OAuth2: Grant Types

Implementa la seguridad de tu API Rest con Spring Boot

Tipos de clientes

- **Clientes confidenciales**: son aquellos capaces de guardar una contraseña sin que esta sea accesible o expuesta (aplicaciones nativas, otra api, ...)
- **Clientes públicos**: son aquellos que no son capaces de guardar una contraseña y mantenerla a salvo (aplicaciones Javascript, Angular, ...)
- En función del tipo de cliente, necesitaremos implementar el flujo de OAuth2 de diferentes formas concretas.

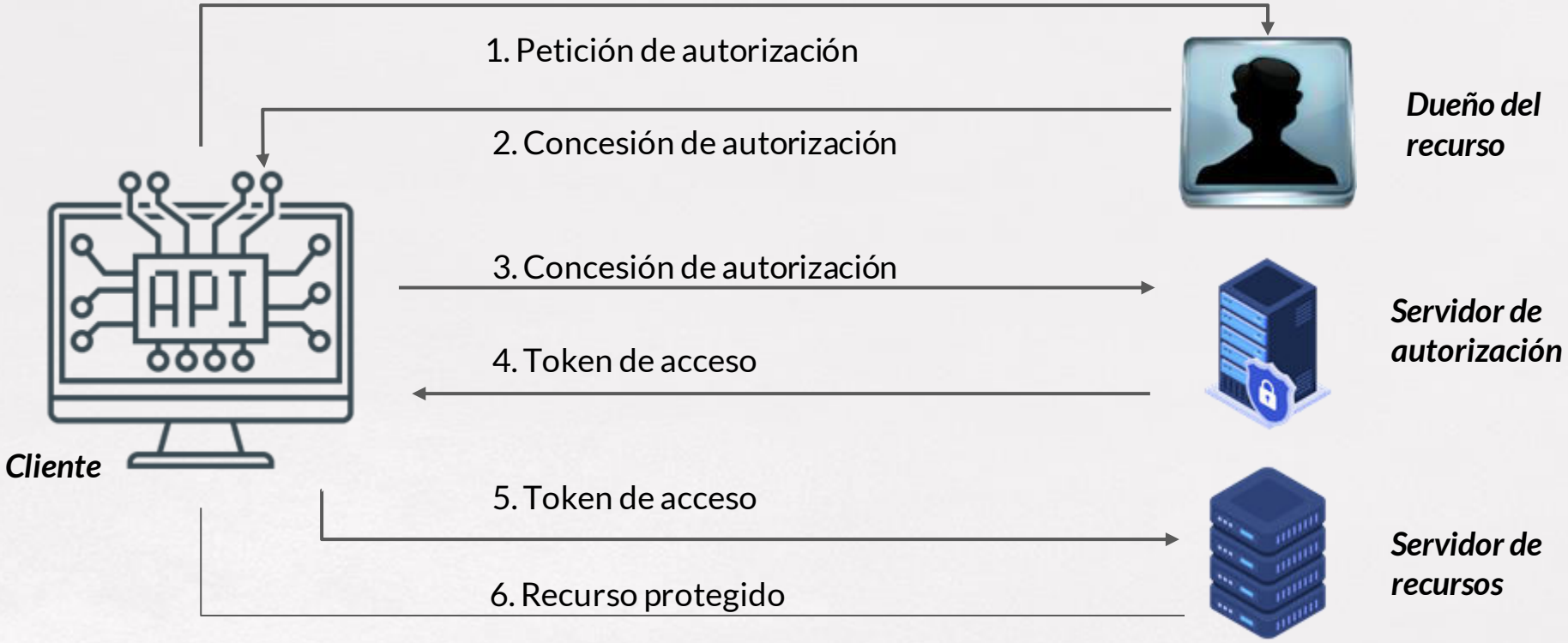
Grant Types o tipos de otorgamiento

- Diferentes formas de obtener el token.
- Surgen a causa de los diferentes tipos de clientes que pueden querer acceder a una serie de recursos.
    - Una aplicación móvil nativa.
    - Una aplicación web con Angular.
    - Una TV con una aplicación en la plataforma X.
    - Un dispositivo IoT como una bombilla inteligente.

Grant Types

- Authorization Code
- Implicit
- Resource Owner Password Credentials
- Client Credentials Flow
- Device Code Flow
- ...

Authorization Code



Authorization Code

- Es el más completo de todos.
- Se utiliza con clientes confidenciales (que son capaces de guardar la contraseña convenientemente).
- Veamos los pasos que se siguen

Authorization Code

- El cliente redirige al usuario al endpoint de autorización, con una serie de parámetros

  *https://autorizacion.servidor.com/authorize?**response_type**=code&**client_id**=the-client-id&**state**=xyz&**redirect_uri**=https://cliente.ejemplo.com/cb&**scope**=api_read*

  - *response_type*: tipo de flujo (code)
  - *client_id*: identificador del cliente
  - *redirect_uri*: url de vuelta a nuestra aplicación
  - *scope*: para qué queremos esta autorización

Authorization Code

- Cuando el cliente es validado, se devuelve una respuesta así:

  *https://cliente.ejemplo.com/cb?**code**=AbCdEfGHiJK12345&**state**=xyz*

  - *code*: código que representa el consentimiento del usuario y su autorización
  - *state*: debe ser igual que en la petición
- Con el código, hacemos una petición POST como la siguiente

Authorization Code

POST /token HTTP/1.1
Host: autorizacion.servidor.com
Authorization: Basic afds8709afs8790asf (client-id:client-secret en base64)

grant_type=authorization_code
&**code**=AbCdEfGHiJK12345
&**redirect_uri**=https://cliente.ejemplo.com/cb

Authorization Code (alternativa)

POST /token HTTP/1.1
Host: autorizacion.servidor.com

grant_type=authorization_code
&**code**=AbCdEfGHiJK12345
&**redirect_uri**=https://cliente.ejemplo.com/cb
&**client_id**=the-client-id
&**client_secret**=qwepuirqewipor09748nmenads

Respuesta (si todo va bien)

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache


```
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"example",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
    "example_parameter":"example_value"
}
```

Implicit

- Se utiliza con clientes públicos (que no son capaces de guardar la contraseña convenientemente).
- Pensado para aplicaciones Javascript, Angular, ...
- Veamos los pasos que se siguen

Implicit

- El cliente redirige al usuario al endpoint de autorización, con una serie de parámetros

  *https://autorizacion.servidor.com/authorize?**response_type**=token&**client_id**=the-client-id&**state**=xyz&**redirect_uri**=https://cliente.ejemplo.com/cb&**scope**=api_read*

  - *response_type*: token
  - *client_id*
  - *redirect_uri*
  - *scope*

Implicit

- Cuando el cliente es validado, se devuelve una respuesta así:

  *https://cliente.ejemplo.com/cb?**access_token**=ABCDEFdaf379489a&**token_type**=example&**expires_in**=3600&**state**=xyz*

  - *access_token:* el token
  - *token_type*: tipo de token
  - expires_in: tiempo de vida
  - *state*: debe ser igual que en la petición

Password

- Apropiado cuando entre el cliente y el servidor de autorización hay una relación de confianza.
- Debería ser usado cuando no se pueda utilizar otra alternativa de flujo.
- Se puede utilizar para migrar desde la autenticación Básica hacia OAuth2

Password

POST /token HTTP/1.1
Host: autorizacion.servidor.com
Authorization: Basic afds8709afs8790asf (client-id:client-secret en base64)
Content-Type: application/x-www-form-urlencoded

grant_type=password
&**username**=luismi
&**password**=AsDf1234

Respuesta (si todo va bien)

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache


```
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"example",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
    "example_parameter":"example_value"
}
```

Client Credentials

- Apropiado cuando no existen usuarios propietarios del recurso. Es decir, si no hay usuarios involucrados.
- Sin haberlos, podemos seguir utilizando OAuth para proteger nuestra API.
- La aplicación cliente, en sí, es el propietario del recurso y no hay usuarios involucrados.

Client credentials

POST /token HTTP/1.1
Host: autorizacion.servidor.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&**client_id**=the-client-id
&**client_secret**=qwepuirqewipor09748nmenads
&**scope**=API_READ

Respuesta (si todo va bien)

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache


```
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"example",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
    "example_parameter":"example_value"
}
```