



WEB ACADEMY

Frameworks Back-end

Daniel Augusto Nunes da Silva

Apresentação

Ementa

- Frameworks Back-end. **Spring Framework**. Injeção de dependência. **Spring Boot**. Persistência de dados com **JPA**, Hibernate e Mapeamento Objeto-Relacional (ORM). Spring Data. **Arquitetura REST e APIs**. Mapeamento de requisições HTTP. Segurança.

Objetivos

- **Geral:** Habilitar o aluno na utilização de **frameworks para desenvolvimento de aplicações WEB voltadas para o back-end**, apoiadas nas ferramentas dos projetos que fazem parte do Spring.
- **Específicos:**
 - Compreender o papel dos frameworks no contexto do desenvolvimento web.
 - Apresentar os principais recursos da família de projetos Spring com ênfase na construção de projetos Spring Boot.
 - Demonstrar como o conjunto de ferramentas do Spring podem otimizar a persistência de dados.
 - Capacitar o aluno na construção de uma API REST baseada em um projeto Spring Boot.

Conteúdo programático

Introdução

- Programação server-side;
- Frameworks web (back-end);
- Spring Framework;
- Inversão de controle e injeção de dependência.

Spring Boot

- Introdução ao Spring Boot;
- Criação de projetos Spring Boot;
- Anotações e meta-anotações;
- Execução da aplicação e deploy no servidor de produção.

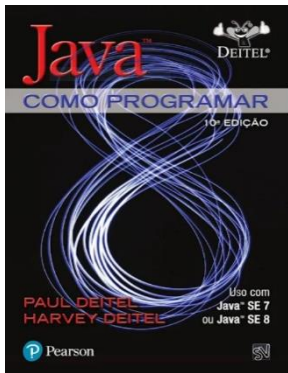
Persistência de dados

- Introdução ao JPA, Hibernate e ORM;
- Estratégias para geração de chaves primárias;
- Relacionamento entre entidades;
- Spring Data.

API

- Introdução à arquitetura REST e construção de APIs.
- Camadas de uma API REST.
- Endpoints e mapeamento de requisições HTTP.
- Segurança: CORS e SSL.

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>
- Spring Boot in Visual Studio Code
 - <https://code.visualstudio.com/docs/java/java-spring-boot>
- Uma visão geral do HTTP
 - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>
- Apostila Java e Orientação a Objetos (Caelum/Alura)
 - <https://www.alura.com.br/apostila-java-orientacao-objetos>
- 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-e147c76f-5808-48bd-9808-8f7315414ed9?access_key=PMAT-01HBEZH1WVE959024Z1V9S5BYS

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/

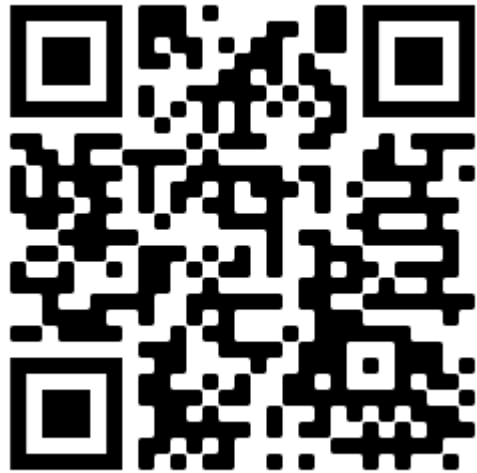
Ferramentas: Maven

- 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>

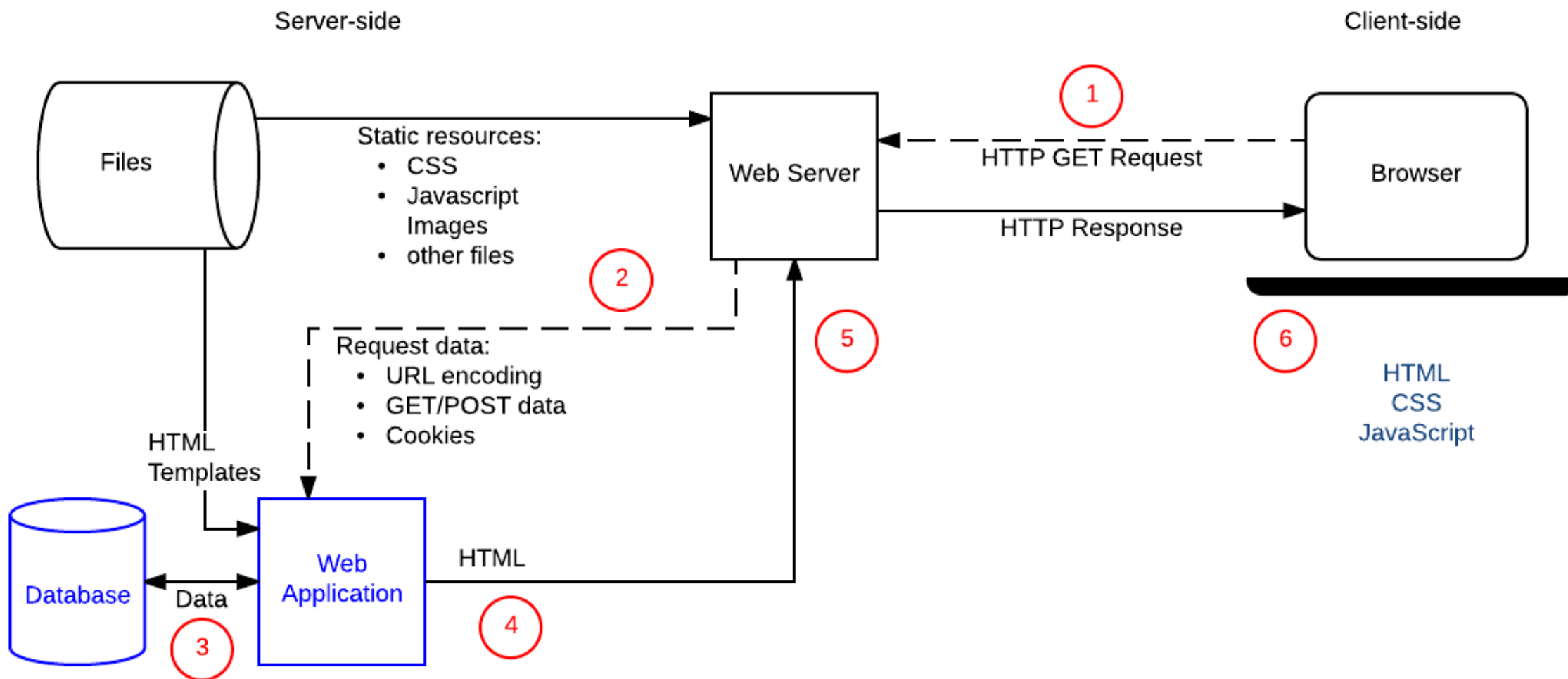
Contato



<https://linkme.bio/danielnsilva/>

Introdução

Programação server-side



Fonte: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/First_steps/Introduction

Programação server-side



Frameworks web (back-end)

- Fornecem ferramentas que **simplificam as operações comuns de desenvolvimento**.
- **Não precisamos de um framework**, mas facilitam muito o trabalho de desenvolvimento.
- Vantagens: produtividade, padronização, reusabilidade, segurança.
- Desvantagens: dependência, segurança (vulnerabilidades), performance.
- Exemplos: Django e Flask (Python), Laravel (PHP), Spring (Java).

Spring


- Originalmente denominado **Spring Framework**.
- Pretendia tornar o desenvolvimento de aplicações J2EE mais fácil.
- O foco do framework não é apenas aplicações web.
- Os recursos para desenvolvimento de aplicações web são baseados em **servlets**.
- Conceitos importantes: **inversão de controle** e **injeção de dependência**.



Inversão de controle e injeção de dependência

- **Inversão de controle permite mudar o fluxo de controle de um programa**, transferindo para um componente externo a responsabilidade de quando executar determinado procedimento.
- A **injeção de dependência** é uma **forma de aplicar a inversão de controle**.
- A dependência não é criada internamente (nova instância de um objeto), mas “injetada” por uma classe externa.

Criação de dependência



```
1. public class Controller {
2.     private PessoaDao dao;
3.     public Controller() {
4.         this.dao = new PessoaDao("mysql");
5.     }
6.     public Pessoa getById(int id) {
7.         return dao.getById(id);
8.     }
9. }
10. Controller c = new Controller();
11. Pessoa pessoa = c.getById(1);
```


Inversão de controle e injeção de dependência

- **Inversão de controle permite mudar o fluxo de controle de um programa**, transferindo para um componente externo a responsabilidade de quando executar determinado procedimento.
- A **injeção de dependência** é uma **forma de aplicar a inversão de controle**.
- A dependência não é criada internamente (nova instância de um objeto), mas “injetada” por uma classe externa.

```
1. public class Controller {  
2.     private PessoaDao dao;  
3.     public Controller(PessoaDao dao) {  
4.         this.dao = dao;  
5.     }  
6.     public Pessoa getById(int id) {  
7.         return dao.getById(id);  
8.     }  
9. }
```

```
10. PessoaDao dao = new PessoaDao("mysql");  
11. Controller c = new Controller(dao);  
12. Pessoa pessoa = c.getById(1);
```

Injeção de dependência



Escopo externo

Inversão de controle e injeção de dependência

- **Inversão de controle permite mudar o fluxo de controle de um programa**, transferindo para um componente externo a responsabilidade de quando executar determinado procedimento.
- A **injeção de dependência** é uma **forma de aplicar a inversão de controle**.
- A dependência não é criada internamente (nova instância de um objeto), mas “injetada” por uma classe externa.

```
1. public class Controller {  
2.     private IDao dao;  
3.     public Controller(IDao dao) {  
4.         this.dao = dao;  
5.     }  
6.     public Pessoa getById(int id) {  
7.         return dao.getById(id);  
8.     }  
9. }
```

```
10. IDao dao = new AlunoDao("mysql");  
11. Controller c = new Controller(dao);  
12. Aluno aluno = c.getById(1);
```

Injeção de dependência

Escopo externo

Inversão de controle e injeção de dependência

- Para saber mais sobre o assunto:
 - <https://engsoftmoderna.info/artigos/injecao-dependencia.html>
 - <https://engsoftmoderna.info/cap6.html#template-method>
 - <https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans>

Spring

- O framework ganhou muitos recursos e foi desmembrado em vários projetos, entre eles:
 - **Spring Framework**: fornece os recursos “básicos”.
 - **Spring Data**: facilita a integração com vários tipos de tecnologias de gerenciamento de dados.
 - **Spring Security**: autenticação e controle de acesso.
 - **Spring Boot**: abstrai a complexidade de configuração de servidores de aplicação.

Spring Boot

Introdução ao Spring Boot

- Facilita o processo de configuração e implantação das aplicações.
 - **Servidor de aplicação embutido.**
 - Gerenciamento de dependências e configurações por meio dos **starters**.
- Responsável por impulsionar a plataforma Spring.



Criando projetos Spring Boot

- É necessário um gerenciador de projetos como o **Maven**.
- A ferramenta **Spring Initializr** (<https://start.spring.io/>) ajuda a criar o projeto com as dependências necessárias.
- O **VS Code também pode fornecer um recurso semelhante** por meio de extensões.
- É um projeto Maven como qualquer outro, exceto pelos **starters** adicionados como dependências ao projeto.
 - Starters: <https://docs.spring.io/spring-boot/docs/current/reference/html/using.html#using.build-systems.starters>
 - Maven: <https://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/htmlsingle/>

Estrutura do projeto

```
+---src
|   +---main
|   |   +---java
|   |   |   \---br
|   |   |       \---ufac
|   |   |           \---exemplospring
|   |   |               |   ExemploSpringApplication.java
|   |   |               \---controller
|   |   |                   ExemploController.java
|   |   \---resources
|   |       |   application.properties
\---target
    |   exemplospring-0.0.1-SNAPSHOT.jar
    +---classes
```

Estrutura do projeto

```
+---src
|   +---main
|   |   +---java
|   |   |   \---br
|   |   |       \---ufac
|   |   |           \---exemplospring
|   |   |               |   ExemploSpringApplication.java
|   |   |               \---controller
|   |   |                   ExemploController.java
|   |   \---resources
|   |       |   application.properties
|   \---target
|       |   exemplospring-0.0.1-SNAPSHOT.jar
+---classes
```

Separação do código Java de outros recursos da aplicação

Estrutura do projeto

```
+---src
|   +---main
|   |   +---java
|   |   |   \---br
|   |   |       \---ufac
|   |   |           \---exemplospring ← - - - - -
|   |   |               |   ExemploSpringApplication.java
|   |   |               \---controller
|   |   |                   ExemploController.java
|   |   \---resources
|   |       |   application.properties
\---target
    |   exemplospring-0.0.1-SNAPSHOT.jar
    +---classes
```

A classe que contém o método main() deve ficar na raiz do pacote principal.

Estrutura do projeto


```
+---src
|   +---main
|   |   +---java
|   |   |   \---br
|   |   |       \---ufac
|   |   |           \---exemplospring
|   |   |               |   ExemploSpringApplication.java
|   |   |               \---controller
|   |   |                   ExemploController.java
|   |   \---resources
|   |       |   application.properties
|   \---target
|       |   exemplospring-0.0.1-SNAPSHOT.jar
+---classes
```

Define propriedades da aplicação, como conexão com banco de dados, segurança, porta TCP, etc.

Estrutura do projeto

```
+---src
|   +---main
|   |   +---java
|   |   |   \---br
|   |   |       \---ufac
|   |   |           \---exemplospring
|   |   |               |   ExemploSpringApplication.java
|   |   |               \---controller
|   |   |                   ExemploController.java
|   |   \---resources
|   |       |   application.properties
\---target
    |   exemplospring-0.0.1-SNAPSHOT.jar
    +---classes
```

Executável JAR contendo a aplicação completa.



Anotações

- Em Java, **uma anotação descreve um componente** (classe, método ou atributo), adicionando metadados ao código.
 - **@SpringBootApplication** identifica a classe principal da aplicação.
- Anotações representam uma **alternativa aos arquivos de configuração XML**.
- Uma parte significativa do funcionamento do **Spring Boot depende de anotações**.
 - <https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-annotation-config>

Anotações

```
@SpringBootApplication  
  
public class Application {  
    public static void main(String[] args) {  
        Application.run(Application.class, args);  
    }  
}
```

Meta-anotações

- Muitas anotações são na verdade meta-anotações (anotações que encapsulam outras anotações).
- **@SpringBootApplication** é uma meta-anotação para:
 - **@Configuration**, que permite registrar *beans* no contexto ou importar classes de configuração adicionais;
 - **@EnableAutoConfiguration**, que habilita a configuração automática do Spring Boot para aplicar configurações baseadas nas dependências que foram adicionadas.
 - **@ComponentScan**, que faz uma busca por outras classes anotadas com @Component.

Executando a aplicação

- A aplicação pode ser inicializada de três formas:
 - **Spring Dashboard.**
 - **Maven:**
 - > mvn spring-boot:run
 - **Executando o pacote (JAR):**
 - > mvn clean package
 - > java -jar target\exemplo.jar
- **Deploy:** o arquivo JAR pode ser executado no servidor de produção.

Persistência de dados

Introdução ao JPA, Hibernate e ORM

- **Java Persistence API (JPA)**, atualmente *Jakarta Persistence*, fornece uma interface comum para persistência de dados.
- JPA define uma forma de representar as entidades de banco de dados relacionais através de classes, utilizando a técnica do **mapeamento objeto-relacional** (ORM, *object-relational mapping*).
- **JPA é apenas uma especificação**, não faz ORM.
- Frameworks ORM, como o **Hibernate**, implementam JPA, gerando as chamadas SQL automaticamente.

Dependência (pom.xml)

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-data-jpa</artifactId>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>mysql</groupId>
```

```
    <artifactId>mysql-connector-java</artifactId>
```

```
    <version>8.0.33</version>
```

```
    <scope>runtime</scope>
```

```
</dependency>
```

Introdução ao JPA, Hibernate e ORM

@Entity

```
public class Especialidade implements Serializable {  
    @Id // Chave primária  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(nullable = false, updatable = false)  
    private Long id;  
    @Column(nullable = false, unique = true)  
    private String nome;  
}
```

Estratégias para geração de chaves primárias

- GenerationType.**IDENTITY**: no MySQL é o mesmo que utilizar AUTO_INCREMENT, mas pode mudar para diferentes SGBD.
- GenerationType.**SEQUENCE**: um *sequence* é um recurso do SGBD para gerar chaves únicas para um grupo (*sequence*), podendo existir vários no banco de dados, mas nem todo SGBD suporta esta funcionalidade.
- GenerationType.**TABLE**: utiliza uma tabela para gerenciar as chaves geradas, sendo uma estratégia compatível com qualquer SGBD, mas que pode afetar o desempenho.
- GenerationType.**AUTO**: o framework ORM (Hibernate) escolhe a estratégia de acordo com o SGBD.

Configurações de conexão (application.properties)

- **Fonte de dados:**

- `spring.datasource.url=jdbc:mysql://localhost:3306/sgcm?createDatabaseIfNotExist=true`
- `spring.datasource.username=root`
- `spring.datasource.password=root`

Configurações de conexão (application.properties)

- **JPA/Hibernate/ORM:**

- `spring.jpa.show-sql=true`
- `spring.jpa.hibernate.ddl-auto=update`
- `spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect`

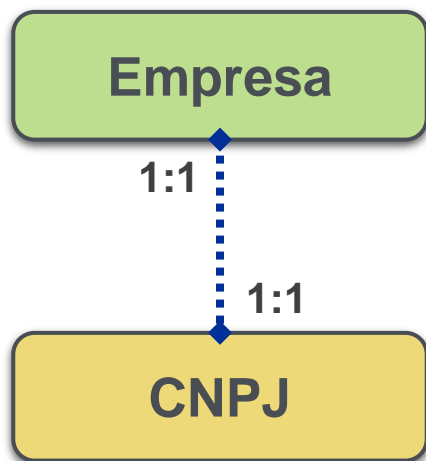
Configurações de conexão (application.properties)

- **Inicialização do banco de dados com scripts SQL:**
 - `spring.jpa.defer-datasource-initialization=true`
 - `spring.sql.init.mode=always`
 - `spring.sql.init.continue-on-error=true`
 - `spring.sql.init.encoding=UTF-8`
- <https://docs.spring.io/spring-boot/docs/current/reference/html/howto.html#howto.data-initialization.using-basic-sql-scripts>

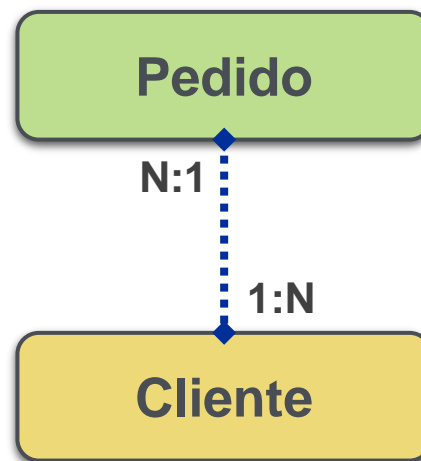
Relacionamento entre entidades

- O **Hibernate facilita o mapeamento de entidades relacionadas**, por meio do ORM, utilizando anotações definidas no JPA.

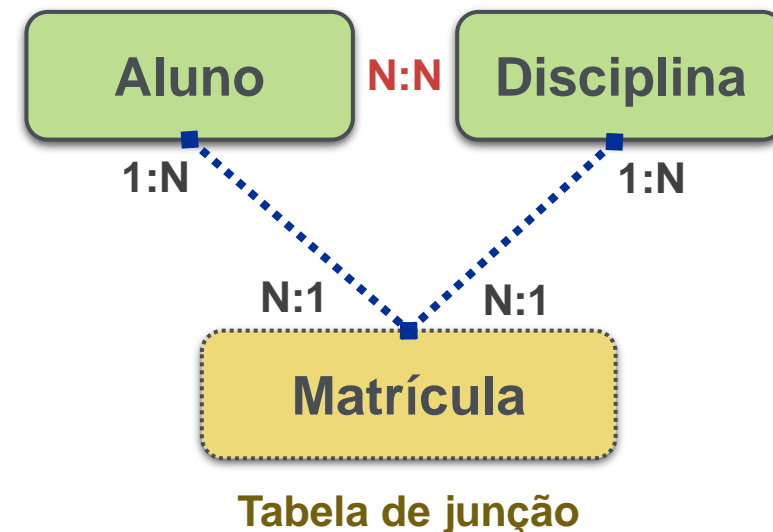
@OneToOne



@ManyToOne @OneToMany



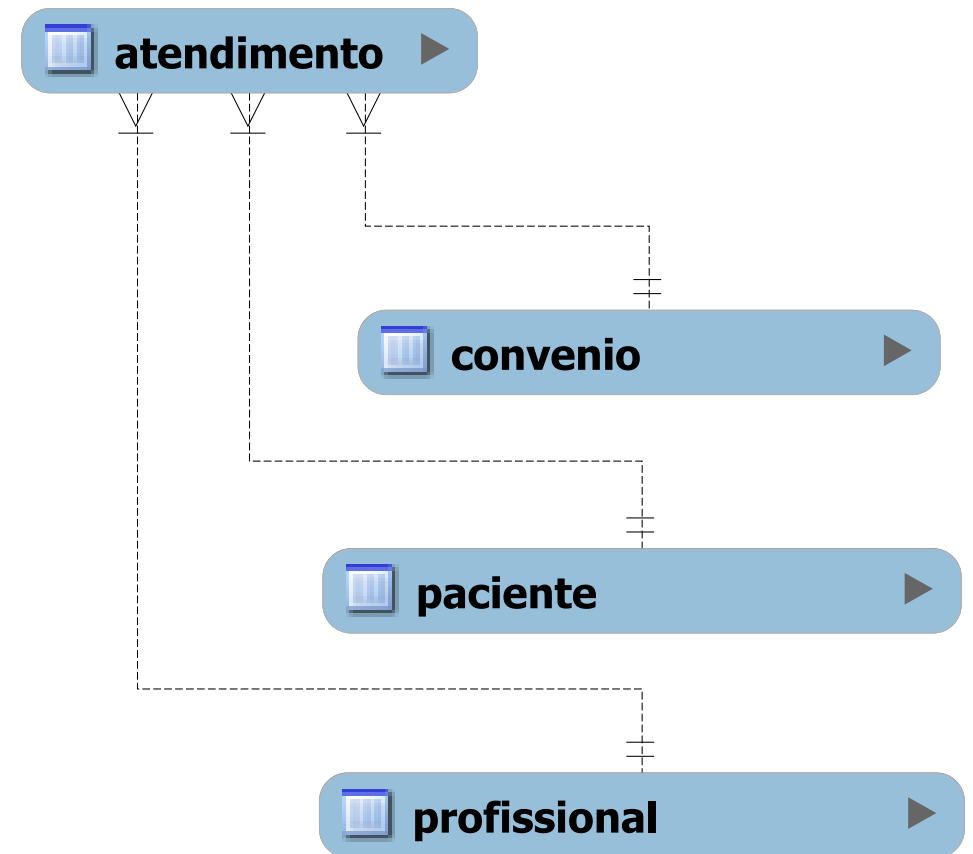
@ManyToMany



Relacionamento entre entidades

@Entity

```
public class Atendimento {  
    @ManyToOne(optional = false)  
    private Profissional profissional;  
    @ManyToOne  
    private Convenio convenio;  
    @ManyToOne(optional = false)  
    private Paciente paciente;  
}
```



Spring Data

- Spring Data fornece um **mecanismo de acesso a dados** de vários tipos diferentes de banco de dados, incluindo relacionais (**JPA**), orientado a documento (MongoDB), grafos (Neo4j) e outros.
- **Spring Data JPA facilita a implementação de repositórios de acesso a dados baseados em JPA**, por meio de uma interface que fornece desde recursos básicos para operações CRUD até funcionalidades avançadas de paginação, consultas customizadas, dentre outros.
- **Dispensa a criação de DAOs** e implementação de métodos específicos para acessos ao banco de dados.

Repositórios e métodos de consulta

```
public interface UnidadeRepository extends JpaRepository<Unidade, Long> {  
    @Query("SELECT u FROM Unidade u WHERE u.nome LIKE %?1%"+  
        " OR u.endereco LIKE %?1%")  
    List<Unidade> findByAll(String termoBusca);  
    List<Unidade> findByNome(String nome);  
    List<Unidade> findByEndereco(String endereco);  
    List<Unidade> findByNomeAndEndereco(String nome, String endereco);  
}
```

Métodos de consulta: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.query-methods.details>

Palavras-chave: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repository-query-keywords>

Injeção de dependência no Spring

```
@Controller
```

```
public class AtendimentoController {  
    private final AtendimentoRepository repo;  
    @Autowired  
    public AtendimentoController(AtendimentoRepository repo) {  
        this.repo = repo;  
    }  
}
```

API

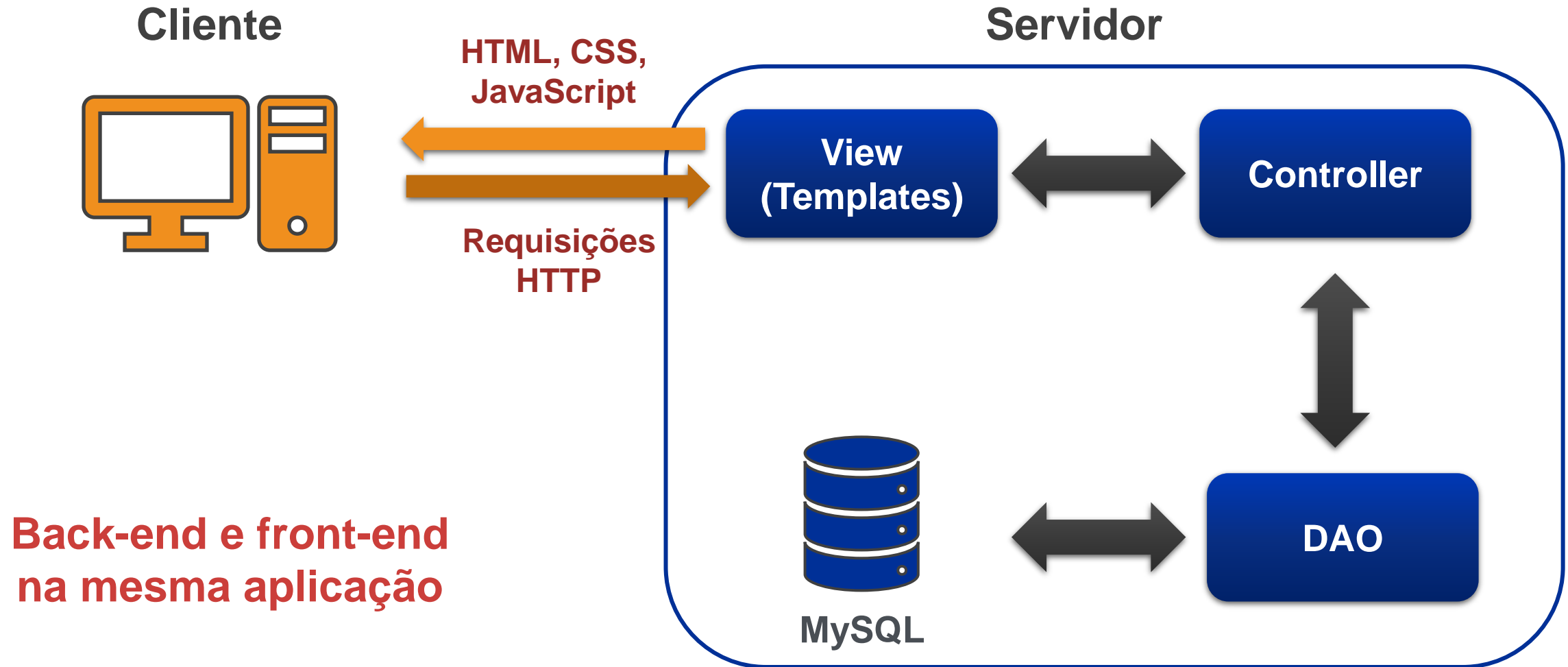
Introdução à arquitetura REST e APIs

- A arquitetura **REST** (***RE**presentational **State** **T**ransfer*) **define um conjunto de restrições** para a criação serviços web.
- Diferente de uma aplicação baseada em RPC (*Remote Procedure Call*), REST não define acesso a métodos/procedimentos, mas sim à recursos (objetos, JSON, XML, etc.), por meio de protocolos como o HTTP e identificadores (URLs) .

Introdução à arquitetura REST e APIs

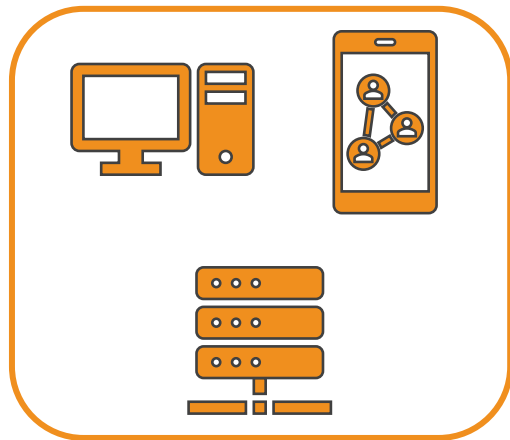
- Uma **API** (***A**pplication **P**rogramming **I**nterface*) é um **conjunto de definições e protocolos** para construção e integração de aplicações, e poder ser baseada na arquitetura REST.
- Por meio de uma API é possível trocar informações com outros softwares **sem precisar saber como eles foram implementados**.
- Recursos de uma API podem ser acessados por meio dos **endpoints** (URLs).
 - Exemplo: <http://localhost:8080/atendimento/> fornece acesso a lista de atendimentos.

Arquitetura de uma aplicação web

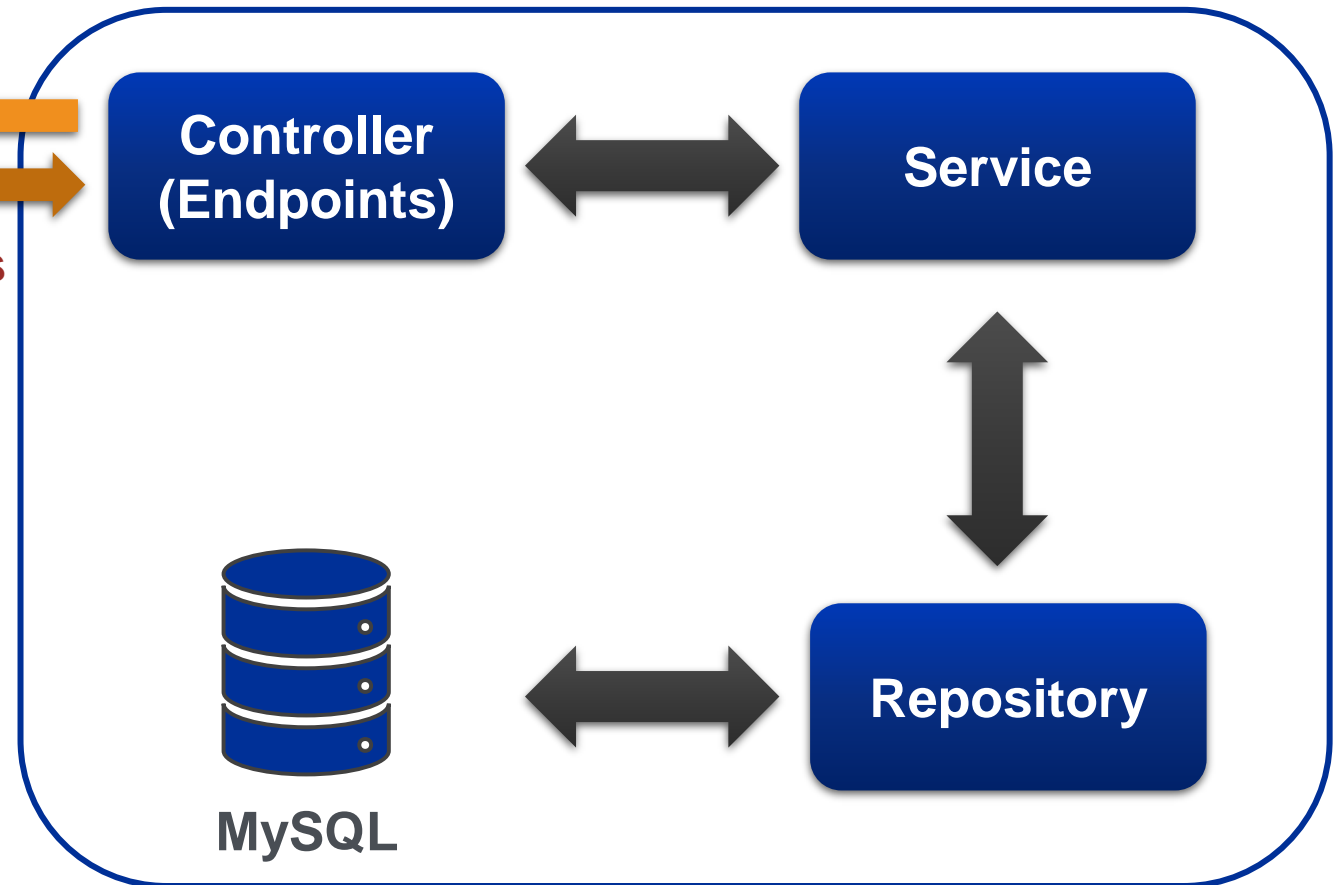


Arquitetura de uma API

Cliente (Front-end)



Servidor (Back-end)



JSON

Requisições
HTTP

Back-end e front-end
separados

Camada de serviço é necessária?

- Nem sempre é necessária, especialmente em aplicações simples.
- Separação de responsabilidades:
 - **Controller**: expõe os **endpoints**.
 - **Service**: lógica de negócio.
 - **Repository**: persistência e acesso aos dados.
- **Lógica de negócios** pode começar simples (operações CRUD), mas **pode ficar mais complexa**.

Mapeamento de requisições HTTP

- No Spring, o **@RequestMapping** é utilizado para **mapear requisições HTTP** feitas para URLs específicas, atribuindo a um método ou classe a tarefa de manipular estas requisições.

```
@RestController
@RequestMapping("/atendimento")
public class AtendimentoController implements IController<Atendimento> {
    @RequestMapping(value =("/{id}", method = RequestMethod.GET)
    public ResponseEntity<Atendimento> getId(@PathVariable("id") Long id) {
        Atendimento registro = servico.getId(id);
        return new ResponseEntity<>(registro, HttpStatus.OK);
    }
}
```

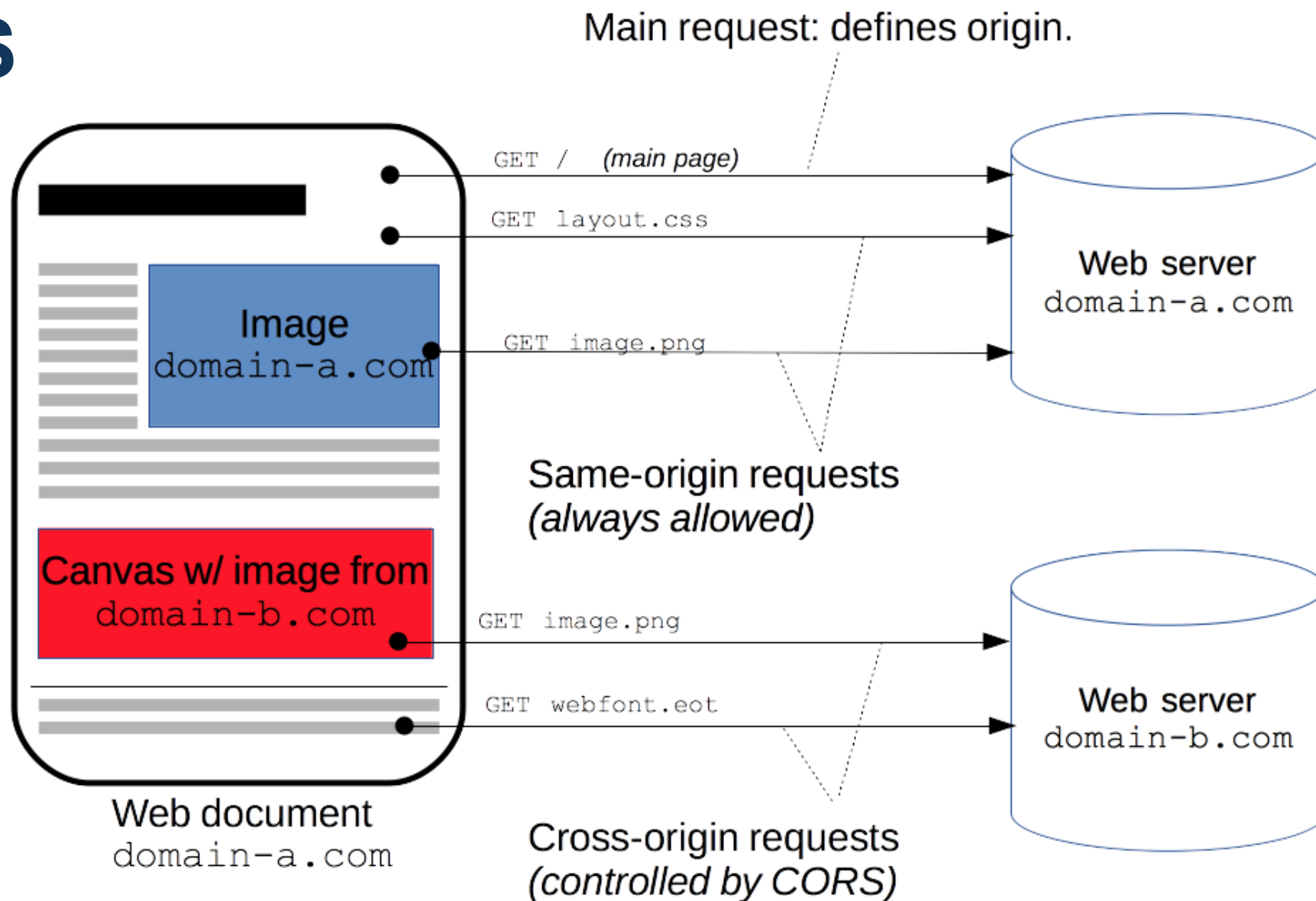
Mapeamento de requisições HTTP

Anotação	CRUD	Atalho para...
@GetMapping	READ	@RequestMapping(method = RequestMethod.GET)
@PostMapping	CREATE	@RequestMapping(method = RequestMethod.POST)
@PutMapping	UPDATE	@RequestMapping(method = RequestMethod.PUT)
@DeleteMapping	DELETE	@RequestMapping(method = RequestMethod.DELETE)

CORS

- **CORS** (**C**ross-**O**rigin **R**esource **S**haring) é um **mecanismo de segurança** que gerencia requisições entre domínios, **impedindo que scripts executem códigos maliciosos**.
- Uma requisição entre domínios é uma solicitação HTTP feita pelo navegador do **dominio-a.com** para o **dominio-b.com** por meio requisições assíncronas (AJAX).
- **Origem** é a combinação do **protocolo + porta + domínio** da solicitação.
 - **http://dominio-a.com:9000/** é diferente de **https://dominio-a.com:9000/**
- CORS é um **padrão em todos os navegadores modernos**.

CORS



Fonte: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS>


Requisições simples

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#simple_requests

Browser (<https://www.site.com>)


Request:

GET <https://www.api.com?q=test>
origin: <https://www.site.com>



Response:

HTTP/1.1 200 OK
access-control-allow-origin: <https://www.site.com>



Server (<https://www.api.com>)

Fonte: <https://www.baeldung.com/cs/cors-preflight-requests>

Requisições com pré-envio

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#preflighted_requests

Browser (<https://www.site.com>)

Pre-flight Request:

OPTIONS <https://www.api.com?q=test>
access-control-request-method: GET
access-control-request-headers: custom-header, ...
origin: <https://www.site.com>

Pre-flight Response:

HTTP/1.1 204 No Content

access-control-allow-origin: <https://www.site.com>
access-control-allow-methods: GET
access-control-allow-headers: custom-header, accept, ...
access-control-max-age: 6000

Request:

GET <https://www.api.com?q=test>

origin: <https://www.site.com>
custom-header: test

Response:

HTTP/1.1 200 OK

access-control-allow-origin: <https://www.site.com>

Server (<https://www.api.com>)

Fonte: <https://www.baeldung.com/cs/cors-preflight-requests>

CORS

@Bean

```
public CorsFilter corsFilter() {  
    CorsConfiguration corsConfig = new CorsConfiguration();  
    corsConfig.setAllowedOrigins(Arrays.asList("http://localhost:5500"));  
    corsConfig.setAllowedMethods(Arrays.asList("*"));  
    corsConfig.setAllowedHeaders(Arrays.asList("*"));  
    UrlBasedCorsConfigurationSource configSource = new UrlBasedCorsConfigurationSource();  
    configSource.registerCorsConfiguration("/**", corsConfig);  
    return new CorsFilter(configSource);  
}
```


Segurança

Usuário e senha sendo capturados no Wireshark

http											
	Time	Source	Destination	Protocol	Length	Info					
81	10.751746988	10.0.2.15	176.28.50.165	HTTP	549	[TCP Prev]					
87	11.023756075	176.28.50.165	10.0.2.15	HTTP	71	HTTP/1.1 2					
89	11.112734635	10.0.2.15	176.28.50.165	HTTP	480	GET /Flash					
92	104.364660492	10.0.2.15	192.16.58.8	OCSP	485	Request					
94	104.405805104	192.16.58.8	10.0.2.15	OCSP	842	Response					
33	104.866913950	10.0.2.15	177.69.134.249	HTTP	342	GET /succe					
35	104.906092022	177.69.134.249	10.0.2.15	HTTP	438	HTTP/1.1 2					
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n											
Accept-Language: en-US,en;q=0.5\r\n											
Accept-Encoding: gzip, deflate\r\n											
Referer: http://testphp.vulnweb.com/login.php\r\n											
Content-Type: application/x-www-form-urlencoded\r\n											
0120	65 6e 3b 71 3d 30 2e 35 0d 0a	41 63 63 65 70 74	en;q=0.5 ··Accept								
0130	2d 45 6e 63 6f 64 69 6e 67 3a	20 67 7a 69 70 2c	-Encodin g: gzip,								
0140	20 64 65 66 6c 61 74 65 0d 0a	52 65 66 65 72 65	deflate ··Refere								
0150	72 3a 20 68 74 74 70 3a 2f 2f	74 65 73 74 70 68	r: http: //testph								
0160	70 2e 76 75 6c 6e 77 65 62 2e	63 6f 6d 2f 6c 6f	p.vulnwe b.com/lo								
0170	67 69 6e 2e 70 68 70 0d 0a 43	6f 6e 74 65 6e 74	gin.php ··Content								
0180	2d 54 79 70 65 3a 20 61 70 70	6c 69 63 61 74 69	-Type: a pplicati								
0190	6f 6e 2f 78 2d 77 77 77 2d 66	6f 72 6d 2d 75 72	on/x-www -form-ur								
01a0	6c 65 6e 63 6f 64 65 64 0d 0a	43 6f 6e 74 65 6e	lencoded ··Conten								
01b0	74 2d 4c 65 6e 67 74 68 3a 20	32 30 0d 0a 43 6f	t-Length : 20 ··Co								
01c0	6f 6b 69 65 3a 20 6c 6f 67 69	6e 3d 74 65 73 74	okie: lo gin=test								
01d0	25 32 46 74 65 73 74 0d 0a 43	6f 6e 6e 65 63 74	%2Ftest ··Connect								
01e0	69 6f 6e 3a 20 6b 65 65 70 2d	61 6c 69 76 65 0d	ion: kee p-alive ·								
01f0	0a 55 70 67 72 61 64 65 2d 49	6e 73 65 63 75 72	·Upgrade -Insecur								
0200	65 2d 52 65 71 75 65 73 74 73	3a 20 31 0d 0a 0d	e-Request: 1 ··								
0210	0a 75 6e 61 6d 65 3d 74 65 73	74 26 70 61 73 73	·uname=t est&pass								
0220	3d 74 65 73 74		=test								
eth0: <live capture in progress>											

Fonte: <https://tavernalinix.com/wireshark-capturando-pacotes-de-login-e-senha-do-telnet-e-http-com-wireshark-3180f7bd2f9>

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).

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`

Fim!



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.], 2023. Disponível em:
<https://developer.mozilla.org/pt-BR/docs/Learn>.
- SPRING (ed.). **Spring Boot Reference Documentation**. [S. l.], 2023. 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.