

Autenticación

dev.*f*
desarrollamos(personas);

dev

Basic Authentication

- Sistema simple de autenticación para el protocolo HTTP
- El Cliente envía una petición HTTP con la cabecera “Authorization” y esta contiene la palabra “**Basic**” seguida por un espacio y en un string “**username:password**” en formato de **base64**
- Por Ejemplo, Para autorizar a “**demo**” **demo:p@55w0rd** el cliente debe mandar la siguiente cabecera: **Authorization: Basic ZGVtbzpwQDU1dzByZA==**
- base64 es fácil de decodificar, Basic authentication debería usarse junto con otros protocolos de seguridad como **HTTPS/SSL**.

OAUTH2

- Es un framework de autorización que le permite a las aplicaciones obtener acceso limitado a cuentas de usuario en un servicio HTTP (Facebook, Google, Github, etc)
- Delega la autenticación del usuario al servicio y autoriza a aplicaciones de terceros el acceso a dicha cuenta de usuario.
- Cuenta con 4 Roles:
 - a. Propietario del recurso: ***Es el "usuario" que da la autorización a una aplicación, para acceder a su cuenta.***
 - b. Cliente: ***Es la aplicación que desea acceder a la cuenta del usuario.***
 - c. Servidor de recursos: ***Servidor que autentifica al cliente y regresa token de acceso***
 - d. Servidor de autorización: ***Servidor que regresa los recursos solicitados al cliente***

OAuth2

Flujo de protocolo abstracto



JWT

- Estándar abierto ([RFC-7519](#)) basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros.
- **JWT** es muy usado para manejar la autenticación en aplicaciones móviles o web.
- Se compone de tres partes **header, payload y signature** todas las partes se concatenan con un **punto** quedando de la siguiente manera:

```
header.payload.signature
```

- Ejemplo de un JWT:
`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjEiLCJ1c2VybmFtZSI6InNlcmdpb2R4YSJ9.Qu7rv5wqk6zGjiMU8ZixwvKQGBNW9hhj55DbSP50b1g`

Header

Payload

Signature

JWT - Header

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

Json con Metadata del JWT el cual especifica el tipo de token y algoritmo de encriptación

- Tipo de token (typ) - Identifica el tipo de token. (siempre debe ser JWT)
- Algoritmo de firmado (alg) - Indica que tipo de algoritmo fue usado para firmar el token.

JWT - Payload

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

Json que puede llevar cualquier tipo de dato, por lo regular se ocupa para mandar datos de la sesión como: email, id, username. Además aquí se ponen otras reglas como la expiración del token.

Algunas reglas son :

- Creador (iss) - Identifica a quien creo el JWT
- Razón (sub) - Identifica la razón del JWT, se puede usar para limitar su uso a ciertos casos.
- Tiempo de expiración (exp) - Una fecha que sirva para verificar si el JWT está vencido y obligar al usuario a volver a autenticarse.
- No antes (nbf) - Indica desde qué momento se va a empezar a aceptar un JWT.
- Creado (iat) - Indica cuando fue creado el JWT.

JWT - Signature

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
) ☐ secret base64 encoded
```

La firma del JWT se genera usando el header y el payload en base64 y una key secreta(Solo la debe saber el servidor que la creó)

Otras Forma de Autenticación

API Keys:

Se generan dos diferentes keys una privada y una pública, ambas son dependientes y por lo regular la llave pública se usa en el frontend y la llave privada en el backend, estas llaves son asociadas a un solo usuario de manera permanente.

API Token:

Se genera un token de caracteres aleatorios y es asociado de manera permanente a un usuario. Es un token simple, solo sirve para identificar el usuario que hace uso de la API.

HMAC:

Conocido como *“hash-based message authentication code”* se envía una versión en hash del password y otros elementos de autenticación al servidor, este recibe varias parte y recompone el password y así dar acceso al usuario.