auto;
margin-top: 30px;
padding: 20px;
background-color: white;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
border-radius: 10px;
}

.admin-container h1, .admin-container h2 {
text-align: center;
}

.user-list-header, .user-details {
display: flex;
justify-content: space-between;
align-items: center;
text-align: center;
font-weight: bold;
margin-bottom: 10px;
padding: 10px;
background-color: #f1f1f1;
border: 1px solid #ddd;
border-radius: 5px;
}

.user-list-header > div, .user-details > div {
flex: 1;
text-align: center; /* Center align text */
}

.user-list {
list-style-type: none;
padding: 0;
}

.user-list li {
justify-content: space-between;
align-items: center;
margin-bottom: 10px;
padding: 10px;
background-color: #f9f9f9;
border: 1px solid #ddd;
border-radius: 5px;
}

.user-details > div {
flex: 1;
margin: 0 10px;
}

```

```

.user-details input, .user-details select {
  width: 100%;
  padding: 5px;
  margin: 5px 0;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.user-actions {
  display: flex;
  align-items: center;
}

.user-actions button {
  margin-left: 10px;
  color: white;
  border: none;
  padding: 5px 10px;
  border-radius: 5px;
  cursor: pointer;
}

.user-actions .edit-button {
  background-color: #007BFF;
}

.user-actions .edit-button:hover {
  background-color: #0056b3;
}

.user-actions .delete-button {
  background-color: #ff4d4d;
}

.user-actions .delete-button:hover {
  background-color: #ff1a1a;
}

.user-actions .cancel-button {
  background-color: #6c757d;
}

.add-user-container {
  margin-top: 20px;
  padding: 20px;
  background-color: #f1f1f1;
  border-radius: 10px;
  display: flex;

```

```

    justify-content: space-between;
  }

.add-user-container > div {
  flex: 1;
  margin-right: 10px;
}

.add-user-container > div:last-child {
  margin-right: 0;
}

.add-user-container input, .add-user-container select, .add-user-
container button {
  width: 20%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 50px;
  box-sizing: border-box;
  height: 50%;
  margin-top: 15px;
}

.add-user-container button {
  background-color: #4CAF50;
  color: white;
  cursor: pointer;
}

.add-user-container button:hover {
  background-color: #45a049;
}

```

Componentes:

Header.js:

```

import React from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useAuth } from './authContext';

const Header = ({ title }) => {
  const { user, signOut } = useAuth();
  const navigate = useNavigate();

  const handleSignOut = () => {
    signOut();
    navigate('/');
  };
};

```

```

return (
  <header className="header">
    <Link to="/" className="header-logo">
      
    </Link>
    <div className="menu-container">
      <ul className="menu">
        <li className="menu-item"><Link to="/">Inicio</Link></li>
        <li className="menu-item"><Link to="/about">Sobre
Nosotros</Link></li>
        <li className="menu-item"><Link
to="/contact">Contacto</Link></li>
      </ul>
    </div>
    <div className="auth-container">
      {user ? (
        <button onClick={handleSignOut} className="signout-
button">Cerrar Sesión</button>
      ) : (
        <Link to="/signin">Iniciar Sesión</Link>
      )}
    </div>
  </header>
);
};

export default Header;

```

Footer:

```

import React from 'react';

const Footer = () => {
  return (
    <footer className="footer">
      <div className="footer-content">
        
        <div className="footer-text">
          <ul className="footer-menu">
            <li className="footer-menu-item"><a href="/about">Sobre
nosotros</a></li>
            <li className="footer-menu-item"><a
href="/contact">Contacto</a></li>
          </ul>
          <p>Instituto Punto Alejandría © 2024</p>

```



```

        </div>
      </div>
    </footer>
  );
};

export default Footer;

```

Con estos componentes es posible agregar el header y footer en todas las páginas que se vayan a utilizar sin tener que copiar y pegar el código múltiples veces.

authContext.js:

```

import React, { createContext, useContext, useState } from 'react';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);

  const signIn = (username, role) => {
    setUser({ username, role });
  };

  const signOut = () => {
    setUser(null);
  };

  return (
    <AuthContext.Provider value={{ user, signIn, signOut }}>
      {children}
    </AuthContext.Provider>
  );
};

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

```

protectedRoute.js:

```

import React from 'react';
import { Navigate } from 'react-router-dom';
import { useAuth } from './authContext';

const ProtectedRoute = ({ children, role }) => {
  const { user } = useAuth();

  if (!user) {
    return <Navigate to="/signin" />;
  }

```

```

    }

    if (role && user.role !== role) {
      return <Navigate to="/" />;
    }

    return children;
  };

export default ProtectedRoute;

```

En el caso de las sesiones, es necesario manejarlos de una manera segura, utilizando `authContext` y `protectedRoute` es posible hacer las navegaciones a las páginas necesarias utilizando la información del usuario que se haya ingresado.

`userContext.js:`

```

import React, { createContext, useState, useContext } from 'react';

const UserContext = createContext();

export const useUser = () => useContext(UserContext);

export const UserProvider = ({ children }) => {
  const [users, setUsers] = useState([
    { username: 'admin', password: 'admin', role: 'admin' },
    { username: 'user', password: 'user', role: 'user' },
    { username: 'test', password: 'test', role: 'user' },
    { username: 'test2', password: 'test2', role: 'user' },
    { username: 'test3', password: 'test3', role: 'user' },
  ]);

  return (
    <UserContext.Provider value={{ users, setUsers }}>
      {children}
    </UserContext.Provider>
  );
};

```

Aquí es donde se tienen los datos de los usuarios y servirá para la demostración del crud, si se quisiera agregar más información, aquí es donde se tiene que agregar.

Páginas:

`Home.js:`

```

import React from 'react';
import '../App.css';

```

```
const Home = () => {
  return (
    <div className="content">
      <div className="imagen-fondo">
        <div className="text-container">
          <h2><strong>Impulsando líderes competentes</strong></h2>
          <h3>Formación es nuestra misión</h3>
        </div>
      </div>
    </div>
  );
}

export default Home;
```



Una página de bienvenida con una imagen para dar más impresión.

About.js:

```
import React from 'react';
import '../App.css';

const About = () => {
  return (
    <div className="content">
      <h1>Sobre Nosotros</h1>
      <p>El Instituto Punto Alejandría es una institución dedicada a la formación y certificación de competencias profesionales. Nuestro objetivo es proporcionar las herramientas y el conocimiento necesario para que los profesionales se destaquen en sus respectivos campos.</p>
      <p>Contamos con un equipo de expertos altamente cualificados y con amplia experiencia en diversas áreas de conocimiento. Nuestro enfoque está en la calidad educativa y en el desarrollo integral de nuestros participantes.</p>
    </div>
  );
}
```

```

    </div>
  );
};

export default About;

```



Una página que describe la empresa.

Contact.js:

```

import React from 'react';
import '../App.css';

const Contact = () => {
  return (
    <div className="content">
      <h1>Contacto</h1>
      <p>Estamos ubicados entre la secretaría de tránsito y el estadio Jalisco en la ciudad de Guadalajara, México.</p>
      <p><strong>Dirección:</strong> Calle Aurelio González #2285, Jardines Alcalde, 44298</p>
      <p><strong>Teléfono:</strong> (333) 191-4694</p>
      <p><strong>Email:</strong> contacto@institutopuntoalejandria.com</p>
      <p>Estamos disponibles de lunes a viernes de 9:00 AM a 6:00 PM. No dudes en contactarnos para cualquier consulta o para más información sobre nuestros programas y certificaciones.</p>
      <div className="map-container">
        <iframe
          src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d14928.184262783387!2d-103.32640405335749!3d20.708353915389946!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x8428b1cbdfdfdf81%3A0xa6c34a3c432980b2!2sAurelio%20Gonz%C3%A1lez%202285%2C%20Jardines%20Alcalde%2C%2044298%20Guadalajara%2C%20Jalisco!5e0!3m2!1sen!2smx!4v1717982691553!5m2!1sen!2smx"
          width="100%"
          height="450"
          style={{ border: 0 }}
          allowFullScreen=""
          loading="lazy"
          referrerPolicy="no-referrer-when-downgrade"
        </iframe>
      </div>
    </div>
  );
};

export default Contact;

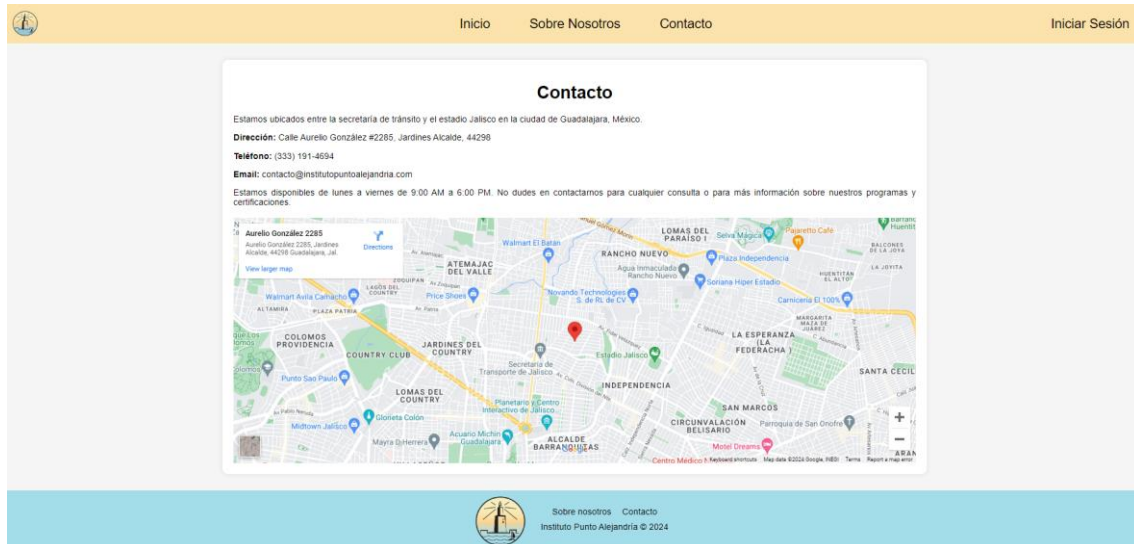
```

```

        title="Google Maps Location"
      <</iframe>
    </div>
  </div>
);
};

export default Contact;

```



Una página donde muestra el lugar de las oficinas con la información en texto y un mapa interactivo.

signIn.js:

```

import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { useAuth } from '../components/authContext';
import { useUser } from '../components/userContext';

const SignIn = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const { signIn } = useAuth();
  const { users } = useUser();
  const navigate = useNavigate();

  const handleSignIn = () => {
    const user = users.find(u => u.username === username && u.password === password);
    if (user) {
      signIn(user.username, user.role);
      navigate(user.role === 'admin' ? '/pagAdmin' : '/pagUser');
    } else {
      alert('Credenciales incorrectas');
    }
  };
};

```

```

    }
  };

  return (
    <div className="form-container">
      <h2>Iniciar Sesión</h2>
      <div className="form-group">
        <label>Nombre de usuario:</label>
        <input
          type="text"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
      </div>
      <div className="form-group">
        <label>Contraseña:</label>
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
      </div>
      <button className="submit-button" onClick={handleSignIn}>Iniciar
Sesión</button>
    </div>
  );
};

export default SignIn;

```

pagUser.js:

```

import React from 'react';
import '../App.css';

const pagUser = () => {
  return (
    <div className="content">
      <h1>Usuario</h1>
      <p>¡Bienvenido Usuario!</p>
    </div>
  );
};

```

```

    </div>
  );
};

export default pagUser;

```



Una página provisional para los usuarios donde pueden ver productos a los que están suscritos.

pagAdmin.js:

```

import React, { useState } from 'react';
import { useUser } from '../components/userContext';

const PagAdmin = () => {
  const { users, setUsers } = useUser();
  const [newUser, setNewUser] = useState({ username: '', password: '',
  role: 'user' });
  const [editUser, setEditUser] = useState(null);

  const addUser = () => {
    if (!newUser.username || !newUser.password) return; // Validación
    básica
    setUsers([...users, newUser]);
    setNewUser({ username: '', password: '', role: 'user' });
  };

  const deleteUser = (username) => {
    setUsers(users.filter(user => user.username !== username));
  };

  const startEditUser = (user) => {
    setEditUser(user);
  };

  const saveEditUser = () => {
    setUsers(users.map(user => (user.username === editUser.username ?
    editUser : user)));
    setEditUser(null);
  };

  return (
    <div className="admin-container">

```

```

<h1>Administrar Usuarios</h1>
<div>
  <h2>Usuarios Existentes</h2>
  <div className="user-list-header">
    <div>Usuario</div>
    <div>Contraseña</div>
    <div>Rol</div>
    <div>Acciones</div>
  </div>
  <ul className="user-list">
    {users.map(user => (
      <li key={user.username}>
        {editUser && editUser.username === user.username ? (
          <div className="user-details">
            <div>
              <input
                type="text"
                value={editUser.username}
                onChange={(e) => setEditUser({ ...editUser,
username: e.target.value })}
              />
            </div>
            <div>
              <input
                type="password"
                value={editUser.password}
                onChange={(e) => setEditUser({ ...editUser,
password: e.target.value })}
              />
            </div>
            <div>
              <select
                value={editUser.role}
                onChange={(e) => setEditUser({ ...editUser, role:
e.target.value })}
              >
                <option value="user">User</option>
                <option value="admin">Admin</option>
              </select>
            </div>
            <div className="user-actions">
              <button onClick={saveEditUser} className="edit-
button">Guardar</button>
              <button onClick={() => setEditUser(null)}
className='cancel-button'>Cancelar</button>
            </div>
          </div>
        ) : (
          <div className="user-details">

```



```

        <div>{user.username}</div>
        <div>{user.password}</div>
        <div>{user.role}</div>
        <div className="user-actions">
            <button onClick={() => startEditUser(user)}
className="edit-button">Editar</button>
            <button onClick={() => deleteUser(user.username)}
className="delete-button">Eliminar</button>
        </div>
    </div>
    )}
  </li>
  )}
</ul>
</div>
<div className="add-user-container">
  <h2>Agregar Usuario</h2>
  <input
    type="text"
    placeholder="Nombre de usuario"
    value={newUser.username}
    onChange={(e) => setNewUser({ ...newUser, username:
e.target.value })}
  />
  <input
    type="password"
    placeholder="Contraseña"
    value={newUser.password}
    onChange={(e) => setNewUser({ ...newUser, password:
e.target.value })}
  />
  <select
    value={newUser.role}
    onChange={(e) => setNewUser({ ...newUser, role: e.target.value
  })}
  >
    <option value="user">User</option>
    <option value="admin">Admin</option>
  </select>
  <button onClick={addUser}>Agregar</button>
</div>
</div>
  );
};

export default PagAdmin;

```

Administrar Usuarios

Usuarios Existentes

Usuario	Contraseña	Rol	Acciones
admin	admin	admin	Editar Eliminar
user	user	user	Editar Eliminar
test	test	user	Editar Eliminar
test2	test2	user	Editar Eliminar
test3	test3	user	Editar Eliminar

Agregar Usuario

Nombre de usuario: Contraseña: Rol: [Agregar](#)

Sobre nosotros Contacto Instituto Punto Alejandria © 2024

Una página en la que el administrador puede ver los datos que existen en userContext.js, puede crear nuevas entradas de datos, editarlas y eliminarlas. Haciendo posible el trabajo del CRUD en esta página.

CRUD en funcionamiento

Desde la página de administrador se tiene la oportunidad de acceder a los datos de los usuarios.

- Create:
Para la creación de un nuevo usuario, está el apartado de abajo donde dice “Agregar Usuario” y existen las entradas de datos para ingresarlas.

Administrar Usuarios

Usuarios Existentes

Usuario	Contraseña	Rol	Acciones
admin	admin	admin	Editar Eliminar
user	user	user	Editar Eliminar
test	test	user	Editar Eliminar
test2	test2	user	Editar Eliminar
test3	test3	user	Editar Eliminar

Agregar Usuario

usuarioPrueba clavePrueba [Agregar](#)

Sobre nosotros Contacto Instituto Punto Alejandria © 2024

The screenshot shows a web application interface for managing users. At the top, there is a navigation bar with links: Inicio, Sobre Nosotros, Contacto, and Cerrar Sesión. The main content area is titled 'Administrar Usuarios' and 'Usuarios Existentes'. It contains a table with the following data:


Usuario	Contraseña	Rol	Acciones
admin	admin	admin	Editar Eliminar
user	user	user	Editar Eliminar
test	test	user	Editar Eliminar
test2	test2	user	Editar Eliminar
test3	test3	user	Editar Eliminar
usuarioPrueba	clavePrueba	user	Editar Eliminar

Below the table, there is a form to 'Agregar Usuario' with fields for 'Nombre de usuario', 'Contraseña', and a dropdown for 'User'. A green 'Agregar' button is at the bottom right of the form.

Se puede ver el nuevo usuario agregado en la lista.

- Read:
Para leer los datos que hay actualmente en la lista, basta con entrar a la página de administrador para observar los datos que hay disponibles.
- Update:
Para actualizar los datos de un usuario, se puede hacer desde la página del administrador, haciendo clic en el botón de “Editar” y se habilitará la opción para editar los campos.

This screenshot shows the same 'Administrar Usuarios' page, but with the 'Editar' form for the 'usuarioPrueba' user active. The form fields are populated with the user's current data: 'usuarioPrueba' for the name, a masked password field, and 'User' for the role. There are 'Guardar' and 'Cancelar' buttons next to the form. The 'Agregar Usuario' form at the bottom remains visible.



[Inicio](#)
[Sobre Nosotros](#)
[Contacto](#)
[Cerrar Sesión](#)

Administrar Usuarios


Usuarios Existentes

Usuario	Contraseña	Rol	Acciones
admin	admin	admin	Editar Eliminar
user	user	user	Editar Eliminar
test	test	user	Editar Eliminar
test2	test2	user	Editar Eliminar
test3	test3	user	Editar Eliminar
usuarioEditado	claveEditada	admin	Editar Eliminar

Agregar Usuario


[Sobre nosotros](#)
[Contacto](#)

- **Delete:**
Para eliminar un usuario o una entrada de datos, desde la página de administrador se hace clic en el botón de “Eliminar” y así el registro se elimina.



[Inicio](#)
[Sobre Nosotros](#)
[Contacto](#)
[Cerrar Sesión](#)


Administrar Usuarios

Usuarios Existentes

Usuario	Contraseña	Rol	Acciones
admin	admin	admin	Editar Eliminar
user	user	user	Editar Eliminar
test	test	user	Editar Eliminar
test2	test2	user	Editar Eliminar
test3	test3	user	Editar Eliminar
usuarioEditado	claveEditada	admin	Editar Eliminar

Agregar Usuario


[Sobre nosotros](#)
[Contacto](#)



[Inicio](#)
[Sobre Nosotros](#)
[Contacto](#)
[Cerrar Sesión](#)

Administrar Usuarios

Usuarios Existentes

Usuario	Contraseña	Rol	Acciones
admin	admin	admin	Editar Eliminar
user	user	user	Editar Eliminar
test	test	user	Editar Eliminar
test2	test2	user	Editar Eliminar
test3	test3	user	Editar Eliminar

Agregar Usuario


[Sobre nosotros](#)
[Contacto](#)

Instituto Punto Alejandria © 2024

¿Cómo funciona React?

1- Componentes:

Se utilizan componentes reutilizables en la interfaz de usuario. Un componente es una pieza de la interfaz que puede tener su propio estado y lógica.

2- JSX:

JSX es una sintaxis que permite escribir estructuras HTML dentro de JavaScript. JSX crea elementos de React para ser utilizados dentro de la aplicación.

3- Estado y props:

Los componentes reciben datos a través de propiedades (props) y también tienen un estado con el que pueden interactuar con otros componentes.

4- Virtual DOM:

React tiene una copia del Modelo de Objetos del Documento (DOM), que se llama Virtual DOM. Cuando se hace un cambio en algún componente, React primero hace el cambio en Virtual DOM y después hace el cambio en el DOM real.

5- Hooks:

Los hooks permiten usar estado y otras características de React en componentes funcionales. Ejemplos incluyen “useState” para el estado y “useEffect” para efectos secundarios.

6- React Router:

El router es la biblioteca para manejar la navegación y el enrutamiento en las aplicaciones de React.

Conclusiones

React es una potente biblioteca de JavaScript que simplifica la creación de interfaces de usuario dinámicas y eficientes mediante el uso de componentes reutilizables y un enfoque declarativo. Su sistema de Virtual DOM optimiza las actualizaciones de la interfaz, mejorando el rendimiento y la experiencia del usuario. Con la introducción de los hooks, React ha facilitado aún más la gestión del estado y los efectos en los componentes funcionales, haciéndolo más flexible y accesible. En conjunto, estas características hacen de React una herramienta esencial para el desarrollo de aplicaciones web modernas y mantenibles.