Programação Back-end

Prof: Rodrigo da Cruz Fujioka

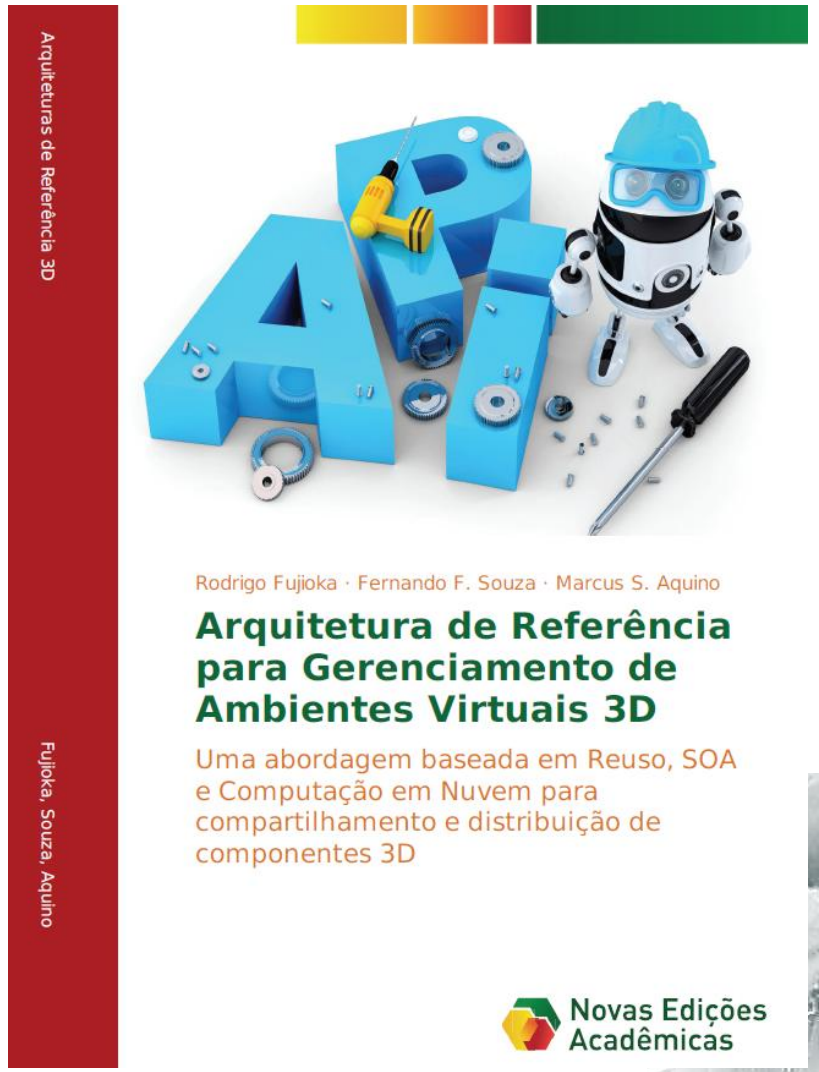
Lattes: http://**lattes**.cnpq.br/0843668802633139

Versão: 1.1 – 29/06/2024

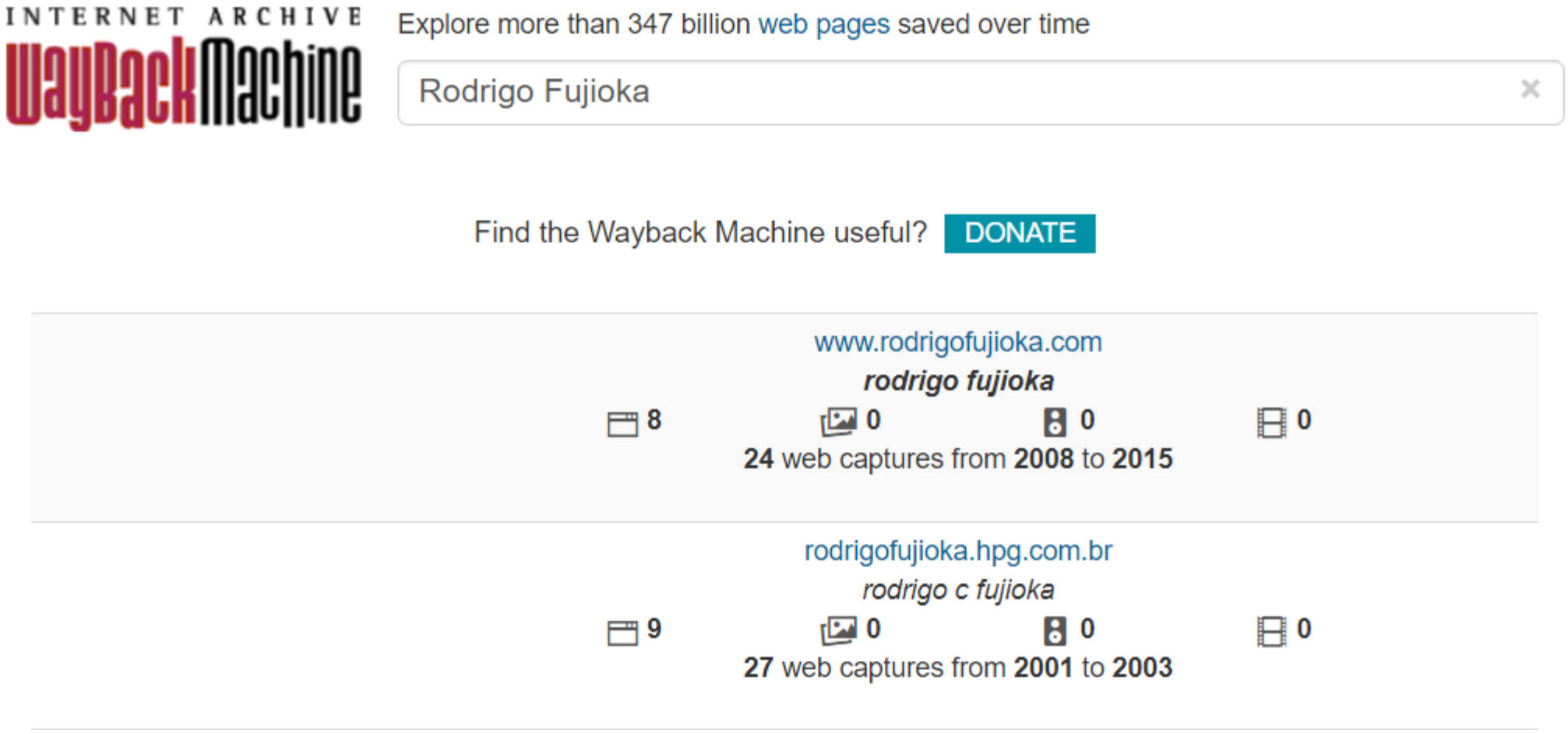
Centro de Informática U·F·P·E

UFPB

FGV management



Profissionalmente com TI desde 2007
Primeira página na internet em **1998** – Geocities
Ferramentas: Frontpage express e notepad



SENADO FEDERAL

Capgemini

ANYMARKET

INEP
Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira

PUCPR
GRUPO MARISTA

alelo

VOX TECNOLOGIA

pagseguro UOL

DB1 GROUP

INPI INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL

invillia
infinite.digital.power

PublicSoft
Desenvolvendo Soluções para o Setor Público

Meta
Digital. Simple. Human.

bradesco

Facisa

UNIPÊ
Centro Universitário de João Pessoa

PROCERGS

SIMPLESTEC
e-government Solutions

MINISTÉRIO DA JUSTIÇA E SEGURANÇA PÚBLICA

Avaty!
TECNOLOGIA E INOVAÇÃO

tryness
Technology

Kanban System Design (KMP I)

Kanban Systems Improvement (KMP II)

Microsoft Certified: Azure Administrator Associate

Microsoft

Certified SAFe® 5 Scrum Master

Scaled Agile Inc

Microsoft Certifiec Azure AI Fundamentals

Microsoft

Microsoft Certified: Azure Fundamentals

Microsoft

Microsoft Certified: Azure Data Fundamentals

Microsoft

Classes Taken

Class Name	
Kanban Coaching Practices	View Certificate
Kanban Maturity Model	View Certificate
Kanban Systems Improvement	View Certificate
Kanban System Design	View Certificate

TOTVS

EBSCO
INFORMATION SERVICES

OpenAthens

Minha Biblioteca .com.br

SODA Virtual
SODA
Digital Agency
Mobile, Web & Social Media
Felicidade, Brazil
www.sodavirtual.com.br

PEARSON
Education

CRM Educacional
Captação, Retenção e Fidelização de alunos.

EMPREL
EMPRESA MUNICIPAL DE INFORMÁTICA

FACIMED

vlex
Toda la información jurídica. Un único sistema de búsqueda
42.700 clientes 105.534.275 documentos 1142 editoriales 13 idiomas 132 países

Sobre o Professor (Redes Sociais)

 @rodrigofujioka

 @fuji_bjj

Conteúdo

Conteúdo

- **10 de Maio**
- Apresentação
- Reforço dos conceitos EndPoints, Apis, Verbos HTTP, Micro Serviços
- Swagger
- @Query
- Beans Validators
- Lombok
- Cache
- Integração com serviços externos

- **07 de Junho**
- Custon Bean Validators
- Handle Exceptions
- Geração de testes unitários com I.A
- DTO e Spring Schedule

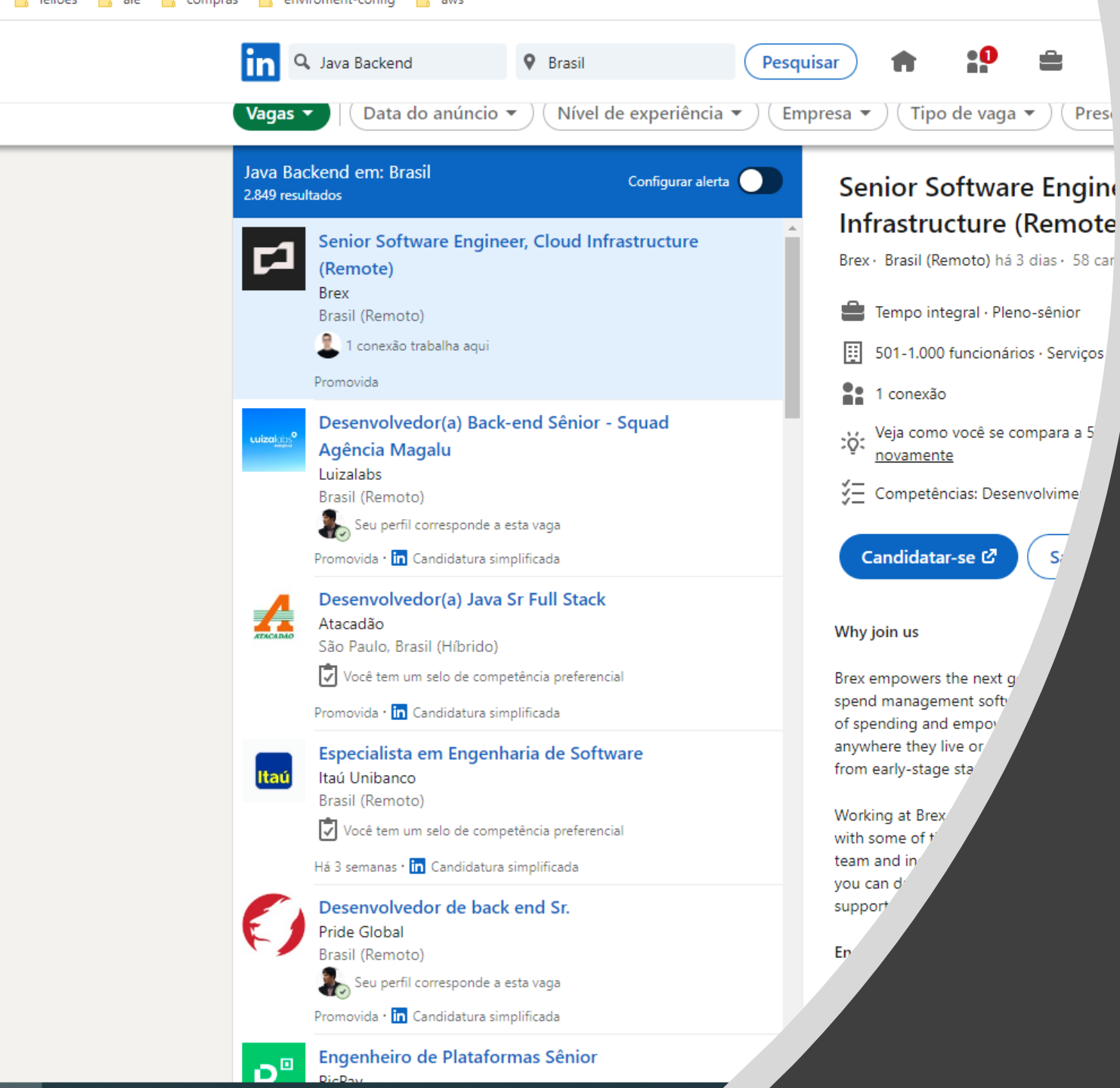
14 de junho

- Boas práticas de programação
- Clean Code e Orientações

Avaliação

1 – Aulas práticas

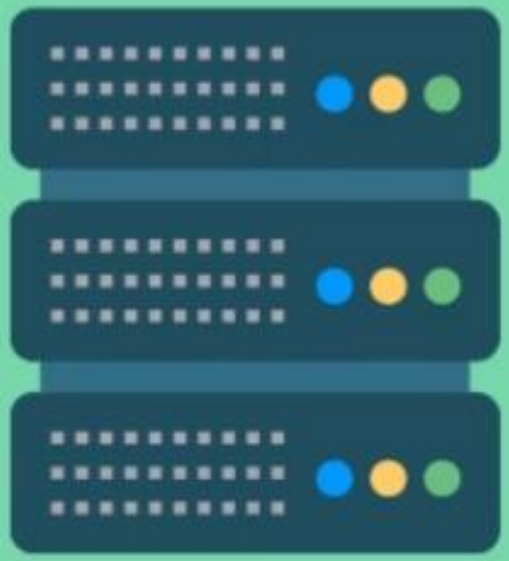
2 – Entrega do repositório git no ambiente virtual.


Java Backend
Brasil
Pesquisar
Vagas
Data do anúncio
Nível de experiência
Empresa
Tipo de vaga
Java Backend em: Brasil
2.849 resultados
Configurar alerta
Senior Software Engineer, Cloud Infrastructure (Remote)
Brex
Brasil (Remoto)
1 conexão trabalha aqui
Promovida
Desenvolvedor(a) Back-end Sênior - Squad Agência Magalu
Luizalabs
Brasil (Remoto)
Seu perfil corresponde a esta vaga
Promovida · Candidatura simplificada
Desenvolvedor(a) Java Sr Full Stack
Atacadão
São Paulo, Brasil (Híbrido)
Você tem um selo de competência preferencial
Promovida · Candidatura simplificada
Especialista em Engenharia de Software
Itaú Unibanco
Brasil (Remoto)
Você tem um selo de competência preferencial
Há 3 semanas · Candidatura simplificada
Desenvolvedor de back end Sr.
Pride Global
Brasil (Remoto)
Seu perfil corresponde a esta vaga
Promovida · Candidatura simplificada
Engenheiro de Plataformas Sênior
Senior Software Engin
Infrastructure (Remote
Brex · Brasil (Remoto) há 3 dias · 58 car
Tempo integral · Pleno-sênior
501-1.000 funcionários · Serviços
1 conexão
Veja como você se compara a 5
novamente
Competências: Desenvolvime
Candidatar-se
Why join us
Brex empowers the next g
spend management soft
of spending and empo
anywhere they live or
from early-stage sta
Working at Brex
with some of t
team and in
you can d
support
En

Motivação

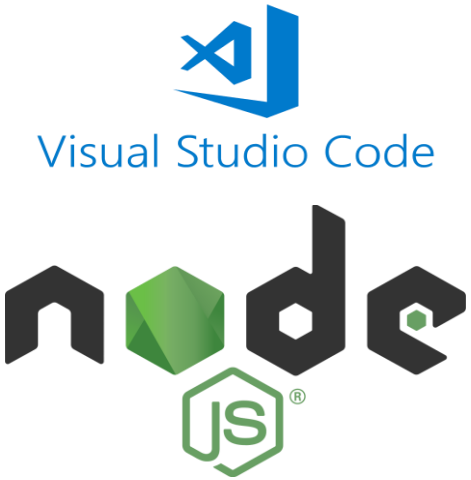
Ué e agora ? Todos sabem essa separação ?

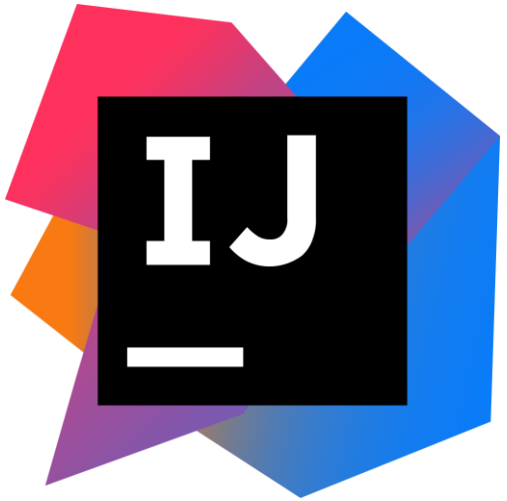
FRONT END



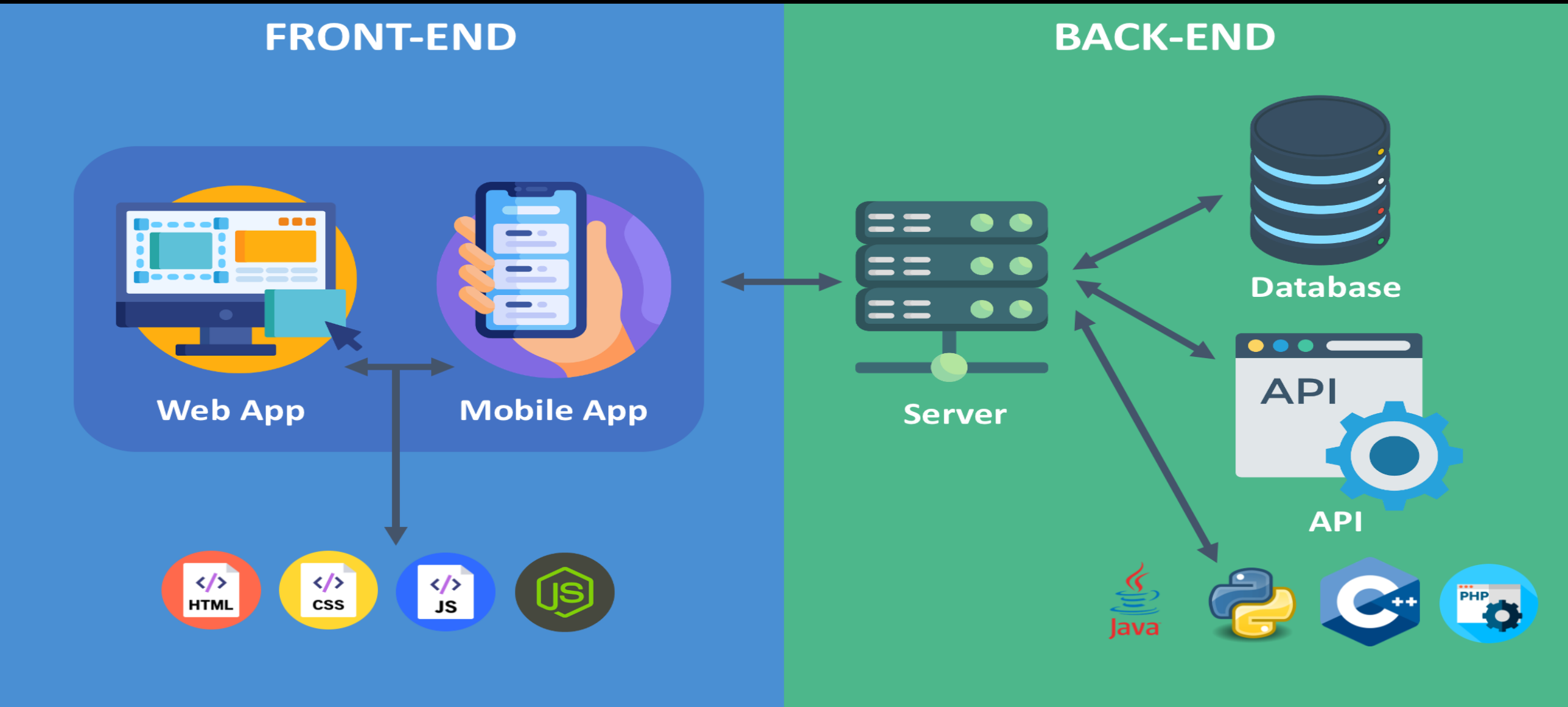
BACK END

Quais ferramentas





Vamos entender mais no que vamos trabalhar!



editor.swagger.io

Arquitetura | gov | EDU | Agile | Certification Exam... | mec | leiloes

https://editor.swagger.io/

Swagger Editor Supported by SMARTBEAR — File ▾ Edit ▾ Insert ▾ Generate Server ▾ Generate Client ▾

```yaml
openapi: 3.0.3
info:
  title: Swagger Petstore - OpenAPI 3.0
  description: |-
    This is a sample Pet Store Server based on the OpenAPI 3.0 specification.  You can find out more about
    Swagger at [https://swagger.io](https://swagger.io). In the third iteration of the pet store, we've switched to the design first approach!
    You can now help us improve the API whether it's by making changes to the definition itself or to the code.
    That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

    _If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](https://editor.swagger.io/?url=https://petstore.swagger.io/v2/swagger.yaml). Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` menu option!_

    Some useful links:
    - [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)
    - [The source API definition for the Pet Store](https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml)
  termsOfService: http://swagger.io/terms/
  contact:
    email: apiteam@swagger.io
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.11
externalDocs:
  description: Find out more about Swagger
  url: http://swagger.io
servers:
  - url: https://petstore3.swagger.io/api/v3
tags:
  - name: pet
    description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: http://swagger.io
  - name: store
    description: Access to Petstore orders
```

Servers: https://petstore3.swagger.io/api/v3 — Authorize

pet Everything about your Pets — Find out more: http://swagger.io

- PUT /pet Update an existing pet
- POST /pet Add a new pet to the store
- GET /pet/findByStatus Finds Pets by status
- GET /pet/findByTags Finds Pets by tags
- GET /pet/{petId} Find pet by ID
- POST /pet/{petId} Updates a pet in the store with form data
- DELETE /pet/{petId} Deletes a pet
- POST /pet/{petId}/uploadImage uploads an image

Vamos aprender a documentar APIS com Swagger

Verbos HTTP

GET

O método `GET` solicita a representação de um recurso específico. Requisições utilizando o método `GET` devem retornar apenas dados.

HEAD

O método `HEAD` solicita uma resposta de forma idêntica ao método **`GET`**, porém sem conter o corpo da resposta.

POST

O método `POST` é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.

Verbos HTTP

PUT

O método `PUT` substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.

DELETE

O método `DELETE` remove um recurso específico.

CONNECT

O método `CONNECT` estabelece um túnel para o servidor identificado pelo recurso de destino.

Verbos HTTP

`OPTIONS`

O método `OPTIONS` é usado para descrever as opções de comunicação com o recurso de destino.

`TRACE`

O método `TRACE` executa um teste de chamada *loop-back* junto com o caminho para o recurso de destino.

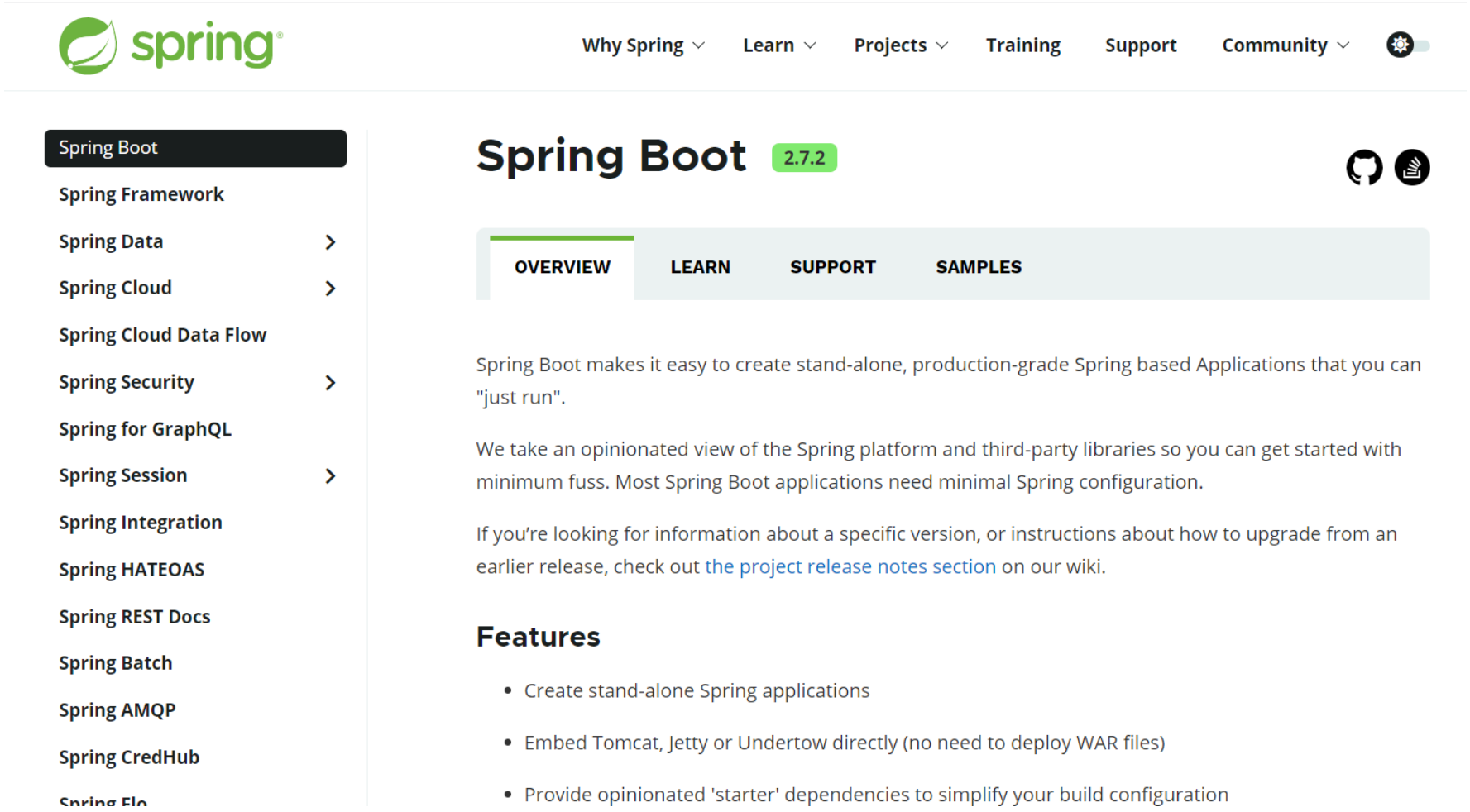
`PATCH`

O método `PATCH` é utilizado para aplicar modificações parciais em um recurso

Back end

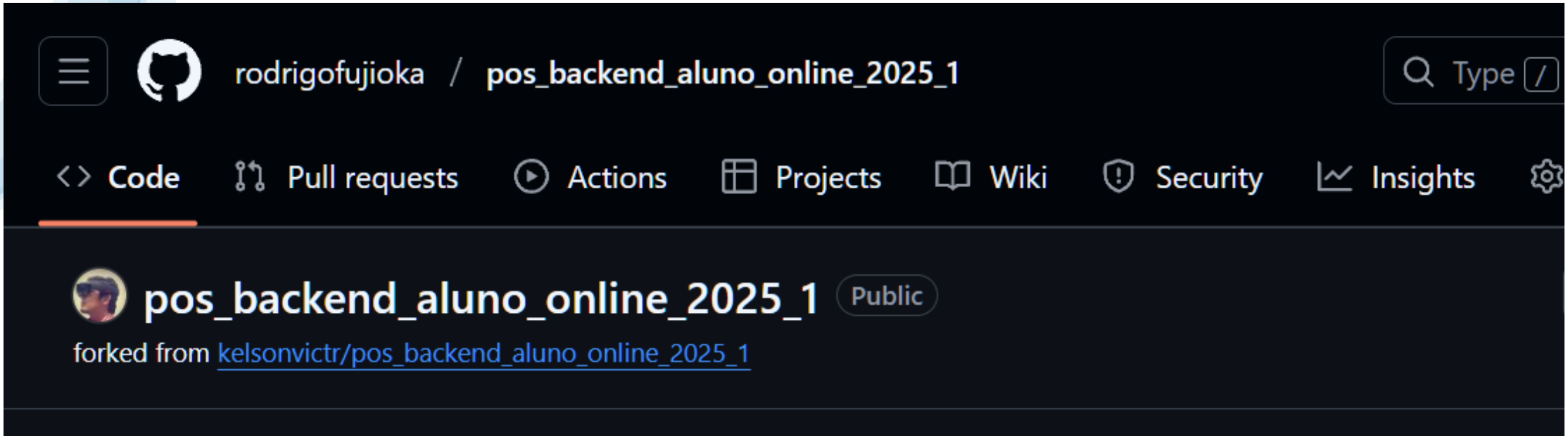
RxJava


Spring Boot
MICRONAUT®
QUARKUS
RxJava
helidon.io
kumuluzEE
Armeria
VERT.X
lagom
Open Liberty
Dropwizard
Spark
payara®
MICRO
AxonFramework



Tecnologia que vamos adotar!

Mão na massa! Prática!



https://github.com/rodrigofujioka/pos_backend_aluno_online_2025_1

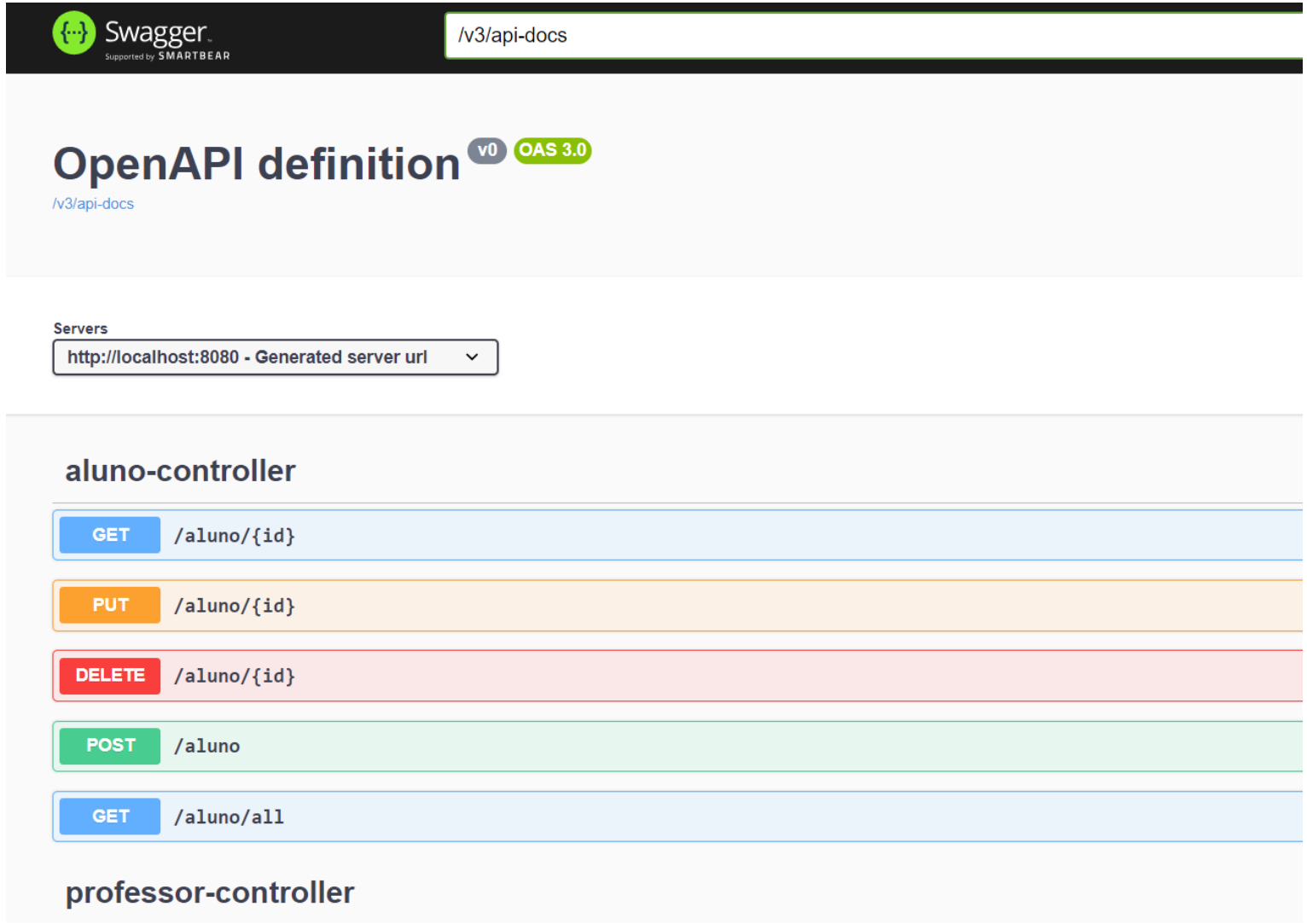
Fork do projeto

```java
@RestController
@RequestMapping("/disciplina")
public class DisciplinaController {

    @Autowired
    DisciplinaService disciplinaService;

    @PostMapping
```

```java
@Slf4j
@AllArgsConstructor
@RestController
@RequestMapping("/aluno")
public class AlunoController {

    private AlunoService alunoService;
```

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
</dependency>

<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.5.0</version>
</dependency>
```

http://localhost:8080/swagger-ui/index.html



Queries JPQL com Spring Data

```java
package br.com.alunoonline.api.repository;

import br.com.alunoonline.api.model.Aluno;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface AlunoRepository extends JpaRepository<Aluno, Long> {

    @Query("SELECT a FROM Aluno a WHERE a.nome = :nome")
    List<Aluno> findByName(@Param("nome") String nome);
}
```

Consultas Avançadas com JPQL

```java
import br.com.alunoonline.api.model.MatriculaAluno;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface MatriculaAlunoRepository extends JpaRepository<MatriculaAluno, Long> {
    List<MatriculaAluno> findByAlunoId(Long alunoId);

    @Query("SELECT m FROM MatriculaAluno m JOIN m.aluno a WHERE a.nome = :nome")
    List<MatriculaAluno> findMatriculasByAlunoNome(@Param("nome") String nome);

}
```

Entidade Base - Produto

```java
@Data
@Entity
public class Produto {
    @Id @GeneratedValue
    private Long id;
    private String nome;
    private Double preco;
    private String categoria;
}
```

```java
public interface ProdutoRepository extends JpaRepository<Produto, Long> {
    // Consultas JPQL e métodos derivados
}
```

Repositório Base

Buscar produtos ordenados por nome

```
1    //Spring Data:
2    List<Produto> findAllByOrderByNomeAsc();
3
4    //JPQL:
5    @Query("SELECT p FROM Produto p ORDER BY p.nome ASC")
6    List<Produto> buscarTodosOrdenadosPorNome();
7
```

Buscar produtos com LIKE no nome

```
1   //Spring Data:
2   List<Produto> findByNomeContainingIgnoreCase(String nome);
3
4   //JPQL:
5   @Query("SELECT p FROM Produto p WHERE LOWER(p.nome) LIKE LOWER(CONCAT('%', :nome, '%'))")
6   List<Produto> buscarPorNomeAproximado(String nome);
7
```

Buscar por categoria e preço mínimo

```java
//Spring Data:
List<Produto> findByCategoriaAndPrecoGreaterThan(String categoria, Double preco);

//JPQL:
@Query("SELECT p FROM Produto p WHERE p.categoria = :categoria AND p.preco > :preco")
List<Produto> buscarPorCategoriaEPrecoMinimo(String categoria, Double preco);

```

Buscar por faixa de preço

```java
//Spring Data:
List<Produto> findByPrecoBetween(Double min, Double max);

//JPQL:
@Query("SELECT p FROM Produto p WHERE p.preco BETWEEN :min AND :max")
List<Produto> buscarPorFaixaDePreco(Double min, Double max);
```

Buscar por nome OU categoria

```java
//Spring Data:
List<Produto> findByNomeContainingOrCategoriaContaining(String nome, String categoria);

//JPQL:
@Query("SELECT p FROM Produto p WHERE p.nome LIKE %:nome% OR p.categoria LIKE %:categoria%")
List<Produto> buscarPorNomeOuCategoria(String nome, String categoria);

```

```java
List<Produto> findByNomeLike(String nome);
```

Dicas para a Aula

- Use banco H2 em memória para testes rápidos

- Ative logs de queries: spring.jpa.show-sql=true

- Compare legibilidade entre JPQL e métodos derivados

Referências

- - Spring Docs: spring.io/spring-data/jpa/
- - Baeldung JPQL: baeldung.com/jpa-queries
- - Query Methods: spring.io/reference/html/#jpa.query-methods

Questão 1: Consulta Simples

Enunciado: Crie uma consulta JPQL para buscar todos os alunos cujo nome é "Maria".

Questão 1: Consulta Simples

Enunciado: Crie uma consulta JPQL para buscar todos os alunos cujo nome é "Maria".

```
@Query("SELECT a FROM Aluno a WHERE a.name=:nome")
List<Aluno> findByName(@Param("nome") String nome);
```

Questão 2: Consulta com Join

Enunciado: Crie uma consulta JPQL para buscar todas as matrículas dos alunos na disciplina de "Matemática".

Questão 2: Consulta com Join

Enunciado: Crie uma consulta JPQL para buscar todas as matrículas dos alunos na disciplina de "Matemática".

```java
@Query("SELECT m FROM MatriculaAluno m JOIN m.disciplina d WHERE d.name = :nome")
List<MatriculaAluno> findMatriculasByDisciplinaNome(@Param("nome") String nome);
```

Questão 3: Consulta com Ordenação

Enunciado: Crie uma consulta JPQL para listar todas as disciplinas ordenadas pelo nome.

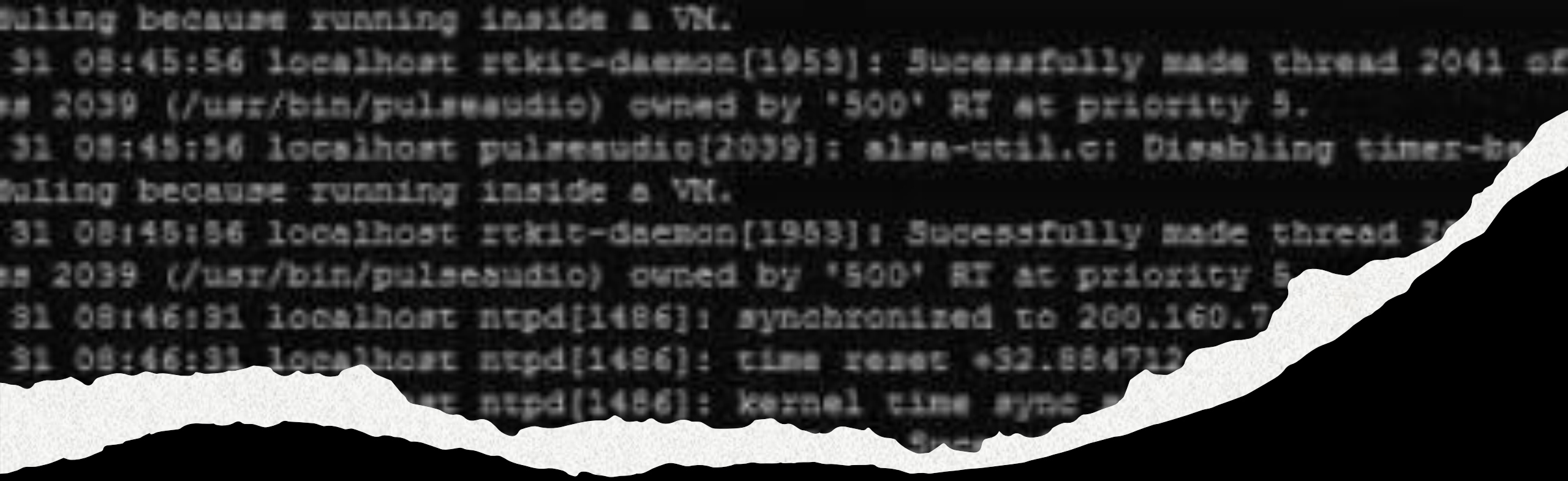
Questão 3: Consulta com Ordenação

Enunciado: Crie uma consulta JPQL para listar todas as disciplinas ordenadas pelo nome.

```
@Query("SELECT d FROM Disciplina d ORDER BY d.name")
List<Disciplina> findAllOrderByNome();
```

Questão 4: Consulta com Condicional

Enunciado: Crie uma consulta JPQL para listar todos os alunos que nasceram em 1982 ou após


uling because running inside a VM.
31 08:45:56 localhost rtkit-daemon[1953]: Successfully made thread 2041 of
es 2039 (/usr/bin/pulseaudio) owned by '500' RT at priority 5.
31 08:45:56 localhost pulseaudio[2039]: alsa-util.c: Disabling timer-ba
uling because running inside a VM.
31 08:45:56 localhost rtkit-daemon[1953]: Successfully made thread 2
es 2039 (/usr/bin/pulseaudio) owned by '500' RT at priority 5.
31 08:46:31 localhost ntpd[1486]: synchronized to 200.160.7
31 08:46:31 localhost ntpd[1486]: time reset +32.884712
ntpd[1486]: kernel time sync

LOGS

Introdução ao @Slf4j

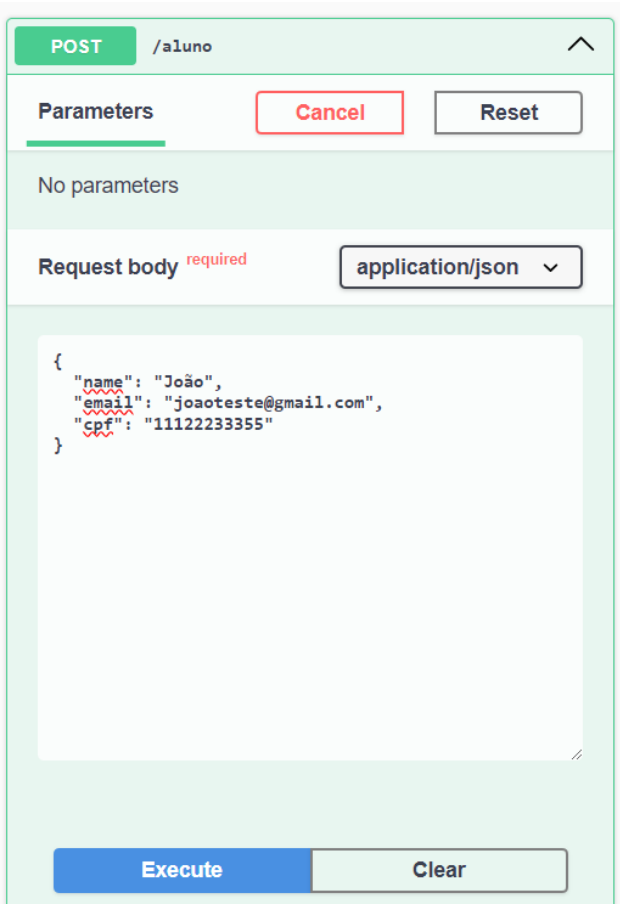
A anotação @Slf4j do Lombok adiciona automaticamente um logger (SLF4J) à sua classe.

O que é SLF4J?

- SLF4J (Simple Logging Facade for Java) é uma API de fachada para várias bibliotecas de logging.

- Oferece uma interface simples e uniforme para diferentes frameworks de logging como Logback, Log4j, e Java Util Logging.

```java
@Slf4j
@RestController
@RequestMapping("/aluno")
public class AlunoController {

    @Autowired
    AlunoService alunoService;

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void create(@RequestBody Aluno aluno) {
        log.info("Iniciando criação de aluno");
        alunoService.create(aluno);
        log.info("Encerrando criação de aluno");
    }
```




```
Hibernate:
    select
        a1_0.id,
        a1_0.cpf,
        a1_0.email,
        a1_0.name
    from
        aluno a1_0
2024-06-29T04:53:39.781-03:00  INFO 2144 --- [nio-8080-exec-2] b.c.a.api.controller.AlunoController     : Iniciando criação de aluno
Hibernate:
    insert
    into
        aluno
        (cpf, email, name, id)
    values
        (?, ?, ?, default)
2024-06-29T04:53:39.810-03:00  INFO 2144 --- [nio-8080-exec-2] b.c.a.api.controller.AlunoController     : Encerrando criação de aluno
```

Níveis de Log

- **Explicação dos Níveis de Log:**

- **trace**: Informação mais detalhada, geralmente usada para depuração muito granular.

- **debug**: Informações de depuração, útil durante o desenvolvimento.

- **info**: Informações gerais do sistema, geralmente usado em produção para rastrear o fluxo de execução.

- **warn**: Alertas sobre potenciais problemas que não interrompem o funcionamento do sistema.

- **error**: Erros que afetam a execução da aplicação.

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

Beans Validators

O que é Bean Validation?

- **Bean Validation** é uma especificação da JSR 380 (Bean Validation 2.0) que define um conjunto de anotações para validar propriedades de beans.

- É usado para garantir que os dados dos beans estejam corretos e consistentes.

Por que usar Bean Validation?

- Facilita a validação de dados.
- Integra-se facilmente com frameworks como Spring Boot.

- **Principais Anotações:**
  - **`@NotNull`**: Valida que o campo não seja nulo.
  - **`@NotEmpty`**: Valida que o campo não seja nulo ou vazio.
  - **`@NotBlank`**: Valida que o campo não seja nulo e contenha pelo menos um caractere não-espaço.
  - **`@Size`**: Valida o tamanho de uma coleção, array, string, etc.
  - **`@Min`** e **`@Max`**: Valida valores numéricos mínimos e máximos.
  - **`@Email`**: Valida se o campo é um e-mail.
  - **`@Pattern`**: Valida se o campo corresponde a um padrão regex.

```java
@NoArgsConstructor
@Data
@Entity
public class Aluno implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank(message = "O nome não pode ser em branco")
    @Size(min = 2, max = 30, message = "O nome deve ter entre 2 e 30 caracteres")
    private String name;

    private String email;

    private String cpf;

    private Integer anoNascimento;
}
```

```java
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/alunos")
public class AlunoController {

    @PostMapping
    public ResponseEntity<?> createAluno(@Valid @RequestBody Aluno aluno, BindingResult result) {
        if (result.hasErrors()) {
            Map<String, String> errors = new HashMap<>();
            result.getFieldErrors().forEach(error -> errors.put(error.getField(), error.getDefaultMessage()));
            return new ResponseEntity<>(errors, HttpStatus.BAD_REQUEST);
        }
        // Código para salvar o aluno
        return new ResponseEntity<>(aluno, HttpStatus.CREATED);
    }
}

```

```
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<Map<String, String>>
handleValidationExceptions(MethodArgumentNotValidException ex) {
        Map<String, String> erros = new HashMap<>();
        ex.getBindingResult().getFieldErrors().forEach(error ->
            erros.put(error.getField(), error.getDefaultMessage()));
        return ResponseEntity.badRequest().body(erros);
    }
}
```

Atividade

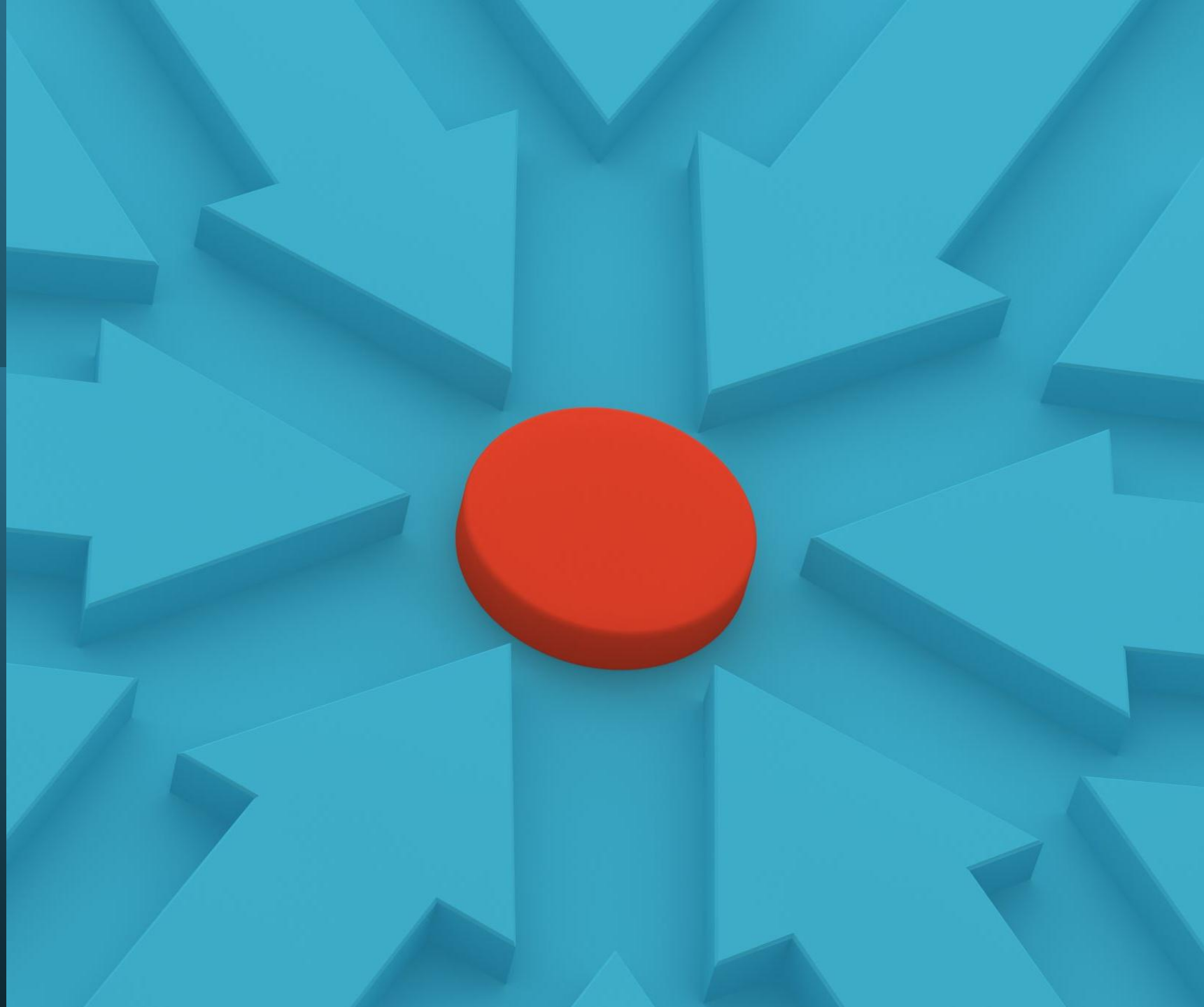
Você e seu grupo (X pessoas)

1. Inclua logs em toda a aplicação.
2. Ajuste toda sua aplicação para remover autowired
3. Incluao beans validators nos locais que achar apropriado dado o contexto da aplicação, por exemplo validação no e-mail ou nomes.

Próximos assuntos

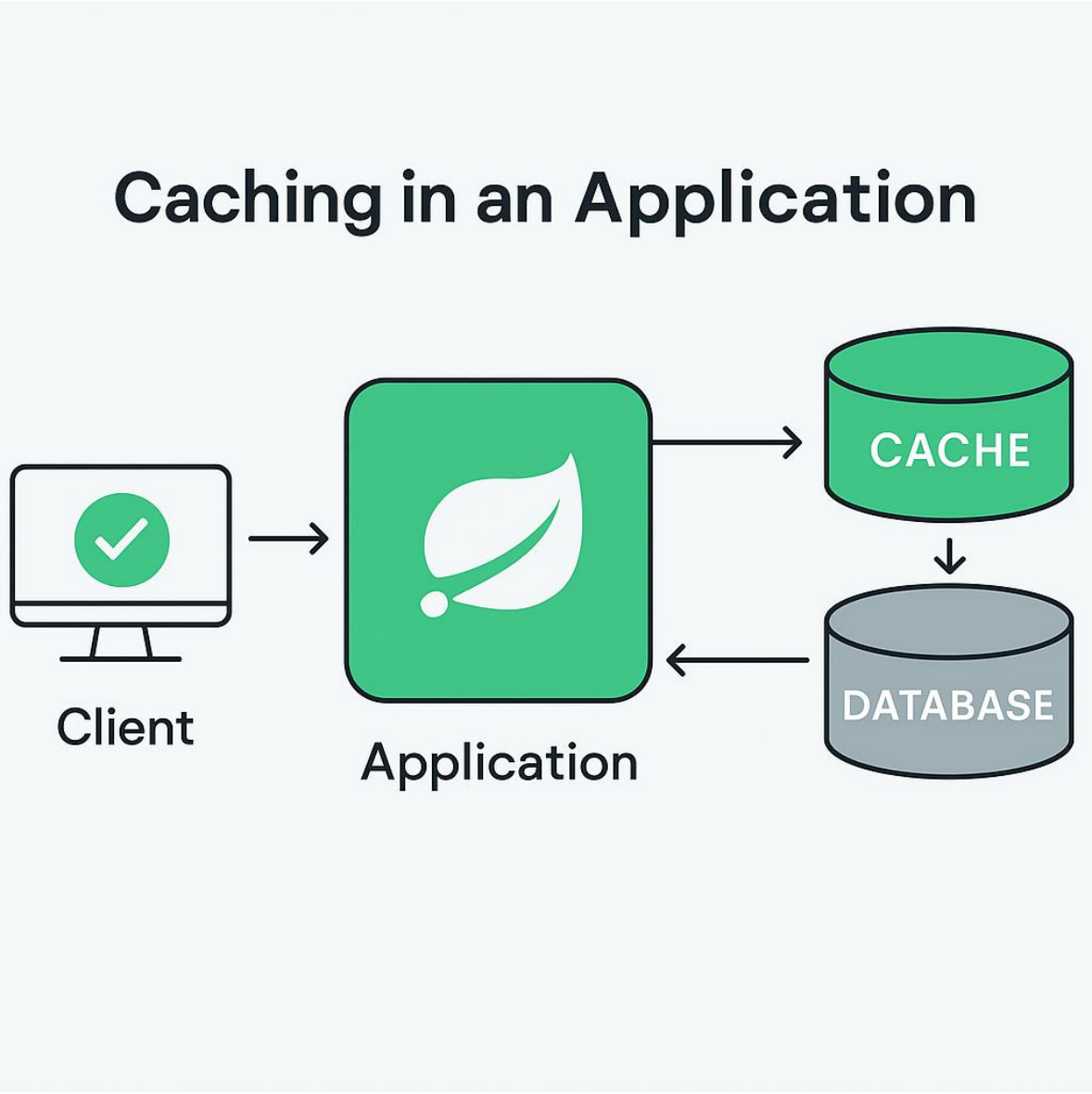
- **Próximos encontros**
- Custon Bean Validators
  Integração com serviços externos
- Handle Exceptions
- **Scheduling Tasks**
- DTO e Boas práticas de programação
- Testes unitários com I.A

Continuação



```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-cache</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
        <version>4.1.0</version>
    </dependency>

</dependencies>
```

CACHE


Caching in an Application
CACHE
DATABASE
Client
Application

```
@SpringBootApplication
@EnableFeignClients
@EnableCaching
public class AlunoOnlineApplication {
```

```java
@Log4j2
@RestController
@RequestMapping("/cache")
public class CacheController {

    @Autowired
    private CacheManager cacheManager;

    @GetMapping
    @Cacheable("CACHE_PALAVRA")
    public String getPalavra(){
        log.info("GET PALAVRA ACIONADO");
        return "Texto Cacheado";
    }

    @DeleteMapping
    public void resetCache(){
        cacheManager.getCache( name: "CACHE_PALAVRA").clear();
    }
```

```java
@Cacheable(value = "categoriasPorGenero", key = "#sexo + '-' + #ano")
public List<String> listarCategoriaClassePorGeneroEAno(String sexo, Integer ano) {
    return repository.listarCategoriaClasseOrdenadaPorIdadeEPeso(sexo, ano).stream()
            .map( String obj -> String.valueOf(obj)) // obj é a categoria_classe
            .collect(Collectors.toCollection(LinkedHashSet::new)) // remove duplicados, preserva ordem
            .stream().toList(); // transforma novamente em List
}

@Cacheable(value = "rankingPorClasseEAno", key = "#categoriaClasse + '-' + #ano")
public List<Ranking> listarRankingPorClasseEAno(String categoriaClasse, Integer ano) {
    return repository.listarPorCategoriaClasseEAno(categoriaClasse, ano);
}
```

```java
@CacheEvict(value = {"categoriasPorGenero", "rankingPorClasseEAno",
        "categoriasPrefixoPorGenero", "categoriasClasseFiltrada",
        "resultados"}, allEntries = true)
public void limparTodosCaches() {
    log.info("Todos os caches foram limpos.");
}
```

Utilização de Caches e Alternativas

Aplique o Cache em toda sua aplicação nos locais onde exista oportunidade, também atualização reset do cache quando um item da lista for atualizado, removido ou incluído.

Introdução ao OpenFeign

Introdução ao OpenFeign - Compreender o que é o OpenFeign e seus benefícios.

Configuração Básica - Configurar o OpenFeign em um projeto Spring Boot.

Criar Clientes Feign - Aprender a criar clientes Feign para consumir APIs externas.

Personalização e Avançado - Explorar personalizações avançadas como interceptores, encoders e decoders.

```
Vantagens do OpenFeign:
- Simplifica a criação de clientes HTTP.
- Integração com sistemas de descoberta de serviços como Eureka.
- Suporta balanceamento de carga com Ribbon.
- Codificação e decodificação automática de mensagens.
```

Introdução ao OpenFeign

O OpenFeign é uma ferramenta de cliente HTTP declarativo, projetada para simplificar a codificação de chamadas HTTP em microserviços. Desenvolvido pelo Netflix, o OpenFeign usa anotações para configurar como as chamadas HTTP devem ser feitas.

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
    <version>3.1.9</version>
</dependency>
```

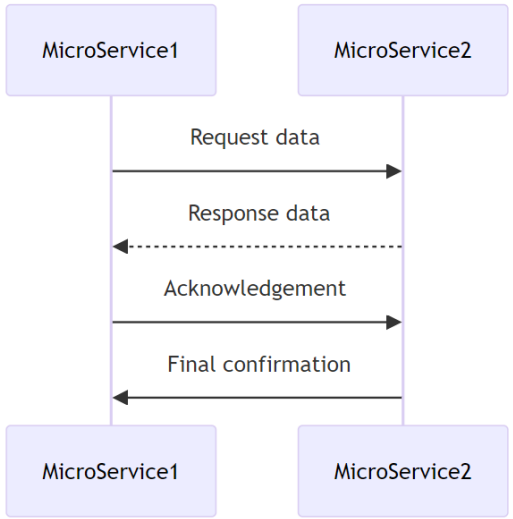
- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-openfeign/3.1.9
- 3.1.9

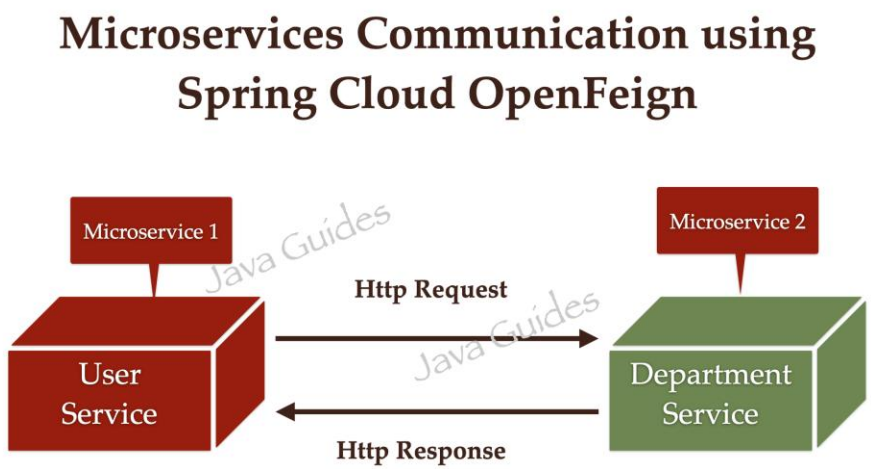
Habilitar Feign Clients na classe principal do aplicativo Spring Boot:

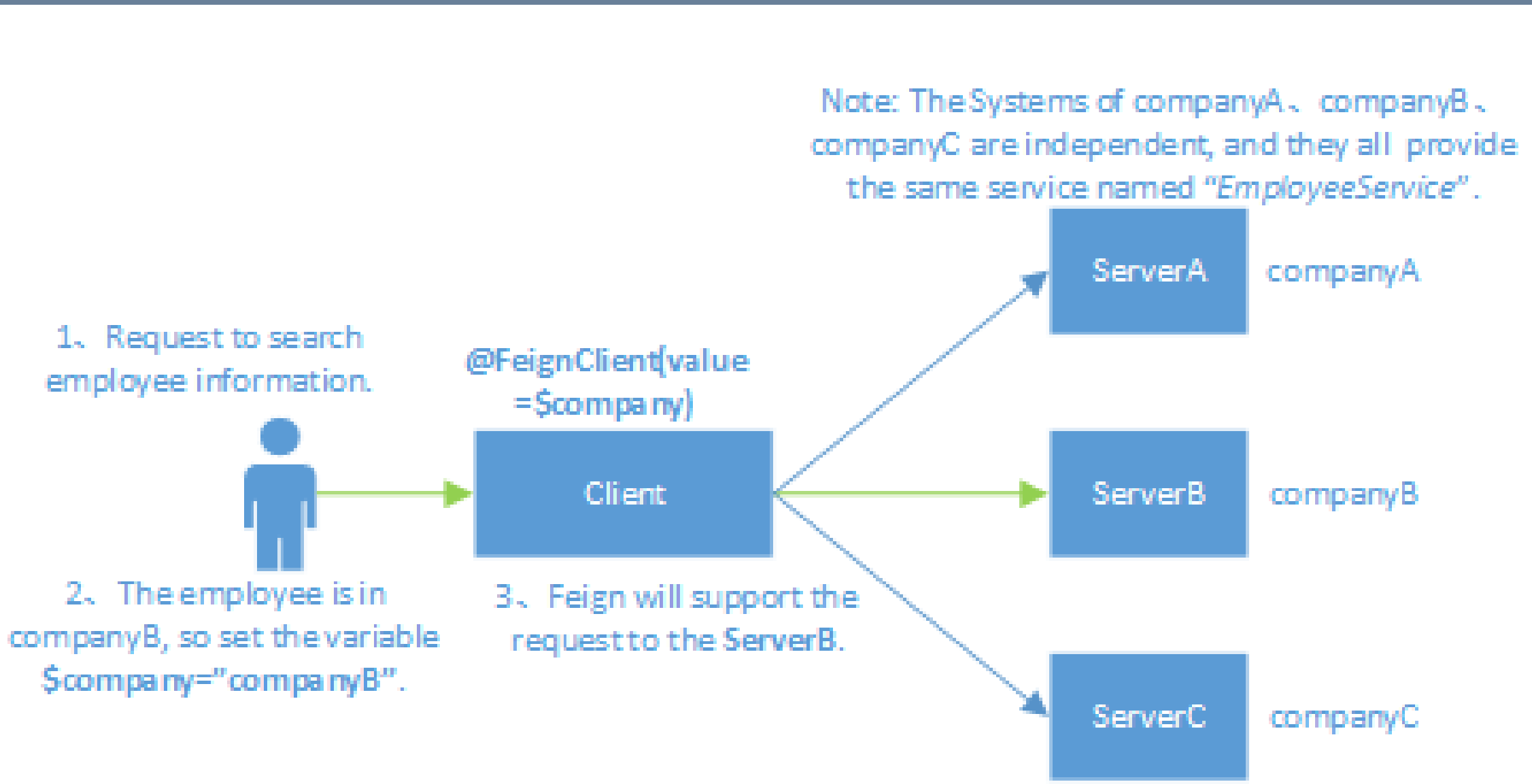
```
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

Exemplo de código

```java
package br.com.fujideia.iesp.tecback.clients;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;

@FeignClient(name = "jsonPlaceholder", url = "https://jsonplaceholder.typicode.com")
public interface JsonPlaceholderClient {
    @GetMapping("/posts")
    List<Post> getPosts();
}
```


MicroService1
MicroService2
Request data
Response data
Acknowledgement
Final confirmation
MicroService1
MicroService2


Microservices Communication using Spring Cloud OpenFeign
Microservice 1
Microservice 2
Java Guides
Http Request
Java Guides
User Service
Department Service
Http Response


Note: TheSystems of companyA、companyB、companyC are independent, and they all provide the same service named "EmployeeService".
1、Request to search employee information.
@FeignClient(value =$company)
Client
ServerA
companyA
ServerB
companyB
ServerC
companyC
2、The employee is in companyB, so set the variable $company="companyB".
3、Feign will support the request to the ServerB.

Personalização e Avançado

```java
package br.com.fujideia.iesp.tecback.model;

import feign.RequestInterceptor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class FeignClientConfig {
    @Bean
    public RequestInterceptor requestInterceptor() {
        return requestTemplate -> {
            requestTemplate.header( name: "Authorization",  ...values: "Bearer your-token");
        };
    }
}
```

Integração com Feign e ViaCEP

Cadastro automático de endereço com base no CEP

1. Dependência no pom.xml

```
<!--
https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-openfeign -->
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-openfeign</artifactId>
   <version>4.2.1</version>
</dependency>
```

2. Ativar Feign na aplicação

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

3. Interface ViaCepClient

```java
package br.com.alunoonline.api.client;

import br.com.alunoonline.api.dtos.EnderecoViaCepDTO;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@FeignClient(name = "viacep", url = "https://viacep.com.br/ws")
public interface ViaCepClient {

    @GetMapping("/{cep}/json/")
    EnderecoViaCepDTO buscarEnderecoPorCep(@PathVariable("cep") String cep);

}
```

4. DTO: EnderecoViaCepDTO

```java
package br.com.alunoonline.api.dtos;

import lombok.Data;

@Data
public class EnderecoViaCepDTO {
    private String cep;
    private String logradouro;
    private String complemento;
    private String bairro;
    private String localidade;
    private String uf;
    private String estado;
    private String regiao;
    private String ibge;
    private String gia;
    private String ddd;
    private String siafi;
}
```

5. DTO: ProfessorRequestDTO

```java
package br.com.alunoonline.api.dtos;

import lombok.Data;

@Data
public class ProfessorRequestDTO {
    private String nome;
    private String email;
    private String cpf;
    private String cep;
    private String numero;
}
```

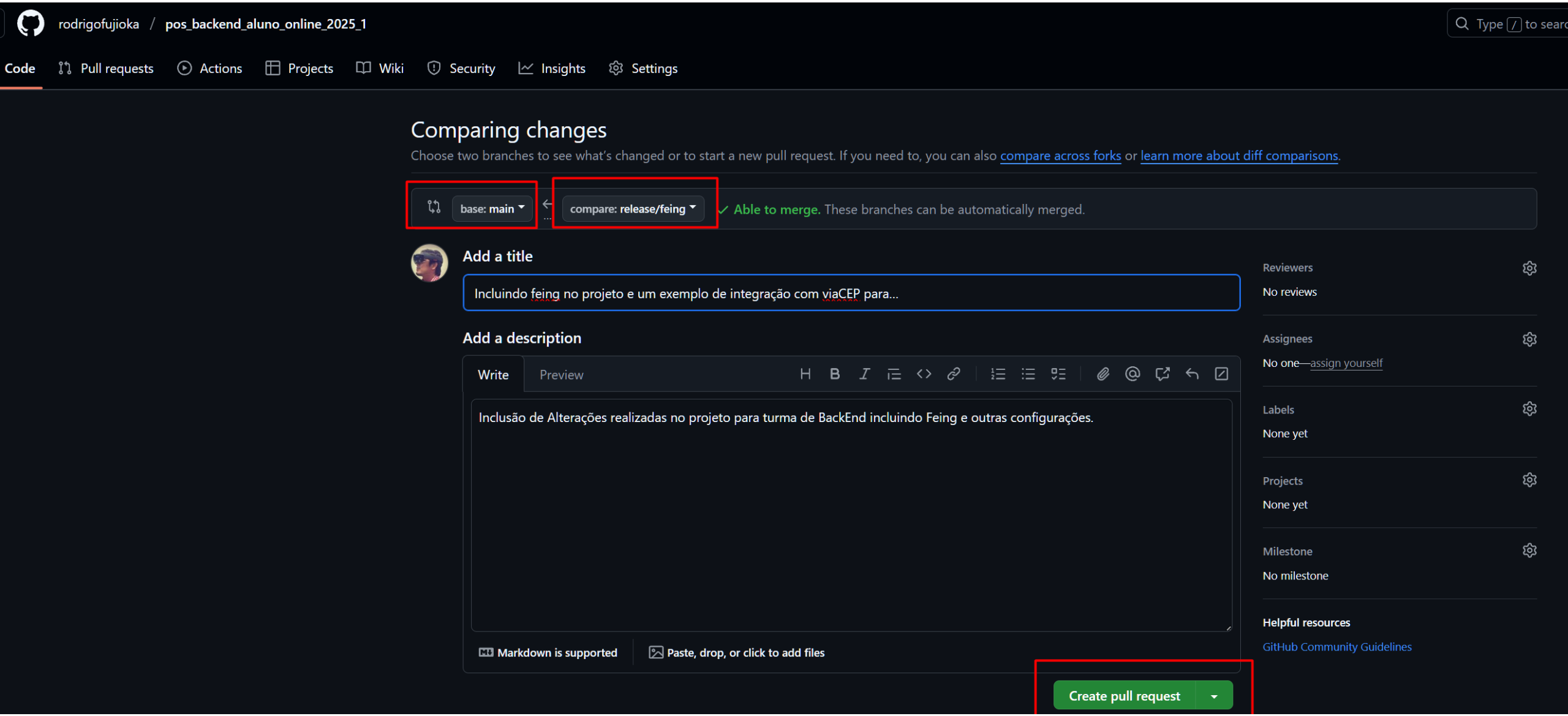
```java
@Slf4j
@RequiredArgsConstructor
@Service
public class ProfessorService {

    private final ProfessorRepository professorRepository;
    private final ViaCepClient viaCepClient;



    public void criarProfessor(ProfessorRequestDTO professorDTO) {
        EnderecoViaCepDTO enderecoDTO = viaCepClient.buscarEnderecoPorCep(professorDTO.getCep());
        log.info("Criando Professor: {}", professorDTO);
        Endereco endereco = Endereco.builder()
                .cep(enderecoDTO.getCep())
                .logradouro(enderecoDTO.getLogradouro())
                .complemento(enderecoDTO.getComplemento())
                .bairro(enderecoDTO.getBairro())
                .localidade(enderecoDTO.getLocalidade())
                .uf(enderecoDTO.getUf())
                .estado(enderecoDTO.getEstado())
                .regiao(enderecoDTO.getRegiao())
                .ibge(enderecoDTO.getIbge())
                .gia(enderecoDTO.getGia())
                .ddd(enderecoDTO.getDdd())
                .siafi(enderecoDTO.getSiafi())
                .numero(professorDTO.getNumero()) // dado manual
                .build();

        Professor professor = new Professor();
        professor.setNome(professorDTO.getNome());
        professor.setEmail(professorDTO.getEmail());
        professor.setCpf(professorDTO.getCpf());
        professor.setEndereco(endereco);

        professorRepository.save(professor);
    }

}
```

6. Service: montar e salvar

7. Controller: endpoint POST

```java
@RestController
@RequestMapping("/professores")
public class ProfessorController {

    @Autowired
    ProfessorService professorService;

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void criarProfessor(@RequestBody ProfessorRequestDTO professorDTO) {
        professorService.criarProfessor(professorDTO);
    }

}
```

- Com base no exemplo em sala de aula, utilize a consulta para o via CEP para consultar o cep do aluno que esta sendo cadastrado e atualize o endereço dele durante o cadastro. Tente fazer essa alteração utilizando uma Branch separada.


rodrigofujioka / pos_backend_aluno_online_2025_1
Type / to searc
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Comparing changes
Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.
base: main
compare: release/feing
Able to merge. These branches can be automatically merged.
Add a title
Incluindo feing no projeto e um exemplo de integração com viaCEP para...
Add a description
Write
Preview
Inclusão de Alterações realizadas no projeto para turma de BackEnd incluindo Feing e outras configurações.
Markdown is supported
Paste, drop, or click to add files
Create pull request
Reviewers
No reviews
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Helpful resources
GitHub Community Guidelines

Mão na massa! Prática!

```java
import jakarta.validation.Constraint;
import jakarta.validation.Payload;

import java.lang.annotation.*;


@Target( { ElementType.FIELD, ElementType.PARAMETER })
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Constraint(validatedBy = EmailValidator.class)
public @interface EmailValidation {
    String message() default "Email não é válido tente fuji.com ";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

```java
public class EmailValidator implements
        ConstraintValidator<EmailValidation, String> {

    private String message;

    @Override
    public void initialize(EmailValidation constraintAnnotation) { message = constraintAnnotation.message(); }


    @Override
    public boolean isValid(String nome, ConstraintValidatorContext constraintValidatorContext) {
        if(nome.contains("@fuji.com")){
            return true;
        }
        constraintValidatorContext.disableDefaultConstraintViolation();
        constraintValidatorContext.buildConstraintViolationWithTemplate( s: message + nome)//
                .addConstraintViolation();
        return false;
    }
}
```

Dinâmica da aula