Aula 8: Introdução à Segurança com Spring Security

September 18, 2025

1 Introdução

Nas aulas anteriores, implementamos boas práticas na API, incluindo tratamento de erros (códigos 400 e 404) e padronização de respostas com `ResponseEntity`. Agora, focaremos na segurança da API, abordando autenticação e autorização com o `Spring Security`. Esta aula apresenta os conceitos fundamentais e diagramas do processo de autenticação (via JWT) e autorização, preparando para as implementações práticas nas próximas aulas. As alterações futuras serão feitas no `Visual Studio Code` ou sua IDE preferida.

2 O que é o Spring Security?

O `Spring Security` é um módulo do ecossistema Spring dedicado à segurança em aplicações, incluindo APIs REST como a nossa. Ele oferece ferramentas para:

- **Autenticação**: Verificar a identidade do usuário (ex.: login com usuário e senha ou tokens).
- **Autorização**: Controlar o acesso a recursos com base em permissões.
- **Proteção contra ataques**: Mitigar vulnerabilidades como CSRF (Cross-Site Request Forgery) e clickjacking.

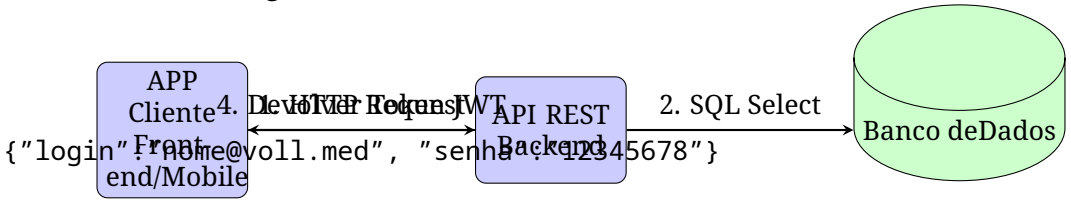
Na nossa API, o objetivo é proteger os endpoints (ex.: `/medicos`, `/pacientes`) para que apenas usuários autenticados e autorizados possam acessá-los, evitando que a API permaneça pública.

3 Autenticação vs. Autorização em APIs REST

Diferentemente de aplicações web tradicionais (stateful), que armazenam sessões no servidor, APIs REST são **stateless**, ou seja, não mantêm estado entre requisições. Em aplicações web, o servidor gerencia sessões com cookies, mas em APIs REST, usamos tokens (como JWT) para autenticação e autorização.

3.1 Autenticação com JWT

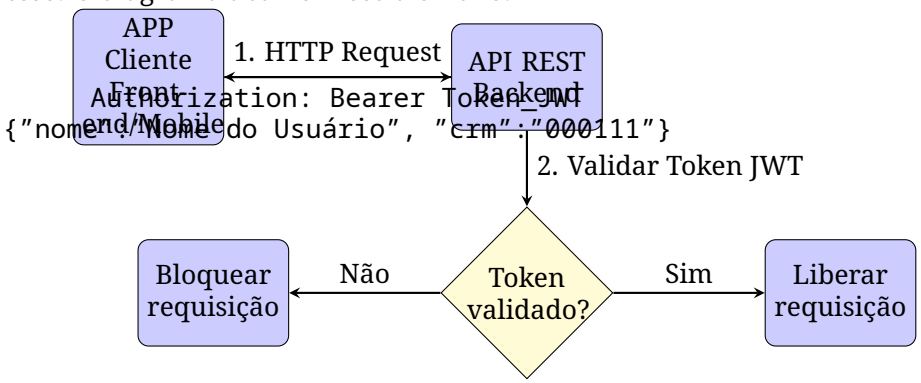
O processo de autenticação envolve verificar as credenciais do usuário e gerar um token JWT (JSON Web Token). O diagrama abaixo ilustra o fluxo:



1. O cliente (ex.: aplicativo mobile) envia uma requisição `POST` para a API (ex.: `/login`) com um JSON contendo `login` e `senha`.
2. A API consulta o banco de dados (tabela de usuários) para verificar as credenciais.
3. Se válidas, a API gera um token JWT (uma string codificada).
4. O token é retornado ao cliente, que o armazena para uso em requisições futuras.

3.2 Autorização com Validação de Token

Na autorização, a API verifica o token JWT em cada requisição para decidir se libera ou bloqueia o acesso. O diagrama abaixo mostra o fluxo:



1. O cliente envia uma requisição (ex.: `POST /medicos`) com o cabeçalho `Authorization: Bearer Token_JWT` e os dados no corpo.
2. A API valida o token JWT para confirmar sua autenticidade.
3. Se o token for inválido, a requisição é bloqueada (ex.: retorna 401 Unauthorized ou 403 Forbidden).
4. Se válido, a requisição é liberada, e o controlador processa os dados.

4 Importância da Segurança na API

Atualmente, nossa API é pública: qualquer pessoa com o endereço (ex.: `http://localhost:8080`) pode enviar requisições via Insomnia ou outro cliente. Após o deploy, isso representa um risco, pois endpoints como `/medicos` ou `/pacientes` podem ser acessados indevidamente. O `Spring Security` permitirá restringir o acesso apenas a funcionários autenticados da clínica, protegendo a API.

5 Próximos Passos

Nas próximas aulas, adicionaremos o `Spring Security` ao projeto, configurando autenticação com JWT (criação de uma tabela de usuários, endpoint de login e geração de tokens) e autorização (validação de tokens em requisições). Continue acompanhando no `Visual Studio Code` ou sua IDE preferida!

6 Dica do Professor

- **Aprofunde-se em Spring Security**: Consulte a documentação oficial do Spring Security (`https://spring.io/projects/spring-security`) para entender autenticação e autorização em APIs REST.

- **Comunidade no X**: Siga perfis como @SpringSecurityPro e @APIAuth no X para dicas sobre segurança em APIs e boas práticas com JWT.
- **Pratique**: Pesquise sobre o formato JWT (`https://jwt.io/introduction`) e como ele codifica informações do usuário, preparando-se para a próxima aula.