Aula 20: Validando o Token JWT no Filtro de Segurança

September 18, 2025

1 Introdução

Na Aula 19, implementamos a recuperação do token JWT do cabeçalho `Authorization` no filtro `SecurityFilter`. Nesta aula, adicionaremos a lógica de validação do token, criando o método `getSubject` na classe `TokenService` para verificar a autenticidade do token e extrair o sujeito (login do usuário). Atualizaremos o `SecurityFilter` para usar esse método e testaremos no Insomnia. As alterações serão feitas no `Visual Studio Code` ou sua IDE preferida.

2 Criando o Método `getSubject` no `TokenService`

No `TokenService` (pacote `med.voll.api.infra.security`), adicionamos o método `getSubject` para validar o token JWT e retornar o sujeito:

```
1  package med.voll.api.infra.security;
2
3  import com.auth0.jwt.JWT;
4  import com.auth0.jwt.algorithms.Algorithm;
5  import com.auth0.jwt.exceptions.JWTCreationException;
6  import com.auth0.jwt.exceptions.JWTVerificationException;
7  import med.voll.api.domain.usuario.Usuario;
8  import org.springframework.beans.factory.annotation.Value;
9  import org.springframework.stereotype.Service;
10 import java.time.Instant;
11 import java.time.LocalDateTime;
12 import java.time.ZoneOffset;
13
14 @Service
15 public class TokenService {
16
17     @Value("${api.security.token.secret}")
18     private String secret;
19
20     public String gerarToken(Usuario usuario) {
21         try {
22             var algoritmo = Algorithm.HMAC256(secret);
23             return JWT.create()
24                     .withIssuer("API Voll.med")
25                     .withSubject(usuario.getLogin())
26                     .withExpiresAt(dataExpiracao())
27                     .sign(algoritmo);
28         } catch (JWTCreationException exception) {
29             throw new RuntimeException("Erro ao gerar token JWT", exception);
30         }
31     }
32
33     public String getSubject(String tokenJWT) {
34         try {
35             var algoritmo = Algorithm.HMAC256(secret);
36             return JWT.require(algoritmo)
```

```
                    .withIssuer("API Voll.med")
                    .build()
                    .verify(tokenJWT)
                    .getSubject();
        } catch (JWTVerificationException exception) {
            throw new RuntimeException("Token JWT inválido ou expirado!",
                exception);
        }
    }

    private Instant dataExpiracao() {
        return LocalDateTime.now().plusHours(2).toInstant(ZoneOffset.of("-03:00"));
    }
}
```

2.1 Explicação do Código

- `getSubject(String tokenJWT)`: Valida o token JWT e retorna o sujeito (sub, ou seja, o login do usuário).
- `Algorithm.HMAC256(secret)`: Usa a mesma chave secreta do método `gerarToken`.
- `JWT.require(algoritmo).withIssuer("API Voll.med")`: Configura a verificação, exigindo o emissor `API Voll.med`.
- `.verify(tokenJWT).getSubject()`: Verifica o token e extrai o sujeito.
- `JWTVerificationException`: Lança uma exceção se o token for inválido ou expirado.

3 Atualizando o `SecurityFilter`

No `SecurityFilter` (pacote `med.voll.api.infra.security`), injetamos o `TokenService` e validamos o token recuperado:

```
package med.voll.api.infra.security;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import java.io.IOException;

@Component
public class SecurityFilter extends OncePerRequestFilter {

    @Autowired
    private TokenService tokenService;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
        HttpServletResponse response, FilterChain filterChain) throws
        ServletException, IOException {
        var tokenJWT = recuperarToken(request);
        var subject = tokenService.getSubject(tokenJWT);
        filterChain.doFilter(request, response);
    }

    private String recuperarToken(HttpServletRequest request) {
```

```
        var authorizationHeader = request.getHeader("Authorization");
        if (authorizationHeader == null) {
            throw new RuntimeException("Token JWT não enviado no cabeçalho
                Authorization!");
        }
        return authorizationHeader.replace("Bearer ", "");
    }
}
```

3.1 Explicação do Código

- `@Autowired private TokenService tokenService`: Injeta o `TokenService` do projeto (pacote `med.voll.api.infra.security`).
- `tokenService.getSubject(tokenJWT)`: Valida o token e obtém o sujeito (login do usuário).
- `filterChain.doFilter(request, response)`: Permite a continuação do fluxo se o token for válido.

4 Testando a Validação do Token

No Insomnia, testamos duas requisições:

- **Detalhar Médico** (`GET /medicos/id`): Sem o token no cabeçalho `Authorization`, retorna erro 500 devido à exceção `Token JWT não enviado no cabeçalho Authorization!`.
- **Listagem de Médicos** (`GET /medicos`): Com o token configurado na aba "Auth" (tipo "Bearer", checkbox "Enabled"), retorna código 200 (OK) e o JSON esperado:

```
[
    {
        "id": 1,
        "nome": "Nome do Médico",
        "email": "medico1@voll.med",
        "crm": "123456",
        "especialidade": "ORTOPEDIA"
    },
    ...
]
```

No console da IDE, o sujeito (ex.: `usuario@voll.med`) foi impresso, confirmando que o token foi validado corretamente. Após a verificação, removemos o `System.out.println(subject)` do `SecurityFilter`.

5 Próximos Passos

Implementamos a validação do token JWT no `SecurityFilter`, mas o filtro ainda não autentica o usuário no contexto do Spring Security. Na próxima aula, configuraremos a autenticação do usuário com base no sujeito extraído, substituindo o log (`System.out.println`) por uma integração com o `SecurityContextHolder`. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

6 Dica do Professor

- **Aprofunde-se em Validação de Tokens**: Consulte a documentação da biblioteca `java-jwt` (`https://github.com/auth0/java-jwt#verify-a-jwt`) para exemplos de validação.

- **Comunidade no X**: Siga perfis como @JWTValidator e @SpringSecPro no X para dicas sobre validação de tokens e segurança.
- **Pratique**: No Insomnia, envie a requisição `GET /medicos` com um token inválido (ex.: altere um caractere do token) e verifique se a exceção `Token JWT inválido ou expirado!` é lançada no console da IDE.