Clase N° 16: Middlewares y autenticación

Crombie

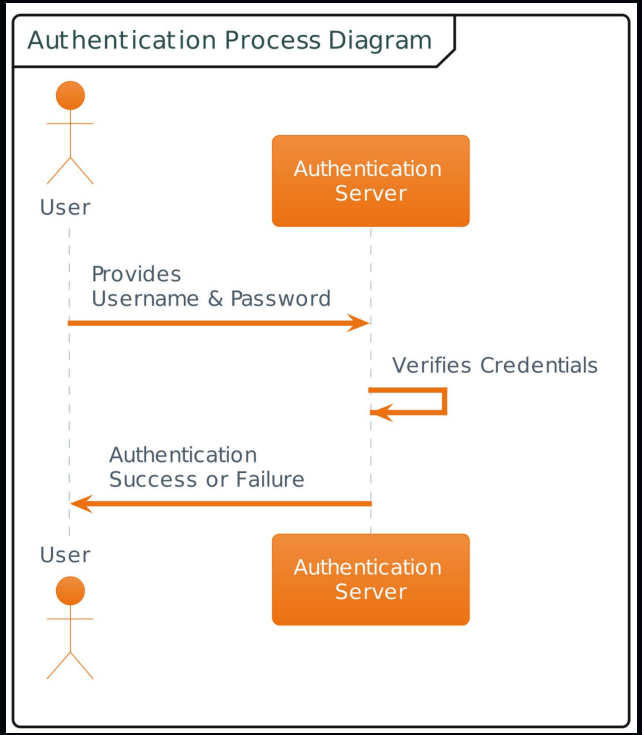
Clase N° 16: Agenda

- ¿Qué es la autenticación?
- Por qué es importante.
- Opciones en Next.js.
- Uso de tokens JWT.
- Integración con bases de datos como MySQL o servicios externos.

Crombie

Qué es la autenticación?

- Proceso de verificar la identidad de un usuario para otorgarle acceso a recursos específicos.



Crombie

Por qué es importante?

- ➔ Protege datos sensibles.
- ➔ Permite personalizar la experiencia del usuario.

Crombie

Algunas formas de autenticación

- Single Sign-On (SSO):
  - Permite a los usuarios acceder a múltiples aplicaciones con una sola credencial.
  - Ejemplo: Iniciar sesión con Google o Microsoft en varias aplicaciones.
  - Ventajas: Simplifica la experiencia del usuario y mejora la seguridad.

- Magic Link (enviado al email):
  - El usuario proporciona su correo electrónico y recibe un enlace único para iniciar sesión.
  - No requiere contraseña, minimizando riesgos asociados a contraseñas débiles o robadas.

Algunas formas de autenticación

- Con credenciales:
  - Método clásico con usuario y contraseña.
  - Requiere almacenamiento seguro de contraseñas (p. ej., usando hashing con bcrypt).

- OAuth:
  - Protocolo estándar para delegar autenticación y autorización.
  - Ejemplo: Iniciar sesión con cuentas de terceros como Google, GitHub o Facebook.
  - Permite obtener acceso controlado a recursos sin compartir contraseñas.

Cómo elegir uno?

- ➔ Facilidad de uso para el usuario.
- ➔ Requisitos de seguridad del proyecto.
- ➔ Compatibilidad con dispositivos y plataformas.
- ➔ Escalabilidad para futuros usuarios o servicios.

Libreria Auth.js

- ➔ Soporta múltiples métodos de autenticación.
- ➔ Fácil gestión de sesiones.
- ➔ Integración a proyectos rápida y sencilla.

¿Qué son los JWT?

- Los JWT (JSON Web Tokens) son un estándar para representar información de forma compacta y segura entre dos partes.
- Contienen tres partes:
  - header
  - payload
  - signature
- Los JWT permiten autenticar usuarios sin mantener estado en el servidor.

JWT - Auth.js

```
import NextAuth from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";

const handler = NextAuth({
  providers: [
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        username: { label: "Username", type: "text" },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        // Lógica para verificar usuario
        const user = { id: 1, name: "Admin", email: "admin@example.com" };
        if (user) return user;
        return null;
      },
    }),
  ],
  session: {
    strategy: "jwt",
  },
});

export { handler as GET, handler as POST };
```

Qué es un middleware?

- Funciones que interceptan solicitudes HTTP antes de que lleguen al endpoint correspondiente.
- Casos de uso:
  - Redirección de usuarios no autenticados.
  - Registro de actividad y auditoría.
  - Validación de datos antes de llegar al backend.

```ts
// @/middleware.ts

import { NextResponse } from "next/server";

export function middleware(request) {
  const token = request.cookies.get("token")?.value;

  if (!token && request.nextUrl.pathname.startsWith("/dashboard")) {
    return NextResponse.redirect(new URL("/login", request.url));
  }

  return NextResponse.next();
}

export const config = {
    matcher: ["/dashboard/:path*"],
  };
```

Pasemos a la práctica

Teniendo en cuenta la documentación oficial de Next.js (versión 15), veamos el proceso de autenticación entre todos.