 La autenticación verifica la identidad del usuario.

 La autorización determina los privilegios y acceso del usuario una vez que ha sido autenticado.

**Elementos Role based Authorization**

* **Roles:** Un rol es un grupo que representa un conjunto de permisos o responsabilidades. Por ejemplo, podrías tener roles como "Administrador", "Usuario", "Editor", etc.
* **Claims:** Son piezas de información sobre el usuario, como su nombre, correo electrónico, roles, etc. En ASP.NET Core, los roles son un tipo de claim (ClaimTypes.Role).
* **Policy:** Es una forma de agrupar requisitos de autorización bajo un nombre específico. Puedes definir políticas que requieran ciertos roles o claims para acceder a una parte de tu aplicación.
* **Authorization:** Es el proceso de determinar si un usuario tiene acceso a un recurso o acción dentro de la aplicación. ASP.NET Core proporciona varias formas de configurar la autorización, incluyendo atributos en los controladores y métodos, así como configuraciones en el archivo Startup.

**Tipos de Autenticación**

1. **Autenticación basada en cookies:**

Características:

* Almacena la información de autenticación en una cookie en el navegador del cliente.
* Es stateful (mantiene el estado en el servidor).
* Ideal para aplicaciones web tradicionales.

Ventajas:

* Fácil de implementar y entender.
* Buena para aplicaciones que se ejecutan en un solo dominio.

Desventajas:

* No es ideal para APIs o aplicaciones móviles.
* Vulnerable a ataques CSRF si no se implementan medidas de seguridad adicionales.

1. **Autenticación basada en tokens (JWT):**

Características:

* Utiliza tokens JWT (JSON Web Tokens) para autenticar solicitudes.
* Es stateless (no mantiene estado en el servidor).
* Ideal para APIs RESTful y aplicaciones de una sola página (SPA).

Ventajas:

* Escalable y eficiente.
* Funciona bien en arquitecturas distribuidas.

Desventajas:

* Los tokens pueden ser más grandes que las cookies.
* La revocación de tokens puede ser más compleja.

1. **Autenticación basada en claims:**

Características:

* Utiliza declaraciones (claims) para representar atributos de identidad.
* Flexible y extensible.
* Se puede usar con cookies o tokens.

Ventajas:

* Permite una autorización más granular y detallada.
* Facilita la implementación de roles y permisos personalizados.

Desventajas:

* Puede ser más complejo de configurar inicialmente.

1. **Autenticación con proveedores externos (OAuth):**

Características clave:

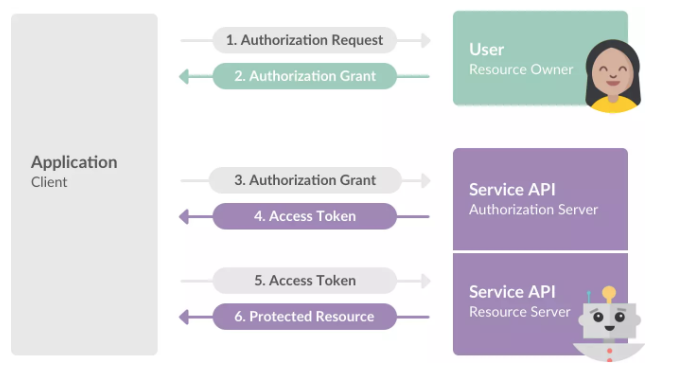
* Permite a los usuarios autenticarse usando cuentas de servicios externos (Google, Facebook, etc.).
* Implementa protocolos estándar como OAuth 2.0

Ventajas:

* Mejora la experiencia del usuario al no requerir nuevas credenciales.
* Delega la responsabilidad de la gestión de contraseñas a proveedores confiables.

Desventajas:

* Depende de servicios externos.
* Puede requerir más configuración y manejo de casos de error.



| **Característica** | **Stateful** | **Stateless** |
| --- | --- | --- |
| Mantenimiento del estado | Servidor | Cliente |
| Implementación | Cookies de sesión, tokens de sesión | JWT |
| Ventajas | Simple, personalizable, seguimiento del estado | Escalable, seguro, sin dependencia del servidor |
| Desventajas | Difícil de escalar, menos seguro, dependencia del servidor | Más complejo, menos personalizable, carga en el cliente |
| Escenarios de uso | Experiencia de usuario personalizada, seguimiento del estado de la sesión | Alta escalabilidad, seguridad, disponibilidad |

Nace para poder satisfacer la necesidad de seguridad de aplicaciones distribuidas. OAuth2 es un protocolo de autorización, es decir que brinda acceso a un recurso. Esto es como si yo tuviera una entrada para asistir a un show en un teatro: a la persona que está en la puerta, lo único que le importa es que yo tenga el ticket para dejarme pasar. Esto es lo que hace OAuth2: no le importa quién soy sino que tenga el “ticket” (token) para darme acceso a un recurso. Provee una delegación de acceso seguro en nombre de un resource owner (dueño del recurso).

**OAuth define cuatro roles:**

"propietario de los recursos":

Una entidad capaz de otorgar acceso a un recurso protegido. Cuando el propietario del recurso es una persona, se lo conoce como usuario final. Esto a veces se abrevia como "RO".

"servidor de recursos":

El servidor que aloja los recursos protegidos, capaz de aceptar y responder a solicitudes de recursos protegidos utilizando tokens de acceso. Esto a veces se abrevia como "RS".

"cliente":

Una aplicación que realiza solicitudes de recursos protegidos en nombre de propietario de los recursos y con su autorización. El término "cliente" sí no implica ninguna característica de implementación particular (por ejemplo., si la aplicación se ejecuta en un servidor, un escritorio u otro dispositivos).

"servidor de autorización":

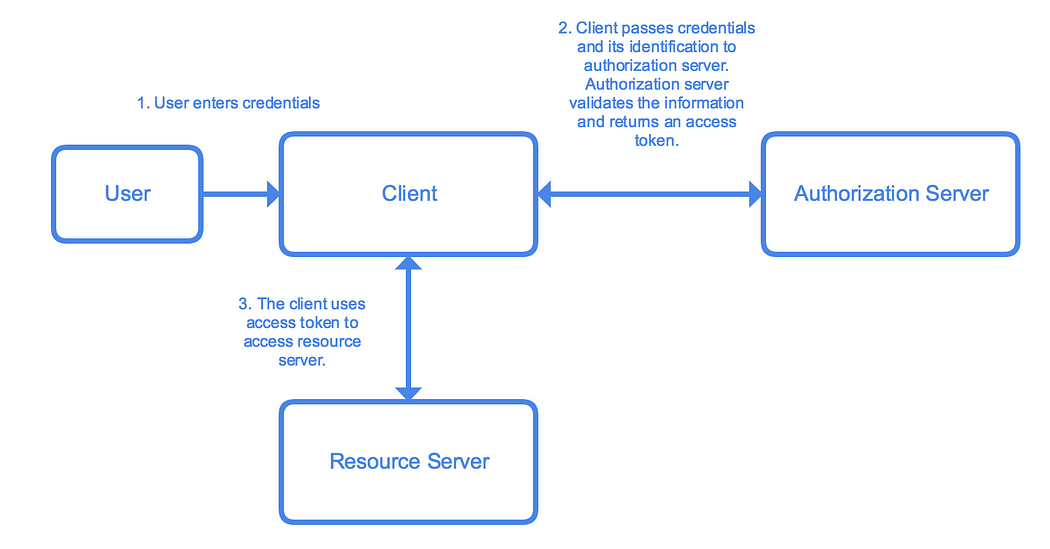
El servidor emite tokens de acceso al cliente después de haber tenido éxito autenticar al propietario del recurso y obtener la autorización. Esto a veces se abrevia como "AS".

La interacción entre el servidor de autorización y el servidor de recursos está más allá del alcance de esta especificación, sin embargo, varias extensiones tienen se ha definido para proporcionar una opción de interoperabilidad entre recursos servidores y servidores de autorización. El servidor de autorización puede ser el mismo servidor que el servidor de recursos o una entidad separada. Un único servidor de autorización puede emitir tokens de acceso aceptados por múltiples servidores de recursos.

Server Recurso 1

Server Recurso 2

Servidor de Autorización



Flujos de Autorizacion

Documentacion OAuth :

<https://auth0.com/es/intro-to-iam/what-is-oauth-2>

<https://blog.postman.com/pkce-oauth-how-to/>

<https://www.linkedin.com/advice/0/what-trade-offs-between-implicit-grant-flow-authorization?lang=es&originalSubdomain=es#:~:text=El%20flujo%20de%20concesi%C3%B3n%20impl%C3%ADcito%20es%20una%20forma%20simplificada%20de,no%20tienen%20un%20servidor%20backend.>

<https://arquitecturaibm.com/todos-los-flujos-de-oauth-2-0/>

<https://www.ibm.com/docs/es/datapower-gateway/10.5.0?topic=support-oauth-flows>