Restringindo o acesso a uma API utilizando o KeyCloak

Utilizando Spring Framework


KEYCLOAK


spring


PostgreSQL


OAUTH
2


POSTMAN


docker


OpenID®

OAuth2

The OAuth 2.0 Authorization Framework - RFC 6749

OAuth 2.0 é o protocolo padrão do para fluxos de autorização sendo seus principais componentes:

- The Authorization Server
- The API: "Resource Server"
- The Third-Party Application: "Client"
- The User: "Resource Owner"

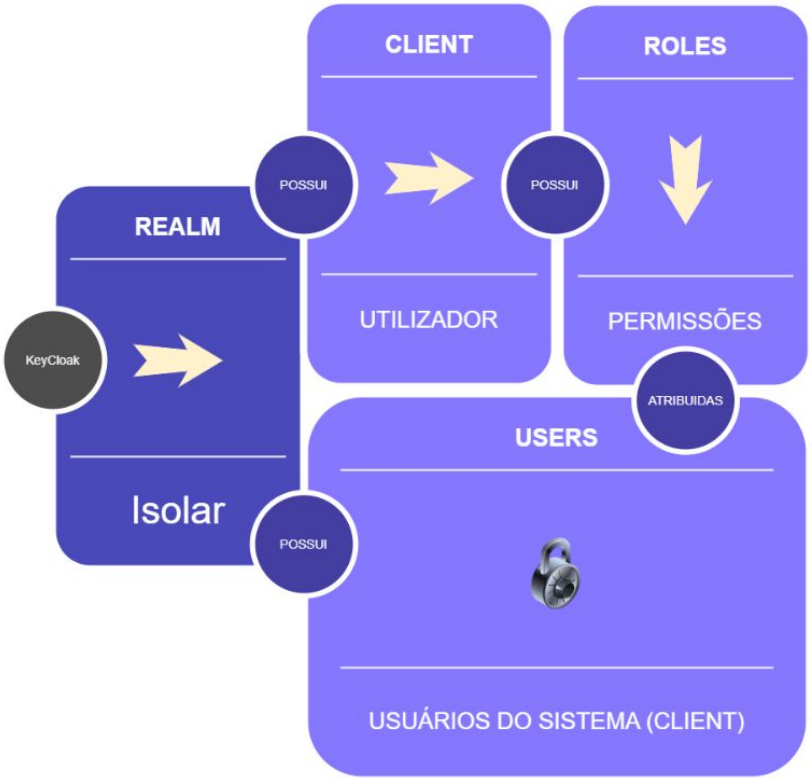
OpenID Connect

OpenID Connect é um protocolo de autenticação interoperável baseado na estrutura de especificações OAuth 2.0, adicionando algumas facilidades como:

- ID Token
- Discovery Endpoint
- UserInfo Endpoint
- Hybrid Authorization Flow

OAuth 2.0 é uma estrutura especificada pela IETF para apoiar o desenvolvimento de protocolos de autenticação e autorização. Ele fornece uma variedade de fluxos de mensagens padronizados baseados em JSON e HTTP; OpenID Connect os usa para fornecer serviços de identidade.

KeyCloak



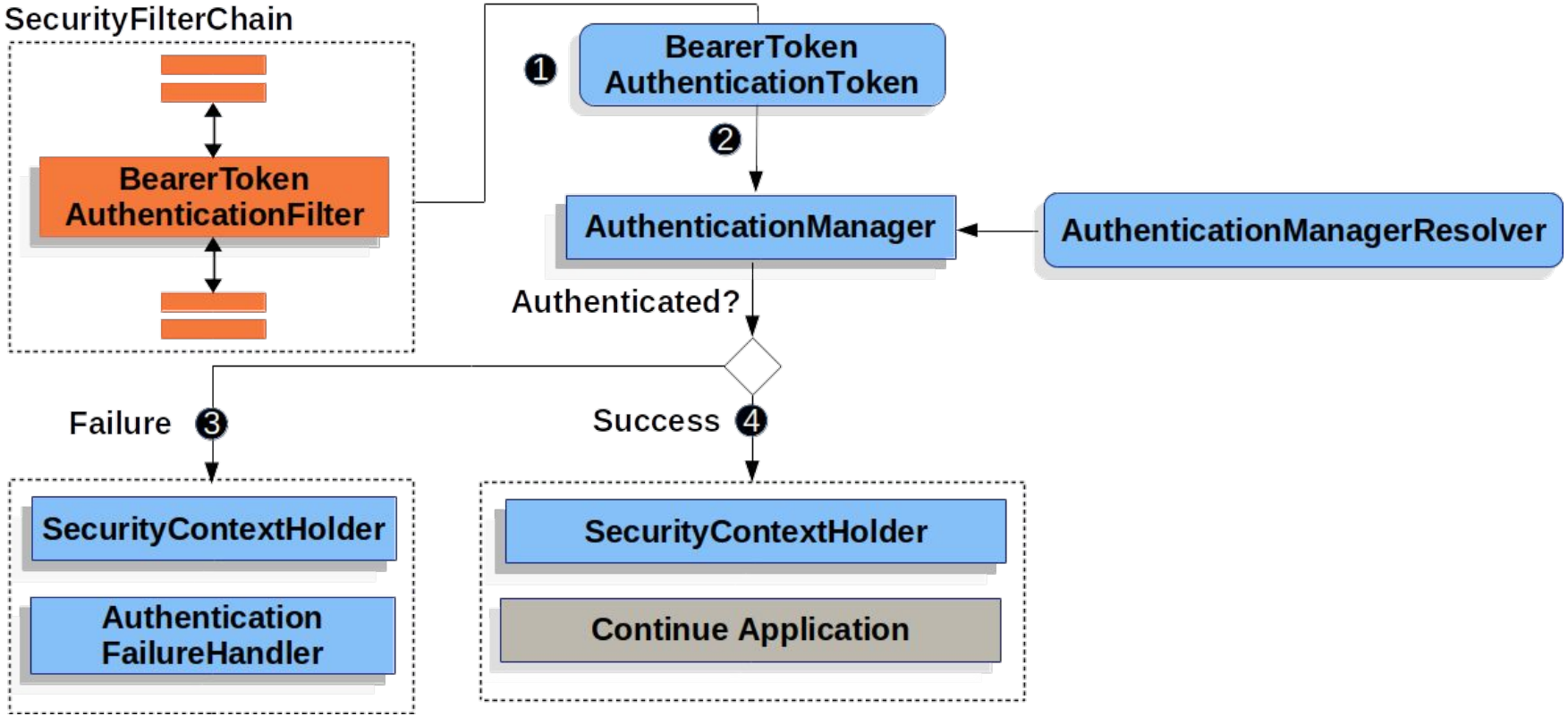
REALM - Espaços isolados onde serão criadas as políticas de segurança e as definições para autenticação/autorização.

CLIENT - representação de uma aplicação que utilizará dos recursos de autenticação/autorização.

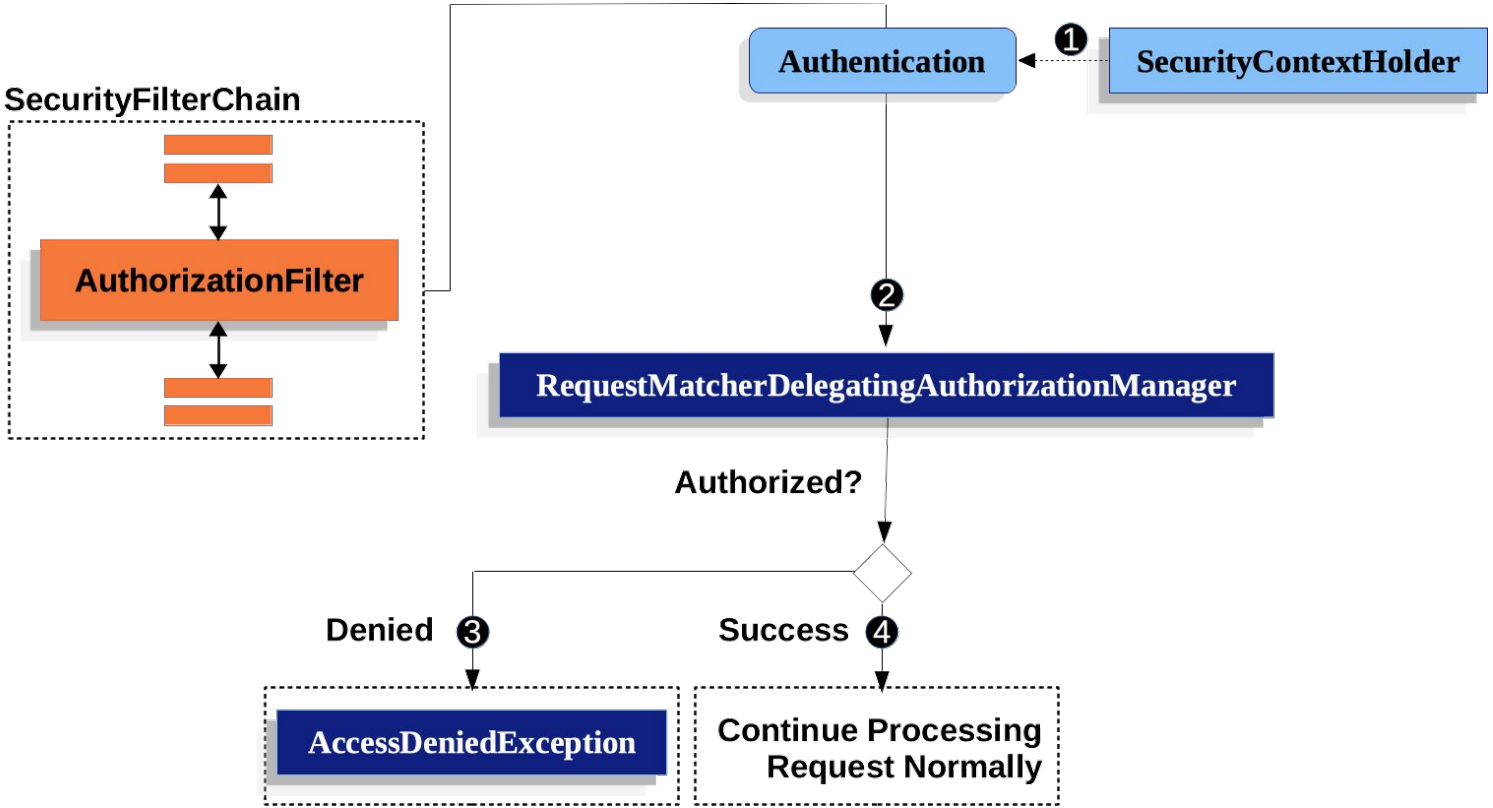
ROLES - representação da autorização que determinado usuário poderá receber.

USERS - usuários que efetivamente irão utilizar do sistema de autenticação/autorização.

Spring Security - OAuth2 Resource Server



Spring Security



Securing GET /api/foo

```
Trying to match request against DefaultSecurityFilterChain
[RequestMatcher = any request, Filters = [
        org.sf.security.web.session.DisableEncodeUrl
        org.sf.security.web.context.request.async.WebAsyncManagerIntegration
        org.sf.security.web.context.SecurityContextHolder
        org.sf.security.web.header.HeaderWriter
        org.sf.security.web.authentication.logout.Logout
        org.sf.security.oauth2.server.resource.web.authentication.BearerTokenAuthentication
        org.sf.security.web.savedrequest.RequestCacheAware
        org.sf.security.web.servletapi.SecurityContextHolderAwareRequest
        org.sf.security.web.authentication.AnonymousAuthentication
        org.sf.security.web.session.SessionManagement
        org.sf.security.web.access.ExceptionTranslation
        org.sf.security.web.access.intercept.Authorization
    ]
]
```

```
Securing GET /api/foo
Invoking DisableEncodeUrlFilter (1/12)
Invoking WebAsyncManagerIntegrationFilter (2/12)
Invoking SecurityContextHolderFilter (3/12)
Invoking HeaderWriterFilter (4/12)
Invoking LogoutFilter (5/12)
Invoking BearerTokenAuthenticationFilter (6/12)
Invoking RequestCacheAwareFilter (7/12)
Invoking SecurityContextHolderAwareRequestFilter (8/12)
Invoking AnonymousAuthenticationFilter (9/12)
Invoking SessionManagementFilter (10/12)
Invoking ExceptionTranslationFilter (11/12)
Invoking AuthorizationFilter (12/12)
Secured GET /api/foo
```

src/main/java/br/edu/biblioteca/entities/Login.java

src/main/java/br/edu/biblioteca/entities/Usuario.java
src/main/java/br/edu/biblioteca/entities/Perfil.java

src/main/java/br/edu/biblioteca/repositories/UsuariosRepository.java
src/main/java/br/edu/biblioteca/repositories/PerfisRepository.java

src/main/java/br/edu/biblioteca/services/AutenticacaoService.java
src/main/java/br/edu/biblioteca/services/TokenService.java

src/main/java/br/edu/biblioteca/configurations/AutenticacaoFilter.java

Pós-Graduação Engenharia DevOps - IFMT

Alunos:

Alison Sacal Lima

Bruno de Oliveira Figueiredo

Edinei Nissola

Eduardo Ormond dos Santos

Francisco Jose Prata Vidal

Jefferson Gonçalves de Oliveira Reis

Fontes:

https://www.keycloak.org/

https://oauth.net/2/

https://openid.net/developers/how-connect-works/

https://docs.spring.io/spring-security/reference/