educación IT
CODE YOUR FUTURE

Autenticación

Autenticación

La autenticación es el proceso de verificar la identidad de un usuario o sistema.

En el contexto de API RESTful, hay varias formas de manejar la autenticación.

Veamos cuáles son en el cuadro de la próxima pantalla.

Autenticación básica (*basic Authentication*)	***Tokens* de autenticación.**	**API *Keys***
• Consiste en enviar las credenciales (usuario y contraseña) codificadas en Base64 dentro del encabezado HTTP. • Es un método simple, pero no seguro si no se utiliza sobre HTTPS, ya que las credenciales pueden ser interceptadas.	• Se utilizan para autenticar solicitudes después de que el usuario se haya autenticado inicialmente. • Ejemplos comunes incluyen JSON Web Tokens (JWT), que contienen información del usuario y caducidad, y son firmados para evitar manipulaciones. • Los *tokens* suelen enviarse en el encabezado `Authorization` como `Bearer Token`.	• Son claves únicas generadas por el servidor que los clientes deben incluir en cada solicitud para autenticarse. • Se envían generalmente como un parámetro en la URL o en los encabezados HTTP.

Método básico de autenticación

Ese método corresponde a la Autenticación Básica (`Basic Authentication`). Es una forma simple de autenticación en la que las credenciales (usuario y contraseña) se codifican en `Base64` y se envían en el encabezado HTTP, generalmente en este formato:

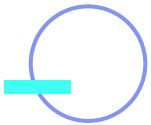
```
Authorization: Basic
[credenciales_codificadas_en_Base64]
```

Ventajas

- Simple de implementar.

Desventajas

- Menor seguridad, especialmente si no se usa HTTPS. Las credenciales se envían con cada solicitud. Esto aumenta el riesgo de exposición.



Utilización de *tokens*

JSON Web Token es un estándar abierto, basado en JSON, para la creación de *tokens* de acceso que permiten la propagación de identidad y privilegios. Por ejemplo, un servidor podría generar un *token* que indique que el usuario tiene privilegios de administrador y proporcionarlo al cliente.

Se trata de un *token* firmado que contiene información del usuario y una fecha de expiración. Generalmente, lo podemos encontrar en este formato:

```
header.payload.signature
```

Ventajas

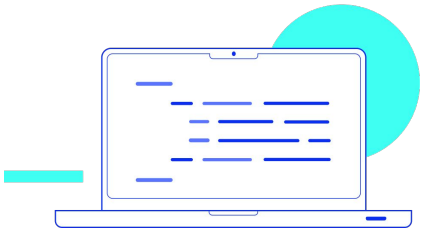
- Es *stateless*, lo que significa que el servidor no mantiene información sobre el estado entre solicitudes, reduciendo así su carga. Además, es fácil de manejar en aplicaciones modernas, facilita la escalabilidad y la eficiencia.

Desventajas

- Si un *token* es robado, se puede usar hasta que expire.

Estándar actual para autorización de acceso

OAuth 2.0 es un estándar para *tokens* de acceso. Es ampliamente utilizado en la actualidad. Los usuarios autorizan aplicaciones de terceros a acceder a sus recursos sin compartir sus credenciales.



Ventajas

- Muy seguro y flexible. Utilizado por grandes plataformas como Google, Facebook, y otros.

Desventajas

- Complejo de implementar.

API *keys*

Una API Key es una cadena única de caracteres generada por un servidor que sirve para identificar a la aplicación o al usuario que hace una solicitud a una API.

Las claves de API se utilizan para autenticar y autorizar a los clientes a interactuar con los servicios de la API, asegurando que las solicitudes provienen de una fuente confiable y que se cuenta con los permisos adecuados.

Ventajas

- Simple de implementar y utilizar.

Desventajas

- Las claves pueden ser robadas y usadas indebidamente. Generalmente menos seguro que otros métodos.

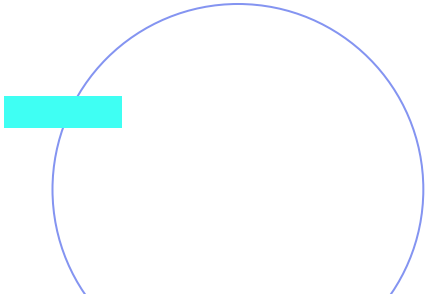
Autorización

Autorización

La autorización es el proceso de determinar si un usuario o sistema tiene permiso para realizar una acción específica en un recurso o conjunto de recursos dentro de una aplicación o sistema.

Después de que un usuario ha sido autenticado (es decir, su identidad ha sido verificada), el siguiente paso es decidir qué puede o no hacer ese usuario, es decir, qué recursos puede acceder y qué operaciones puede ejecutar sobre esos recursos.

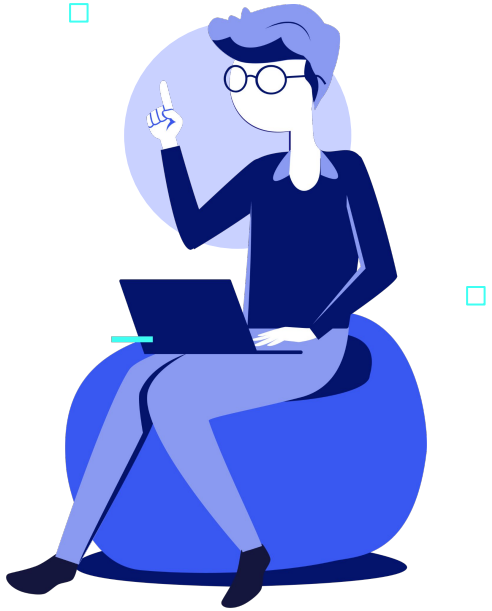
En el contexto de una API RESTful, **la autorización es fundamental para garantizar que los usuarios solo puedan acceder a los recursos o realizar las acciones que tienen permitido según su rol, contexto o atributos.**



Verificación de Identidad y control de acceso

La autenticación garantiza que un usuario es quien dice ser, y en las API RESTful se pueden emplear diversos métodos para lograrlo, como la autenticación básica, los tokens (JWT, OAuth 2.0) y las API Keys.

La autorización asegura que un usuario tiene los permisos necesarios para realizar acciones específicas. Las opciones incluyen roles y permisos, OAuth 2.0 con scopes, ACL y ABAC.



Diferencia clave entre autenticación y autorización

Autenticación	Autorización
Verifica quién es el usuario (por ejemplo, con una contraseña o un token).	Determina qué puede hacer ese usuario una vez que ha sido autenticado (por ejemplo, leer un recurso, escribir datos).

Técnicas de autorización

- **Roles y permisos:** a los usuarios se les asignan roles con permisos específicos. Por ejemplo, un rol de "admin" puede tener permisos para crear, leer, actualizar y eliminar recursos.
- **OAuth 2.0:** utiliza scopes para definir lo que puede hacer un token de acceso. Los scopes limitan el acceso de la aplicación a ciertas partes de la API.
- **ACL (*Access Control Lists*):** las listas de control de acceso definen permisos específicos para cada usuario o grupo de usuarios en un recurso.
- **ABAC (*Attribute-Based Access Control*):** las decisiones de acceso se basan en atributos del usuario, del recurso, y del entorno. Por ejemplo, un usuario puede tener acceso a recursos solo durante ciertas horas del día.