Módulo 5
Desarrollo de aplicaciones Front-End con React

Seguridad en un Aplicativo Front-End

Adalid
aspasia
LA FORMACIÓN DE TU FUTURO

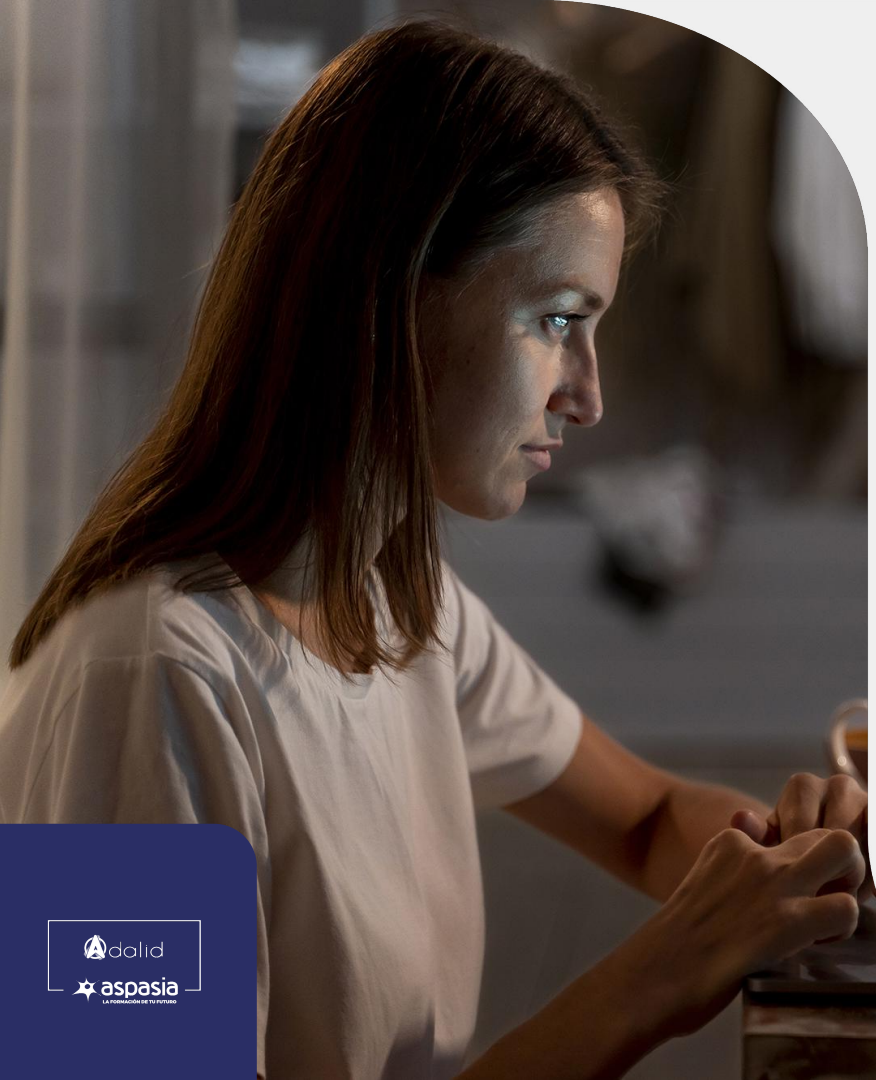
Módulo 5

AE 3.2

OBJETIVOS

Entender los riesgos de seguridad en aplicaciones web y React, aprender a proteger rutas, manejar roles, consumir servicios REST con Api Key/JWT, asegurar autenticación, y aplicar encriptación para datos sensibles.

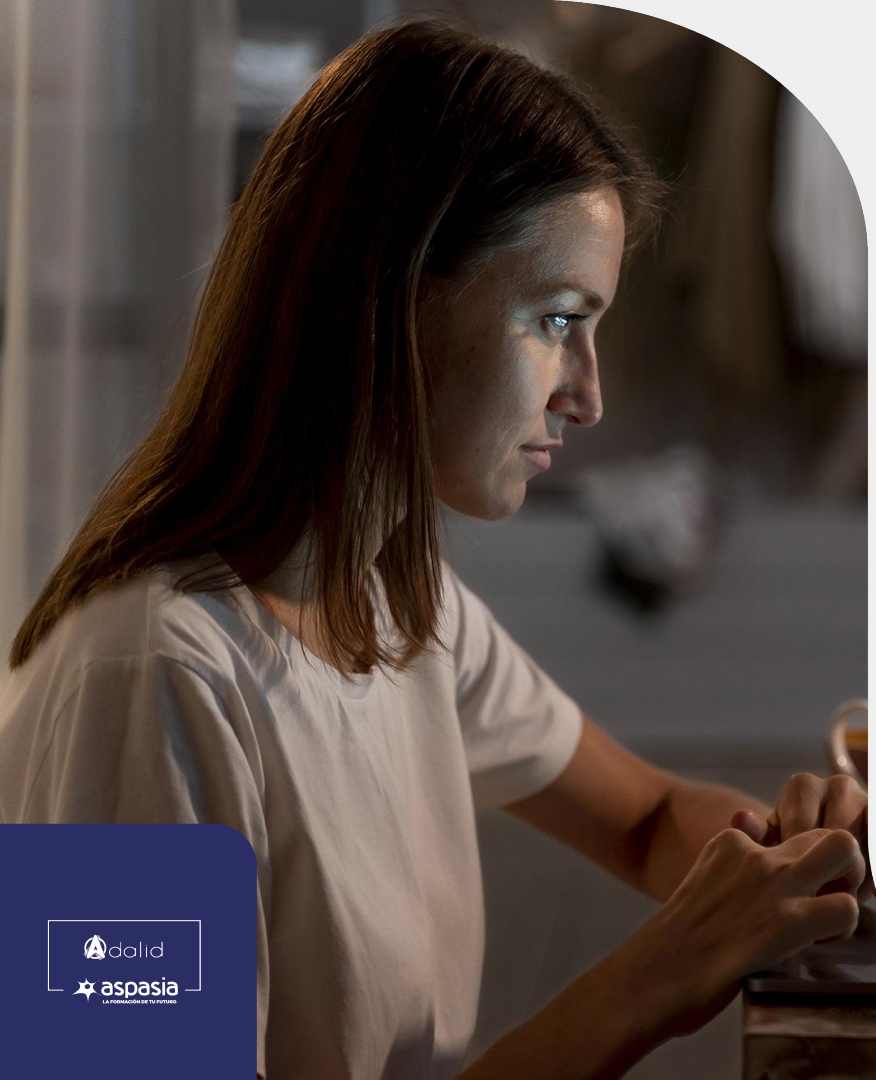
Chile
dalid
aspasia
LA FORMACIÓN DE TU FUTURO



¿QUÉ VAMOS A VER?

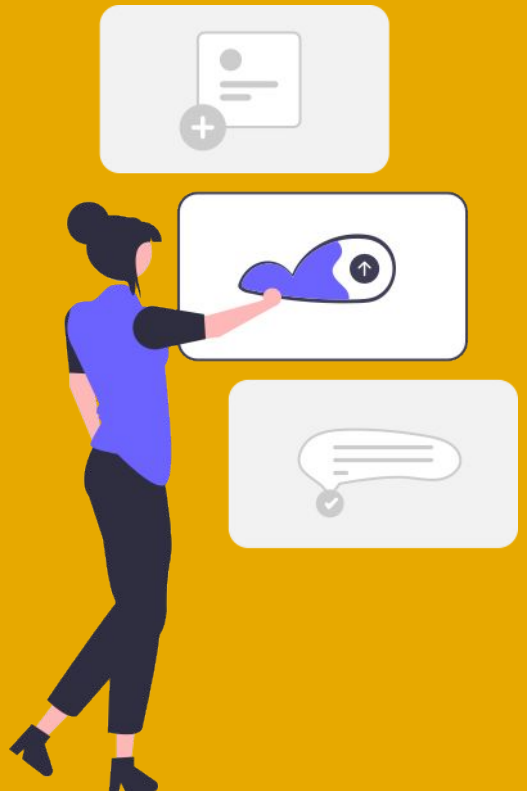
- Seguridad en un Aplicativo Front-End.
- Conceptos básicos de seguridad en aplicaciones Web (Clickjacking, Ataque XSS, SQL Injection, Ataque DoS).
- Recomendaciones de seguridad en una aplicación Web.
- Recomendaciones de seguridad en una aplicación ReactJs.
- Identificando vulnerabilidades en una aplicación ReactJs.

adalid aspasia



¿QUÉ VAMOS A VER?

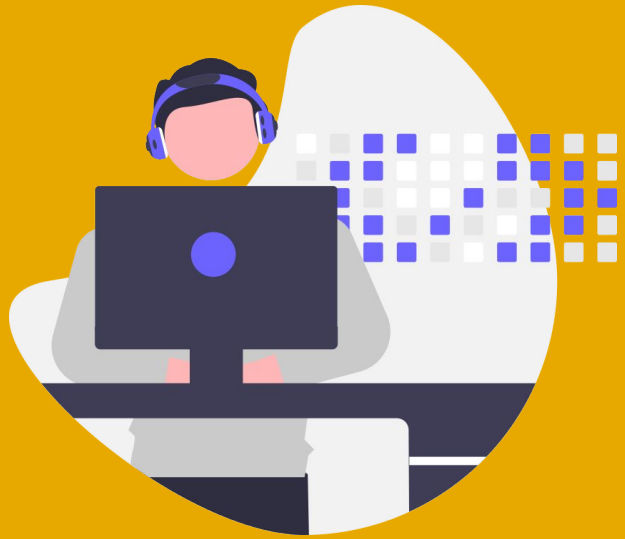
- Consumiendo servicios REST con Api Key y JWT.
- Cómo proteger rutas con React Router DOM.
- Implementando seguridad por Roles en React.
- La seguridad en la autenticación de usuarios.
- Encriptación de datos en el front.



Seguridad en un Aplicativo Front-End

Adalid
aspasia

Pongamos a prueba lo aprendido 😊!!!



Adalid aspasia

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

Desarrollaremos una aplicación React con Vite para gestionar reservas de vuelo, enfocándonos en implementar seguridad, manejo de estado y Hooks personalizados. Durante el ejercicio, cada avance será observable en el navegador y se garantizará el cumplimiento de los requisitos solicitados. Utilizaremos Bootstrap para asegurar un diseño atractivo y funcional.

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

1. Crea un Proyecto React

Crea el proyecto con Vite:

```
npm create vite@latest flight-reservation-system
```

Navega al directorio del proyecto e instala las dependencias:

```
cd flight-reservation-system
npm install
npm install bootstrap
npm install react-router-dom
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

2. Limpia la Estructura Base

Elimina archivos innecesarios para simplificar el proyecto.

1. Elimina los siguientes archivos en src:
    a. src/assets/
    b. src/App.css
    c. src/index.css
2. Modifica index.html para actualizar el título de la página

```
<title>Flight Reservation System</title>
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

3. Modifica el archivo App.jsx:

- Usamos React Router DOM para manejar las rutas.
- Creamos tres rutas básicas:
  - /: Página de inicio.
  - /booking: Formulario de reservas.
  - /dashboard: Panel principal con información de vuelos.

```jsx
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Home from './components/Home';
import BookingForm from './components/BookingForm';
import Dashboard from './components/Dashboard';

const App = () => {
  return (
    <Router>
      <div className="container my-5">
        <h1 className="text-center">Sistema de Reservas de Vuelo ✈️</h1>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/booking" element={<BookingForm />} />
          <Route path="/dashboard" element={<Dashboard />} />
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

4. Crea Componentes Iniciales:

- Crea el Componente **Home**
- **Ubicación:** src/components/Home.jsx

```jsx
import React from 'react';
import { Link } from 'react-router-dom';

const Home = () => {
  return (
    <div className="text-center">
      <h2>Bienvenido al Sistema de Reservas</h2>
      <p>Reserva vuelos de forma fácil y rápida.</p>
      <Link to="/booking" className="btn btn-primary">
        Realizar una Reserva
      </Link>
    </div>
  );
};

export default Home;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

4. Crea Componentes Iniciales:

- Crea el Componente **BookingForm**
- **Ubicación:** src/components/BookingForm.jsx

```
import React, { useState } from 'react';

const BookingForm = () => {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [flight, setFlight] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Reserva confirmada para ${name}. Vuelo: ${flight}`);
  };
```

```
return (
    <form onSubmit={handleSubmit} className="mx-auto" style={{
maxWidth: '400px' }}>
      <h3>Formulario de Reserva</h3>
      <div className="mb-3">
        <label className="form-label">Nombre</label>
        <input
          type="text"
          className="form-control"
          value={name}
          onChange={(e) => setName(e.target.value)}
          required
        />
      </div>
      <div className="mb-3">
        <label className="form-label">Correo Electrónico</label>
        <input
          type="email"
          className="form-control"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
      </div>
```

```
      <div className="mb-3">
        <label className="form-label">Vuelo</label>
        <select
          className="form-select"
          value={flight}
          onChange={(e) => setFlight(e.target.value)}
          required
        >
          <option value="">Selecciona un vuelo</option>
          <option value="Vuelo 101">Vuelo 101</option>
          <option value="Vuelo 202">Vuelo 202</option>
          <option value="Vuelo 303">Vuelo 303</option>
        </select>
      </div>
      <button type="submit" className="btn btn-success w-100">
        Confirmar Reserva
      </button>
    </form>
  );
};

export default BookingForm;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

4. Crea Componentes Iniciales:

- Crea el Componente **Dashboard**
- **Ubicación:** src/components/Dashboard.jsx

```jsx
import React from 'react';

const Dashboard = () => {
  return (
    <div className="text-center">
      <h2>Panel de Información de Vuelos</h2>
      <p>Aquí podrás consultar la disponibilidad de vuelos.</p>
    </div>
  );
};

export default Dashboard;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

5. Actualiza el Componente Dashboard:

Modificaremos este componente para que:

1. Realice una solicitud **HTTP** al cargar el componente.
2. Utilice **useEffect** para manejar el efecto secundario.
3. Muestre los vuelos obtenidos o un mensaje de error si ocurre un problema.

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

5. Actualiza el Componente Dashboard:

```
import React, { useEffect, useState } from 'react';

const Dashboard = () => {
  const [flights, setFlights] = useState([]); // Estado para almacenar los vuelos
  const [error, setError] = useState(''); // Estado para manejar errores

  useEffect(() => {
    const fetchFlights = async () => {
      try {
        const response = await fetch('https://jsonplaceholder.typicode.com/posts');
// API simulada
        if (!response.ok) {
          throw new Error('Error al obtener los vuelos.');
        }
        const data = await response.json();
        setFlights(data.slice(0, 10)); // Limitamos a 10 vuelos para simplicidad
      } catch (err) {
        setError(err.message);
      }
    };
    fetchFlights();
  }, []); // Se ejecuta solo una vez al montar el componente

  return (
    <div className="mt-4">
      <h2 className="text-center">Vuelos Disponibles</h2>
      {error ? (
        <p className="text-danger text-center">{error}</p>
      ) : (
        <ul className="list-group">
          {flights.map((flight) => (
            <li key={flight.id} className="list-group-item">
              <h5>{flight.title}</h5>
              <p>{flight.body}</p>
            </li>
          ))}
        </ul>
      )}
    </div>
  );
};

export default Dashboard;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

5. Actualiza el Componente Dashboard:

- **Explicación del Código**
  - **useEffect:**
    - Se utiliza para realizar una solicitud HTTP al montar el componente.
    - La dependencia [] asegura que se ejecute solo una vez.
  - **Manejo de Errores:**
    - Si la solicitud falla, el estado error se actualiza con un mensaje descriptivo.
  - **Listado de Vuelos:**
    - Se muestra una lista de vuelos obtenida de la API, con un límite de 10 elementos.

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

6. Crear un Hook Personalizado (useForm)

- Crea el Archivo del **Hook**
- Ubicación: src/hooks/useForm.js
- Implementaremos un Hook que:
  - Gestione el estado de formularios.
  - Maneje errores de validación.
  - Devuelva métodos y datos necesarios para interactuar con el formulario.

```
import { useState } from 'react';

const useForm = (initialValues, validate, onSubmit) => {
  const [values, setValues] = useState(initialValues);
  const [errors, setErrors] = useState({});
  const [isSubmitting, setIsSubmitting] = useState(false);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setValues({ ...values, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    const validationErrors = validate(values);
    setErrors(validationErrors);
    if (Object.keys(validationErrors).length === 0) {
      setIsSubmitting(true);
      onSubmit(values);
      setIsSubmitting(false);
    }
  };

  return { values, errors, isSubmitting, handleChange, handleSubmit };
};

export default useForm;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

7. Usar useForm en BookingForm

- Actualizaremos el componente BookingForm para usar el Hook personalizado.

```
import React from 'react';
import useForm from '../hooks/useForm';

const BookingForm = () => {
  const validate = (values) => {
    const errors = {};
    if (!values.name) errors.name = 'El nombre es obligatorio.';
    if (!values.email) errors.email = 'El correo es obligatorio.';
    if (!values.flight) errors.flight = 'Debes seleccionar un vuelo.';
    return errors;
  };
```

```
  const onSubmit = (values) => {
    alert(`Reserva confirmada para ${values.name}. Vuelo: ${values.flight}`);
  };

  const { values, errors, handleChange, handleSubmit } = useForm(
    { name: '', email: '', flight: '' },
    validate,
    onSubmit
  );
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

7. Usar useForm en BookingForm

- Actualizaremos el componente BookingForm para usar el Hook personalizado.

```
return (
  <form onSubmit={handleSubmit} className="mx-auto" style={{ maxWidth: '400px' }}>
    <h3>Formulario de Reserva</h3>
    <div className="mb-3">
      <label className="form-label">Nombre</label>
      <input
        type="text"
        name="name"
        value={values.name}
        onChange={handleChange}
        className="form-control"
      />
      {errors.name && <p className="text-danger">{errors.name}</p>}
    </div>
    <div className="mb-3">
      <label className="form-label">Correo Electrónico</label>
      <input
        type="email"
        name="email"
        value={values.email}
        onChange={handleChange}
        className="form-control"
      />
      {errors.email && <p className="text-danger">{errors.email}</p>}
    </div>
    <div className="mb-3">
      <label className="form-label">Vuelo</label>
      <select
        name="flight"
        value={values.flight}
        onChange={handleChange}
        className="form-select"
      >
        <option value="">Selecciona un vuelo</option>
        <option value="Vuelo 101">Vuelo 101</option>
        <option value="Vuelo 202">Vuelo 202</option>
        <option value="Vuelo 303">Vuelo 303</option>
      </select>
      {errors.flight && <p className="text-danger">{errors.flight}</p>}
    </div>
    <button type="submit" className="btn btn-success w-100">
      Confirmar Reserva
    </button>
  </form>
 );
};

export default BookingForm;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

8. Crea Servicio Centralizado para API

- **Ubicación:** src/services/apiService.js
- Este archivo manejará las solicitudes HTTP para garantizar consistencia y facilitar el manejo de errores.

```
const BASE_URL = 'https://jsonplaceholder.typicode.com';

export const fetchFlights = async () => {
  try {
    const response = await fetch(`${BASE_URL}/posts`);
    if (!response.ok) {
      throw new Error('No se pudieron cargar los vuelos. Inténtalo más tarde.');
    }
    const data = await response.json();
    return data.slice(0, 10); // Devuelve los primeros 10 vuelos simulados
  } catch (error) {
    console.error(error.message);
    throw error; // Propaga el error para que el componente lo maneje
  }
};
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

9. Actualizar el Componente Dashboard

- Modificaremos Dashboard para usar el servicio de API y manejar errores adecuadamente.

```
import React, { useEffect, useState } from 'react';
import { fetchFlights } from '../services/apiService';

const Dashboard = () => {
  const [flights, setFlights] = useState([]);
  const [error, setError] = useState('');

  useEffect(() => {
    const loadFlights = async () => {
      try {
        const data = await fetchFlights();
        setFlights(data);
      } catch (err) {
        setError(err.message);
      }
    };

    loadFlights();
  }, []);
```

```
  return (
    <div className="mt-4">
      <h2 className="text-center">Vuelos Disponibles</h2>
      {error ? (
        <p className="text-danger text-center">{error}</p>
      ) : (
        <ul className="list-group">
          {flights.map((flight) => (
            <li key={flight.id} className="list-group-item">
              <h5>{flight.title}</h5>
              <p>{flight.body}</p>
            </li>
          ))}
        </ul>
      )}
    </div>
  );
};

export default Dashboard;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

10. Proteger Rutas con React Router DOM

- Crear el Componente **ProtectedRoute**
- **Ubicación:** src/components/ProtectedRoute.jsx
- Este componente redirigirá a los usuarios no autenticados a la página de inicio de sesión.

```
import React from 'react';
import { Navigate, Outlet } from 'react-router-dom';

const ProtectedRoute = () => {
  const isAuthenticated = localStorage.getItem('authToken') !== null;

  return isAuthenticated ? <Outlet /> : <Navigate to="/" />;
};

export default ProtectedRoute;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

11. Actualiza App.jsx para Usar Rutas Protegidas

```jsx
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Home from './components/Home';
import BookingForm from './components/BookingForm';
import Dashboard from './components/Dashboard';
import ProtectedRoute from './components/ProtectedRoute';
```

- **ProtectedRoute** permite el acceso solo si hay un token de autenticación en localStorage.

```jsx
const App = () => {
  return (
    <Router>
      <div className="container my-5">
        <h1 className="text-center">Sistema de Reservas de Vuelo ✈️</h1>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/booking" element={<BookingForm />} />
          {/* Ruta protegida */}
          <Route element={<ProtectedRoute />}>
            <Route path="/dashboard" element={<Dashboard />} />
          </Route>
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

12. Crea el Componente RoleBasedRoute

- Este componente permitirá el acceso a ciertas rutas solo a usuarios con un rol específico.
- **Ubicación:** src/components/RoleBasedRoute.jsx

```
import React from 'react';
import { Navigate, Outlet } from 'react-router-dom';

const RoleBasedRoute = ({ allowedRoles }) => {
  const userRole = localStorage.getItem('userRole');

  return allowedRoles.includes(userRole) ? <Outlet /> : <Navigate to="/" />;
};

export default RoleBasedRoute;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

13. Simula Autenticación y Roles

- Para propósitos de prueba, simularemos un sistema básico de autenticación con roles.
- Modifica el componente **Home** para permitir que los usuarios se autentiquen con un rol.

```
import React from 'react';
import { useNavigate } from 'react-router-dom';

const Home = () => {
  const navigate = useNavigate();

  const handleLogin = (role) => {
    localStorage.setItem('authToken', 'mockToken123');
    localStorage.setItem('userRole', role);
    navigate('/dashboard');
  };
```

```
  return (
    <div className="text-center">
      <h2>Bienvenido al Sistema de Reservas</h2>
      <p>Inicia sesión como:</p>
      <button
        className="btn btn-primary mx-2"
        onClick={() => handleLogin('user')}
      >
        Usuario
      </button>
      <button
        className="btn btn-secondary mx-2"
        onClick={() => handleLogin('admin')}
      >
        Administrador
      </button>
    </div>
  );
};

export default Home;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

14. Crea el Componente AdminPanel

- Página exclusiva para usuarios con rol de administrador.
- **Ubicación:** src/components/AdminPanel.jsx

```jsx
import React from 'react';

const AdminPanel = () => {
  return (
    <div className="mt-4">
      <h2 className="text-center">Panel de Administración</h2>
      <p>Acceso exclusivo para administradores.</p>
    </div>
  );
};

export default AdminPanel;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

15. Protege el Acceso a AdminPanel

- Incluye la ruta para el panel de administración con protección basada en roles.

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Home from './components/Home';
import BookingForm from './components/BookingForm';
import Dashboard from './components/Dashboard';
import ProtectedRoute from './components/ProtectedRoute';
import RoleBasedRoute from './components/RoleBasedRoute';
import AdminPanel from './components/AdminPanel';
```

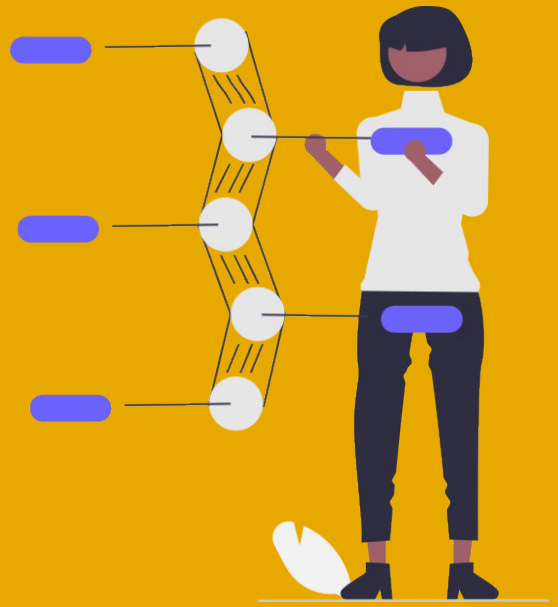
```
const App = () => {
  return (
    <Router>
      <div className="container my-5">
        <h1 className="text-center">Sistema de Reservas de Vuelo ✈️</h1>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/booking" element={<BookingForm />} />

          {/* Ruta protegida */}
          <Route element={<ProtectedRoute />}>
            <Route path="/dashboard" element={<Dashboard />} />

            {/* Ruta basada en roles */}
            <Route element={<RoleBasedRoute allowedRoles={['admin']} />}>
              <Route path="/admin" element={<AdminPanel />} />
            </Route>
          </Route>
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

Ejercicio Guiado: Sistema de Reservas de Vuelo con Seguridad y Gestión de Hooks

Puedes añadir funcionalidades adicionales:

- Navegación para las rutas como:
  - dashboard
  - booking
- Integra de forma correcta el adminPanel

Resumen de lo aprendido 🧑‍🎓



Adalid aspasia

Resumen de lo aprendido

- Aprendiste los riesgos comunes en aplicaciones web, como XSS, SQL Injection y DoS, y cómo mitigarlos con validación de entradas, cabeceras de seguridad y HTTPS.
- Implementaste medidas de seguridad en React, como proteger rutas con React Router DOM, roles de usuario y manejo seguro de tokens con JWT y API Keys.
- Descubriste cómo identificar vulnerabilidades en aplicaciones React y asegurar datos sensibles mediante encriptación con librerías como crypto-js.
- Aplicaste buenas prácticas en autenticación, como almacenamiento seguro de tokens y validación en el backend, para garantizar la integridad y seguridad del usuario.

GRACIAS POR TU ATENCIÓN

Nos vemos en la próxima clase

Chile
Adalid
aspasia
LA FORMACIÓN DE TU FUTURO