Residência em Software

Módulo Programação JAVA (Avançado)

MÊS 01

INSTITUIÇÃO EXECUTORA
cepedi UESC

COORDENADORA
MCTI FUTURO Softex

APOIO
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Autenticação JWT com **Spring Security**

Spring Security

Spring Security é um framework poderoso e altamente personalizável que gerencia a autenticação e autorização em aplicações Java. Quando utilizado com outras bibliotecas como Spring Boot, Spring MVC, e o JJWT (uma biblioteca popular para criar e verificar JWTs), Spring Security pode facilmente implementar autenticação baseada em token.

Autenticação JWT com **Spring Security**

Spring Security - Autenticação via token

Integração com JWT:
Spring Security não possui um suporte in-built para JWT, mas pode ser facilmente integrado com bibliotecas externas como JJWT para gerenciar a criação e verificação de tokens JWT. Esta integração é usada para implementar Stateless Session Management, onde o servidor não precisa manter o estado da sessão do usuário.

Spring Security - Autenticação via token

Customização de Filtros:
Spring Security permite a customização de filtros de segurança. Por exemplo, você pode criar um filtro personalizado para interceptar requisições, extrair o JWT do cabeçalho Authorization, e autenticar o pedido antes de ele chegar ao seu controlador.

Autenticação JWT com **Spring Security**

Spring Security - Autenticação via token

Configuração Declarativa:
Usando anotações e configuração baseada em Java, você pode declarar regras de segurança aplicadas automaticamente a suas rotas/APIs. Isso inclui especificar URLs que podem ser acessadas sem autenticação, ou que requerem um usuário autenticado com certos papéis.

Spring Security - Autenticação via token

Suporte para OAuth2 e OpenID Connect:
Além de JWT, Spring Security oferece suporte extensivo para OAuth2 e OpenID Connect, padrões modernos para autenticação e autorização que também utilizam tokens para segurança.

Autenticação JWT com **Spring Security**

Spring Security - Frameworks alternativos

Embora Spring Security seja a escolha mais popular, existem outras ferramentas e frameworks que também podem ser usadas para implementar autenticação via token em aplicações Java:

- **Apache Shiro:** Uma alternativa poderosa e flexível para Spring Security, oferece suporte a autenticação, autorização, criptografia e gerenciamento de sessão.
- **Keycloak:** Keycloak é uma solução de Identity and Access Management que pode ser usada em conjunto com Java applications para gerenciar autenticação e autorização, incluindo o suporte a tokens.

Autenticação JWT com **Spring Security**

Spring Security - Frameworks alternativos

- **Auth0 Java SDK:** Para desenvolvedores que preferem usar uma solução como serviço (SaaS), Auth0 oferece SDKs que podem ser integrados com aplicações Java para lidar com autenticação e autorização usando tokens.

Escolher entre essas opções depende das necessidades específicas do projeto, experiência da equipe e a complexidade da infraestrutura de segurança desejada.

Autenticação JWT com **Spring Security**

O que é um AUTH token?

Um AUTH token (token de autenticação) é uma ferramenta usada em ambientes de programação web para gerenciar a autenticação e a autorização de usuários e sistemas. Esse token geralmente é um conjunto de caracteres que serve como credencial, substituindo nomes de usuário e senhas tradicionais.

Autenticação JWT com **Spring Security**

Processo típico de uma autenticação de API com AUTH token

1. Autenticação do Usuário
O usuário primeiro se autentica na aplicação utilizando um método convencional (como usuário e senha). Este passo é necessário para verificar se o usuário é quem ele diz ser.

Autenticação JWT com **Spring Security**

Processo típico de uma autenticação de API com AUTH token

2. Geração do Token
Após a autenticação ser bem sucedida, o servidor gera um token de autenticação. Este token é criado com base em informações do usuário, juntamente com um segredo que apenas o servidor conhece (isso pode ser feito usando algoritmos como HMAC ou RSA). O token pode incluir informações como:

Identidade do Usuário: Quem é o usuário.
Timestamp de Emissão: Quando o token foi emitido.
Prazo de Validade: Até quando o token pode ser usado.

Autenticação JWT com **Spring Security**

Processo típico de uma autenticação de API com AUTH token

3. Envio do Token ao Usuário
O token é então enviado para o usuário, normalmente como resposta de uma requisição HTTP. O usuário armazenará esse token, geralmente no navegador ou no aplicativo cliente.

Autenticação JWT com **Spring Security**

Processo típico de uma autenticação de API com AUTH token

4. Uso do Token
Para realizar futuras requisições à API, o usuário deve enviar esse token com a requisição. Isso geralmente é feito por meio de um cabeçalho HTTP (como Authorization: Bearer <token>). Este token serve como prova de autenticação e autorização.

Autenticação JWT com **Spring Security**

Processo típico de uma autenticação de API com AUTH token

5. Validação do Token
Quando o servidor recebe uma requisição com um token, ele verifica a sua validade antes de permitir acesso à API. O servidor pode verificar coisas como:

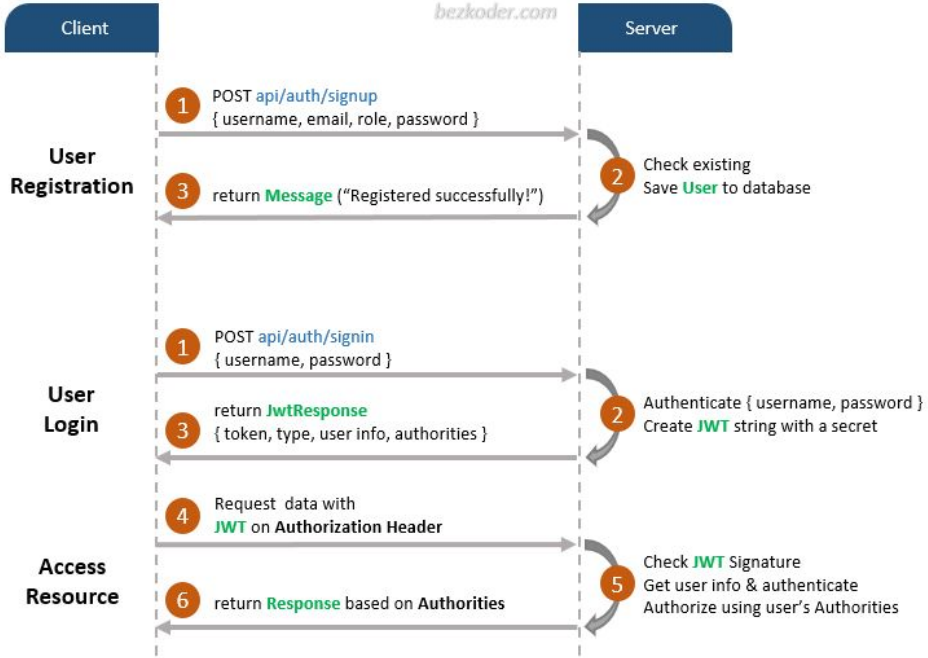
Integridade: Se o token não foi alterado.
Validade: Se continua no prazo de validade.
Origem: Se foi emitido pelo servidor.

Se o token for válido, o servidor processa a requisição. Se não, ele pode retornar um erro de autenticação.

Residência em Software

Autenticação JWT com **Spring Security**

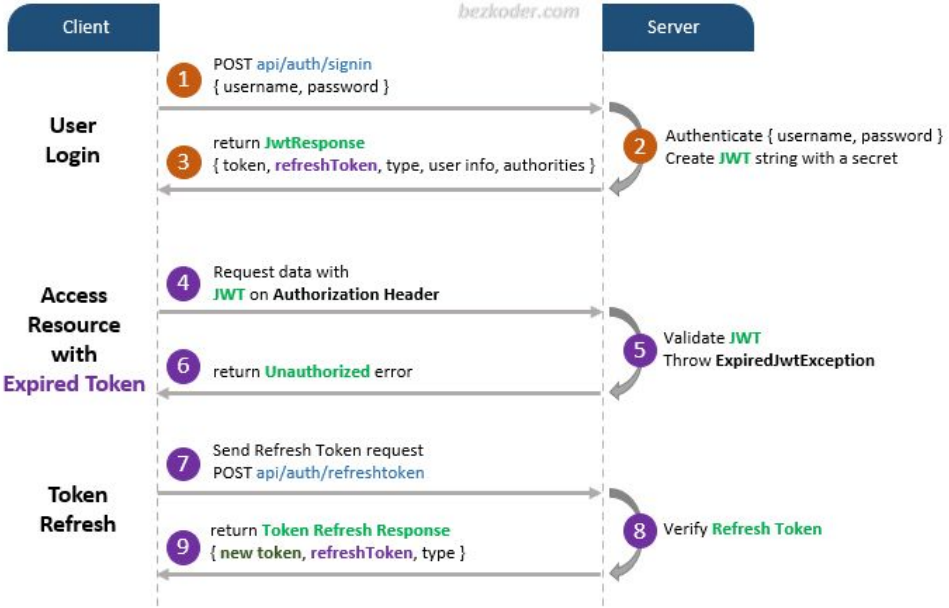
Processo típico de uma autenticação de API com AUTH token



Residência em Software

Autenticação JWT com **Spring Security**

Processo típico de uma autenticação de API com AUTH token



Autenticação JWT com **Spring Security**

Vantagens do Uso de Tokens em uma autenticação de API

- Segurança: Reduz o risco de exposição de credenciais.
- Estado: O token não requer que o servidor mantenha um estado, o que é ideal para APIs escaláveis.
- Desempenho: Reduz a necessidade de verificar a senha do usuário a cada requisição.
- Conveniência: Os tokens podem ser facilmente distribuídos entre diferentes sistemas e serviços.

Autenticação JWT com **Spring Security**

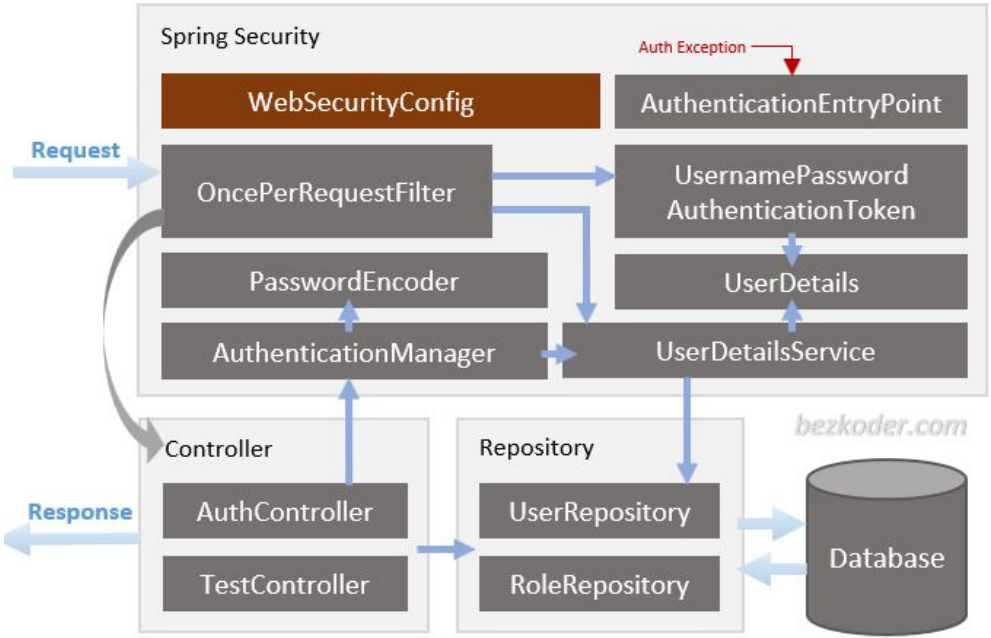
Spring Security - Processo para proteger uma API com token

- **Configuração do Spring Security:** Inicialmente, configura-se o Spring Security no projeto para usar JWT, definindo classes de segurança que incluem filtros e gerenciadores de autenticação.
- **Geração de Tokens:** Quando um usuário se autentica, um token JWT é gerado usando uma chave secreta. Este token enviado ao cliente contém informações do usuário.
- **Gerenciamento de Usuários e Papéis:** O sistema utiliza o **Spring Data JPA** para gerenciar usuários e seus papéis, permitindo diferentes níveis de acesso.
- **Autorização de Requisições:** Cada requisição subsequente deve incluir o token JWT no cabeçalho. O servidor verifica esse token antes de permitir acesso a recursos protegidos.

Residência em Software

Autenticação JWT com **Spring Security**

Spring Security - Autenticação JWT em uma API



Autenticação JWT com **Spring Security**

Spring Security - Autenticação JWT em uma API

Passo 1: Configuração Inicial
Dependências: Adicione as dependências de Spring Web, Spring Security, Spring Data JPA, e JWT.

Passo 2: Configuração de Segurança
WebSecurityConfig: Crie uma classe para configurar as regras de segurança, estendendo WebSecurityConfigurerAdapter.
AuthenticationManager: Configure o AuthenticationManager para usar um provedor de autenticação personalizado.

Spring Security - Autenticação JWT em uma API

Passo 3: Modelo de Usuário e Papéis
User e Role: Defina as entidades de usuário e papéis utilizando JPA.
UserDetails: Implemente a interface UserDetails para representar um usuário autenticado.

Passo 4: Autenticação e Geração de Token
AuthTokenFilter: Crie um filtro que verifica se o pedido possui um JWT válido.
JWTUtils: Desenvolva uma classe utilitária para gerar e parsear o JWT.

Autenticação JWT com **Spring Security**

Spring Security - Autenticação JWT em uma API

Passo 5: Endpoints de Autenticação
AuthController: Implemente um controlador para lidar com **login e registro**.
Login: Crie um endpoint que autentique os usuários e **retorne o JWT**.

Passo 6: Teste e Validação
Testar Endpoints: Escreva testes integrados e utilize ferramentas como **Postman** para testar os endpoints de autenticação.
Segurança: Verifique se os acessos estão corretamente restritos com base no token JWT.
Cada etapa envolve configurações específicas e codificação detalhada.

Autenticação JWT com **Spring Security**

Exemplo de implementação

pom: pom.xml

- https://gist.github.com/rog3r/d3f7d3addfb76e7cc27611ef564ef53a

config:

- https://gist.github.com/rog3r/a4de4185a2eddff3de15f479f3c9f6d0
- https://gist.github.com/rog3r/accb5d77070e692e6d1dbf97203ce71d
- **controller:**
- https://gist.github.com/rog3r/8fde871f5cc7e701d09316b48849c499
- https://gist.github.com/rog3r/d4a374e4cfa4d1cdfe7b19cfda1596fd

Autenticação JWT com **Spring Security**

Exemplo de implementação

- **mapper:**
- https://gist.github.com/rog3r/695dc9f5aade7212a2419aef57148ea0
- **dto:**
- https://gist.github.com/rog3r/aee2fa03eb15af785500a5fbf4243f75
- https://gist.github.com/rog3r/53984d774fc596ef8bc991e9f39daf29
- https://gist.github.com/rog3r/a8551b156fd0610291713abf4ee7a7a9
- https://gist.github.com/rog3r/acdb41ae106574b321f065fafe330dd8
- **entity:**
- https://gist.github.com/rog3r/65e5aa15dc04c6f30a17e5bc32af7e77

Autenticação JWT com **Spring Security**

Exemplo de implementação

- **repository:**
- https://gist.github.com/rog3r/2ed6b79193c8f7c58abaf9c4d815e68c
- **security:**
- https://gist.github.com/rog3r/78141a0e97e4ceb3bac7929e02131a88
- **service:**
- https://gist.github.com/rog3r/8fa59b23dfce05dd851e45d56ee544f5
- https://gist.github.com/rog3r/b532cc00144745863b4418152dde8780
- **validation:**
- https://gist.github.com/rog3r/b85e222a4fc3ec85b8cbb76700330710
- https://gist.github.com/rog3r/f013b142e781f1d393ed64335e4acf0b

Autenticação JWT com **Spring Security**

Exemplo de implementação

- **db/migrate:**
- https://gist.github.com/rog3r/ac5437c89dc346893fb5e1e77327c4f7

- **teste:**
- https://gist.github.com/rog3r/ac5437c89dc346893fb5e1e77327c4f7

arquivos com credenciais:
https://drive.google.com/file/d/1cDmIRuvnfEFhirh6R7x2r1xbpwHIsGdg/view?usp=sharing

Autenticação JWT com **Spring Security**

Exemplo de utilização - Postman

POST http://localhost:8080/api/auth/register

```
{
    "username":"rogerio",
    "password":"JAVA!@#ResTIC18",
    "email":"rogerio@gmail.com"
}
```

POST http://localhost:8080/api/auth/login

```
{
    "username":"rogerio",
    "password":"JAVA!@#ResTIC18"
}
```

Autenticação JWT com **Spring Security**

Exemplo de utilização - Postman

GET http://localhost:8080/api/dashboard
Authorization: "Bearer " +
eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwic3ViIjoicm9nZXJpbyIsImV4cCI6MTcxMzkxNTU3OSwiaWF0IjoxNzEzOTExOTc5LCJzY29wZSI6IlJPTEVfVVNFUiJ9.QkWaDo4T4hYWjIr940BveSfQIjcKXgDMFxyzLJwR5gU5YxksjSbBVOATlqJz-YFXP8epMMuKKFJ1xRdc8JNiBq879_nNubVPZSeBFIVAuvmxepUFGOkLARxKi9yHOYKs6IT_BPtNRmU7H_4HUmEvkMS3wemDoGhzFMDmw1uo-lrezBl4aK6VbsyO3NfBZM7o9swdaNF2RarzP8cvUhd6gl0jwKRLrWZWxrrYjypprGfq1KZcrRIXOlBFpZwzKoUtdb0WKcL1CP11kKDt6FARqBZyLwnMbnuOb_kHA9cBaSwXVRmf0B69y3ycX-vylFLPDmJLzQjoxzvuwc7dmICSLQ"

Autenticação JWT com **Spring Security**

Exemplo de utilização - CURL

curl -X GET "http://localhost:8080/api/dashboard" -H "Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwic3ViIjoicm9nZXJpbyIsImV4cCI6MTcxMzkxNTU3OSwiaWF0IjoxNzEzOTExOTc5LCJzY29wZSI6IlJPTEVfVVNFUiJ9.QkWaDo4T4hYWjIr940BveSfQIjcKXgDMFxyzLJwR5gU5YxksjSbBVOATlqJz-YFXP8epMMuKKFJ1xRdc8JNiBq879_nNubVPZSeBFIVAuvmxepUFGOkLARxKi9yHOYKs6IT_BPtNRmU7H_4HUmEvkMS3wemDoGhzFMDmw1uo-lrezBl4aK6VbsyO3NfBZM7o9swdaNF2RarzP8cvUhd6gl0jwKRLrWZWxrrYjypprGfq1KZcrRIXOlBFpZwzKoUtdb0WKcL1CP11kKDt6FARqBZyLwnMbnuOb_kHA9cBaSwXVRmf0B69y3ycX-vylFLPDmJLzQjoxzvuwc7dmICSLQ"

Residência em Software

Contato

rogerio.jesus@cepedi.org.br

https://moodle.residenciatic18.cepedi.org.br/