KEYCLOAK

Wilder García
Felipe Rivas
Catalina Bernal

TABLA DE
CONTENIDO

EL PROBLEMA

GESTIONAR IDENTIDADES Y ACCESOS

¿Cómo centralizamos la identidad y el acceso?

¿QUÉ ES KEYCLOAK?

Plataforma de gestión de identidad y acceso (IAM) de código abbierto

Permite:

- Autenticación centralizada
- Autorización
- Gestión de usuarios y roles
- SSO
- Emisión y gestión de tokens

IAM: AUTENTICACIÓN VS AUTORIZACIÓN

Gestión de las identidades digitales y de los permisos de acceso a recursos

Autenticación

¿Quién eres?
Usuario + contraseña
Verifica identidad

Autorización

¿Qué puedes hacer?
Roles y permisos
Controla acceso

CONCEPTOS FUNDAMENTALES

Realm

User

Client

Role

CÓMO FUNCIONA KEYCLOAK

OAuth 2.0

Framework de autorización

OpenID Connect

Utilizado para la autenticación, extiende OAuth 2.0

TOKENS GENERADOS POR KEYCLOAK

01

Access token

Acceso a recursos o servicios protegidos

02

Refresh token

Obtención de nuevo access token

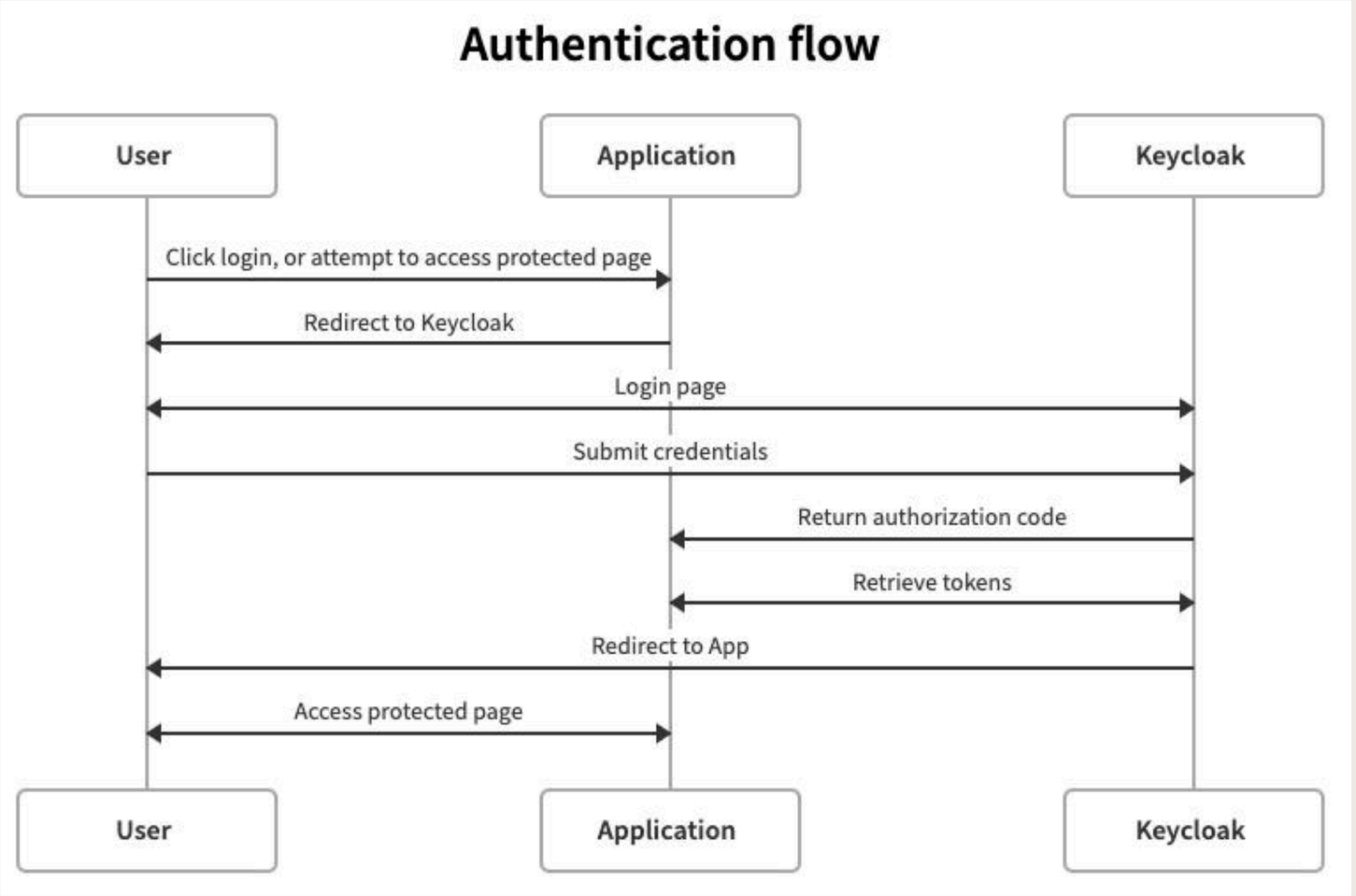
03

Token ID

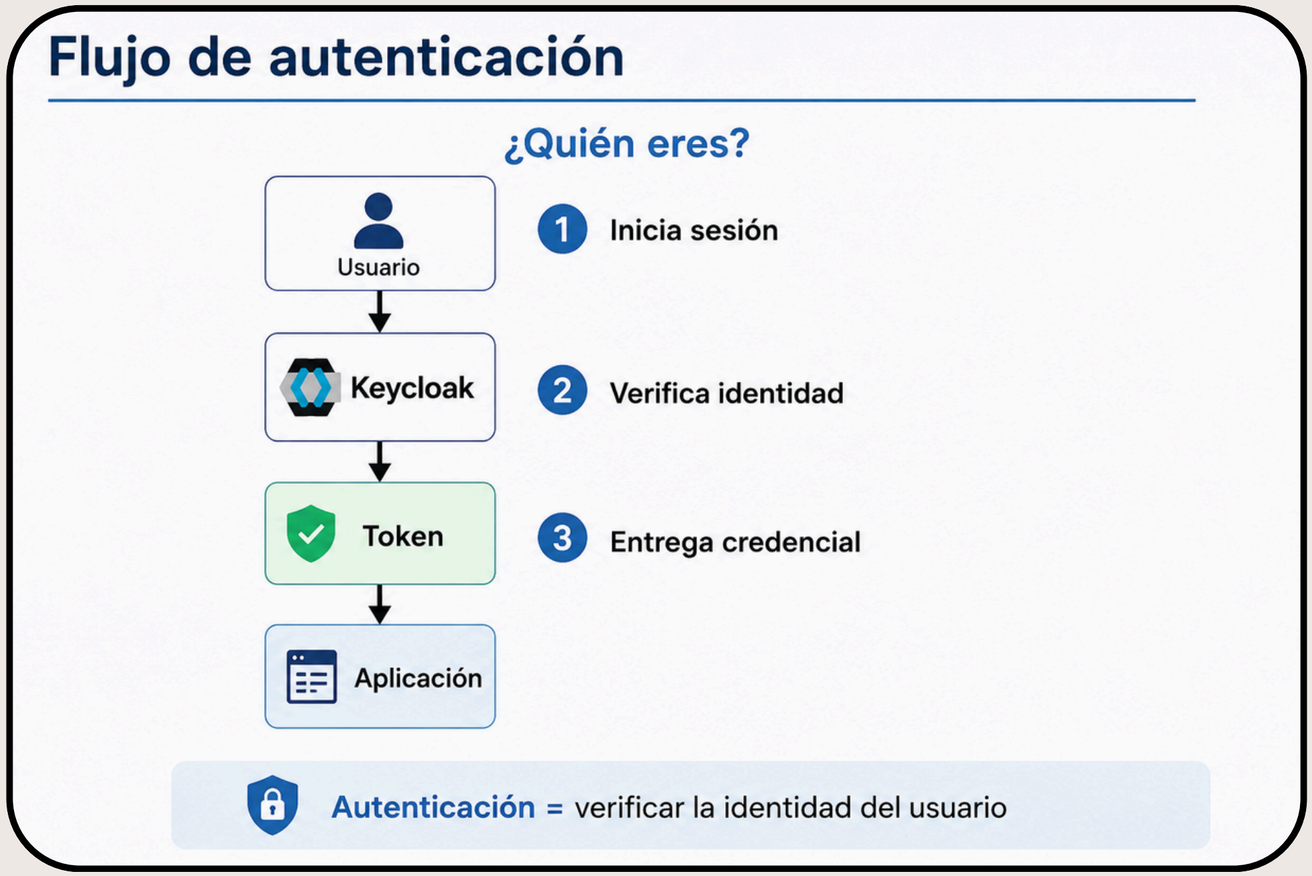
Información de identidad del usuario

FLUJO DE AUTENTICACIÓN

1. Solicitud
2. Redirección
3. Autenticación
4. Tokens
5. Acceso

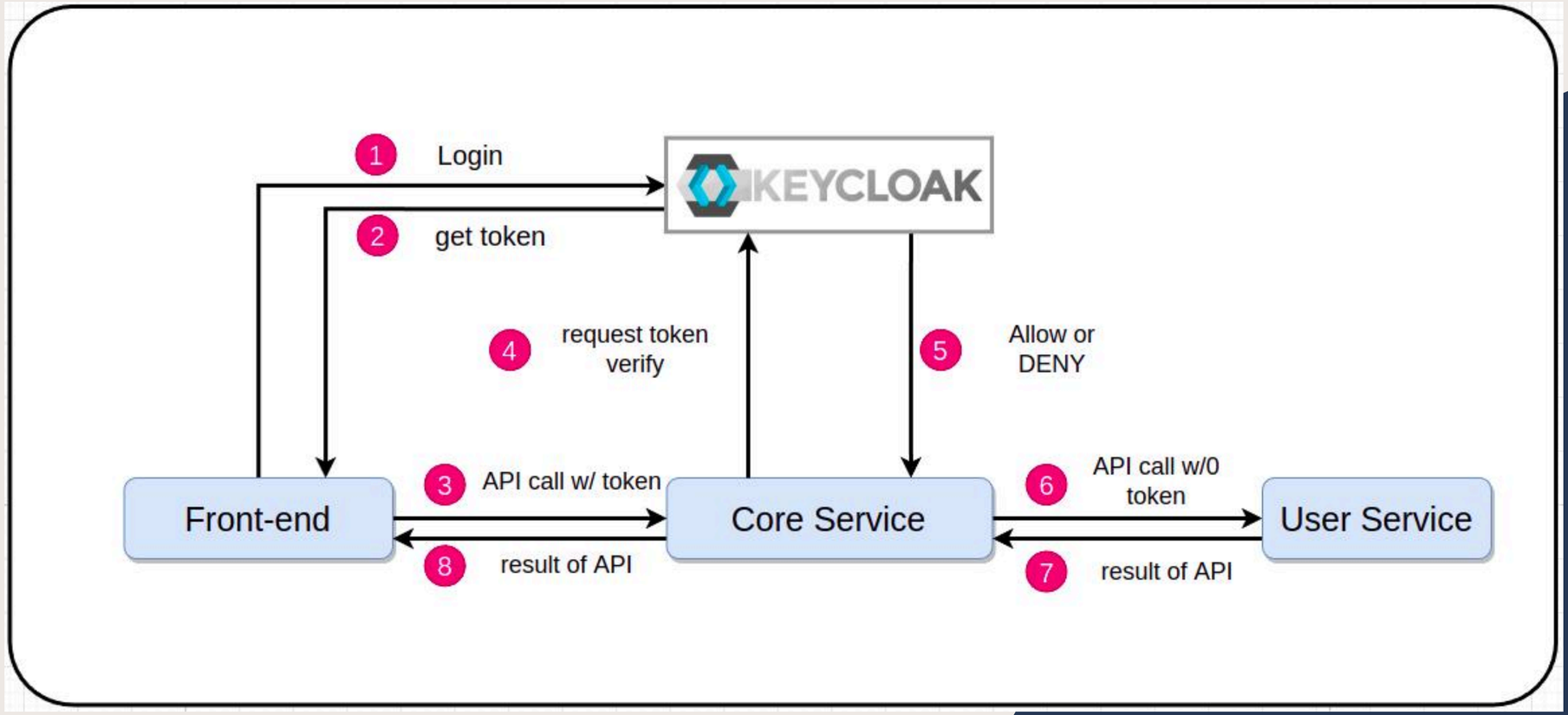


FLUJO DE AUTENTICACIÓN

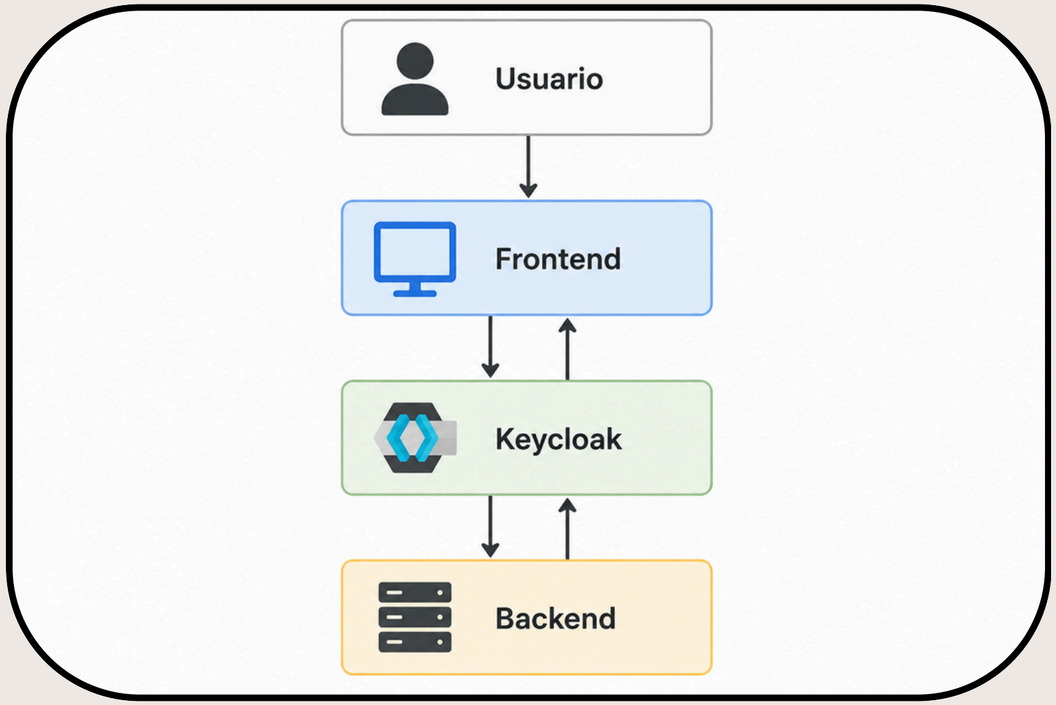


- **AUTENTICACIÓN** = ¿QUIÉN ERES?
- **AUTORIZACIÓN** = ¿QUÉ PUEDES HACER?

ARQUITECTURA DE KEYCLOAK



INTEGRACIÓN DE KEYCLOAK CON UNA APLICACIÓN



- Servicio externo de identidad
- Keycloak centraliza la gestión de acceso.
- Keycloak no reemplaza al frontend ni al backend. Se integra con ellos para centralizar la autenticación y la gestión del acceso.

1. FRONTEND → KEYCLOAK: CONFIGURACIÓN Y AUTENTICACIÓN

```
const keycloak = new Keycloak({
  url: "http://localhost:8080",
  realm: "demo",
  clientId: "keycloak-demo",
});

keycloak.init({
  onLoad: "login-required"
});
```

El frontend se conecta con Keycloak y solicita que el usuario se autentique antes de acceder a la aplicación.

2. FRONTEND → BACKEND: ENVÍO DEL ACCESS TOKEN

```
fetch("http://localhost:3000/api/profile", {
  headers: {
    Authorization: `Bearer ${keycloak.token}`
  }
});
```

Después de autenticarse, el frontend utiliza el Access Token **emitido por Keycloak** para identificarse ante el backend y acceder a un recurso protegido.

SEGURIDAD **DE LA INTEGRACIÓN**

Tokens firmados

Permiten verificar que la información de autenticación no fue alterada.
(fuente confiable y contenido no fue modificado.)

Expiración de tokens

Limita el tiempo durante el cual una credencial puede utilizarse.

MFA (Multi-Factor Authentication)

Añade una segunda capa de verificación de identidad.

REFERENCIAS

Keycloak. Keycloak Documentation. https://www.keycloak.org/documentation

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749, IETF. https://www.rfc-editor.org/rfc/rfc6749

MUCHAS GRACIAS