AUTENTICACIÓN EN ASP.NET CORE

CURSO .NET – IT SCHOOL

IT SCHOOL LA POSTA!

CONCEPTOS BÁSICOS DE SEGURIDAD EN WEB

- La Importancia de Proteger Datos y Recursos
  - En el contexto de aplicaciones web backend, proteger datos y recursos es fundamental para asegurar la confidencialidad, integridad y disponibilidad de la información. La protección de datos sensibles, como información personal, financiera o médica, es crucial para evitar el robo de identidad, fraudes y otras formas de abuso. Los usuarios confían en que sus datos estarán seguros, lo que incrementa su lealtad y uso de la aplicación. Además, las regulaciones como el Reglamento General de Protección de Datos (GDPR) y la Ley de Privacidad del Consumidor de California (CCPA) exigen medidas estrictas de protección de datos, y el incumplimiento puede resultar en sanciones significativas.

CONCEPTOS BÁSICOS DE SEGURIDAD EN WEB

- La Importancia de Proteger Datos y Recursos
  - Proteger los recursos de una aplicación también garantiza que solo los usuarios autorizados puedan acceder y manipular información crítica. Esto previene la manipulación malintencionada de datos y funciones, manteniendo la integridad de la aplicación. Además, mantener la seguridad de los datos y recursos ayuda a preservar la reputación de la empresa, diferenciando a la aplicación en un mercado competitivo.

CONCEPTOS BÁSICOS DE SEGURIDAD EN WEB

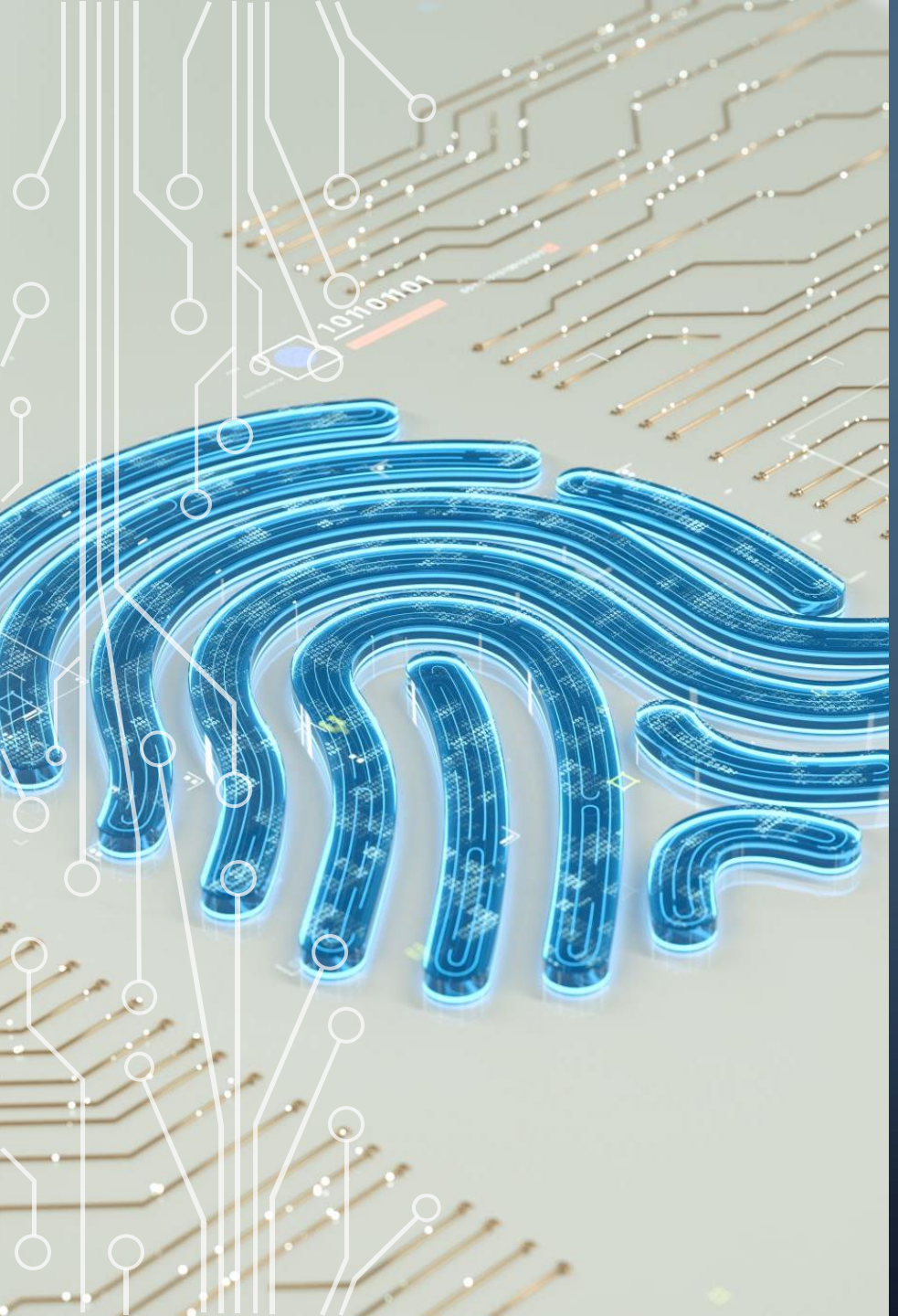
- Principios Fundamentales de Seguridad Web
  - Autenticación: La autenticación es el proceso de verificar la identidad de un usuario o sistema que intenta acceder a un recurso. Este proceso asegura que quien accede es realmente quien dice ser. Los métodos de autenticación pueden variar desde el uso de contraseñas y tokens hasta biometría y autenticación multifactor (MFA), que combina varios métodos de autenticación para incrementar la seguridad.

CONCEPTOS BÁSICOS DE SEGURIDAD EN WEB

- Principios Fundamentales de Seguridad Web
  - Autorización: Una vez autenticado el usuario, la autorización determina qué acciones puede realizar o a qué recursos puede acceder. Esto se puede implementar a través de controles basados en roles (Role-Based Access Control, RBAC), donde los permisos se asignan a roles específicos, y controles basados en políticas (Policy-Based Access Control, PBAC), donde las políticas definen qué permisos están disponibles para cada usuario o grupo en determinadas condiciones.

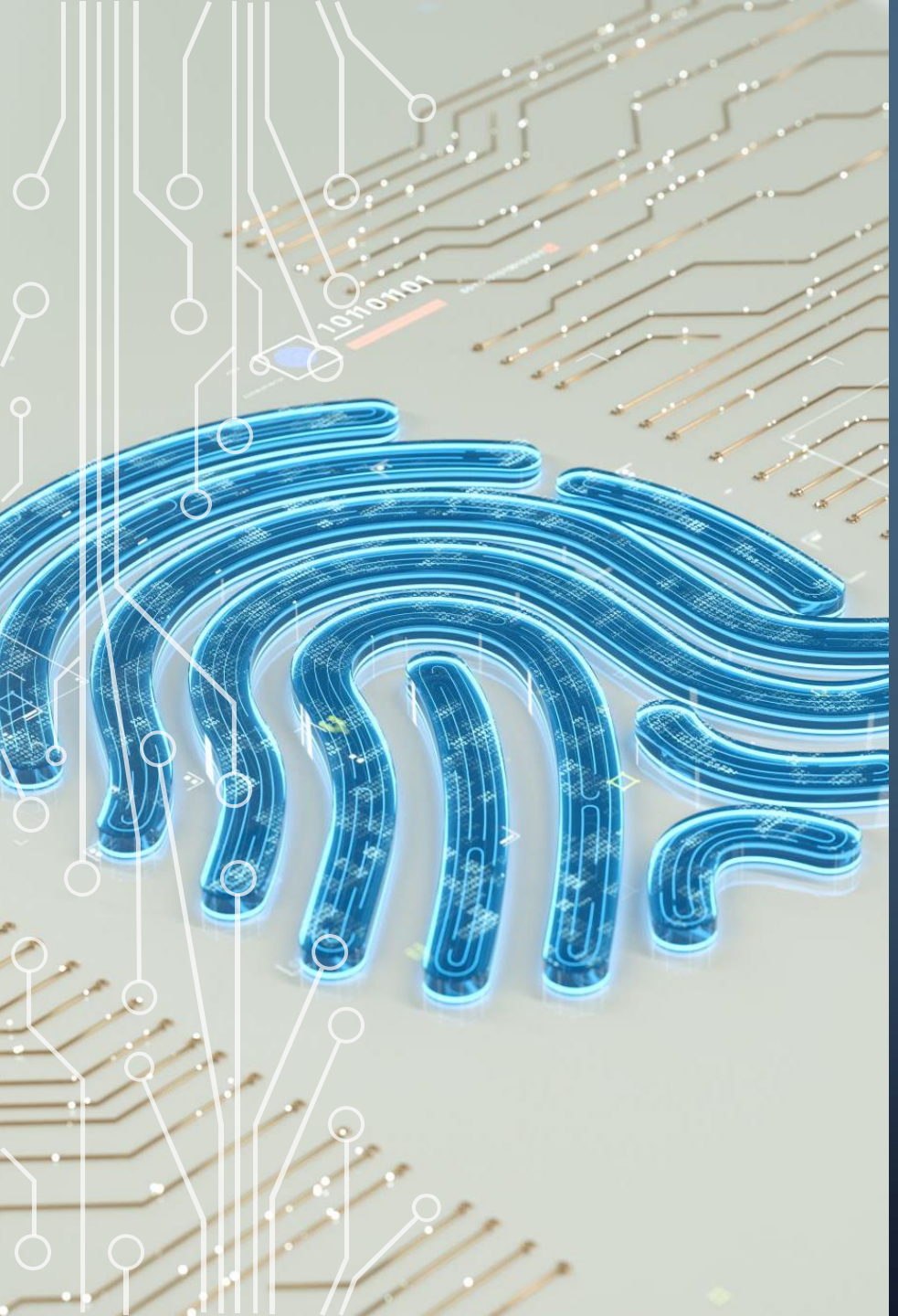
CONCEPTOS BÁSICOS DE SEGURIDAD EN WEB

- Principios Fundamentales de Seguridad Web
  - Principio de Mínimo Privilegio: El principio de mínimo privilegio establece que los usuarios y sistemas solo deben tener los permisos necesarios para realizar sus funciones, y nada más. Este principio minimiza la superficie de ataque potencial, reduciendo el riesgo de accesos no autorizados y daños potenciales en caso de compromisos de seguridad.



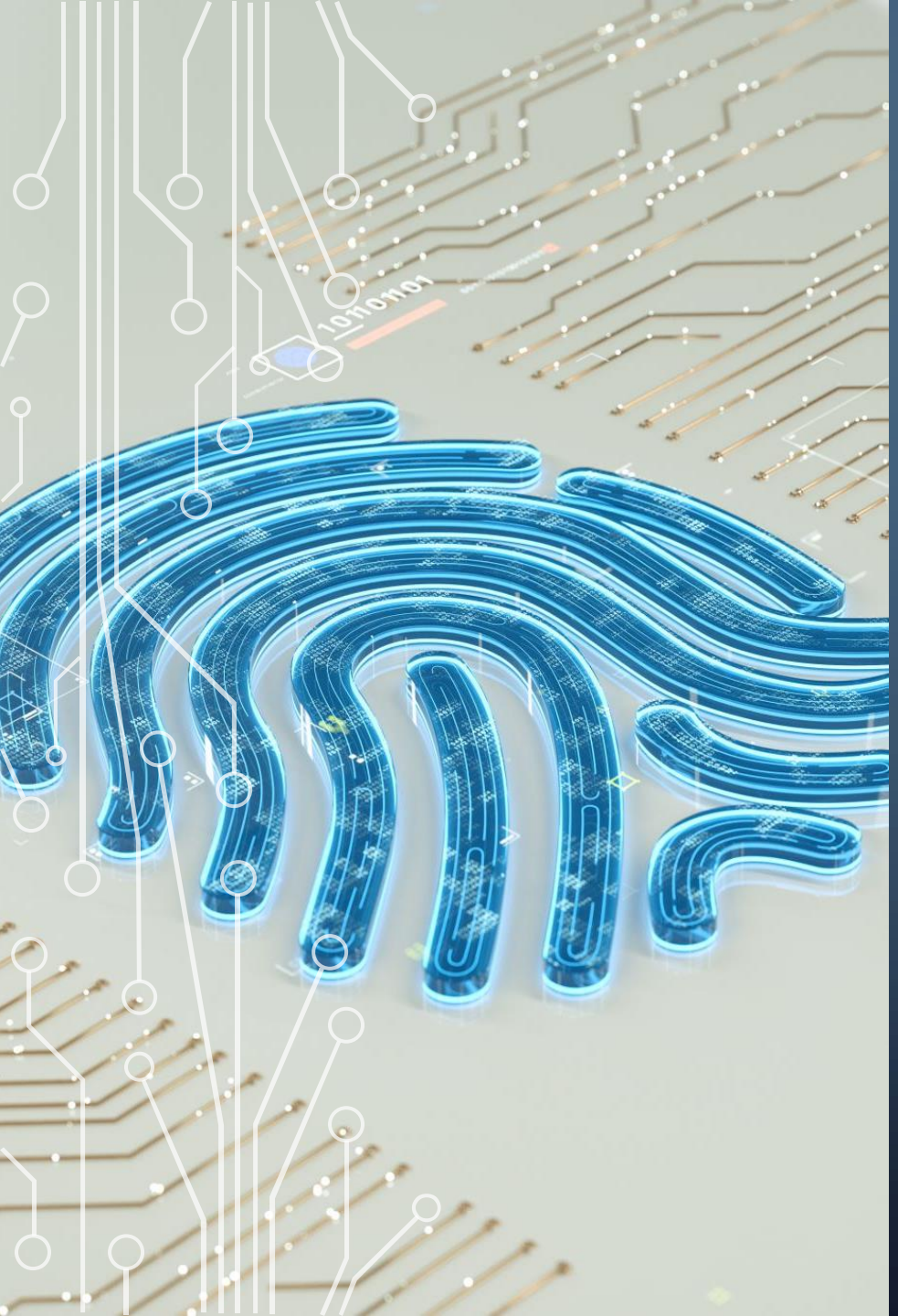
DIFERENCIAS ENTRE AUTENTICACIÓN Y AUTORIZACIÓN

- Autenticación
  - Definición: La autenticación es el proceso mediante el cual se verifica la identidad de un usuario o sistema que intenta acceder a un recurso o servicio. En otras palabras, es el mecanismo que confirma que el usuario es quien dice ser.
  - Propósito: La autenticación responde a la pregunta "¿Quién eres?". Su propósito principal es garantizar que los usuarios son quienes afirman ser antes de permitirles acceder a sistemas, aplicaciones o datos.



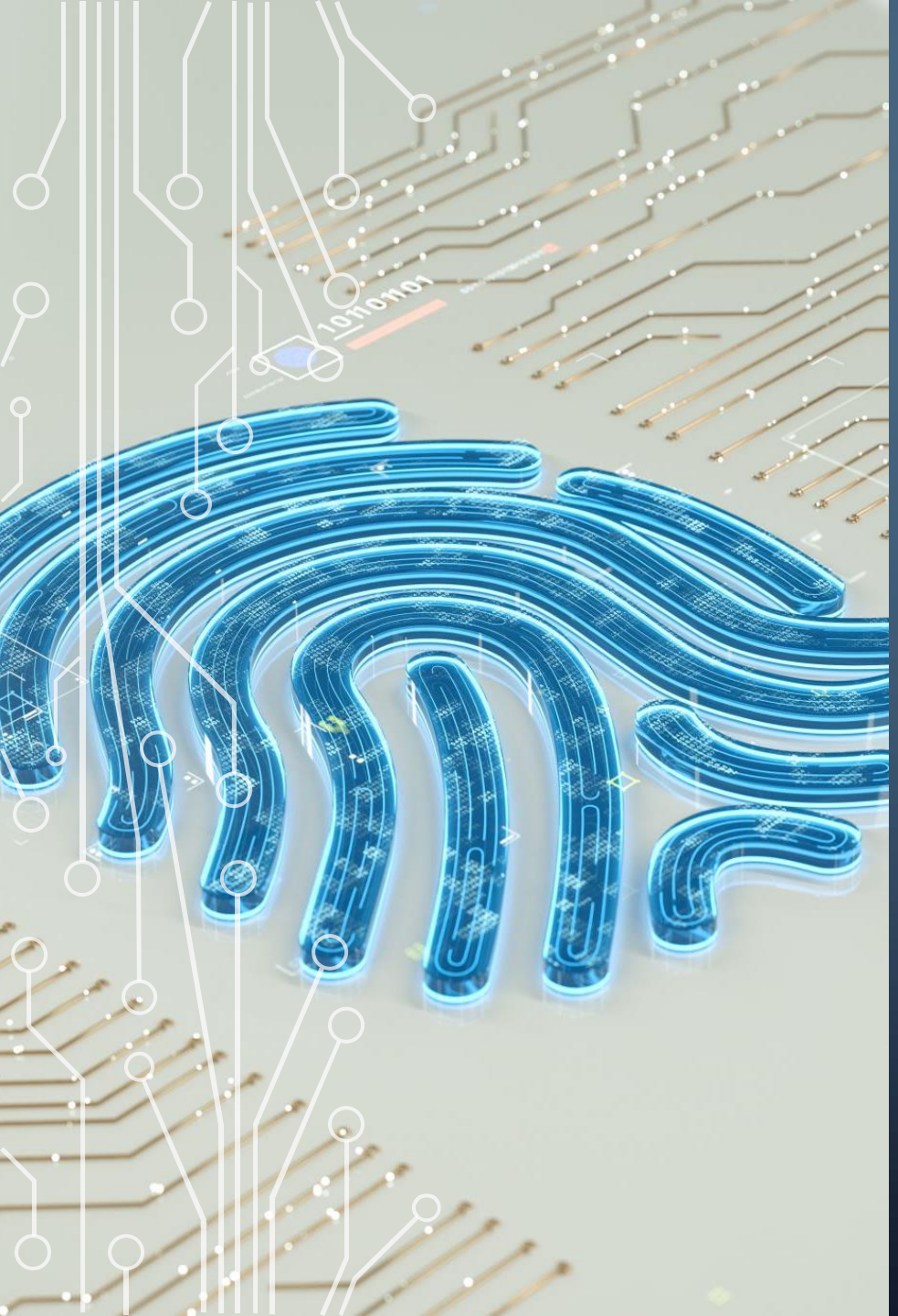
DIFERENCIAS ENTRE AUTENTICACIÓN Y AUTORIZACIÓN

- Métodos de Autenticación:
  - Usuario/Contraseña: Es el método más común y tradicional. Los usuarios se autentican proporcionando un nombre de usuario y una contraseña. Sin embargo, este método puede ser vulnerable a ataques de fuerza bruta, phishing y otros tipos de ciberataques si no se implementan adecuadamente políticas de seguridad robustas, como la complejidad de contraseñas y la limitación de intentos de inicio de sesión.
  - Tokens: Los tokens, como los JSON Web Tokens (JWT), se utilizan ampliamente en aplicaciones modernas, especialmente en APIs. Un token es un string generado y firmado por el servidor que contiene información sobre la identidad del usuario y sus permisos. Los tokens son portátiles y pueden ser utilizados por el cliente para autenticarse en múltiples solicitudes sin necesidad de reenviar las credenciales de usuario.
  - Certificados: Los certificados digitales utilizan criptografía de clave pública para autenticar usuarios o dispositivos. Este método es más seguro que las contraseñas, ya que un certificado es difícil de falsificar y puede ser revocado si se detecta un uso indebido.



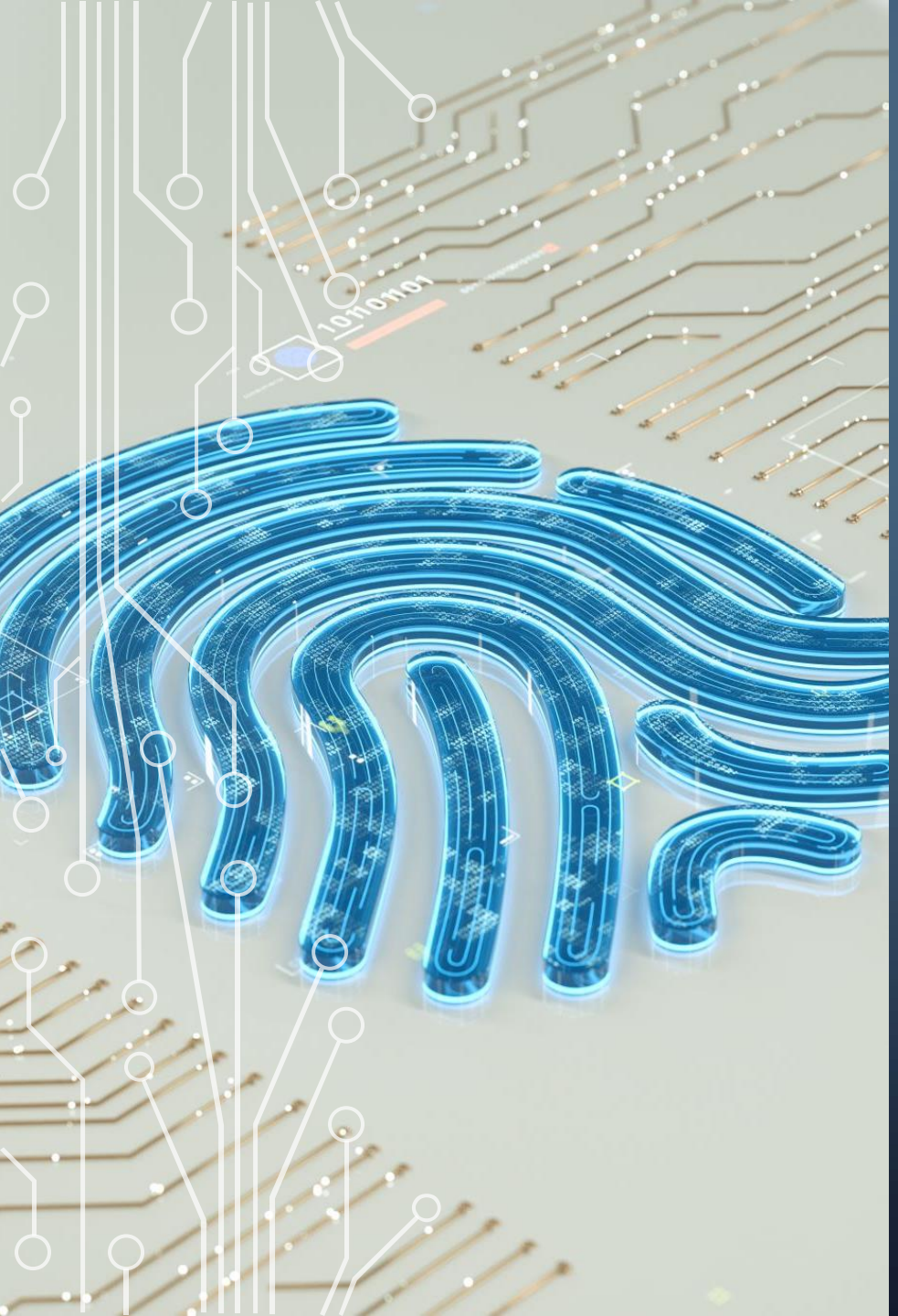
DIFERENCIAS ENTRE AUTENTICACIÓN Y AUTORIZACIÓN

- Proceso de Autenticación:
  - El usuario envía sus credenciales (usuario/contraseña, token, certificado) al servidor,
  - El servidor verifica las credenciales comparándolas con las almacenadas o validándolas mediante un tercero de confianza.
  - Si las credenciales son válidas, el usuario es autenticado y se le concede acceso temporal mediante una sesión o un token.



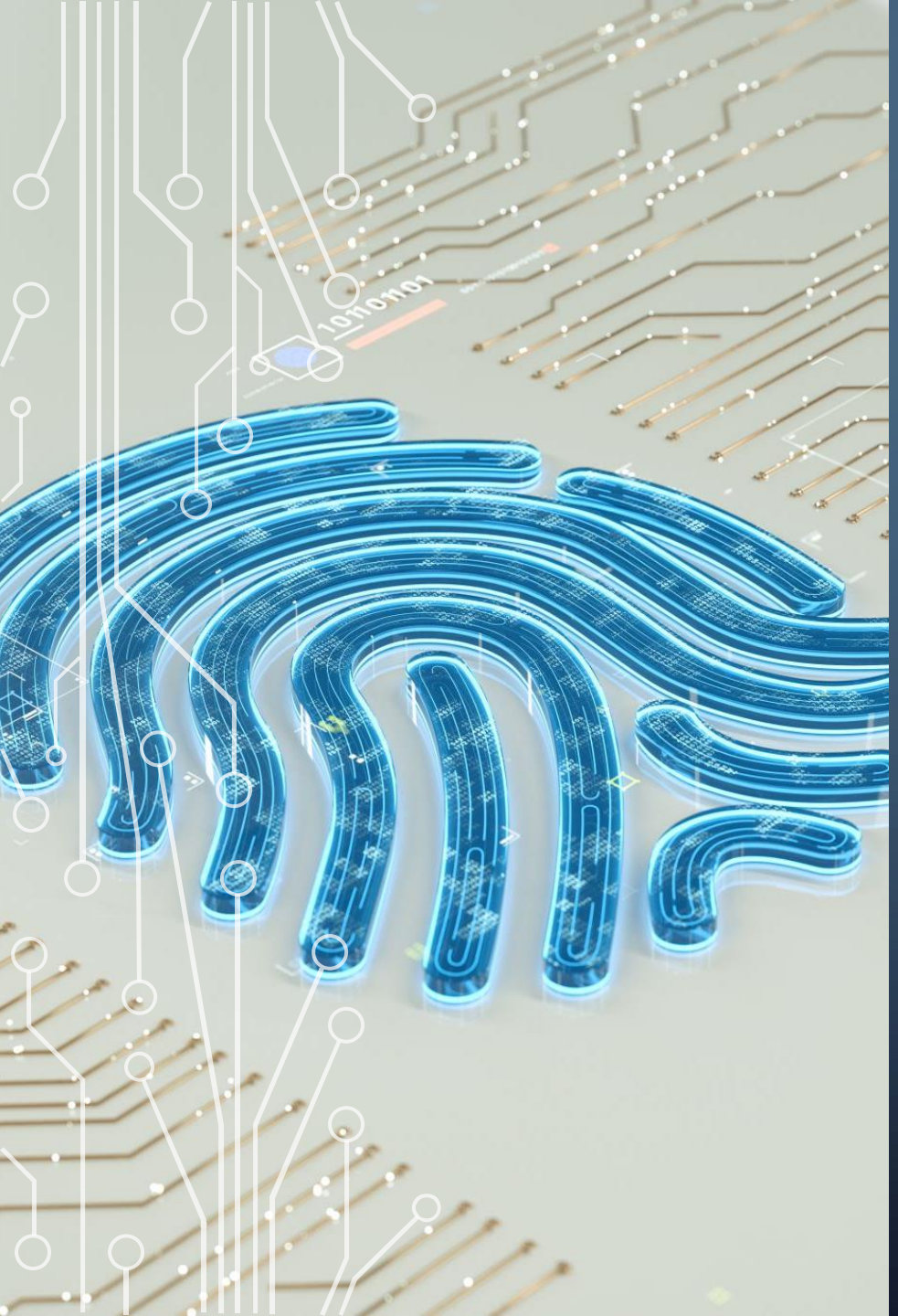
DIFERENCIAS ENTRE AUTENTICACIÓN Y AUTORIZACIÓN

- Autorización
  - Definición: La autorización es el proceso de determinar qué acciones puede realizar un usuario autenticado y a qué recursos puede acceder. Es un paso posterior a la autenticación.
  - Propósito: La autorización responde a la pregunta "¿Qué puedes hacer?". Su propósito es asegurar que los usuarios solo puedan realizar acciones y acceder a datos para los cuales tienen permisos explícitos.



DIFERENCIAS ENTRE AUTENTICACIÓN Y AUTORIZACIÓN

- Métodos de Autorización:
  1. Roles: En el control de acceso basado en roles (Role-Based Access Control, RBAC), los permisos se asignan a roles específicos. Los usuarios se asignan a estos roles, heredando sus permisos. Por ejemplo, un rol "Administrador" podría tener acceso completo al sistema, mientras que un rol "Usuario" solo tiene permisos limitados.
  2. Políticas: El control de acceso basado en políticas (Policy-Based Access Control, PBAC) utiliza reglas y condiciones específicas para otorgar permisos. Las políticas pueden ser más granulares que los roles, permitiendo decisiones de autorización basadas en atributos como la ubicación del usuario, la hora del día, o el tipo de dispositivo utilizado.
  3. Claims: Las claims son declaraciones sobre el usuario (como su nombre, rol, edad, etc.) que se utilizan para tomar decisiones de autorización. Por ejemplo, una aplicación podría usar una claim que indique que el usuario es mayor de edad para permitir el acceso a contenido restringido.



DIFERENCIAS ENTRE AUTENTICACIÓN Y AUTORIZACIÓN

- Proceso de Autorización:
  - Una vez que el usuario está autenticado, el sistema consulta las reglas de autorización definidas para determinar sus permisos.
  - Basado en el rol del usuario, las políticas aplicables, y las claims proporcionadas, el sistema decide qué acciones puede realizar el usuario y a qué recursos puede acceder.
  - Si el usuario intenta realizar una acción para la cual no tiene permiso, el acceso es denegado.

MÉTODOS COMUNES DE AUTENTICACIÓN

- Usuario y Contraseña
  - Descripción:
    - El método de usuario y contraseña es el sistema de autenticación más tradicional y comúnmente utilizado. Consiste en que un usuario proporciona un nombre de usuario y una contraseña para acceder a un sistema o aplicación.
  - Funcionamiento:
    - El usuario introduce su nombre de usuario y contraseña en el formulario de inicio de sesión.
    - El sistema verifica la contraseña comparándola con la almacenada en su base de datos.
    - Si coinciden, el usuario es autenticado.

MÉTODOS COMUNES DE AUTENTICACIÓN

- Usuario y Contraseña
  - Ventajas:
    - Simplicidad y familiaridad: La mayoría de los usuarios están acostumbrados a este método.
    - Fácil de implementar: No requiere infraestructura compleja.
  - Desventajas:
    - Vulnerabilidad a ataques de fuerza bruta: Los atacantes pueden intentar adivinar la contraseña mediante múltiples intentos.
    - Susceptible a ataques de phishing: Los usuarios pueden ser engañados para revelar sus credenciales.
    - Problemas de gestión de contraseñas: Las contraseñas pueden ser débiles o reutilizadas en múltiples sitios, lo que aumenta el riesgo.

MÉTODOS COMUNES DE AUTENTICACIÓN

- Usuario y Contraseña
  - Medidas de Mitigación:
    - Uso de contraseñas fuertes y políticas de cambio regular.
    - Implementación de autenticación multifactor (MFA) para agregar una capa adicional de seguridad.
    - Uso de técnicas de hash y sal para almacenar contraseñas de manera segura.

MÉTODOS COMUNES DE AUTENTICACIÓN

- Tokens de Sesión
  - Descripción:
    - Los tokens de sesión, especialmente los JSON Web Tokens (JWT), se utilizan ampliamente en aplicaciones modernas, especialmente en aplicaciones de una sola página (SPA) y aplicaciones móviles. Los tokens son piezas de información que se emiten al usuario después de una autenticación exitosa y se utilizan para autenticar solicitudes posteriores.
  - Funcionamiento:
    - Después de que el usuario se autentica con éxito (por ejemplo, mediante usuario y contraseña), el servidor emite un token firmado (JWT).
    - El cliente almacena este token (usualmente en localStorage o sessionStorage en el navegador).
    - En cada solicitud posterior, el cliente envía el token al servidor en el encabezado de la autorización.
    - El servidor verifica la firma del token para autenticarse sin necesidad de volver a verificar las credenciales del usuario.

MÉTODOS COMUNES DE AUTENTICACIÓN

- Tokens de Sesión
  - Ventajas:
    - Stateless: El servidor no necesita mantener el estado de la sesión, lo que mejora la escalabilidad.
    - Portabilidad: Los tokens pueden ser utilizados en diferentes dominios y servicios.
    - Seguridad: Los tokens pueden contener claims que agregan información útil sobre el usuario.
  - Desventajas:
    - Almacenamiento seguro: El almacenamiento incorrecto de tokens en el cliente puede llevar a vulnerabilidades.
    - Revocación: Dificultad para revocar tokens antes de que expiren.

MÉTODOS COMUNES DE AUTENTICACIÓN

- Tokens de Sesión
  - Medidas de Mitigación:
    - Almacenar tokens en lugares seguros (evitar localStorage si es posible).
    - Implementar técnicas de expiración y renovación de tokens.
    - Utilizar HTTPS para proteger la transmisión de tokens.

MÉTODOS COMUNES DE AUTENTICACIÓN

- OAuth2 y OpenID Connect
  - Descripción:
    - OAuth2 es un marco de autorización que permite a las aplicaciones obtener acceso limitado a los recursos de un usuario en un servidor sin exponer las credenciales del usuario. OpenID Connect es una capa de autenticación que se construye sobre OAuth2 para proporcionar autenticación federada.
  - Funcionamiento:
    - OAuth2: Permite a una aplicación (cliente) obtener acceso limitado a los recursos del usuario en un servidor (servidor de recursos) mediante un intermediario de autorización (servidor de autorización). Los usuarios pueden autorizar a la aplicación a acceder a sus datos sin compartir sus credenciales.
    - OpenID Connect: Extiende OAuth2 para autenticar usuarios y obtener información sobre ellos (claims). Utiliza el flujo de autorización de OAuth2 para proporcionar autenticación.

MÉTODOS COMUNES DE AUTENTICACIÓN

- OAuth2 y OpenID Connect
  - Ventajas:
    - Seguridad: Los usuarios no necesitan compartir sus credenciales con aplicaciones de terceros.
    - Federated Identity: Permite a los usuarios autenticarse utilizando sus cuentas de Google, Facebook, etc.
    - Versatilidad: Soporta varios flujos de autorización para diferentes casos de uso.
  - Desventajas:
    - Complejidad: Implementar y configurar OAuth2 y OpenID Connect puede ser complejo.
    - Dependencia de terceros: Las aplicaciones dependen de proveedores de identidad externos.

MÉTODOS COMUNES DE AUTENTICACIÓN

- OAuth2 y OpenID Connect
  - Medidas de Mitigación:
    - Usar bibliotecas y servicios probados para implementar OAuth2 y OpenID Connect.
    - Realizar auditorías de seguridad regulares y monitoreo de actividades.

TOKEN-BASED AUTHENTICATION

- Descripción:
  - La autenticación basada en tokens es una técnica popular para manejar la autenticación en aplicaciones web y móviles modernas, especialmente en el contexto de APIs backend. En este esquema, se utiliza un token como medio para autenticar solicitudes posteriores, eliminando la necesidad de mantener el estado de la sesión en el servidor.
- Es el método elegido de este curso para ser implementado en ASP.NET Core.

TOKEN-BASED AUTHENTICATION

- JWT: Autocontenido y Portátil
  - JSON Web Tokens (JWT): Un JWT es un token compactado y autocontenido que puede ser utilizado para autenticar y autorizar usuarios en aplicaciones. Se compone de tres partes: el encabezado, el payload (carga útil) y la firma.

TOKEN-BASED AUTHENTICATION

- Estructura de un JWT:
    1. Encabezado (Header): El encabezado típicamente incluye dos partes: el tipo de token (JWT) y el algoritmo de firma (por ejemplo, HMAC SHA256).

Ejemplo: {"alg": "HS256", "typ": "JWT"}

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

TOKEN-BASED AUTHENTICATION

- Estructura de un JWT:
  2. Carga útil (Payload): La carga útil contiene las claims, que son declaraciones sobre una entidad (usualmente el usuario) y datos adicionales. Existen tres tipos de claims:
     - Registered claims: Claims estándar como iss (emisor), exp (expiración), sub (sujeto) y aud (audiencia).
     - Public claims: Claims definidas libremente por la comunidad y no estandarizadas.
     - Private claims: Claims personalizadas que se utilizan para compartir información entre partes que acuerdan su uso.

Ejemplo: {"sub": "1234567890", "name": "John Doe", "admin": true}

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

TOKEN-BASED AUTHENTICATION

- Estructura de un JWT:
  3. Firma (Signature): La firma se crea tomando el encabezado codificado en base64, la carga útil codificada en base64, un secreto y el algoritmo especificado en el encabezado. Se utiliza para verificar que el mensaje no haya sido alterado.

Ejemplo: HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)
```

IMPLEMENTACIÓN EN ASP.NET CORE

- Configuración de JWT:
  - Primero, se deben instalar los paquetes necesarios:

```
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer
```

IMPLEMENTACIÓN EN ASP.NET CORE

- Configuración de JWT:
  - Luego, en el archivo **Program.cs** o **Startup.cs**, se configura el servicio de autenticación JWT:

```
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = "your-issuer",
        ValidAudience = "your-audience",
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("your-secret-
key"))
    };
});
```

IMPLEMENTACIÓN EN ASP.NET CORE

- Generación de Tokens:
  - Para generar un JWT, se puede crear un método en un controlador que emita tokens:

```
[HttpPost("login")]
public IActionResult Login([FromBody] UserLoginDto userLogin)
{
    // Validar las credenciales del usuario (omitido por simplicidad)
    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, userLogin.Username),
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
    };

    var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("your-secret-key"));
    var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

    var token = new JwtSecurityToken(
        issuer: "your-issuer",
        audience: "your-audience",
        claims: claims,
        expires: DateTime.Now.AddMinutes(30),
        signingCredentials: creds);

    return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });
}
```

IMPLEMENTACIÓN EN ASP.NET CORE

- Protección de Rutas:
  - Para proteger rutas específicas en la aplicación, se pueden utilizar atributos de autorización:

```
[Authorize]
[HttpGet("protected")]
public IActionResult GetProtected()
{
    return Ok("This is a protected endpoint.");
}
```

MATERIAL DE ESTUDIO

- Vídeos tutoriales:
  - ¿Qué es Autenticación? ¿Qué es Autorización? - Te lo explico en 15 minutos y NO TE LO OLVIDAS MÁS! (youtube.com)
  - JWT en 10 minutos - ¿Qué es JWT? ¿Para qué sirve? ¿Cuándo usarlo? ¿Cómo se usa? (youtube.com)
  - JWT para Programar Backend Seguro (youtube.com)
- Documentación oficial:
  - Información general sobre la autenticación de ASP.NET Core | Microsoft Learn

iTSCHOOL LA POSTA!