Aula 11: Criando o Repository e Serviço de Autenticação

September 18, 2025

1 Introdução

Na Aula 10, criamos a entidade JPA `Usuario` e a migration Flyway para a tabela `usuarios`. Nesta aula, continuaremos a implementação da autenticação com `Spring Security`, criando a interface `UsuarioRepository` para acesso ao banco de dados e a classe `AutenticacaoService` para gerenciar a lógica de autenticação. Essas classes são essenciais para consultar usuários durante o processo de login. As alterações serão feitas no `Visual Studio Code` ou sua IDE preferida.

2 Criando a Interface `UsuarioRepository`

Para acessar a tabela `usuarios`, criamos uma interface de repositório no pacote `med.voll.api.domain.usuari` A interface estende `JpaRepository` e inclui um método personalizado para buscar usuários pelo login.

O código da interface é:

```
1 package med.voll.api.domain.usuario;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.security.core.userdetails.UserDetails;
5
6 public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
7     UserDetails findByLogin(String login);
8 }
```

2.1 Explicação do Código

- `extends JpaRepository<Usuario, Long>`: Configura a interface para gerenciar a entidade `Usuario` com chave primária do tipo `Long`.
- `UserDetails findByLogin(String login)`: Método personalizado que consulta um usuário pelo campo `login` (e-mail). Retorna `UserDetails`, interface do `Spring Security` usada no processo de autenticação.

3 Criando a Classe `AutenticacaoService`

Para implementar a lógica de autenticação, criamos a classe `AutenticacaoService` no mesmo pacote `med.voll.api.domain.usuario`. Esta classe é um serviço do Spring que implementa a interface `UserDetailsService`, usada pelo `Spring Security` para carregar dados do usuário durante o login.

O código da classe é:

```java
package med.voll.api.domain.usuario;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class AutenticacaoService implements UserDetailsService {

    @Autowired
    private UsuarioRepository repository;

    @Override
    public UserDetails loadUserByUsername(String username) throws
        UsernameNotFoundException {
        return repository.findByLogin(username);
    }
}
```

3.1 Explicação do Código

- `@Service`: Marca a classe como um componente de serviço do Spring, permitindo que seja carregada automaticamente.
- `implements UserDetailsService`: Implementa a interface do `Spring Security` que define o método `loadUserByUsername`, chamado automaticamente durante o processo de autenticação.
- `@Autowired private UsuarioRepository repository`: Injeta o repositório para acessar a tabela `usuarios`.
- `loadUserByUsername(String username)`: Consulta o usuário pelo `login` (e-mail) usando o método `findByLogin` do repositório. Retorna um `UserDetails` ou lança `UsernameNotFoundExcep` se o usuário não for encontrado.

4 Integração com o Spring Security

O `Spring Security` detecta automaticamente a classe `AutenticacaoService` por causa da anotação `@Service` e da implementação de `UserDetailsService`. Quando uma requisição de login for feita, o Spring chamará o método `loadUserByUsername`, passando o login informado (ex.: e-mail do usuário). A consulta ao banco de dados é realizada via `UsuarioRepository`, garantindo que o processo de autenticação seja baseado nos dados da tabela `usuarios`.

5 Próximos Passos

Com o repositório e o serviço de autenticação configurados, estamos prontos para criar o endpoint de login e implementar a geração de tokens JWT. Na próxima aula, configuraremos o `Spring Security` para suportar autenticação stateless e adicionaremos a lógica de autenticação. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

6 Dica do Professor

- **Aprofunde-se em Spring Security**: Consulte a documentação do Spring Security sobre `UserDetailsService` (`https://docs.spring.io/spring-security/reference/features/authentication/user-details.html`) para entender como personalizar a autenticação.

- **Comunidade no X**: Siga perfis como @SpringSecGuru e @AuthExpert no X para dicas sobre autenticação e boas práticas com Spring Security.
- **Pratique**: Adicione um usuário à tabela `usuarios` (ex.: via MySQL Workbench) e teste o método `findByLogin` manualmente, verificando se o `UsuarioRepository` retorna os dados corretamente.