Feature 4: autenticação e autorização com Spring Security

Esta feature tem como objetivo blindar um projeto API RESTful, integrando os pilares de segurança: **autenticação** e **autorização**, utilizando o poderoso framework **Spring Security**. Mais do que apenas adicionar uma camada de segurança, a ideia é aplicar os princípios de design seguro, como o **Princípio do Menor Privilégio** e a **Segurança por Padrão**, para garantir que apenas usuários legítimos acessem o sistema e que suas ações sejam estritamente controladas por suas permissões.

Objetivo

Aplicar o Spring Security em um projeto individual existente (desenvolvido nas features anteriores ou em outra disciplina) para implementar: a) **Autenticação:** utilizando o método HTTP Basic; b) **Autorização:** implementando um controle de acesso baseado em Roles (ADMIN e USER) através de regras configuradas via URL/Request Matching, protegendo pelo menos dois "contextos" ou conjuntos de endpoints distintos dentro da API.

Desafio

Escolha do projeto base:

Selecionem um dos projetos desenvolvidos por vocês anteriormente (pode ser o projeto que já possui as features de TDD, Integração com APIs externas ou Modularização) para ser a base dessa implementação de segurança. O foco será adicionar as camadas de autenticação e autorização a esta API existente.

Configuração do Spring Security:

Para iniciar a proteção de sua API, o primeiro passo é configurar o Spring Security para gerenciar o acesso às suas aplicações. Esta etapa inicial é fundamental e estabelece a base sobre a qual construiremos as regras de autorização. Ela envolve a inclusão das dependências necessárias e a criação de uma classe central que orquestrará as políticas de segurança. Aqui, definiremos como os usuários serão autenticados e quais credenciais básicas serão utilizadas, para que possamos, então, avançar para as regras de autorização específicas: a) Incluam a devida dependência no arquivo `pom.xml` do projeto; b) Criem uma classe de configuração; c) Realize a devida configuração para que todas as requisições exijam autenticação e utilizem o mecanismo de autenticação HTTP Basic; d) Gerenciamento de usuários em memória: um usuário com a role `ADMIN`, outro com a role `USER`; garanta que as senhas sejam codificadas.

Implementação da autorização baseada em URL/Request Matching:

Após estabelecer a base de autenticação e definir os usuários fundamentais, o próximo passo crucial é implementar a autorização. Esta fase visa garantir que, mesmo após um usuário ser autenticado, ele só possa acessar os recursos e executar as operações para as quais possui permissão. Para isso, utilizaremos a poderosa abordagem de Autorização Baseada em URL/Request Matching do Spring Security, que permite definir regras de acesso de forma centralizada e granular para diferentes "contextos" da sua API: a) Selecionem pelo menos dois conjuntos de endpoints distintos dentro da API do projeto; b) Para cada contexto, configurem as regras de autorização.

Exemplos interessantes a serem utilizados nos seus contextos:

- Operações de criação, atualização e exclusão em `/api/vendedores/**` devem ser permitidas **apenas para usuários com a role `ADMIN`**. Operações de leitura devem ser permitidas para **usuários com as roles `ADMIN` ou `USER`**.
- Operações de criação e leitura em `/api/comentarios/**` devem ser permitidas para **usuários com as roles `ADMIN` ou `USER`**. Operações de atualização e exclusão em `/api/comentarios/**` devem ser permitidas **apenas para usuários com a role `ADMIN`**.

Segurança por Padrão:

Certifiquem-se de que todas as outras requisições (não especificamente mapeadas nas regras acima) exijam autenticação, implementando assim o conceito de **Segurança por Padrão**.

Refatoração e limpeza (Opcional, mas recomendado):

Se houver alguma lógica de segurança manual nos controladores (como verificações de role dentro dos métodos), removam-na, confiando totalmente no Spring Security para o controle de acesso centralizado.

Testes da implementação:

Utilizem ferramentas como o Postman para testar exaustivamente as regras de segurança, validando cenários de sucesso (acesso permitido) e cenários de falha (recebimento de `401 Unauthorized` para falta de autenticação e `403 Forbidden` para falta de autorização para usuários com roles inadequadas).

Critérios de avaliação

- **Configuração do Spring Security e autenticação (30%)**

  Qualidade da configuração, implementação correta do HTTP Basic e dos usuários em memória com senhas codificadas.

- **Implementação da autorização por URL (40%)**

  Aplicação eficaz das regras de autorização para os dois contextos selecionados, garantindo que as permissões de `ADMIN` e `USER` estejam devidamente segregadas conforme o especificado, e uso correto de `HttpMethod` nas regras.

- **Princípios de segurança aplicados (20%)**

  Demonstração clara da compreensão e aplicação do Princípio do Menor Privilégio e da Segurança por Padrão na solução.

- **Documentação e testes (10%)**

  Clareza e completude do relatório no `Readme.md`, incluindo a justificativa das escolhas e as evidências dos testes que validam a segurança.

Dicas para o Sucesso

Comecem com a base: garanta que a autenticação HTTP Basic esteja funcionando antes de adicionar as regras de autorização. **Teste incrementalmente:** adicione uma regra de autorização por vez e teste-a imediatamente com diferentes usuários para verificar se ela funciona como esperado. **Atenção à ordem das regras:** lembrem-se que as regras são avaliadas na ordem de sua declaração. Regras mais específicas devem vir antes das mais genéricas. **Entendam as roles:** o Spring Security automaticamente adiciona o prefixo `ROLE_` quando você usa `hasRole()`. **Falhem de forma segura:** o objetivo é que, se algo não for explicitamente permitido, seja implicitamente negado.