Módulo 5
Desarrollo de aplicaciones Front-End con React

Seguridad en un Aplicativo Front-End

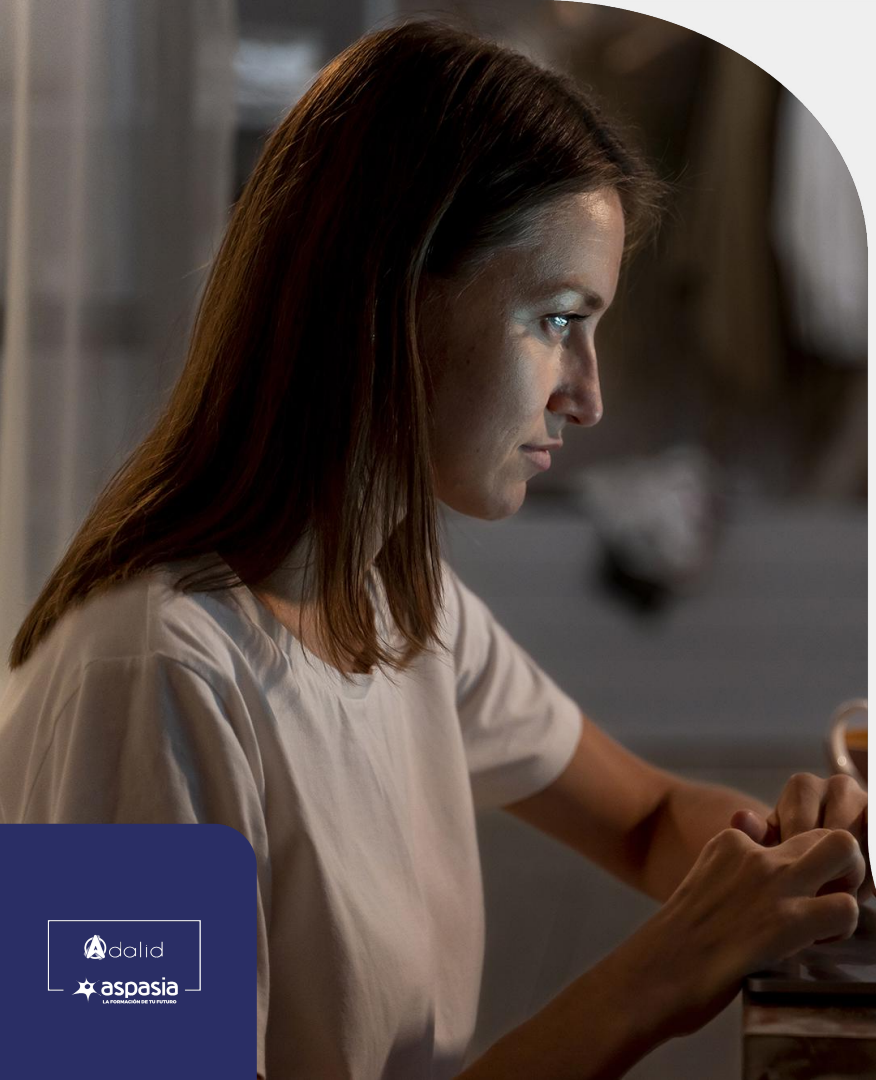
Adalid
aspasia
LA FORMACIÓN DE TU FUTURO

Módulo 5

AE 3.1

OBJETIVOS

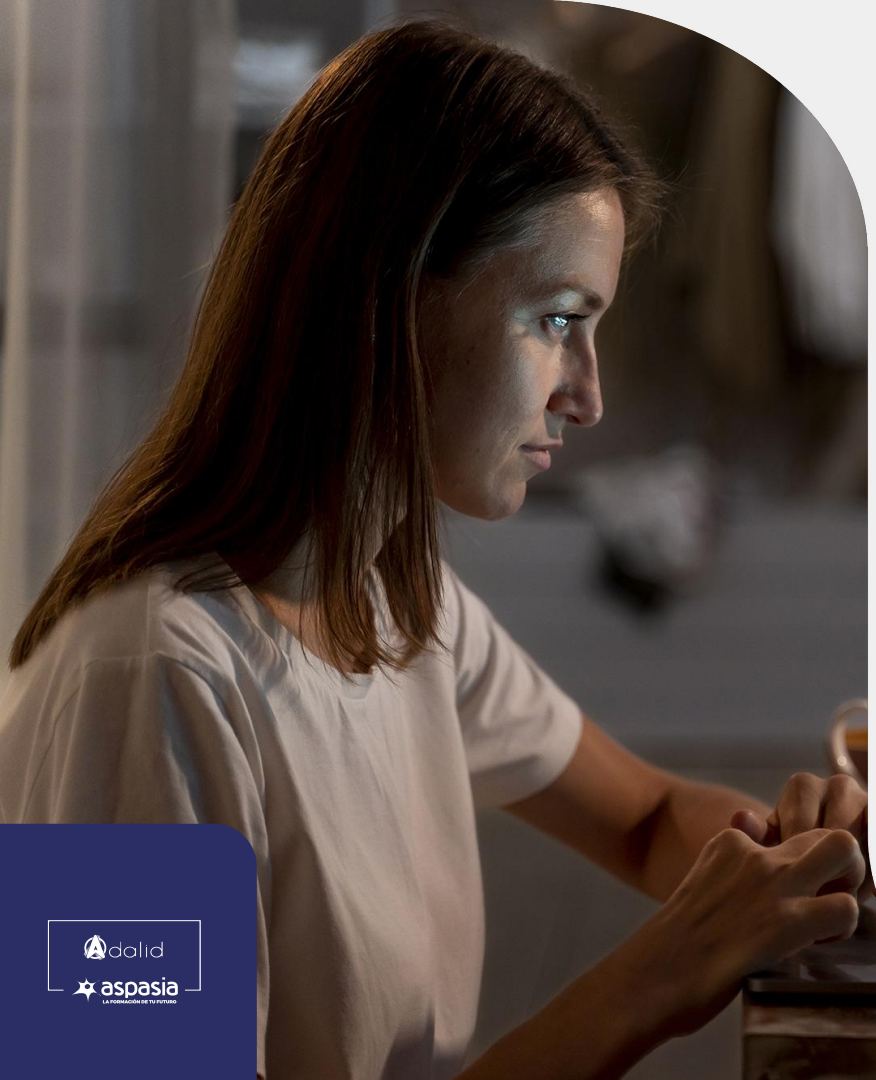
Entender los riesgos de seguridad en aplicaciones web y React, aprender a proteger rutas, manejar roles, consumir servicios REST con Api Key/JWT, asegurar autenticación, y aplicar encriptación para datos sensibles.



¿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.

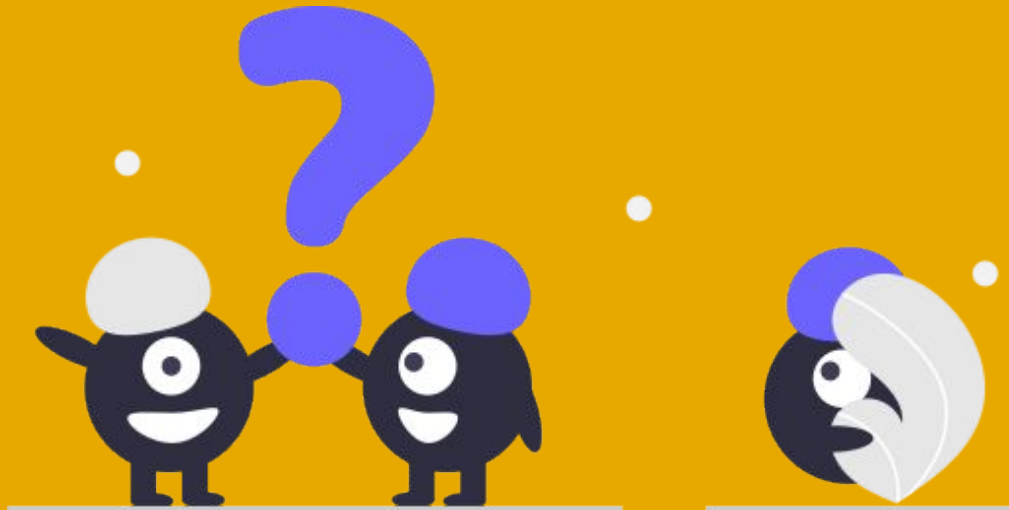
adalid aspasia



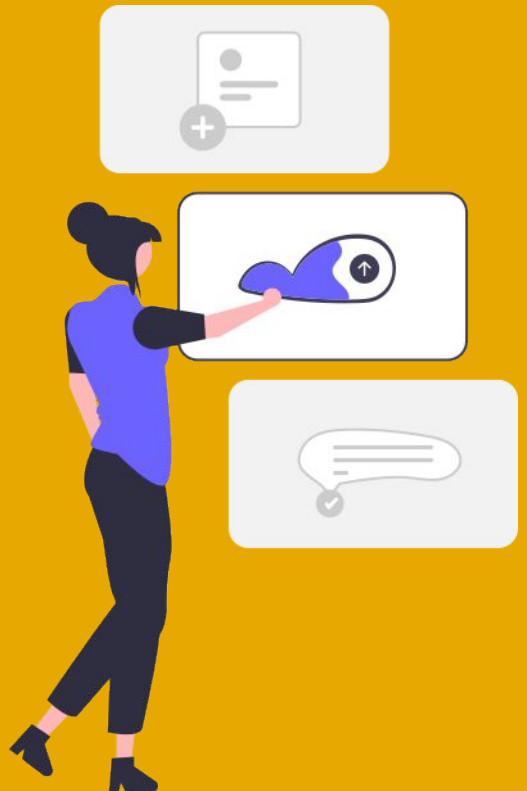
¿QUÉ VAMOS A VER?

- 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.

¿Conoces los riesgos de seguridad de React?



Adalid
aspasia

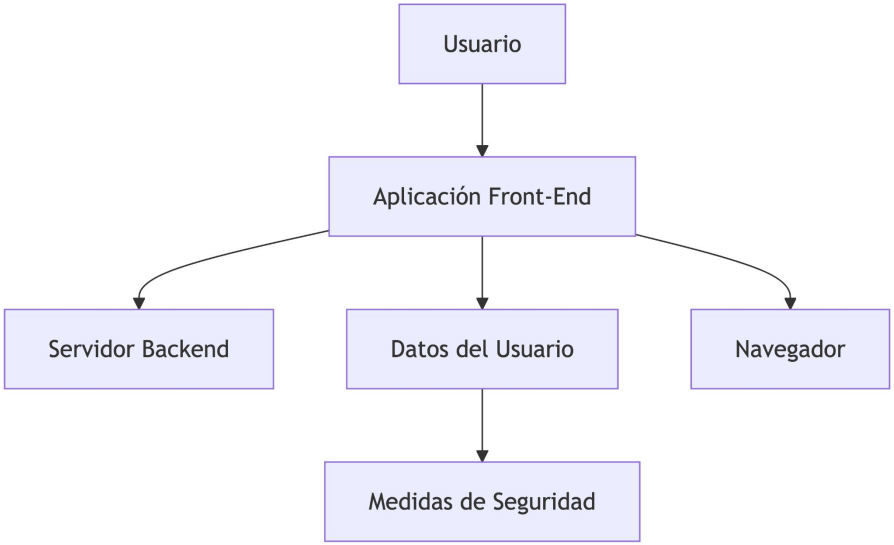


Seguridad en un Aplicativo Front-End

Adalid
aspasia

Seguridad en un Aplicativo Front-End

La seguridad en el front-end busca **proteger los datos del usuario**, las interacciones y la comunicación con el backend. Aunque muchas vulnerabilidades se originan en el servidor, **el front-end puede ser el punto de entrada para ataques**.



Conceptos Básicos de Seguridad en Aplicaciones Web

Clickjacking

- El atacante carga una aplicación legítima dentro de un iframe oculto para capturar clics del usuario.
- **Solución:** Configurar la cabecera HTTP X-Frame-Options en el servidor.

```
// Ejemplo en Express.js
res.setHeader('X-Frame-Options', 'DENY');
```

Ataque XSS (Cross-Site Scripting)

- Inyección de scripts maliciosos en el cliente.
- **Solución:** Escapar contenido dinámico y usar librerías como DOMPurify.

```
import DOMPurify from 'dompurify';

const sanitizedHTML = DOMPurify.sanitize(userInput);
```

Conceptos Básicos de Seguridad en Aplicaciones Web

SQL Injection

- Inyección de código SQL en formularios para manipular la base de datos.
- **Solución:** Utilizar consultas preparadas en el backend.

Ataque DoS (Denegación de Servicio)

- Sobrecarga de recursos mediante solicitudes masivas.
- **Solución:** Limitar las solicitudes desde una IP con herramientas como express-rate-limit.

Recomendaciones de Seguridad en una Aplicación Web

- **Configurar HTTPS:** Garantiza una conexión cifrada entre el cliente y el servidor.
- **Validación de Entradas:** Verifica que todos los datos enviados por el usuario sean válidos.
- **Manejo de Sesiones:** Usa tokens seguros como JWT y configura tiempos de expiración.
- **Cabeceras HTTP:** Configura cabeceras de seguridad como Content-Security-Policy y Strict-Transport-Security.

Recomendaciones de Seguridad en una Aplicación ReactJS

- **Evitar HTML peligroso:** Nunca uses dangerouslySetInnerHTML sin sanitización.
- **Protección contra XSS:** Escapa todo contenido dinámico.
- **Manejo de tokens:** Almacena tokens en cookies seguras con la opción httpOnly habilitada.
- **Variables de entorno:** Usa variables de entorno para configurar claves sensibles.

Identificando Vulnerabilidades en una Aplicación ReactJS

Uso de herramientas:

- **React Developer Tools:** Identifica componentes inseguros.
- **OWASP ZAP:** Escanea tu aplicación en busca de vulnerabilidades comunes.

Ejemplo: Detectar dependencias vulnerables, ejecuta en la consola:

```
npm audit
```

Consumiendo Servicios REST con Api Key y JWT

Uso de API Key

Envía una clave en los encabezados de las solicitudes.

```
fetch('https://api.example.com/data', {
  headers: {
    'Authorization': 'Api-Key your_api_key_here',
  },
});
```

Uso de JWT (JSON Web Tokens)

Autentica solicitudes con un token firmado.

```
const token = localStorage.getItem('authToken');
fetch('https://api.example.com/protected', {
  headers: {
    'Authorization': `Bearer ${token}`,
  },
});
```

Cómo Proteger Rutas con React Router DOM

Protección de rutas:

Redirige usuarios no autenticados a una página de inicio de sesión.

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

const ProtectedRoute = ({ isAuthenticated }: { isAuthenticated: boolean }) => {
  return isAuthenticated ? <Outlet /> : <Navigate to="/login" />;
};
```

Implementando Seguridad por Roles en React

Ejemplo práctico:

- Define roles en el estado:

```
const user = { role: 'admin' };

const isAuthorized = (role: string) => user.role
=== role;
```

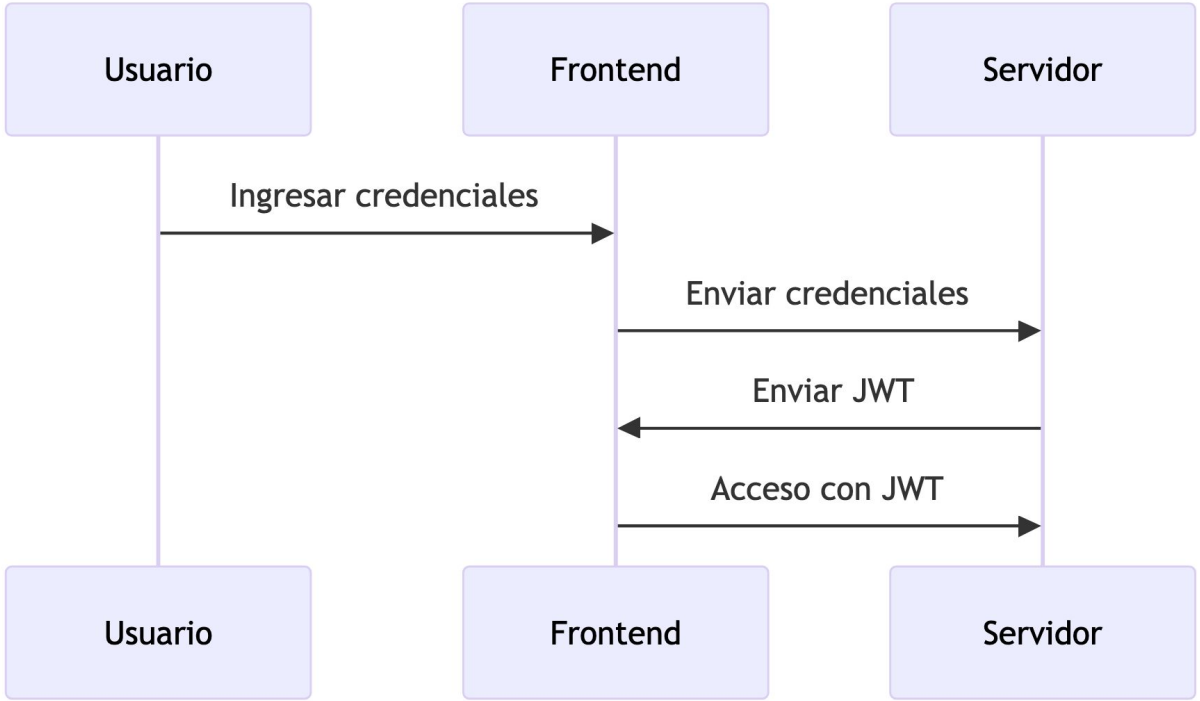
- Controla el acceso:

```
{isAuthorized('admin') ? <AdminPanel /> :
<Unauthorized />}
```

La Seguridad en la Autenticación de Usuarios

- **Almacenamiento seguro de credenciales:** Usa cookies con httpOnly para evitar accesos desde JavaScript.
- **Validación en el backend:** Verifica siempre los tokens enviados desde el cliente.

La Seguridad en la Autenticación de Usuarios



Encriptación de Datos en el Front

Usa la librería crypto-js para encriptar datos sensibles antes de enviarlos al servidor.

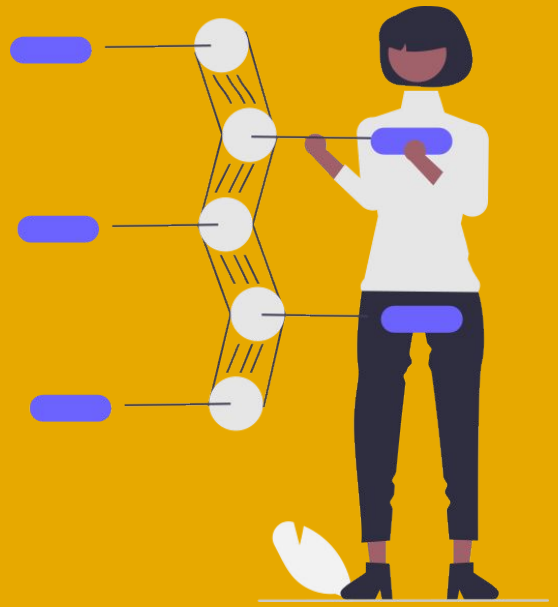
```
import CryptoJS from 'crypto-js';

const encrypted = CryptoJS.AES.encrypt('datos sensibles', 'clave_secreta').toString();
const decrypted = CryptoJS.AES.decrypt(encrypted, 'clave_secreta').toString(CryptoJS.enc.Utf8);

console.log(decrypted); // datos sensibles
```

Acceso al Repositorio

En el siguiente enlace podrás acceder a un repositorio relacionado con la temática propuesta.

https://github.com/adalid-cl/ESPECIALIZACION_FRONTEND_M5_AE3

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