



WEB ACADEMY

Programação Avançada Back-end

Daniel Augusto Nunes da Silva

Apresentação

Ementa

- Mecanismos de segurança e controle de acesso: Autenticação por senha e JWT. Utilização de Data Transfer Objects (DTO). Validação de dados. Paginação e ordenação. Documentação de REST APIs. Acesso a APIs externas. Gerenciamento de cache.

Objetivos

- **Geral:** Capacitar os alunos para o desenvolvimento avançado de aplicações back-end, por meio da exploração de tecnologias, práticas e padrões essenciais, preparando-os para os desafios do ambiente profissional.
- **Específicos:**
 - Aplicar técnicas de autenticação e segurança, para garantir a proteção adequada dos recursos das aplicações.
 - Explorar técnicas avançadas para melhorar a eficiência e a qualidade das soluções back-end.
 - Abordar métodos de integração de aplicações back-end e estratégias de otimização de desempenho.

Conteúdo programático

Autenticação e Segurança

- Introdução ao Spring Security;
- Autenticação de usuários e autorização de acesso;
- Autenticação JWT.
- Segurança na comunicação via HTTP.

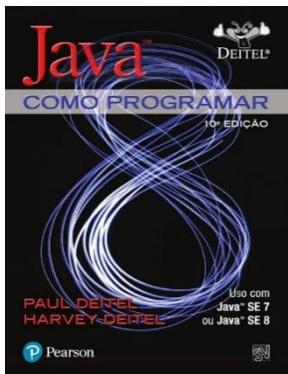
Práticas Avançadas

- Padrão de projeto DTO;
- DTO com Java Records;
- Mapeamento de DTOs;
- Paginação e ordenação de resultados;
- Validações no back-end;
- Documentação de REST APIs com Swagger.

Integração e otimização

- Acesso a APIs externas: RestTemplate, RestClient e WebClient;
- Implementação de cache: conceitos e estratégias.

Bibliografia



Java: Como Programar.

Paul Deitel e Harvey Deitel
10ª Edição – 2016
Editora Pearson
ISBN 9788543004792



Spring in Action

Craig Walls
6ª Edição – 2021
Editora Manning
ISBN 9781617297571



Engenharia de Software Moderna

Marco Tulio Valente
<https://engsoftmoderna.info/>



Sites de referência

- Spring Boot Reference Documentation:
 - <https://docs.spring.io/spring-boot/docs/current/reference/html/index.html>
- Spring Getting Started Guides:
 - <https://spring.io/guides#getting-started-guides>
- Swagger Documentation:
 - <https://swagger.io/docs/>
- Baeldung:
 - <https://www.baeldung.com/>

Ferramentas

- **Visual Studio Code:** <https://code.visualstudio.com/Download>
- **Extension Pack for Java (Extensão do VS Code):**
<https://marketplace.visualstudio.com/items?vscjava.vscode-java-pack>
- **Spring Boot Extension Pack (Extensão do VS Code):**
<https://marketplace.visualstudio.com/items?itemName=pivotal.vscode-boot-dev-pack>
- **XML (Extensão do VS Code):** <https://marketplace.visualstudio.com/items?itemName=redhat.vscode-xml>
- **Postman:** <https://www.postman.com/downloads/>
 - Link para download da coleção compartilhada: https://api.postman.com/collections/19704449-afc96c2c-7ce9-4370-b280-30b4bf2e67a7?access_key=PMAT-01HZR284JMDFHP76NYMWK1AJCD
 - Para importar a coleção no Postman, clique em “**Import**” e cole o link acima.

Ferramentas: JDK 17

- Verificar versão do JDK instalada: **javac -version**
- https://download.oracle.com/java/17/archive/jdk-17.0.6_windows-x64_bin.msi
- Criar a variável de ambiente JAVA_HOME configurada para o diretório de instalação do JDK. Exemplo: “C:\Program Files\Java\jdk-17”.
- Adicionar “%JAVA_HOME%\bin” na variável de ambiente PATH.
- Tutorial de configuração: https://mkyong.com/java/how-to-set-java_home-on-windows-10/

Ferramentas: Maven

- Verificar versão do Maven instalada: **mvn -version**
- Link para download: <https://dlcdn.apache.org/maven/maven-3/3.8.8/binaries/apache-maven-3.8.8-bin.zip>
- Adicionar o diretório de instalação do Maven na variável de ambiente PATH.
Exemplo: “C:\apache-maven\bin”.
- Tutorial de instalação: <https://mkyong.com/maven/how-to-install-maven-in-windows/>

Ferramentas: MySQL

- Verificar se o MySQL está funcionando:
 - `mysql -u root -p`
 - Tentar acessar com senha em branco ou senha igual ao nome de usuário (root).
 - Tutorial para resetar a senha de root: <https://dev.mysql.com/doc/mysql-windows-excerpt/8.0/en/resetting-permissions-windows.html>
- Remova o banco de dados **sgcm**, se existir:
 - No prompt de comandos digite: `mysql -u root -p`
 - Ao conectar no MySQL, execute a seguinte instrução SQL: **DROP DATABASE sgcm;**
- Se necessário, realizar a instalação:
 - Link para download: <https://dev.mysql.com/downloads/file/?id=516927>
 - Tutorial de instalação: <https://github.com/webacademyufac/tutoriais/blob/main/mysql/mysql.md>

Contato



<https://github.com/danielnsilva>

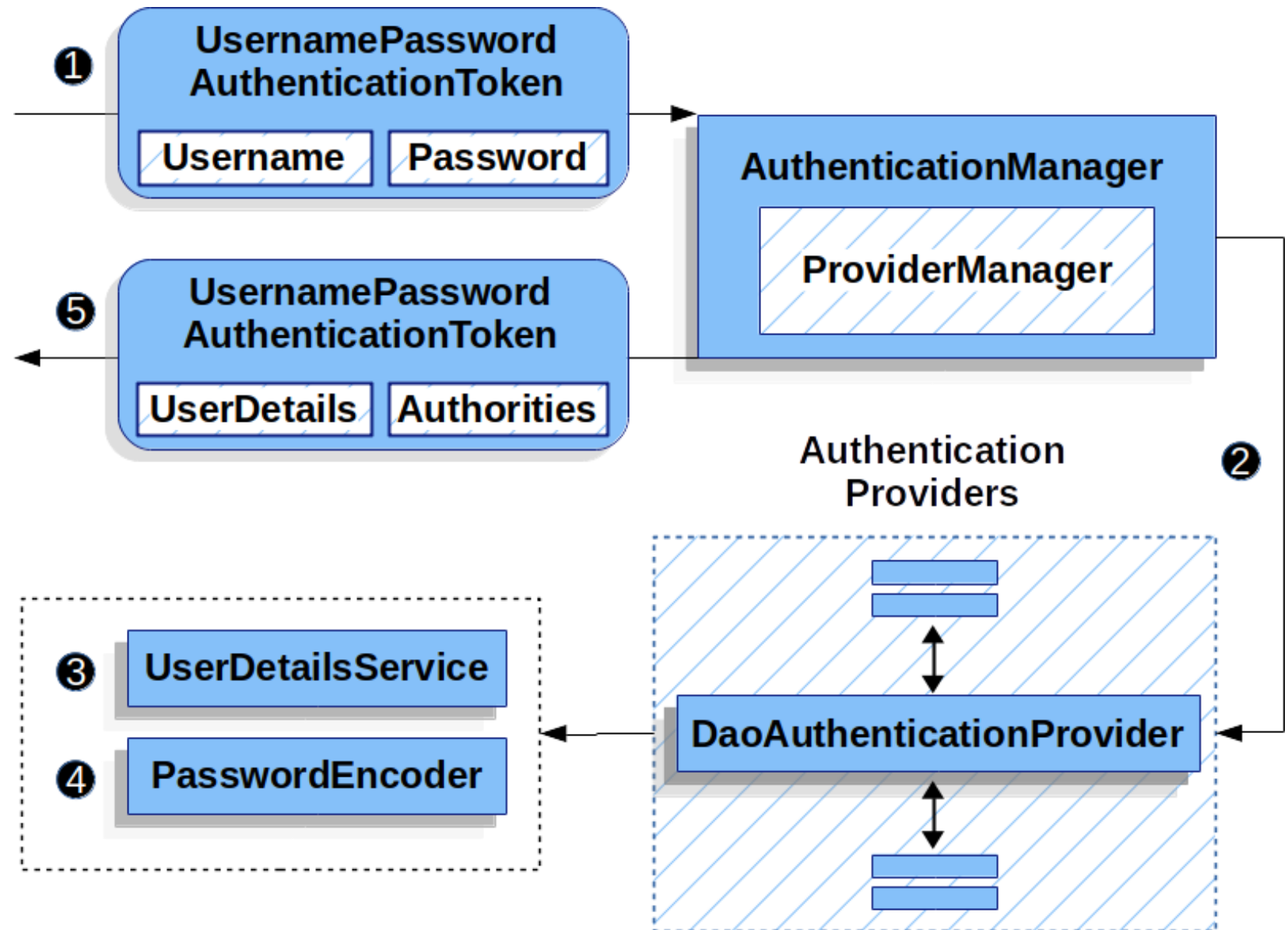
Autenticação e Segurança

Introdução ao Spring Security

- **Spring Security** fornece recursos para **autenticação**, **autorização de acesso** e proteção contra ataques comuns.
- **Autenticação**: verifica a identidade do usuário.
 - Exemplos: usuário/senha, certificado digital X.509, protocolo CAS para autenticação SSO (Single Sign-On).
- **Autorização**: verifica as permissões do usuário para realizar determinadas ações ou acessar recursos específicos.
- Possui suporte ao **OAuth2** (login via Google, Facebook, GitHub, etc.).

Autenticação de usuários

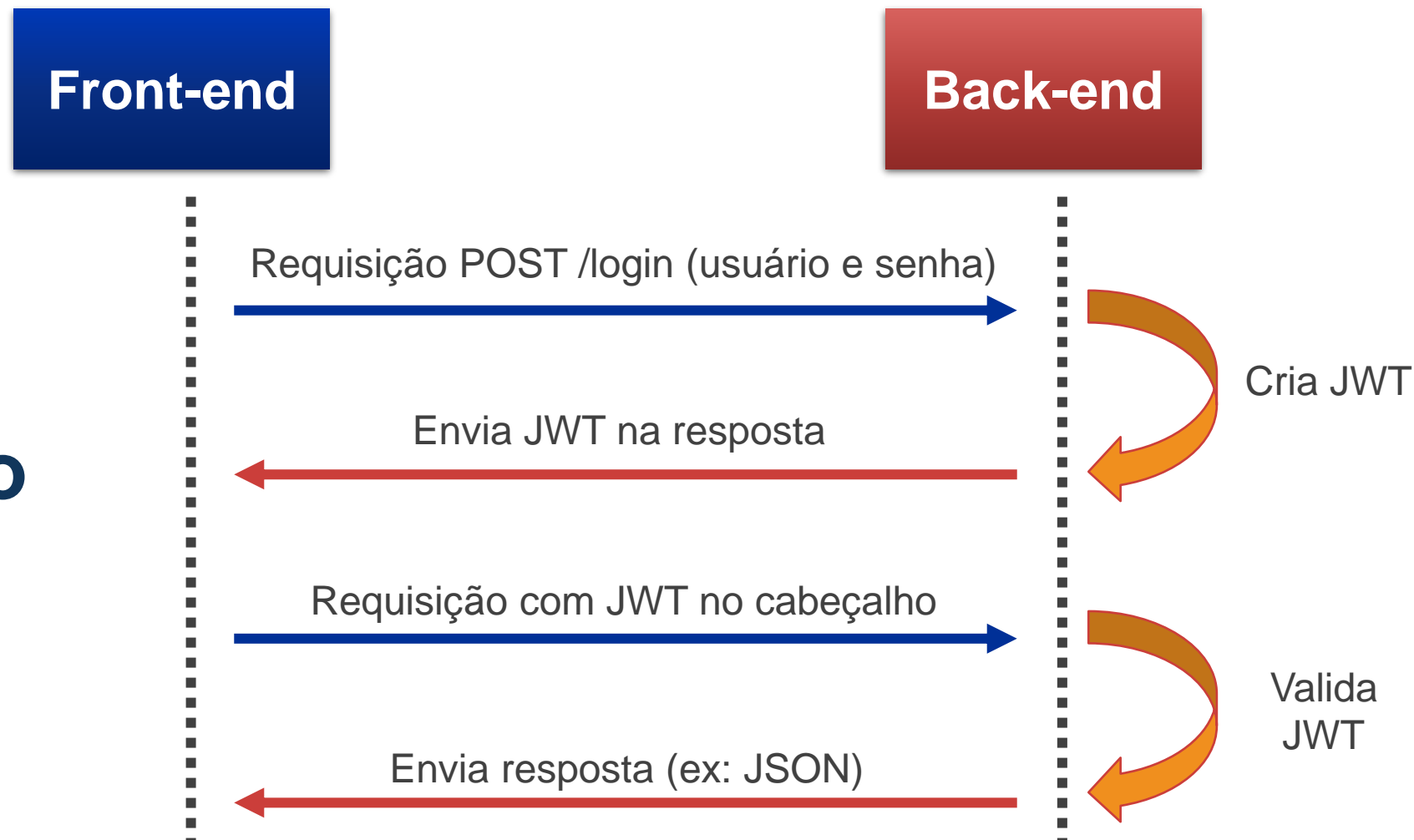
Provedor de autenticação que implementa um DAO para recuperar informações do usuário.



Fonte: <https://docs.spring.io/spring-security/reference/6.2/servlet/authentication/passwords/dao-authentication-provider.html>

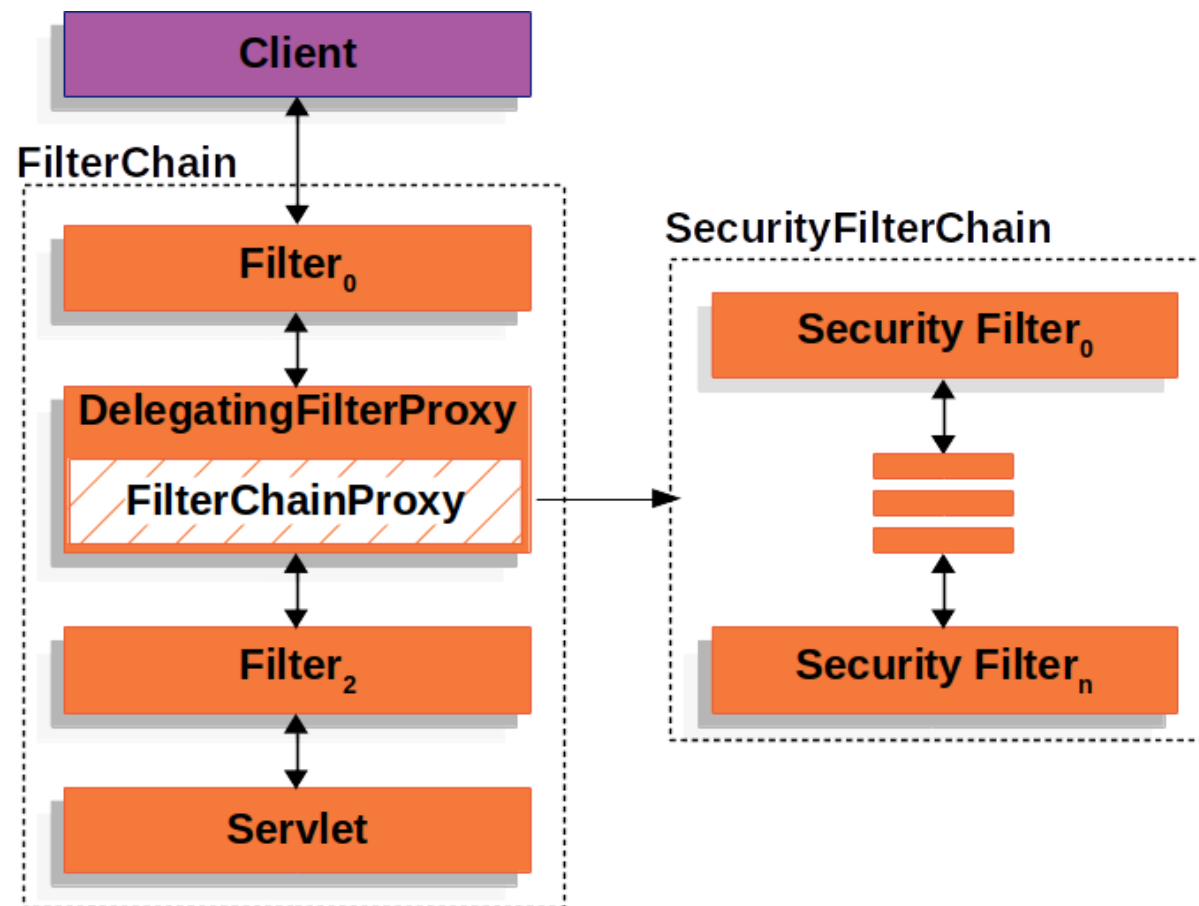
Autenticação JWT

JSON Web Token



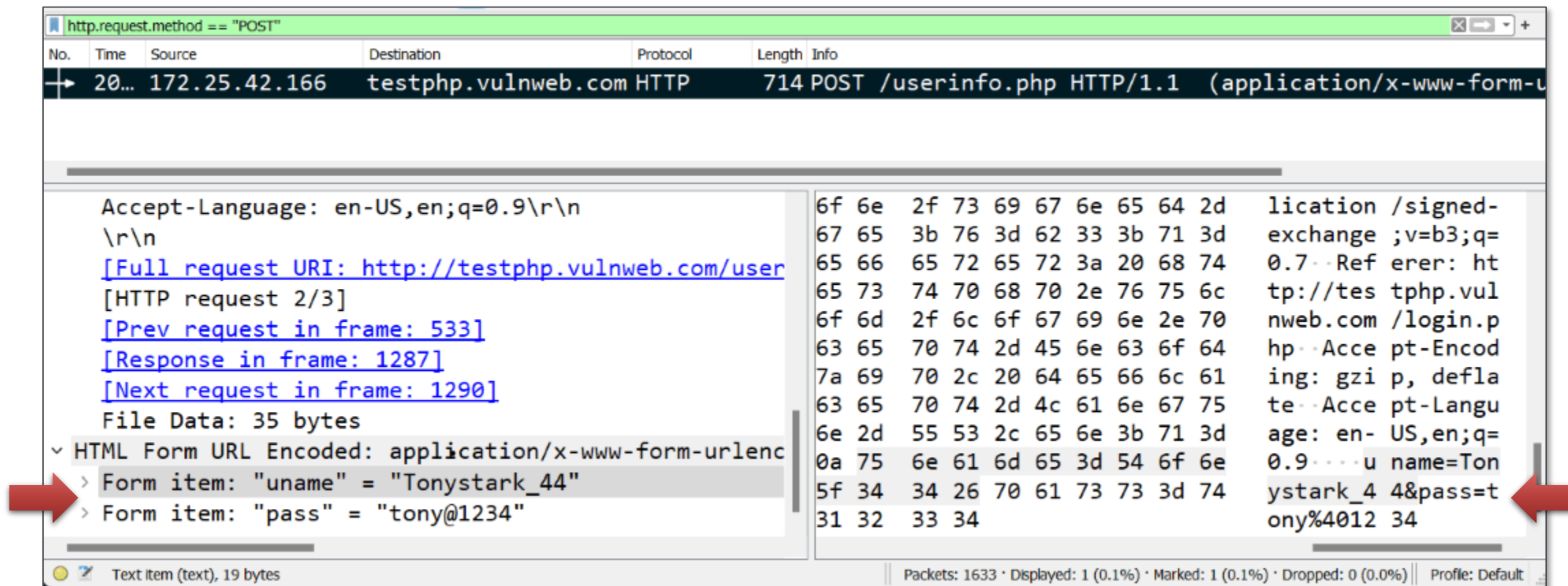
Cadeia de Filtros de Segurança no Spring Security

- ○ Spring Security aplica uma **série de filtros de segurança específicos** para verificar autenticação, autorização e outras políticas de segurança.
- Após passar por todos esses filtros a requisição chega ao **Servlet** para ser processada normalmente.



Segurança na comunicação via HTTP

Usuário e senha sendo capturados no Wireshark:



Fonte: <https://www.geeksforgeeks.org/sniffing-of-login-credential-or-password-capturing-in-wireshark/>

SSL/TLS

- **SSL** (**S**ecure **S**ockets **L**ayer) permite o tráfego de dados pela rede de forma segura, estabelecendo um **canal de comunicação entre aplicações onde as informações são criptografadas**.
- **TLS** (**T**ransport **L**ayer **S**ecurity) é o **successor do SSL** e funciona de forma semelhante.
 - Apesar do termo SSL ser mais popular, na maioria das vezes o termo correto que deveria ser utilizado é TLS.
- O protocolo **HTTPS** é uma implementação do HTTP com uma camada adicional de segurança (HTTPS = HTTP + SSL/TLS).

Certificados autoassinados

- São **certificados gerados por entidade própria**, sem validação por uma Autoridade Certificadora (CA), e são normalmente usados em ambientes de desenvolvimento, testes internos e **sistemas locais onde a validação externa não é essencial**.
- Possuem a **vantagem da redução de custo**, mas sem validação de uma CA confiável, **aumentam as chances de problemas de segurança**.
- **Serviços de nuvem normalmente fornecem acesso seguro** (HTTPS), e a há ainda a possibilidade de utilizar certificado próprio.

Habilitar SSL no Spring Boot

- **Criar certificado**

```
keytool -genkeypair -alias SGCM -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore certificado.p12  
-validity 3650 -dname "CN=SGCM, OU=localhost, O=UFAC, L=Rio Branco, S=AC, C=BR" -ext  
san=dns:localhost
```

```
keytool -export -keystore certificado.p12 -alias SGCM -file certificado.crt
```

- Os arquivos **certificado.p12** e **certificado.crt** devem ser colocados no diretório **src/main/resources/**

- **application.properties**

```
server.ssl.key-store=classpath:certificado.p12
```

```
server.ssl.key-store-password=webacademy
```

```
server.ssl.key-store-type=PKCS12
```

Práticas Avançadas

Padrão de projeto DTO

- O padrão **DTO** (*Data Transfer Object*) representa um **formato utilizado na transferência de dados** entre sistemas diferentes (ou entre camadas de um mesmo sistema).
- Não é o mesmo que classes da camada de modelo: **DTO não representa um objeto que será persistido.**

```
public class UsuarioDto {  
    // Mesmos atributos da camada  
    // de modelo, mas sem a senha.  
    private Long id;  
    private String nomeCompleto;  
    private String nomeUsuario;  
    private String papel;  
    private boolean ativo;  
    // Getters e Setters  
}
```


DTO com Java Records

```
public class UsuarioDto {  
    private Long id;  
    private String nomeCompleto;  
    private String nomeUsuario;  
    private String papel;  
    private boolean ativo;  
  
    // Getters e Setters  
  
}
```

```
public record UsuarioDto(  
    Long id,  
    String nomeCompleto,  
    String nomeUsuario,  
    String papel,  
    boolean ativo  
) {}
```

```
// Acessar um atributo  
dto.nomeCompleto();
```

Java Records são imutáveis, pois uma vez criado, o estado de um objeto não podem ser alterado. Não há como definir valor para os atributos após criação do objeto (sem *setters*).

Validação de dados no back-end

- Validação é uma forma de **verificação de qualidade dos dados**, garantindo que as informações fornecidas seguem as regras definidas.
- A **validação no back-end ajuda a garantir a segurança e a integridade dos dados**, independentemente da verificação feita no front-end, que pode ser manipulada.
 - Validação no front-end tem objetivo diferente: melhorar a experiência do usuário.
- Princípio ***Fail Fast***: retornar o mais rápido possível uma falha, interrompendo a operação atual, e evitando uso de recursos desnecessários.

Validação de dados no back-end

- Spring Boot **Starter Validation** inclui no projeto recursos do Java Bean Validation, que permitem realizar validação por meio de anotações.
- Exemplos de validadores: @NotNull, @NotBlank e @Email.
- Lista de validadores:
 - <https://jakarta.ee/specifications/bean-validation/3.0/jakarta-bean-validation-spec-3.0.html#builtinconstraints>
- Também é possível criar **validadores customizados**.

Paginação e ordenação de resultados

- **Paginação de resultados** é uma técnica comum em aplicações que acessam bancos de dados, com objetivo de melhorar a **performance**, **usabilidade**, além de **reduzir a carga sobre o servidor da aplicação**.
- O **Spring Data** fornece a interface **Pageable**, que facilita a implementação de **paginação e ordenação** em consultas a banco de dados.

```
public ResponseEntity<Page<Atendimento>> get(Pageable page) {  
    Page<Atendimento> registros = servico.get(page);  
    return ResponseEntity.ok(registros);  
}
```

- Exemplo de requisição: `GET /atendimento/consultar?page=0&size=10&sort=data,desc&sort=hora,asc`

Documentação de APIs

- A documentação é uma **parte essencial na implementação de REST APIs**, seja para servir a um front-end ou mesmo para integrar com outras aplicações.
- É importante descrever a especificação de uma API para evitar **redundâncias**, garantir **comunicação clara**, facilitar a **integração**, assegurar **consistência**, e possibilitar **escalabilidade** e **reuso**.
- Exemplo: <https://petstore.swagger.io/>

order-controller Order Controller		
DELETE	/orders/{orderId}	deleteOrder
user-controller User Controller		
POST	/users/	Create a user from input JSON body
PATCH	/users/{userId}	patch
GET	/users/{userId}	Get user info by userId in URL path
HEAD	/users/{userId}	head
PUT	/users/{userId}	put

Swagger

- Swagger é um **conjunto de ferramentas** de código aberto **para projetar, documentar e usar APIs**.
 - Exemplos: Swagger Editor, Swagger UI e Swagger Codegen.
- Utiliza a **especificação OpenAPI**, que padroniza a forma de descrever APIs.
- OpenAPI foi originalmente desenvolvido como **Swagger Specification** (até versão 2), e depois padronizado e mantido pela OpenAPI Initiative (versão 3).



Integração do Swagger com Spring Boot

- O trabalho de construir e manter atualizada a documentação é difícil de ser realizado de forma manual, mas a **integração com frameworks permite automatizar o processo**.
- No Spring, a integração pode ser feita com auxílio de uma ferramenta mantida pelo projeto [springdoc-openapi](https://springdoc.org).

```
<dependency>  
  <groupId>org.springdoc</groupId>  
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
  <version>2.6.0</version>  
</dependency>
```

Continua...

Referências

- DEITEL, Paul; DEITEL, Harvey. **Java: Como Programar**. 10. ed. São Paulo: Pearson, 2016. 968 p.
- MARCO TULIO VALENTE. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**, 2020. Disponível em:
<https://engsoftmoderna.info/>
- MOZILLA (ed.). **MDN Web Docs: Aprendendo desenvolvimento web**. [S. l.], 2024. Disponível em:
<https://developer.mozilla.org/pt-BR/docs/Learn>.
- SPRING (ed.). **Spring Boot Reference Documentation**. [S. l.], 2024. Disponível em:
<https://docs.spring.io/spring-boot/docs/current/reference/html/index.html>.
- WALLS, Craig. **Spring in Action**. 6. ed. Shelter Island: Manning, 2021. 520 p.