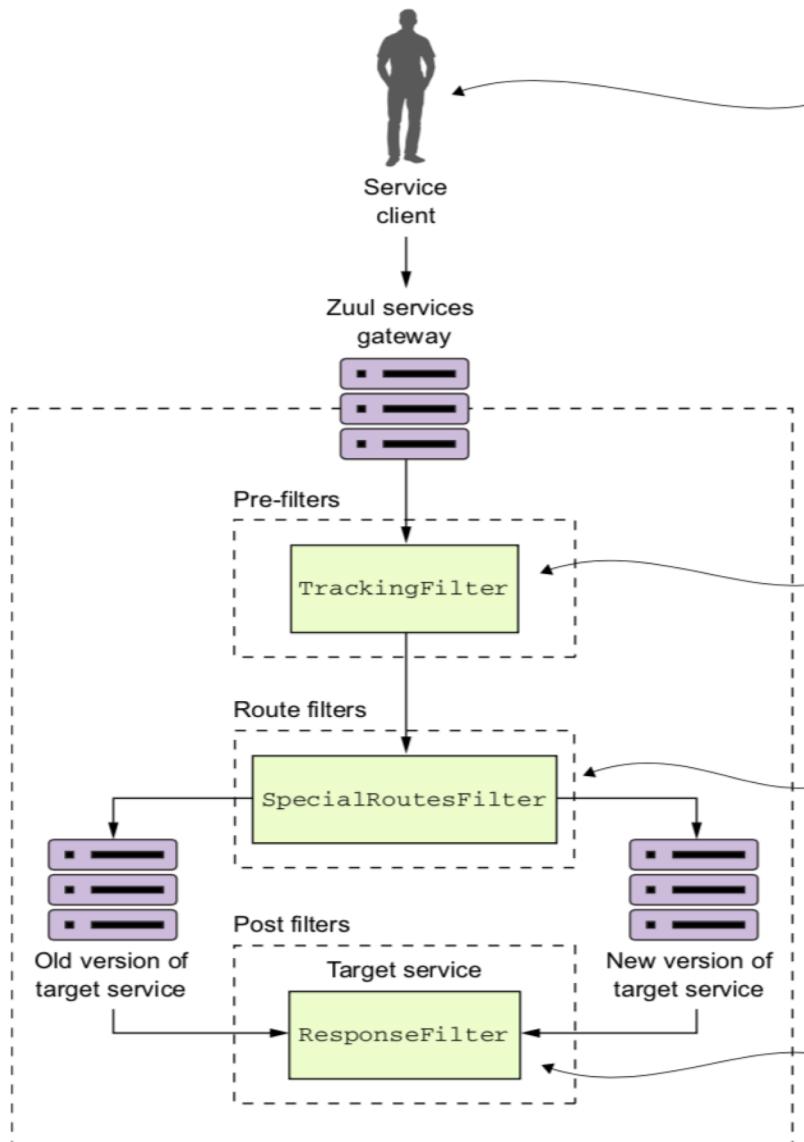


Arquitectura e Implementación de Microservicios

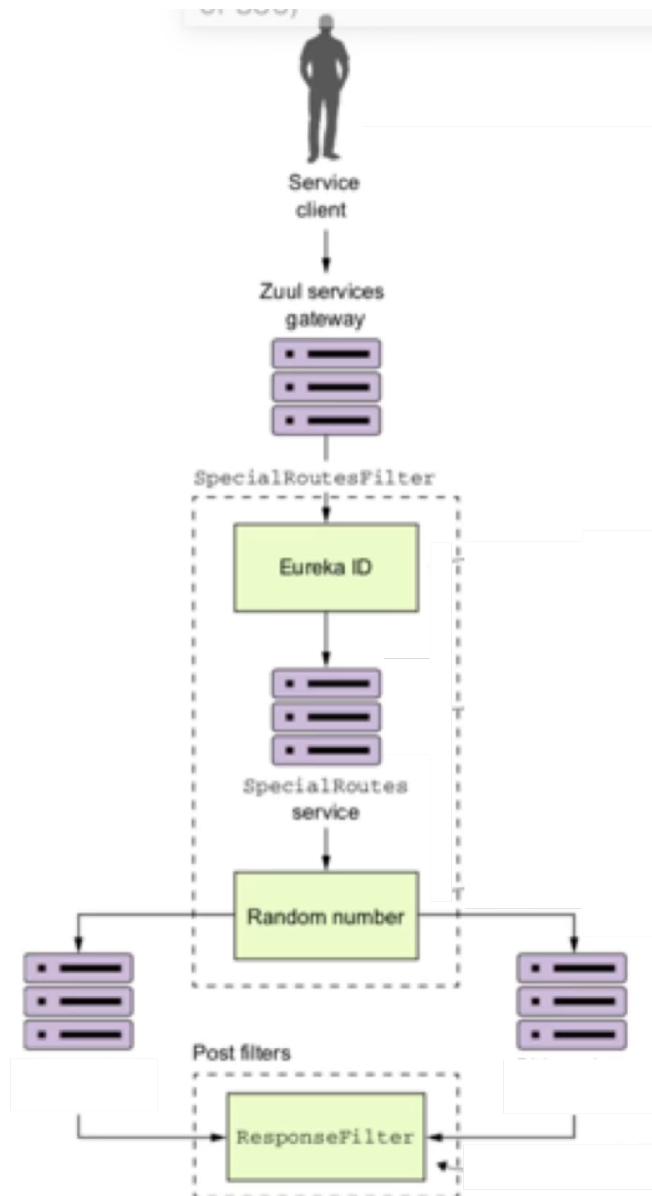
API Gateway con Netflix Zuul

Filters de ejemplo



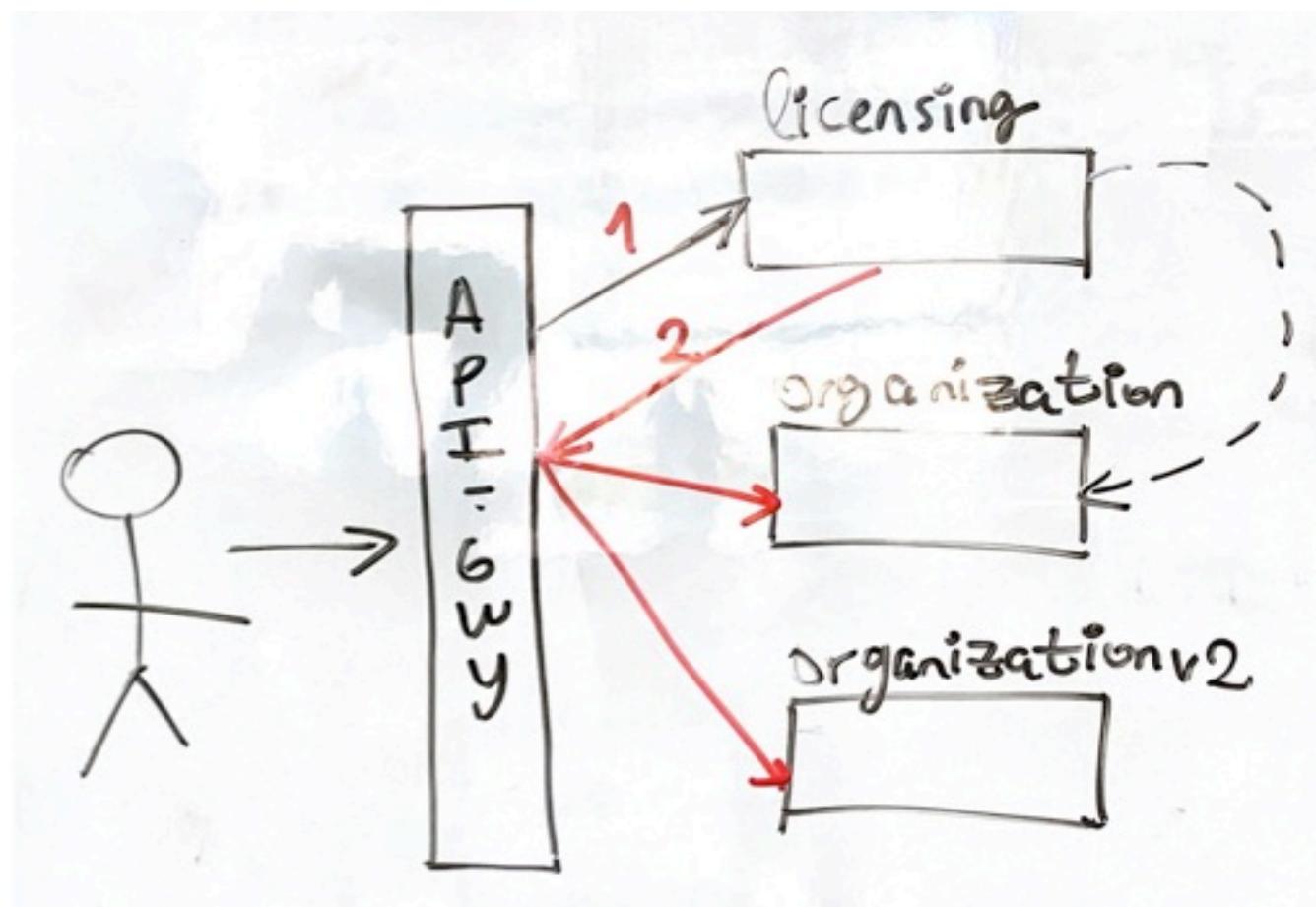
Route filter

- Ruteo inteligente
 - A/B testing / despliegue progresivo
 - 50% nueva version
 - 50% Antigua version

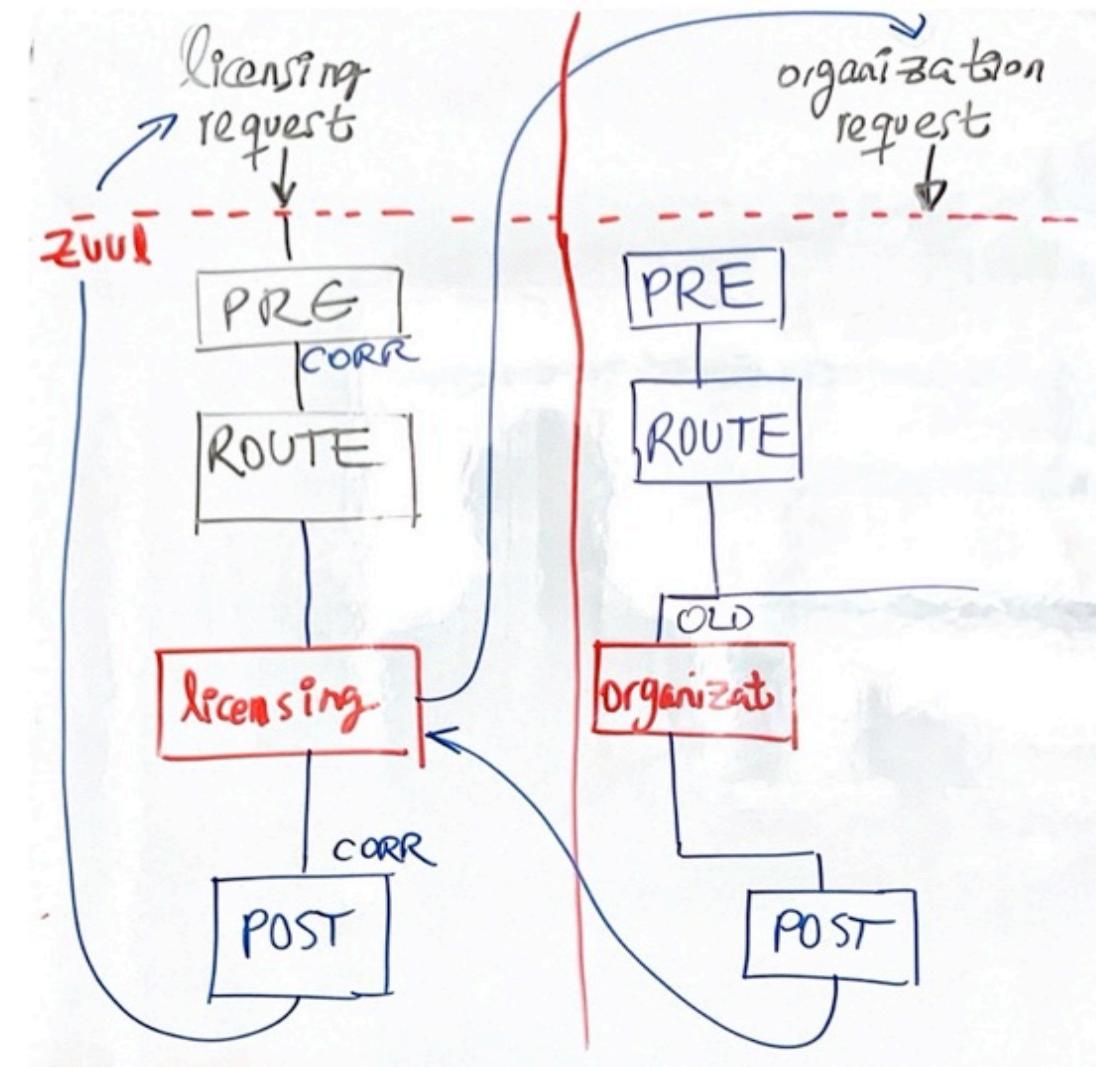


Laboratorio Route Filter

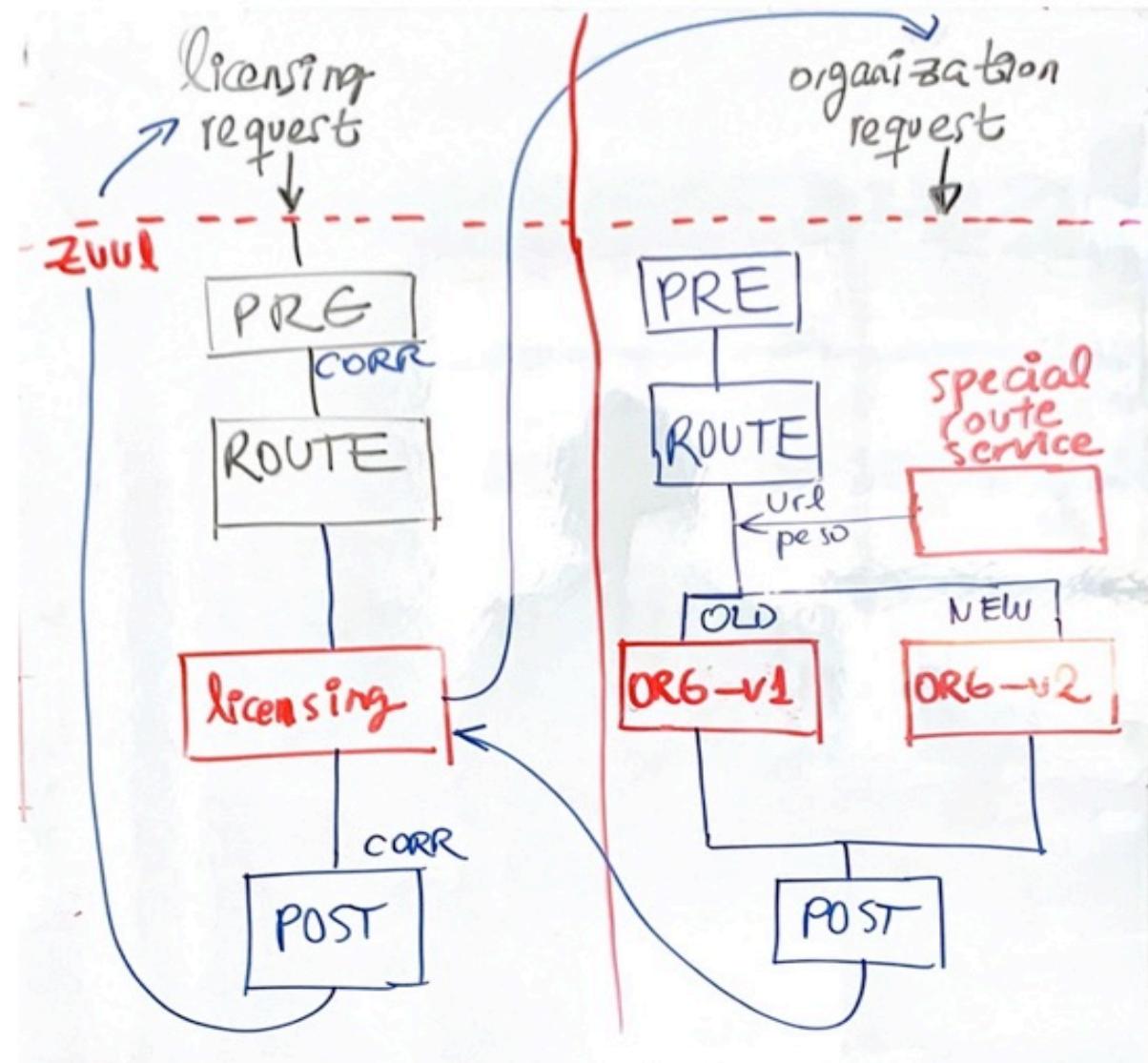
Arquitectura ejemplo



Flujo de invocación



Flujo de invocación completo



Seguridad en microservicios con Oauth2

Capas de protección

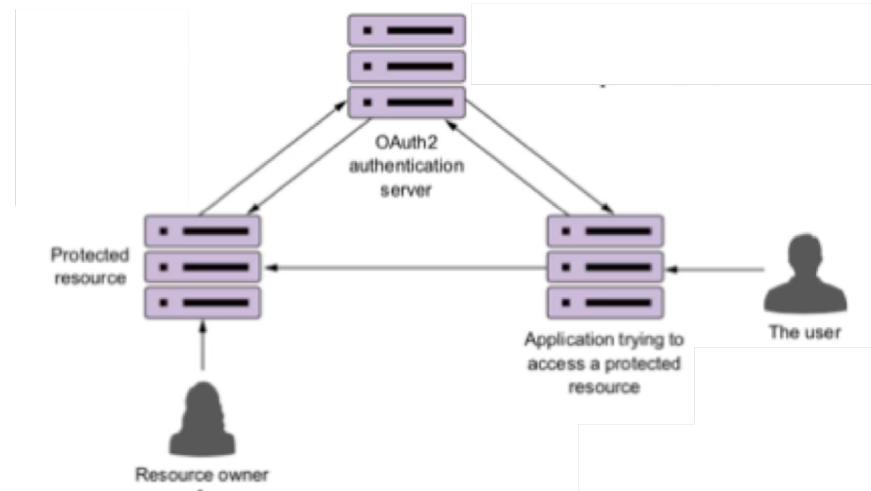
- Controles de usuario que validan:
 - Quien?
 - Tiene permisos?
- Infraestructura actualizada
 - Minimizar vulnerabilidades
- Controles de acceso
 - Puertos bien definidos
 - Servers autorizados
- Alcance:
 - Autenticacion
 - Autorizacion
 - Spring cloud security y ouath2

Oauth2

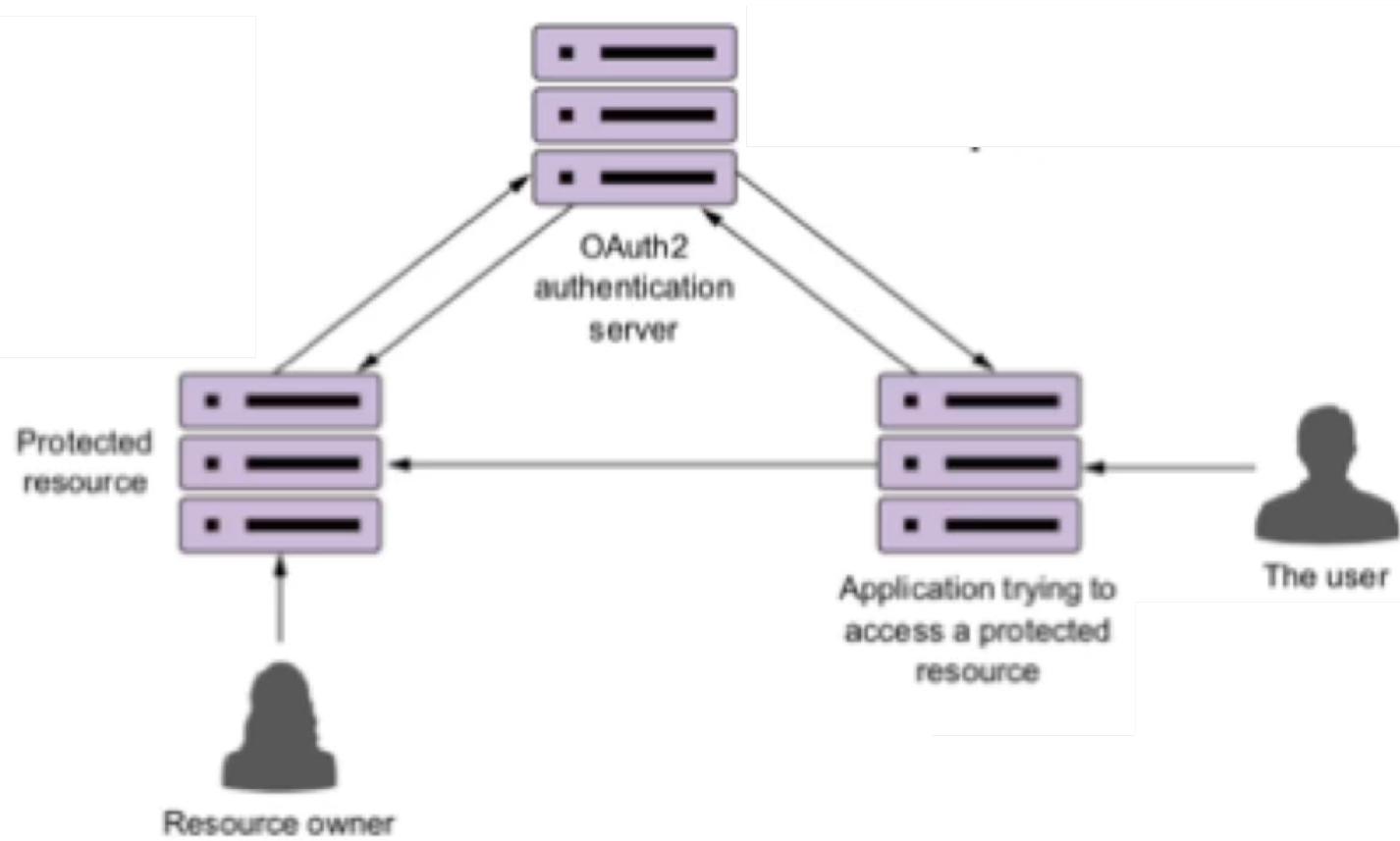
- Framework basado en tokens
 - Usuarios se autentica en servicio de autenticacion
 - Usuario se autentica. Recibe un token
 - Token se valida con servicio de autenticacion
- Objetivo
 - Multiples servicios autenticados sin credenciales cada vez
 - Single sign-on: SSO
 - Permite integrarse con Facebook, github,google sin login constant
- Spring cloud implementacion de un servidor oauth2.

Componentes

- Recurso protegido
- Resource owner
 - Aplicaciones
 - Usuarios y Acciones
 - app-name/app-secret
 - Parte de las credenciales
- Application
 - Llama al recurso en nombre del usuario
 - Usuarios no llaman directamente
 - Ejemplo: web-app, mobile app
- Oauth2 authentication server
 - Intermediario application y recurso protegido
 - Permite usuario autenticarse



Autenticacion de usuario

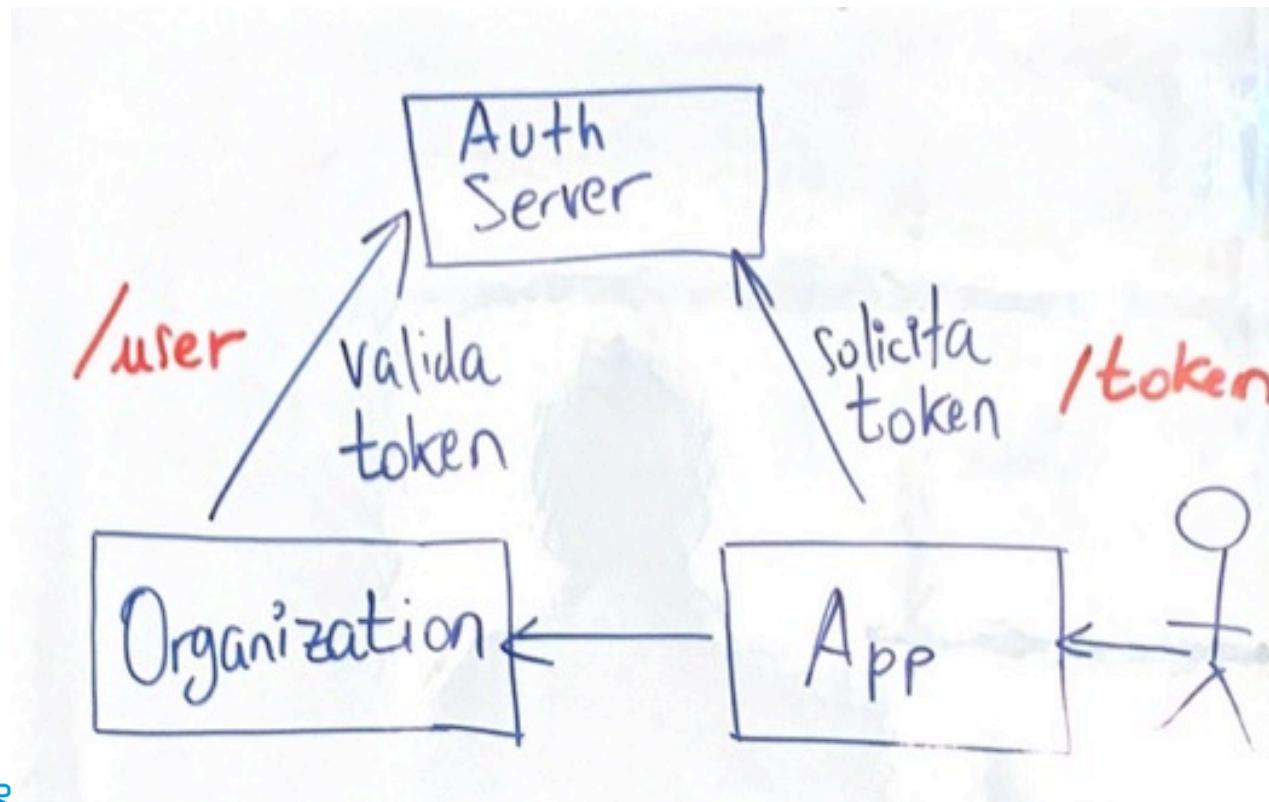


Grants

- Esquemas de seguridad: Grants
 - Password
 - Client credential
 - Authorization code
 - implicit

Laboratorio

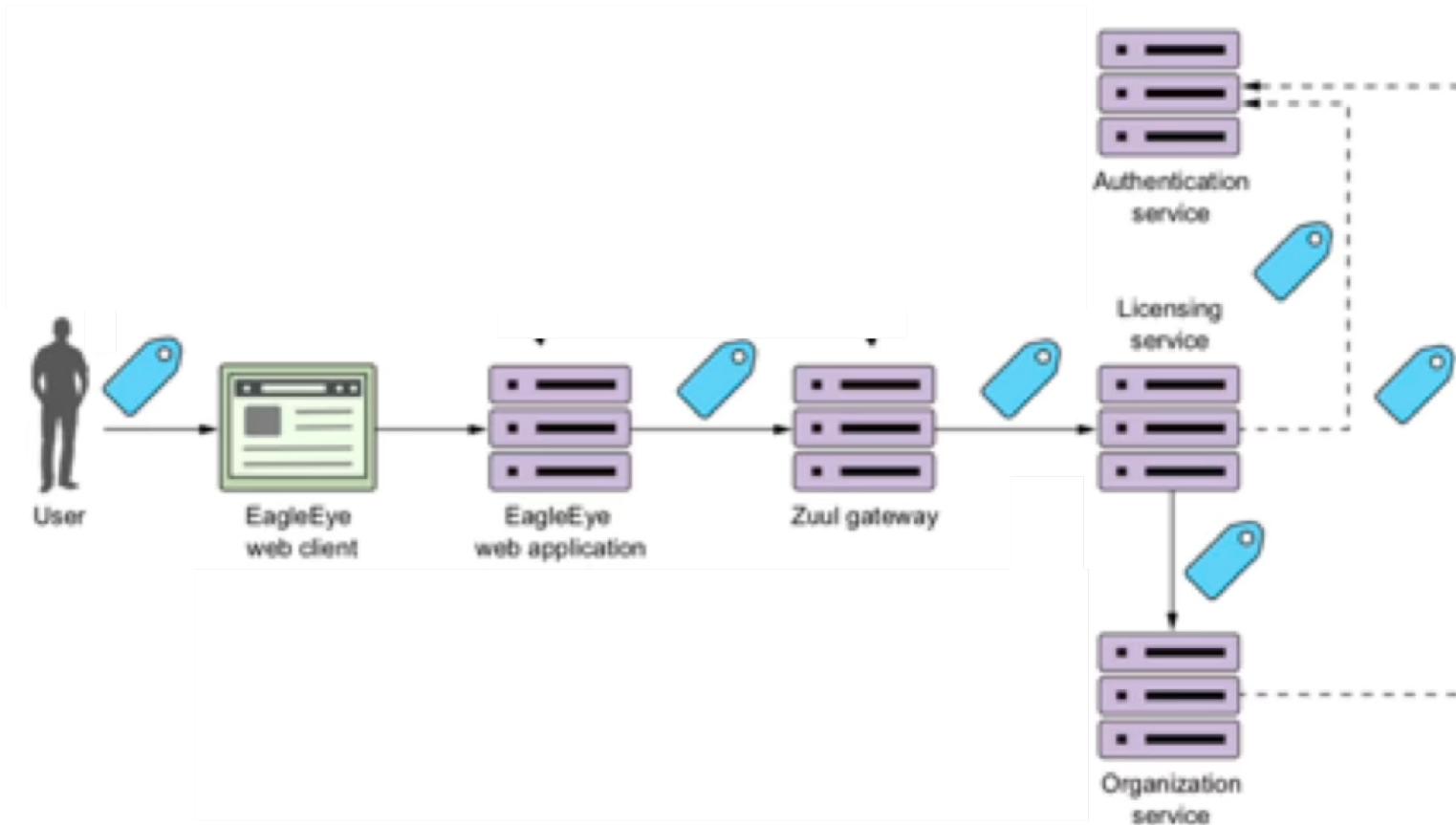
Security endpoints



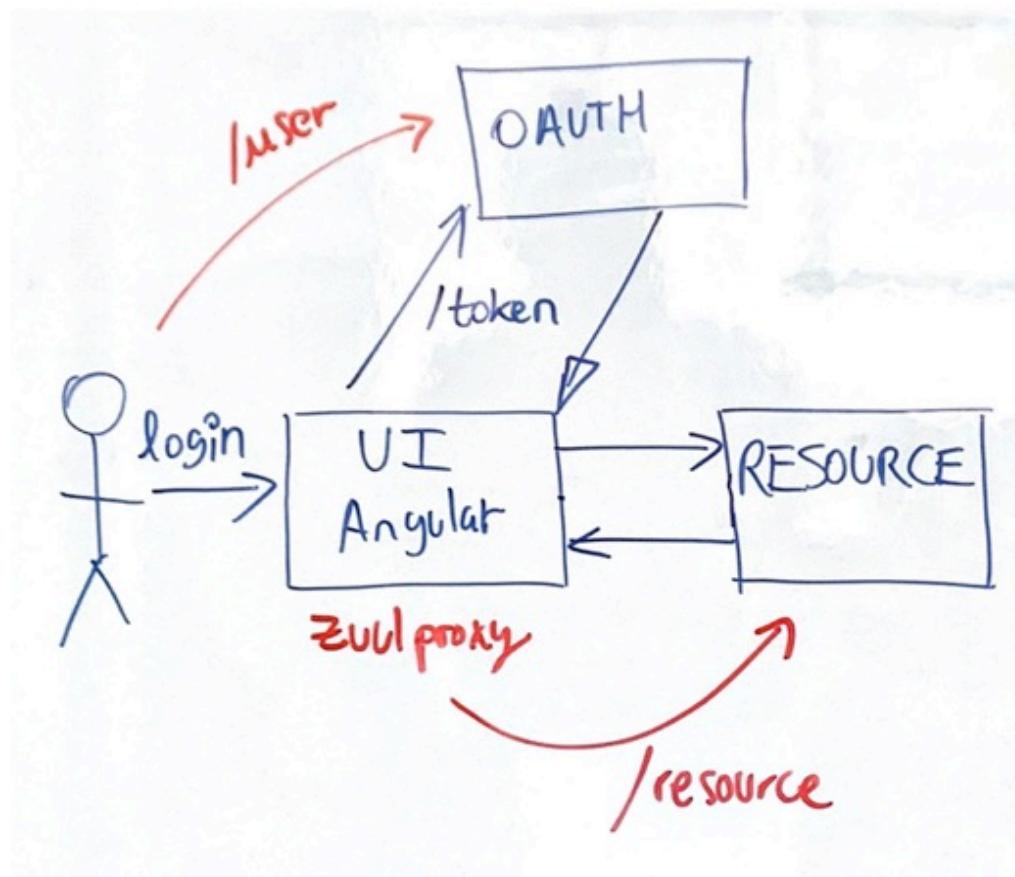
Spring security expressions

- .antMatchers("/auth/admin/*").hasRole("ADMIN")
- .antMatchers("/auth/*").hasAnyRole("ADMIN","USER")
- antMatchers("/*").permitAll() : usuarios anonimos y autenticados
- antMatchers("/*").denyAll() :
- antMatchers("/*").authenticated() : autenticados
- antMatchers("/*").anonymous(): anonimos

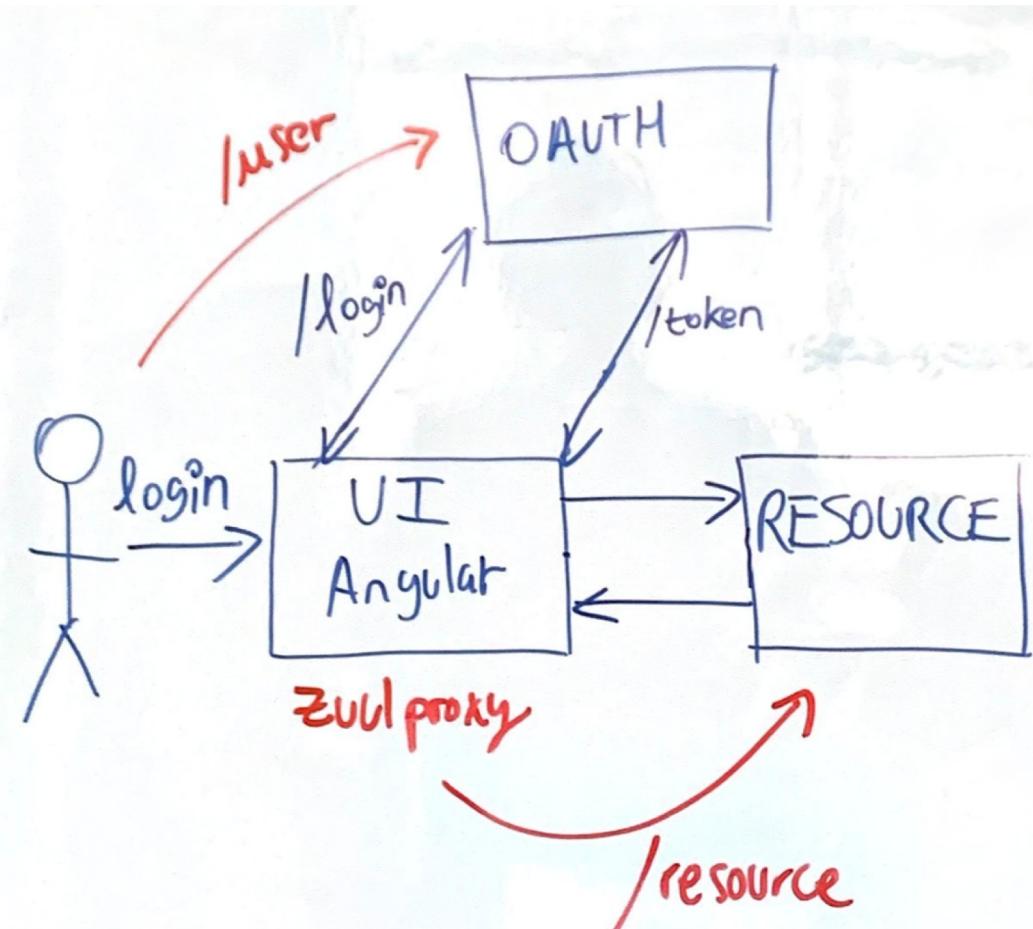
Propagando token auth2



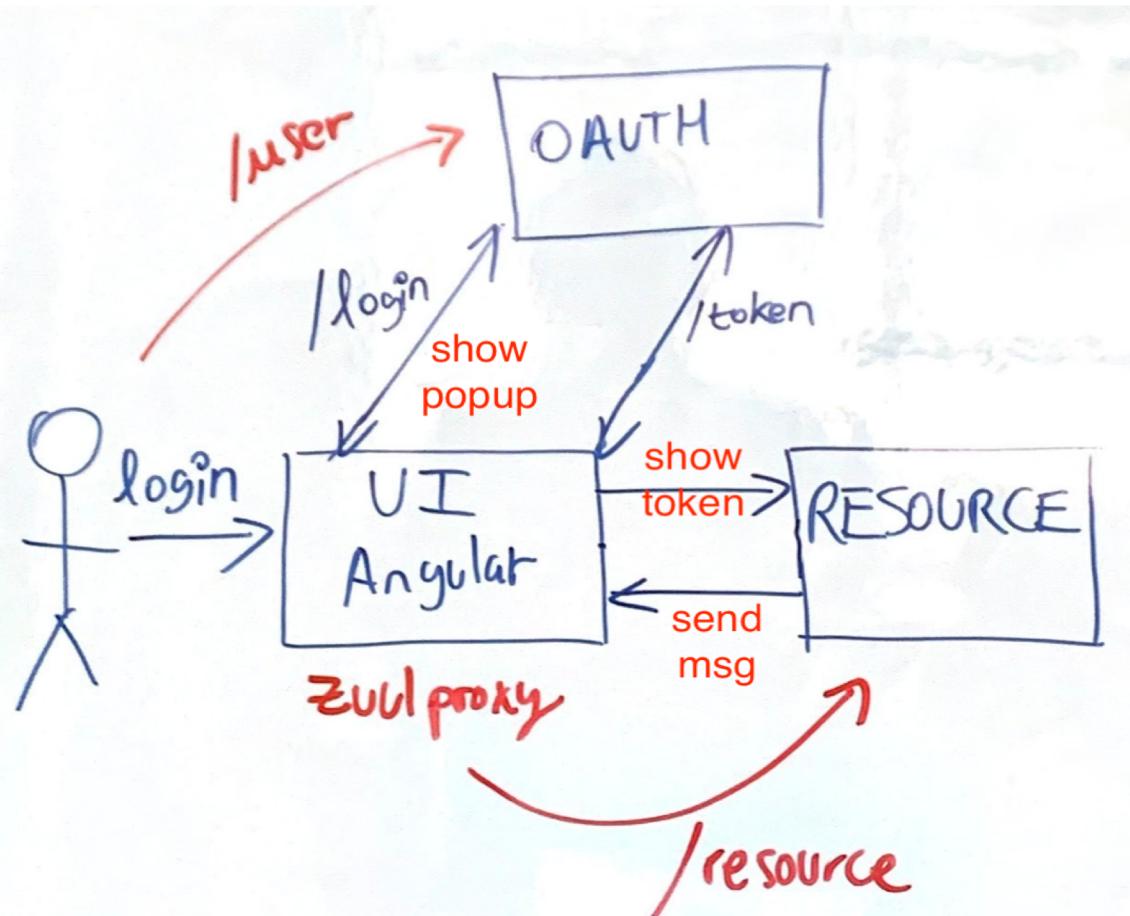
Autenticacion con frontend



Oauth2 Client



Popup de autenticacion



Seguridad en microservicios con Oauth2/JWT

JWT y oauth2

- Oauth2 no hay estandar para token
 - Json web tokens: JWT
- JWT estructura para oauth2 tokens
- Caracteristicas:
 - Pequeños
 - Base64. pasados por http url,header,parameters
 - Firmados criptograficamente
 - Firmado por authentication server
 - Garantiza no esta alterado
 - Auto-contenido
 - Microservicio sabe token es valido con la firma.
 - no llama a authentication server para validar
 - Token validado solo por el microservicio
 - Simplifica autenticacion
 - Extensible
 - Informacion adicional al generar token
 - Microservicio decodifica y recupera informacion

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4

JWT

- Usos:
 - Autorizacion
 - comun
 - Intercambio de informacion
 - No alterado
- Estructura
 - Header
 - Tipo
 - Algoritmo firma
 - Payload
 - Claims: data acerca del usuario
 - Signature/firma
 - Verifica contenido no alterado
 - Cifrado utiliza una clave

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

header

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

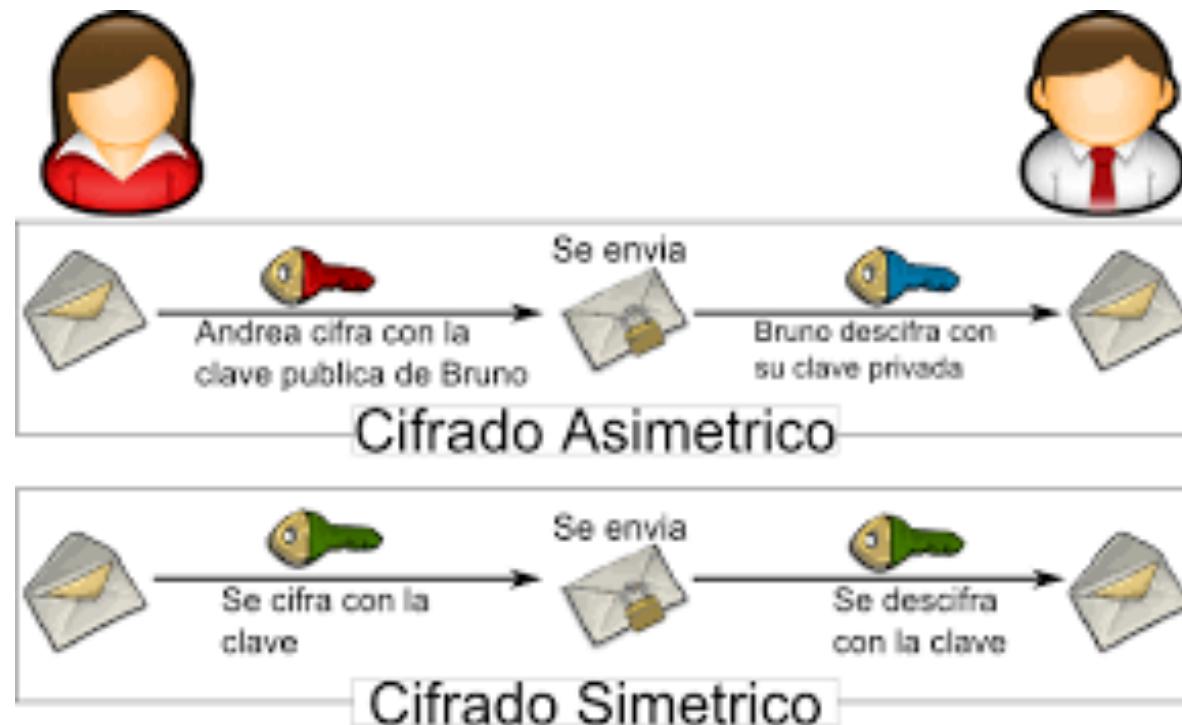
payload

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret)
```

signature

Ciprado en JWT

- 2 tipos de cifrado
 - Asimetrico
 - Simetrico



JWT

- 3 cadenas en base64 separados por .

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
- Jwt firmados: no pueden alterarse , pero si leidos
 - no poner secretos en header y payload
 - Token encriptados: JWE
 - <https://tools.ietf.org/html/rfc7516>

Autenticacion con JWT

