Sesión 04

Seguridad de aplicaciones

Instructor:

ERICK ARÓSTEGUI
earostegui@galaxy.edu.pe

GALAXY TRAINING

8

NET

FULL-STACK DEVELOPER

ÍNDICE

GALAXY TRAINING

01

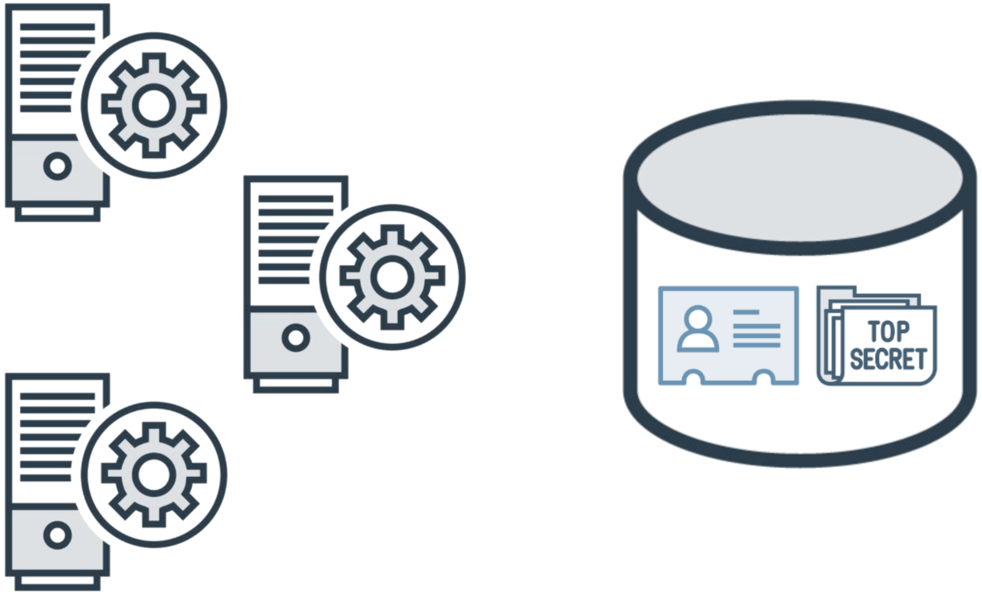
Desafíos con la seguridad perimetral

GALAXY
TRAINING

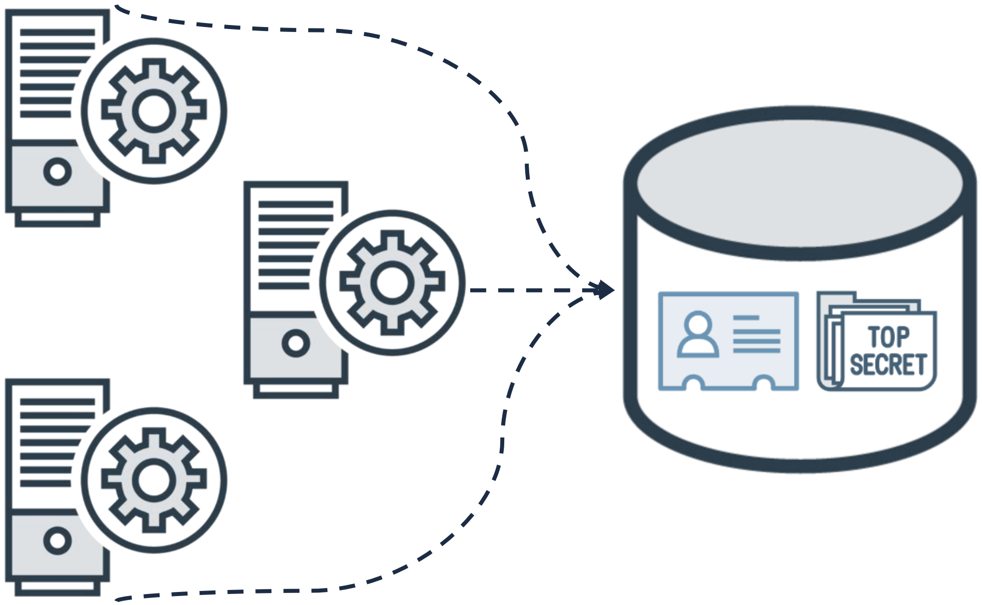
Desafíos con la seguridad perimetral

Desafíos con la seguridad perimetral

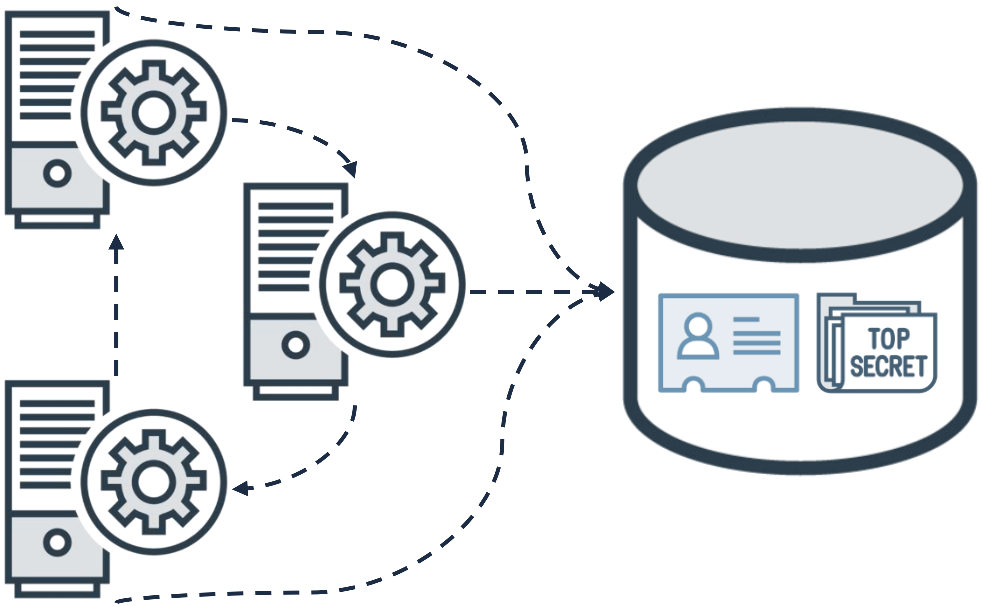
Desafíos con la seguridad perimetral



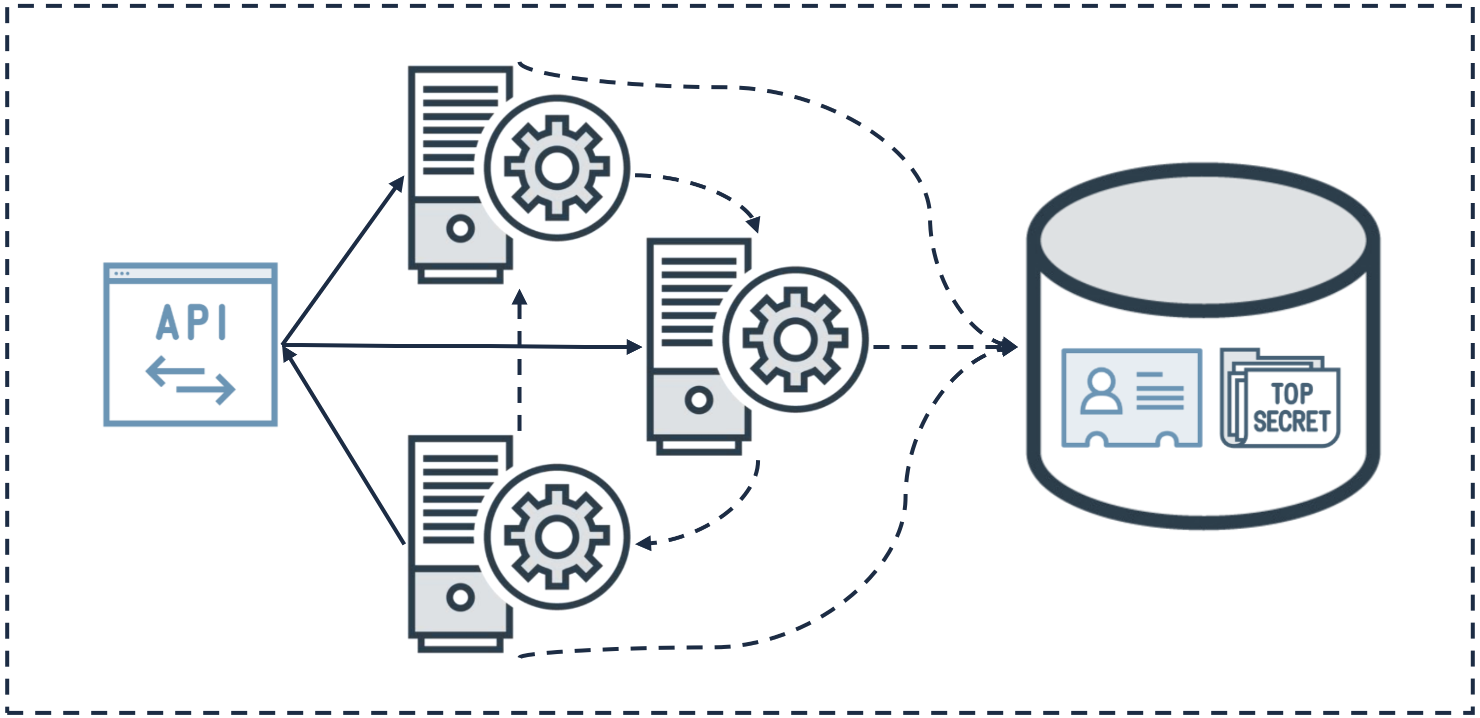
Desafíos con la seguridad perimetral



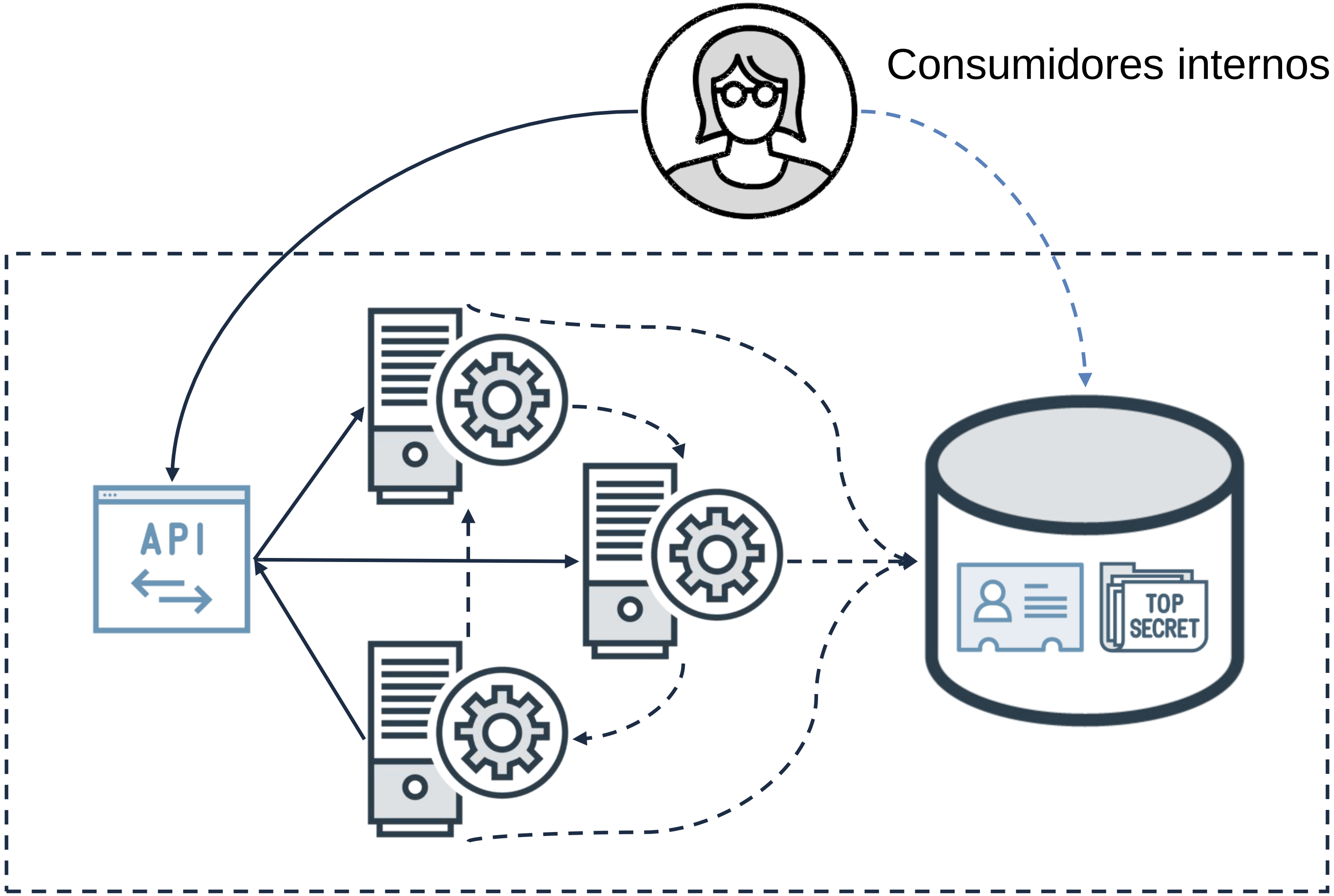
Desafíos con la seguridad perimetral



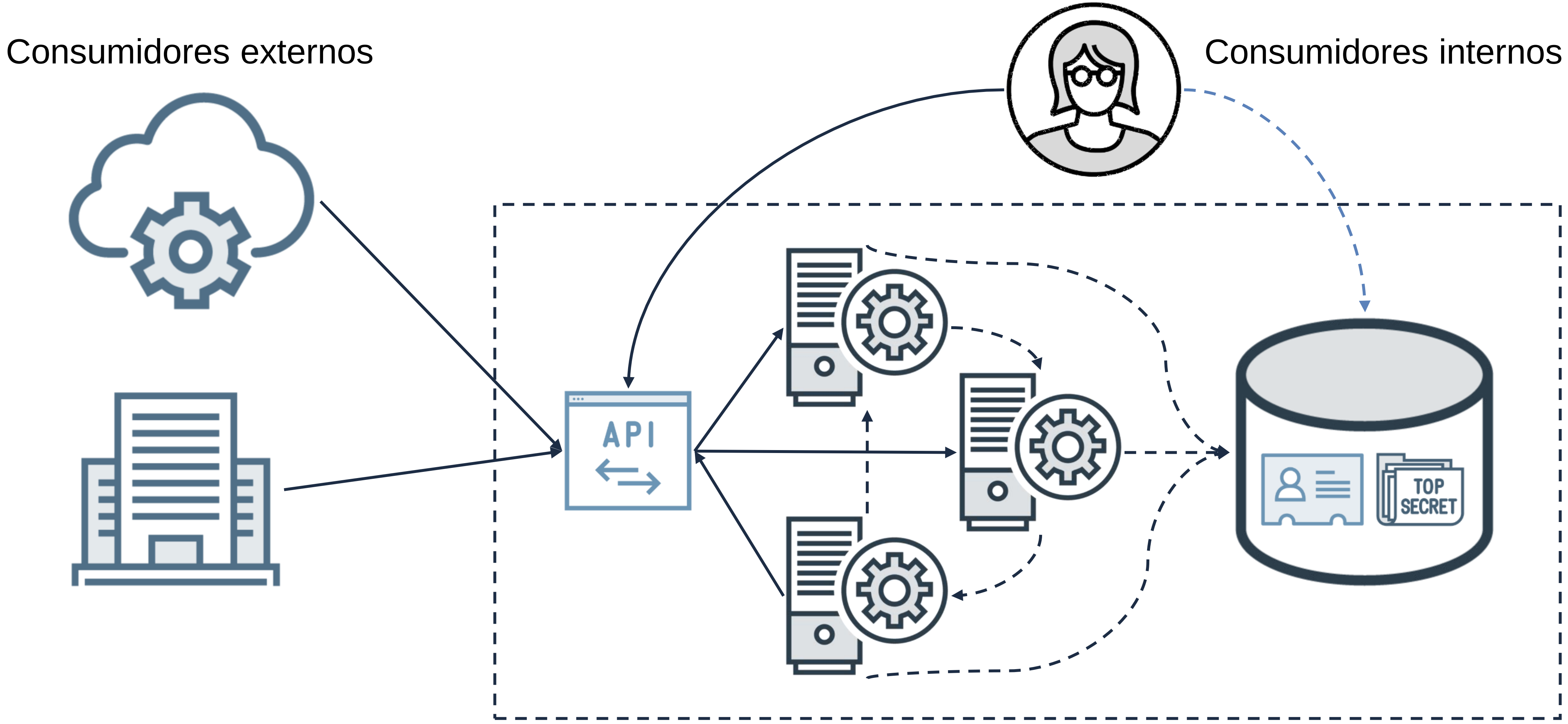
Desafíos con la seguridad perimetral



Desafíos con la seguridad perimetral

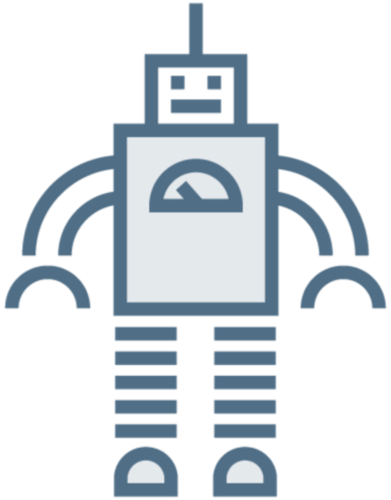


Desafíos con la seguridad perimetral



Tipo de consumidor

Humano (Usuario)



No humano
(Servicio o sistema)

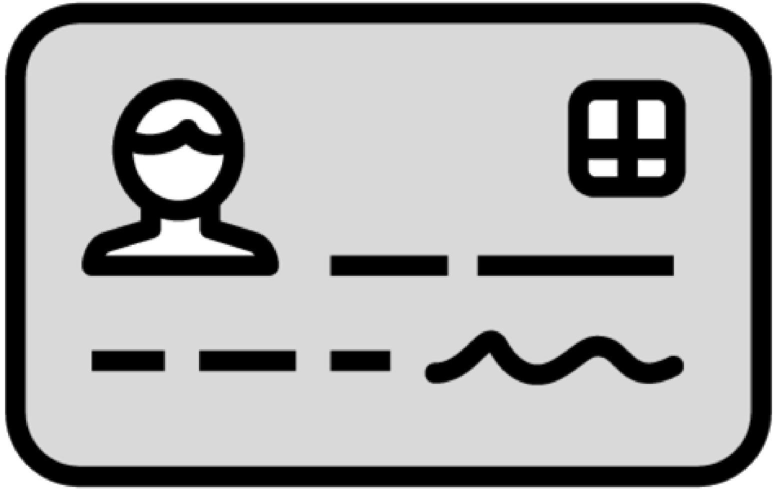
02

Autenticación / Autorización

GALAXY
TRAINING

Autenticación / Autorización

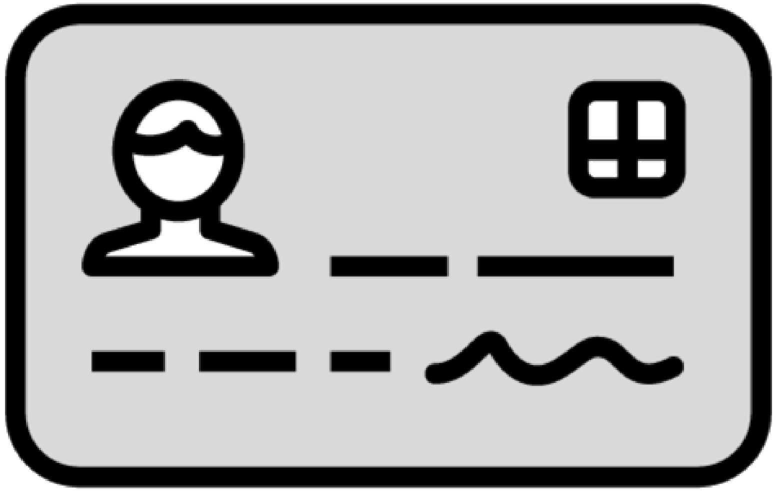
Autenticación / Autorización

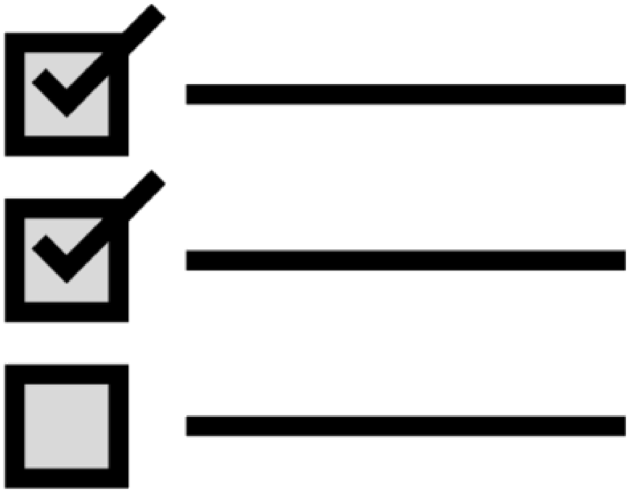


Autenticación

El proceso de identificar quién es el consumidor.

Autenticación / Autorización





Autenticación

El proceso de identificar quién es el consumidor.

Autorización

El proceso de identificar lo que el consumidor puede hacer.

03

Oauth2, OpenID Connect y JWT

GALAXY
TRAINING

OAuth 2.0

OAuth 2.0 (**Open Authorization**) es un protocolo de autorización que permite a una aplicación **obtener acceso limitado a los recursos** de un usuario en otro **servicio web**.

Protocolo que permite a los usuarios **compartir** su **información** almacenada **en un sitio web con otro sitio web sin necesidad de proporcionar sus credenciales** de inicio de sesión.

En lugar de compartir contraseñas, las aplicaciones obtienen tokens de acceso que pueden ser utilizados para acceder a ciertos recursos en nombre del usuario. Estos tokens de acceso son **emitidos por un servidor de autorización** después de que el usuario ha proporcionado su consentimiento.

OAuth 2.0

OAuth 2.0 es ampliamente **utilizado** en la web por servicios como **Google, Facebook, Twitter y Microsoft** para permitir el uso de funcionalidades de sus plataformas en otras aplicaciones. Por ejemplo, una aplicación puede utilizar OAuth 2.0 para obtener permiso para acceder a los datos de tu calendario en Google, o para publicar actualizaciones de estado en tu nombre en Twitter.

Es importante destacar que OAuth 2.0 es un protocolo de autorización, no de autenticación. Aunque a menudo se utiliza en combinación con protocolos de autenticación como OpenID Connect, OAuth 2.0 por sí solo no proporciona formas de autenticar la identidad de los usuarios.

Flujos de concesión OAuth2

Authorization Code Grant

Este es el flujo más común y se utiliza para aplicaciones que se ejecutan en un servidor web. El proceso implica redirigir al usuario a un servidor de autorización, que luego redirige de vuelta a la aplicación con un código de autorización. Este código se intercambia por un token de acceso.

Implicit Grant

Este flujo fue diseñado para aplicaciones basadas en JavaScript que se ejecutan en el navegador del usuario. En lugar de recibir un código de autorización que debe ser intercambiado por un token de acceso, la aplicación recibe directamente el token de acceso. Este flujo se considera menos seguro y ya no se recomienda su uso.

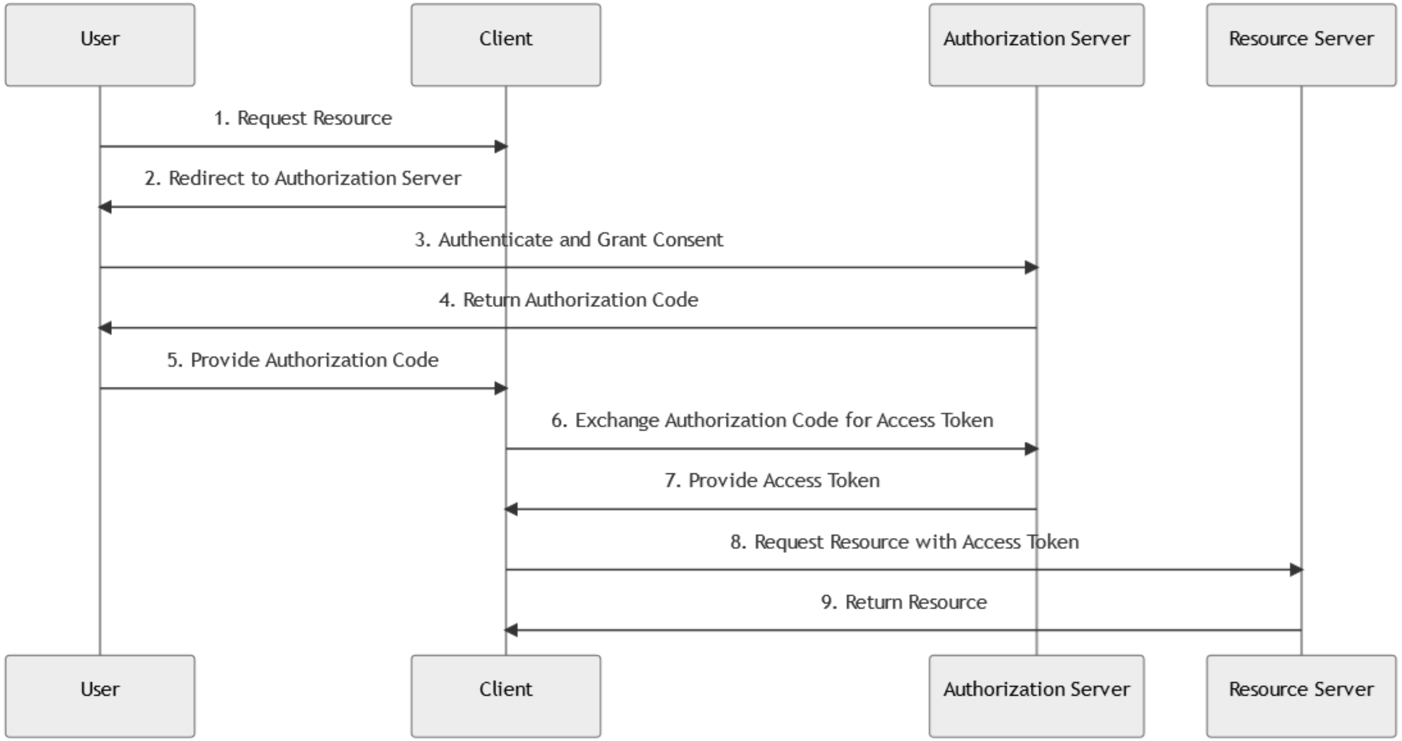
Resource Owner Password Credentials

Este flujo permite a la aplicación recoger las credenciales del usuario (nombre de usuario y contraseña) y usarlas para obtener un token de acceso directamente. Este flujo solo se recomienda para aplicaciones de confianza, como las creadas por el mismo servicio que proporciona la API.

Client Credentials Grant

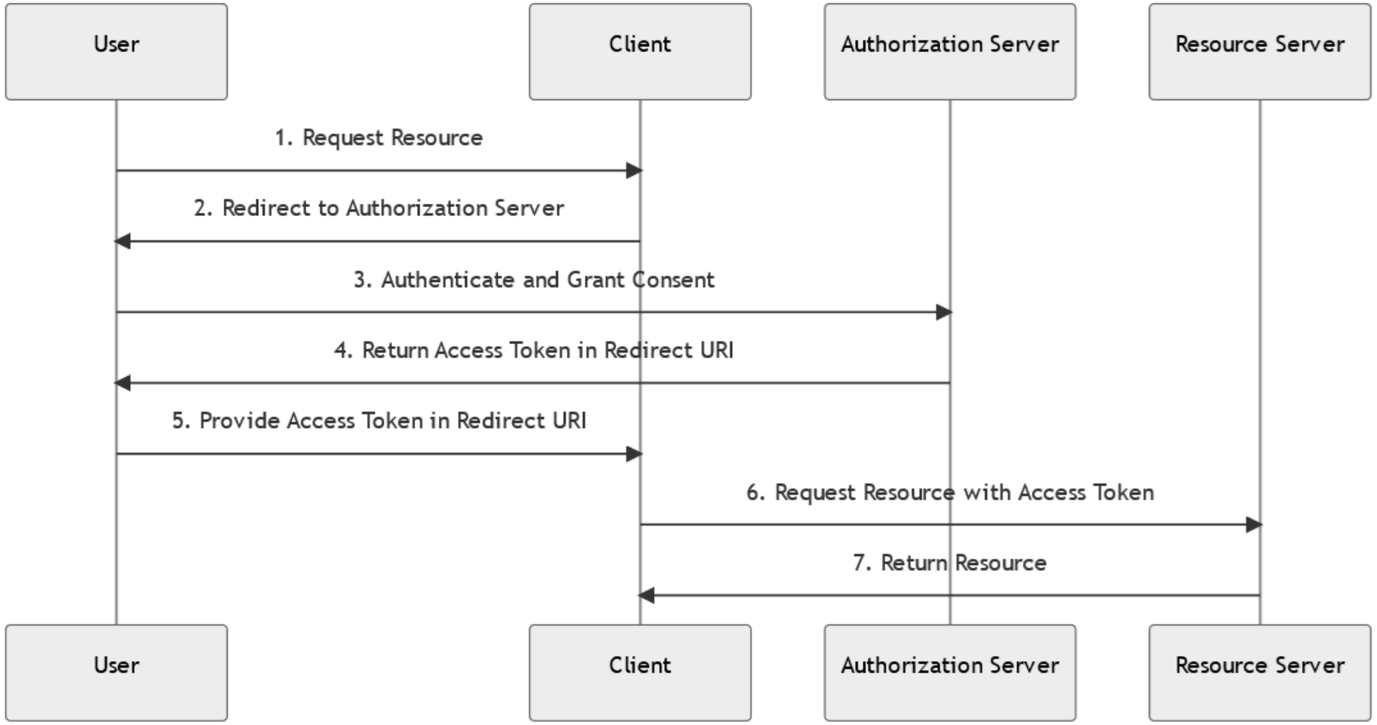
Este flujo es para aplicaciones que acceden a su propio servicio, no en nombre de un usuario. La aplicación utiliza sus propias credenciales para obtener un token de acceso.

Authorization Code Grant



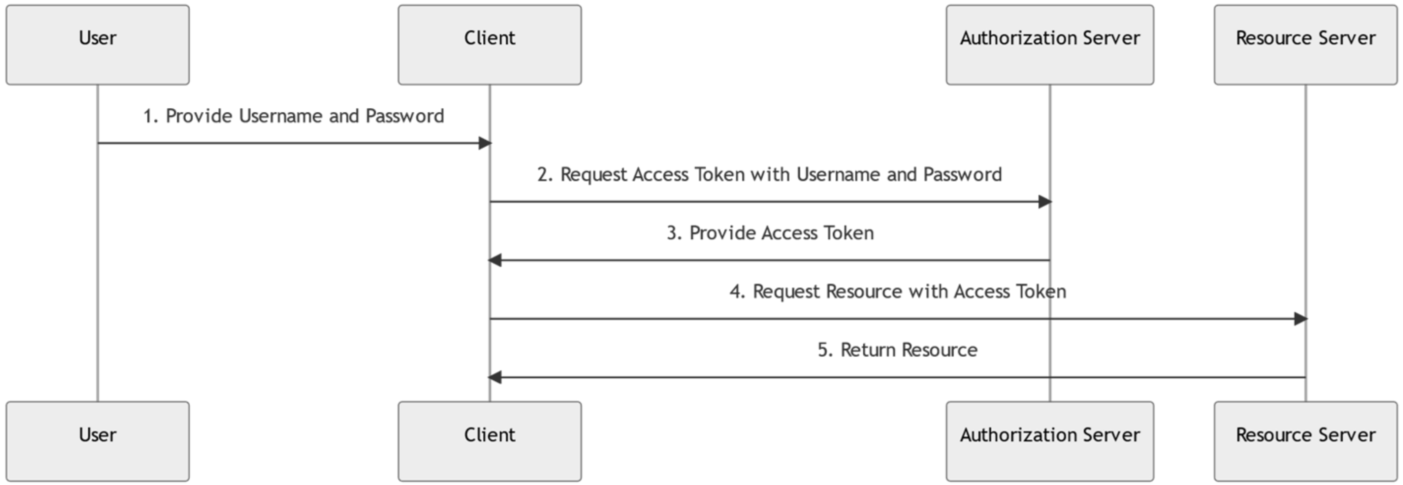
1. El Usuario solicita un recurso al Cliente.
2. El Cliente redirige al Usuario al Servidor de Autorización.
3. El Usuario se autentica y concede el consentimiento en el Servidor de Autorización.
4. El Servidor de Autorización devuelve un código de autorización al Usuario.
5. El Usuario proporciona el código de autorización al Cliente.
6. El Cliente intercambia el código de autorización por un token de acceso con el Servidor de Autorización.
7. El Servidor de Autorización proporciona un token de acceso al Cliente.
8. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
9. El Servidor de Recursos devuelve el recurso al Cliente.

Implicit Grant



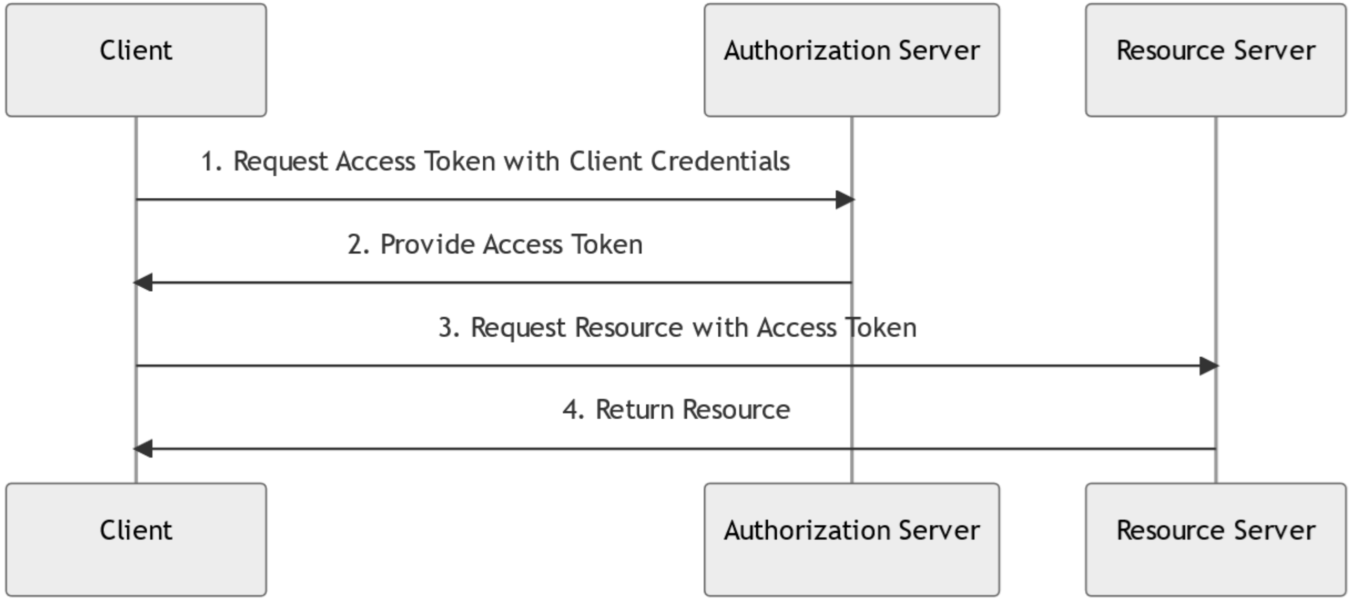
1. El Usuario solicita un recurso al Cliente.
2. El Cliente redirige al Usuario al Servidor de Autorización.
3. El Usuario se autentica y concede el consentimiento en el Servidor de Autorización.
4. El Servidor de Autorización devuelve un token de acceso en la URI de redirección al Usuario.
5. El Usuario proporciona el token de acceso en la URI de redirección al Cliente.
6. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
7. El Servidor de Recursos devuelve el recurso al Cliente.

Resource Owner Password Credentials Grant



1. El Usuario proporciona su nombre de usuario y contraseña al Cliente.
2. El Cliente solicita un token de acceso al Servidor de Autorización utilizando el nombre de usuario y la contraseña del Usuario.
3. El Servidor de Autorización proporciona un token de acceso al Cliente.
4. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
5. El Servidor de Recursos devuelve el recurso al Cliente.

Client Credentials Grant



1. El Cliente solicita un token de acceso al Servidor de Autorización utilizando sus propias credenciales.
2. El Servidor de Autorización proporciona un token de acceso al Cliente.
3. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
4. El Servidor de Recursos devuelve el recurso al Cliente.

OpenID Connect

OpenID Connect es un **estándar de autenticación** que se construye sobre el protocolo OAuth 2.0. Mientras que OAuth 2.0 se utiliza para la autorización, **OpenID Connect introduce el concepto de un ID token**, que **permite a la aplicación cliente verificar la identidad del usuario**.

El ID token contiene información sobre el usuario. Esta información **puede incluir el nombre de usuario**, la dirección de correo electrónico, la imagen de **perfil y otros datos** que el usuario ha acordado compartir.

OpenID Connect

OpenID Connect también introduce un nuevo emdpoint, el **enpdoint de userinfo**, que la aplicación cliente puede utilizar para obtener más información sobre el usuario una vez que ha obtenido un token de acceso.

OpenID Connect se utiliza para autenticar al usuario y proporcionar información sobre su identidad a la aplicación. Esto lo convierte en una solución ideal para el **inicio de sesión único (SSO)**, donde un usuario puede **iniciar sesión en varias aplicaciones con una sola cuenta.**

OpenID Connect

Authorization Code Flow

Este es el flujo más común y se utiliza para aplicaciones que se ejecutan en un servidor web. Es similar al flujo de código de autorización en OAuth 2.0, pero en este caso, además del token de acceso, el servidor de autorización también devuelve un ID token.

Implicit Flow

Este flujo fue diseñado para aplicaciones basadas en JavaScript que se ejecutan en el navegador del usuario. En lugar de recibir un código de autorización que debe ser intercambiado por un token de acceso, la aplicación recibe directamente el token de acceso y el ID token.

Hybrid Flow

Este flujo combina elementos del flujo de código de autorización y del flujo implícito. Permite a la aplicación recibir algunos tokens (acceso, ID o ambos) en un redireccionamiento y solicitar otros tokens en un intercambio de código de autorización.

OpenID Connect

Authorization Code Flow

Este es el flujo más común y se utiliza para aplicaciones que se ejecutan en un servidor web. Es similar al flujo de código de autorización en OAuth 2.0, pero en este caso, además del token de acceso, el servidor de autorización también devuelve un ID token.

Implicit Flow

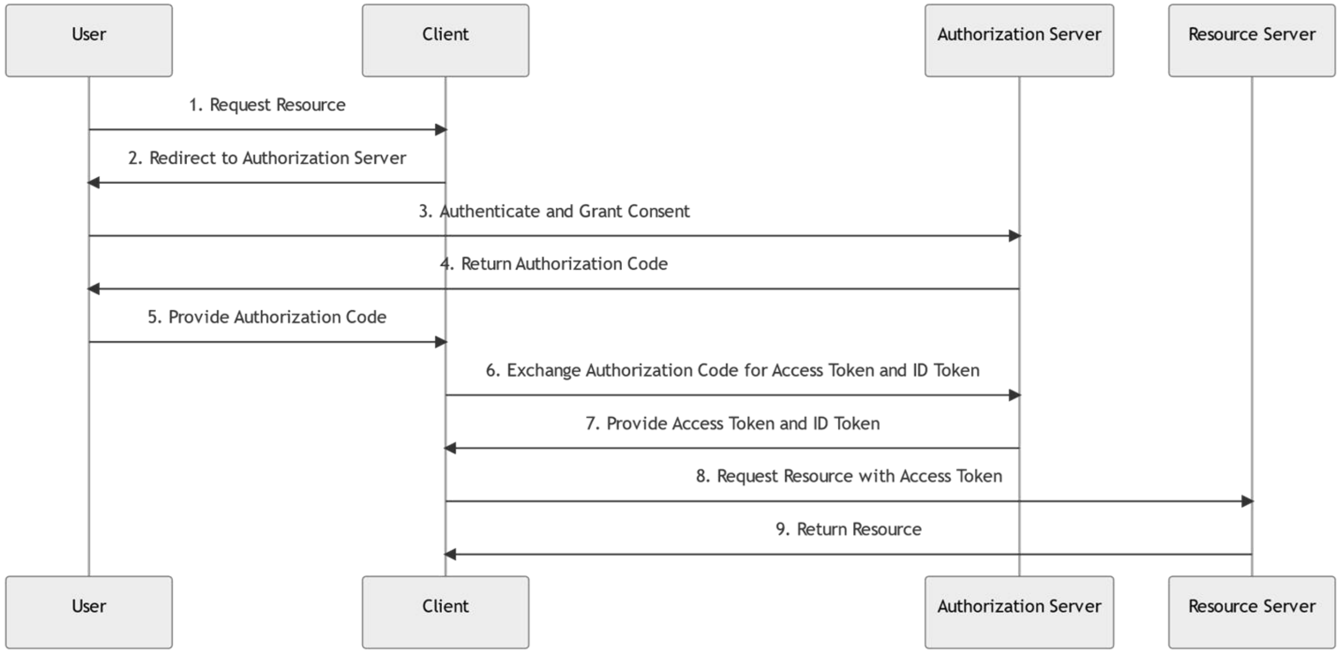
Este flujo fue diseñado
JavaScript que se ejecutan
lugar de recibir un código
intercambiado por un token de acceso, la aplicación recibe directamente el token de acceso y el ID token.

Este flujo se considera menos seguro y ya no se recomienda su uso.

Hybrid Flow

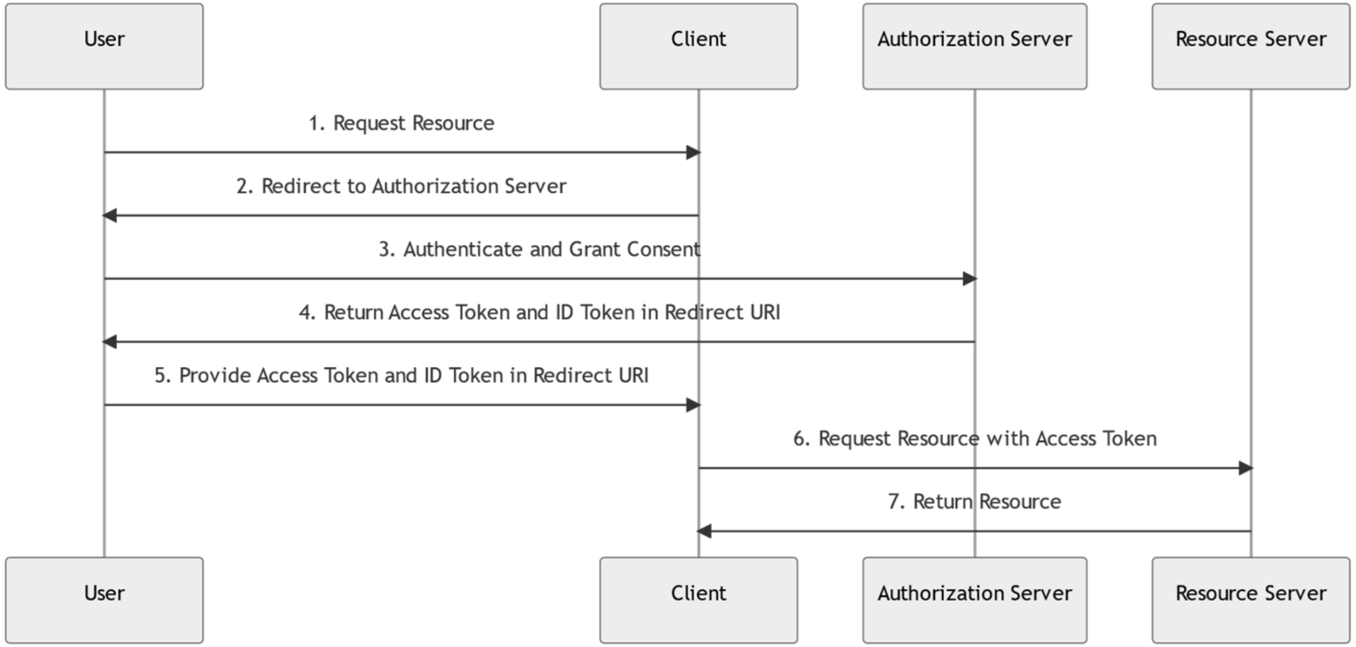
Este flujo combina elementos del flujo de código de autorización y del flujo implícito. Permite a la aplicación recibir algunos tokens (acceso, ID o ambos) en un redireccionamiento y solicitar otros tokens en un intercambio de código de autorización.

Authorization Code Flow



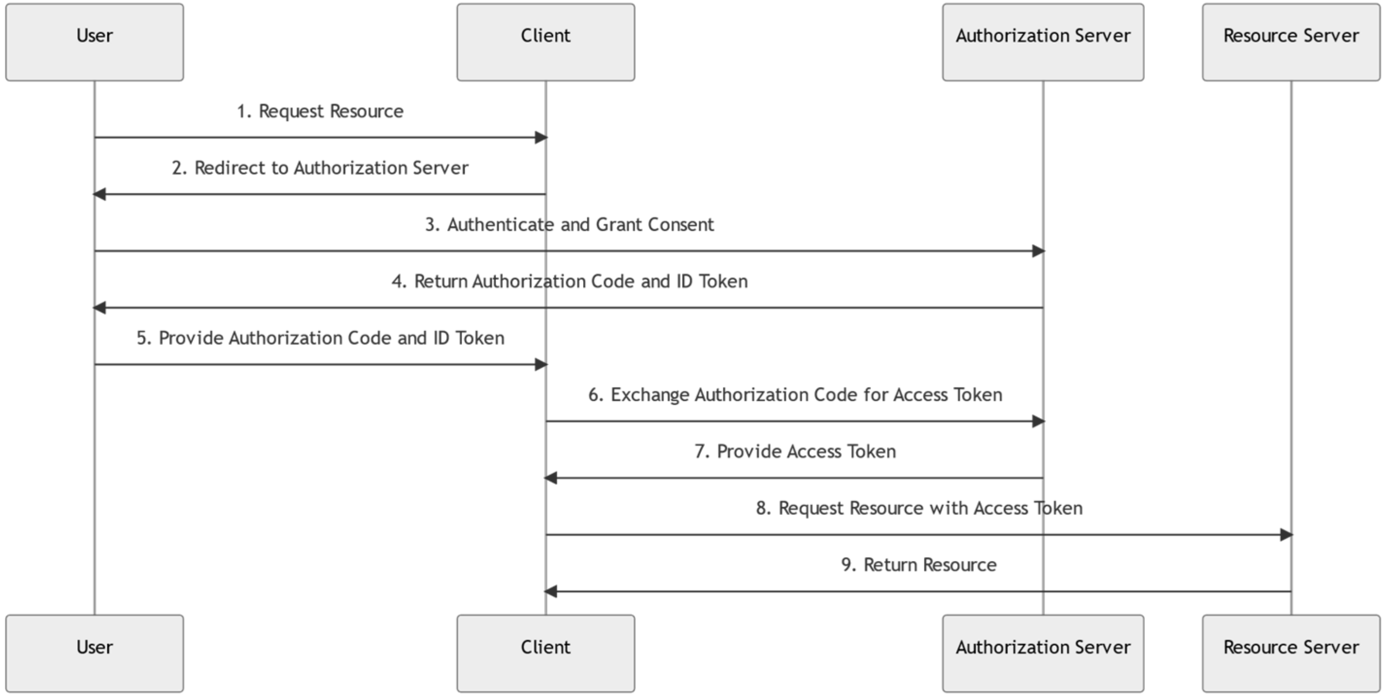
1. El Usuario solicita un recurso al Cliente.
2. El Cliente redirige al Usuario al Servidor de Autorización.
3. El Usuario se autentica y concede el consentimiento en el Servidor de Autorización.
4. El Servidor de Autorización devuelve un código de autorización al Usuario.
5. El Usuario proporciona el código de autorización al Cliente.
6. El Cliente intercambia el código de autorización por un token de acceso y un ID token con el Servidor de Autorización.
7. El Servidor de Autorización proporciona un token de acceso y un ID token al Cliente.
8. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
9. El Servidor de Recursos devuelve el recurso al Cliente.

Implicit Flow



1. El Usuario solicita un recurso al Cliente.
2. El Cliente redirige al Usuario al Servidor de Autorización.
3. El Usuario se autentica y concede el consentimiento en el Servidor de Autorización.
4. El Servidor de Autorización devuelve un token de acceso y un ID token en la URI de redirección al Usuario.
5. El Usuario proporciona el token de acceso y el ID token en la URI de redirección al Cliente.
6. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
7. El Servidor de Recursos devuelve el recurso al Cliente.

Hybrid Flow



1. El Usuario solicita un recurso al Cliente.
2. El Cliente redirige al Usuario al Servidor de Autorización.
3. El Usuario se autentica y concede el consentimiento en el Servidor de Autorización.
4. El Servidor de Autorización devuelve un código de autorización y un ID token al Usuario.
5. El Usuario proporciona el código de autorización y el ID token al Cliente.
6. El Cliente intercambia el código de autorización por un token de acceso con el Servidor de Autorización.
7. El Servidor de Autorización proporciona un token de acceso al Cliente.
8. El Cliente solicita el recurso al Servidor de Recursos utilizando el token de acceso.
9. El Servidor de Recursos devuelve el recurso al Cliente.

Tokens

Un token es una cadena de caracteres generada por un servidor en respuesta a una **solicitud válida de autenticación**. Este token **representa la identidad del usuario** y los **permisos de acceso**, y se utiliza para acceder a recursos protegidos.

Un token es como un **pase de acceso digital** que permite al usuario acceder a ciertos recursos y realizar ciertas operaciones. Es una parte esencial de los sistemas de autenticación y autorización modernos.

GALAXY TRAINING

Características de un token

Identificación del usuario

Un token representa a un usuario específico. Contiene información (llamada "claims" o afirmaciones) que identifica al usuario ante el servidor. Esta información puede incluir el ID del usuario, el nombre de usuario, el correo electrónico y otros datos.

Autorización

Además de identificar al usuario, un token también puede contener información sobre los permisos del usuario. Esto permite al servidor determinar qué recursos puede acceder el usuario y qué operaciones puede realizar.

Seguridad

Son emitidos por un servidor de autorización después de que el usuario se ha autenticado correctamente. Los tokens pueden ser firmados digitalmente para garantizar su integridad y autenticidad, y también pueden ser cifrados para proteger la información que contienen.

Formato

Los tokens pueden estar en varios formatos, como JWT (JSON Web Tokens), SAML (Security Assertion Markup Language) tokens, tokens opacos, y PASETO (Platform-Agnostic Security Tokens). Cada formato tiene sus propias características y usos.

Vida útil

Los tokens tienen una vida útil limitada. Una vez que un token ha expirado, ya no puede ser utilizado para acceder a recursos, y el usuario debe autenticarse de nuevo para obtener un nuevo token.

Formatos de un token

JWT (JSON Web Tokens)

Este es un formato de token muy común que se utiliza en OAuth 2.0 y OpenID Connect. Los JWT son tokens compactos y autocontenidos que pueden contener una variedad de afirmaciones (claims) sobre un usuario. Los JWT pueden ser firmados digitalmente para garantizar su integridad y también pueden ser cifrados.

SAML (Security Assertion Markup Language) Tokens

Aunque SAML es en realidad un protocolo de autenticación y autorización separado, en algunos casos, los tokens SAML pueden ser utilizados con OAuth 2.0. Un token SAML es un tipo de token de seguridad que contiene información sobre un usuario y sus derechos de acceso. Los tokens SAML son más grandes y más verbosos que los JWT.

Opaque Tokens

A diferencia de los JWT y los tokens SAML, los tokens opacos son tokens de acceso que no contienen ninguna información legible por el usuario. En su lugar, cualquier información necesaria se almacena en el servidor y se asocia con el token. Los tokens opacos pueden ser una cadena de caracteres aleatoria.

Formatos de un token

JWT (JSON Web Tokens)

Este es un formato de token muy común que se utiliza en OAuth 2.0 y OpenID Connect. Los JWT son tokens compactos y autocontenidos que pueden contener una variedad de afirmaciones (claims) sobre un usuario. Los JWT pueden ser firmados digitalmente para garantizar su integridad y también pueden ser cifrados.

SAML (Security Assertion Markup Language) Tokens

Aunque SAML es en realidad un protocolo de autenticación y autorización separado, en algunos casos, los tokens SAML pueden ser utilizados con OAuth 2.0. Un token SAML es un tipo de token de seguridad que contiene información sobre un usuario y sus derechos de acceso. Los tokens SAML son más grandes y más verbosos que los JWT.

Opaque Tokens

A diferencia de los JWT y los tokens SAML, los tokens opacos son tokens de acceso que no contienen ninguna información legible por el usuario. En su lugar, cualquier información necesaria se almacena en el servidor y se asocia con el token. Los tokens opacos pueden ser una cadena de caracteres aleatoria.

PASETO (Platform-Agnostic Security Tokens)

Este es un formato de token relativamente nuevo que se propone como una alternativa más segura a JWT. Al igual que los JWT, los PASETO pueden contener una serie de afirmaciones sobre un usuario y pueden ser firmados y cifrados.

Formatos de un token

JWT (JSON Web Tokens)

Este es un formato de token muy común que se utiliza en OAuth 2.0 y OpenID Connect. Los JWT son tokens compactos y autocontenidos que pueden contener una variedad de afirmaciones (claims) sobre un usuario. Los JWT pueden ser firmados digitalmente para garantizar su integridad y también pueden ser cifrados.

SAML (Security Assertion Markup Language) Tokens

Aunque SAML es en realidad un protocolo de autenticación y autorización separado, en algunos casos, los tokens SAML pueden ser utilizados con OAuth 2.0. Un token SAML es un tipo de token de seguridad que contiene información sobre un usuario y sus derechos de acceso. Los tokens SAML son más grandes y más verbosos que los JWT.

Opaque Tokens

A diferencia de los JWT y los tokens SAML, los tokens opacos son tokens de acceso que no contienen ninguna información legible por el usuario. En su lugar, cualquier información necesaria se almacena en el servidor y se asocia con el token. Los tokens opacos pueden ser una cadena de caracteres aleatoria.

PASETO (Platform-Agnostic Security Tokens)

Este es un formato de token relativamente nuevo que se propone como una alternativa más segura a JWT. Al igual que los JWT, los PASETO pueden contener una serie de afirmaciones sobre un usuario y pueden ser firmados y cifrados.

GALAXY TRAINING

¿Qué es un JWT?

JSON Web Token es un estándar abierto de la industria que se usa para compartir información entre dos entidades, generalmente un cliente (como el front-end de su aplicación) y un servidor (el backend de su aplicación).

Contienen objetos JSON que tienen la información que necesita ser compartida. Cada JWT también se firma mediante criptografía (hashing) para garantizar que el contenido JSON (también conocido como notificaciones JWT) no pueda ser alterado por el cliente o una parte malintencionada.

Estructura de un JWT

Un JWT contiene tres partes:

- **Cabecera**: Consta de dos partes:
  - El algoritmo de firma que se está utilizando.
  - El tipo de token, que, en este caso, es principalmente "JWT".
- **Carga útil**: la carga contiene las notificaciones o el objeto JSON.
- **Firma**: Una cadena que se genera a través de un algoritmo criptográfico que se puede utilizar para verificar la integridad de la carga útil JSON.

Estructura de un JWT

Estructura de un JWT

CABECERA

GALAXY TRAINING

Estructura de un JWT

CABECERA

CARGA UTIL

Estructura de un JWT

CABECERA	CARGA UTIL	FIRMA

GALAXY TRAINING

Estructura de un JWT

CABECERA | CARGA UTIL | FIRMA

```
{
    “alg” : “HS246” ,
    “typ“ : “JWT”
}
```

GALAXY TRAINING

Estructura de un JWT

CABECERA | CARGA UTIL | FIRMA

```
{
    “alg” : “HS246” ,
    “typ“ : “JWT”
}
```

```
{
    “sub” : “1234567890” ,
    “name” : “Erick Arostegui” ,
    “admin” : true ,
    “iat” : 1516239022
}
```

GALAXY TRAINING

Estructura de un JWT

CABECERA	CARGA UTIL	FIRMA
{ "alg" : "HS246" , "typ" : "JWT" }	{ "sub" : "1234567890" , "name" : "Erick Arostegui" , "admin" : true , "iat" : 1516239022 }	Base64URLSafe (HMACSHA256 (<header>.<payload> , <secret key>))

04

Keycloak, Características, Instalación y configuración

GALAXY
TRAINING

Keycloak

Es un **servidor de autenticación y gestión de identidad** de código abierto que proporciona autenticación de usuarios con una gran variedad de características.

Fue **desarrollado por Red Hat** y es ampliamente utilizado para la autenticación en aplicaciones modernas.

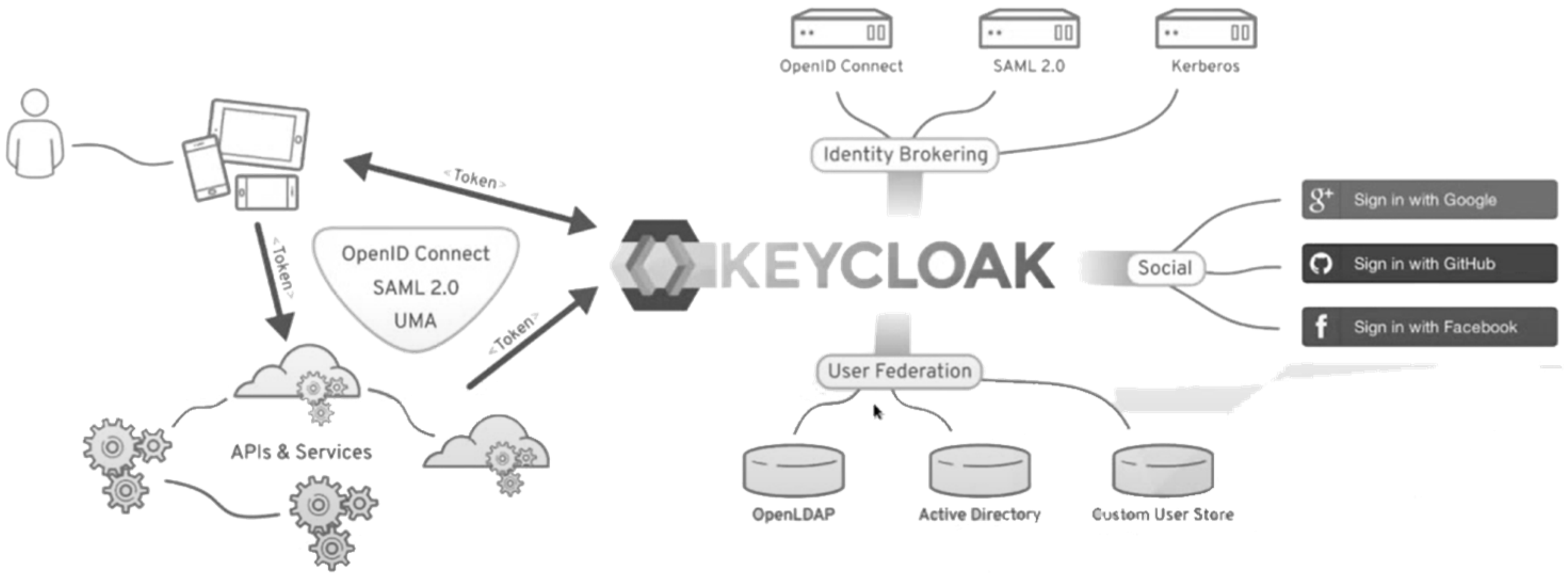
Soporta protocolos estándar de la industria como **OpenID Connect, OAuth 2.0, JWT, SAML** entre otros . Esto significa que puede integrarse con una gran cantidad de aplicaciones y servicios sin necesidad de implementar su propia solución de autenticación.

Características de Keycloak

Single-Sign On Inicie sesión una vez en varias aplicaciones	**Protocolos estándar** OpenID Connect, OAuth 2.0 y SAML 2.0	**Gestión centralizada** Para administradores y usuarios	**Adaptadores** Proteja las aplicaciones y los servicios fácilmente
LDAP y Active Directory Conectarse a directorios de usuario existentes	**Social Login** Habilite fácilmente el inicio de Social Login	**Identity Brokering** OpenID Connect or SAML 2.0 IdPs	**Alto rendimiento** Ligero, rápido y escalable
Clustering Escalabilidad y disponibilidad	**Themes** Personaliza la apariencia	**Extensible** Personalizar a través del código	**Políticas de contraseñas** Personalizar directivas de contraseñas

Keycloak, Instalación y configuración

DEMO

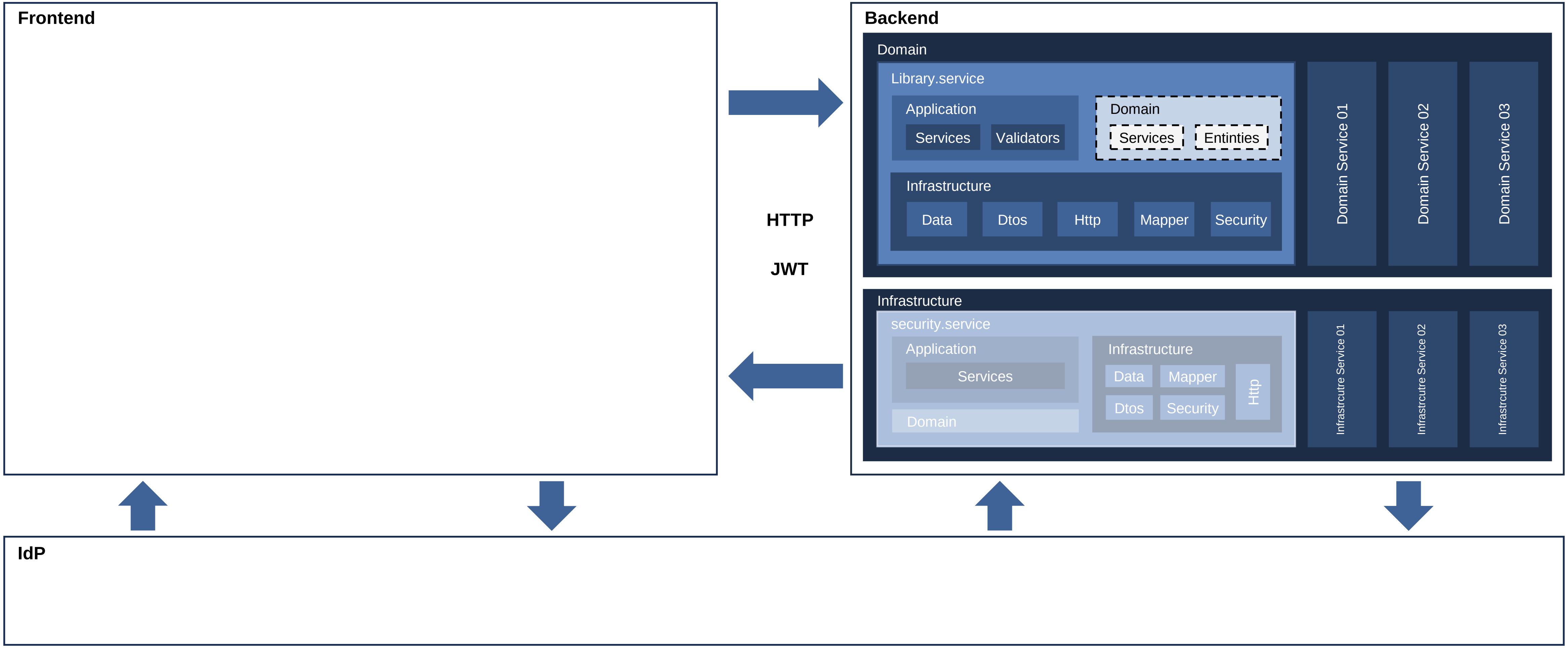


05

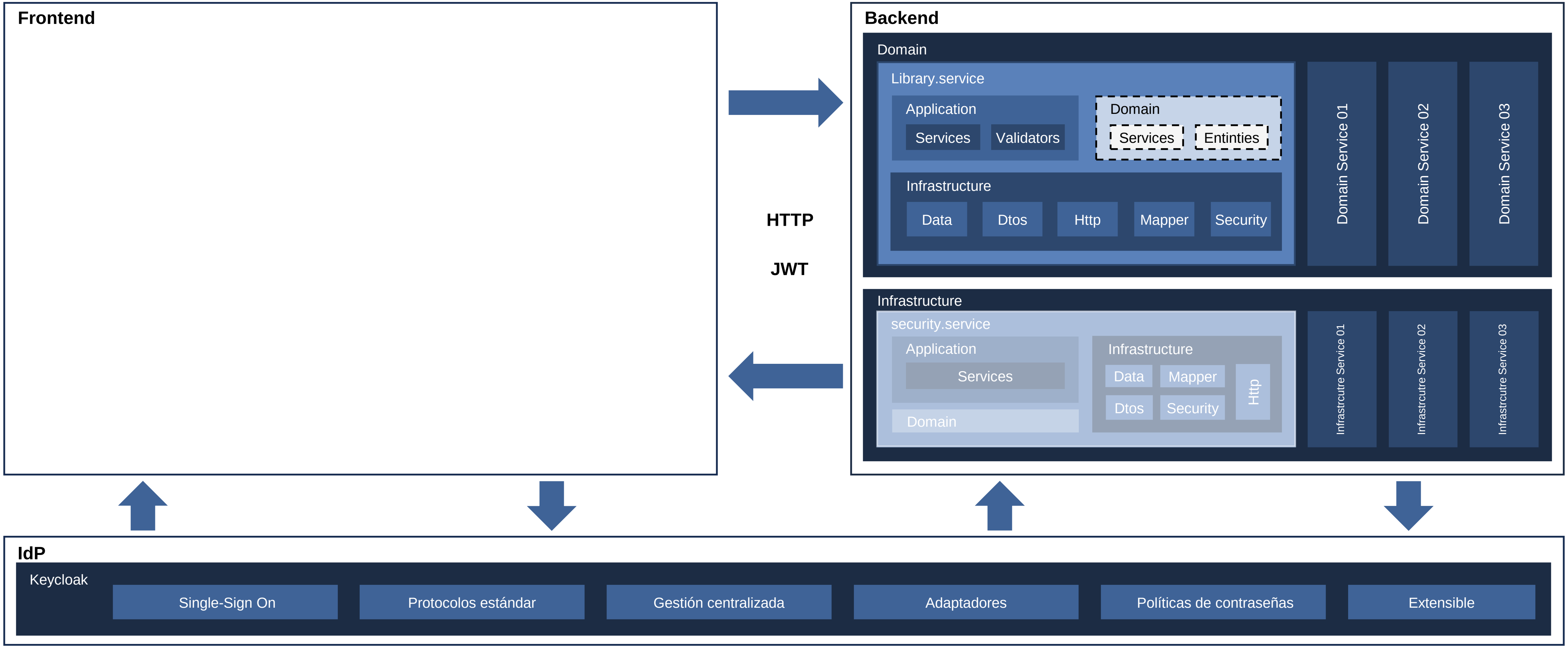
Implementación de seguridad con ASP.NET

GALAXY TRAINING

Arquitectura de Aplicación



Arquitectura de Aplicación



DEMO

Implementación de seguridad con ASP.NET

GALAXY TRAINING

GRACIAS

POR SU PREFERENCIA

GALAXY
TRAINING