

53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

RESIDÊNCIA DE SOFTWARE

**CAPACITAR
TREINAR
EMPREGAR**

TRANSFORMAR



Desenvolvimento de API Restful

@Alexandre Paixão | aopaixao@gmail | aopaixao@outlook.com | (24) 988497769

Graduado em Análise de Sistemas
MBA em Marketing Digital
Mestre em Informática

Atuação Profissional

Tech Lead / Desenvolvedor Full Stack / @Tic.Social
Professor / @ Residência Software
Consultor de TI / @

Áreas de Interesse

/Gestão de TI
/Projetos e Negócios
/BPMN

/Análise de Sistemas
/Desenvolvimento Full Stack
/Desenvolvimento Mobile

/Mineração de Dados
/Data Science
/BI

Conteúdos Formativos	
Capacidades Técnicas	Conhecimentos
<ul style="list-style-type: none"> Entender os protocolos HTTP e REST. Saber utilizar uma ferramenta de automação, compilação, gerenciamento de dependências e empacotamento de projeto. Criar um projeto utilizando framework backend. Criar uma api REST (controllers, mappings, rotas, etc). Entender e saber aplicar os conceitos de inversão de controle e injeção de dependências. Tratar erros do backend e códigos de status HTTP. Fazer o mapeamento objeto-relacional. Utilizar linguagem de consultas objeto relacional. Habilitar a documentação com Swagger. Saber acessar outras APIs REST a partir do backend. Entender o protocolo HTTPS. Gerenciar autenticações. Fazer upload de arquivos. Criar recursos para envio de e-mails. Aplicar ferramentas para gestão e controle de versão 	<ul style="list-style-type: none"> Métodos HTTP GET POST PUT PATCH DELETE HTTPS Ferramenta para automatizar tarefas Ferramenta para compilar Ferramenta para gerenciamento de dependências Ferramenta para empacotamento de projetos ORM Dependency Injection Frameworks Estrutura Padrões de projeto Persistência de dados Gerenciamento de dependências Tratamento de erros Requisições através do backend Autenticação e autorização Upload de arquivos Tratamento de arquivos Armazenamento Bibliotecas Api's Upload de múltiplos arquivos Envio de e-mails Criação de serviços para envio de e-mails Anexos Bibliotecas Api's Controle de versão Definição, criação e documentação de projetos

Aula	Data	Conteúdo
1	16/05	Conceitos de API e RESTFUL; Framework Spring
2	17/05	Spring: Criando a Entidade (Prática)
3	18/05	Spring: Criando o Repositório (Prática)
4	19/05	Spring: Criando o Serviço (Prática)
5	20/05	Spring: Criando o Controller (Prática)
6	23/05	Spring: Tratamento de Erros / Código de Status HTTP
7	24/05	Spring: Validação de Dados
8	25/05	Mapeamento Entidade x VO/DTO

Aula	Data	Conteúdo
9	26/05	Documentação com Swagger
10	27/05	Consumo de APIs REST
11	30/05	Envio de E-mails
12	31/05	Prática: Trabalho final da disciplina
13	01/06	Prática: Trabalho final da disciplina
14	02/06	Prática: Trabalho final da disciplina
15	03/06	Prática: Trabalho final da disciplina
16	06/06	Prática: Trabalho final da disciplina

Aula	Data	Conteúdo
17	07/06	Apresentação do trabalho final da disciplina

AVALIAÇÕES

Data		Tipo	Formato	Pontuação
1	20/05/2021	Técnica	Projeto API Editora	10 pontos
2	27/05/2021	Técnica	Projeto API Editor + Erros / Validação / VO	10 pontos
3	03/06/2021	Técnica	Projeto API DVD Rental Consumo API Externa / Swagger	10 pontos
3	16/06/2021	Técnica	Projeto API E-commerce	60 pontos
4	--	Social	Aspectos Gerais	10 pontos

Material de Apoio:

- Documentação do Spring Boot

<https://spring.io/quickstart>

- Baeldung

<https://www.baeldung.com/spring-tutorial>

Ferramentas:

- Spring Tool Suite 4
<https://spring.io/tools>
- Postman / Insomnia
- Postgresql
- DBEaver

Aula 1:

- Conceitos de API e Restful;
- Controle de Dependências com Maven
- Versionamento de Código – Git
- Framework Spring

API:

Application Programming Interface

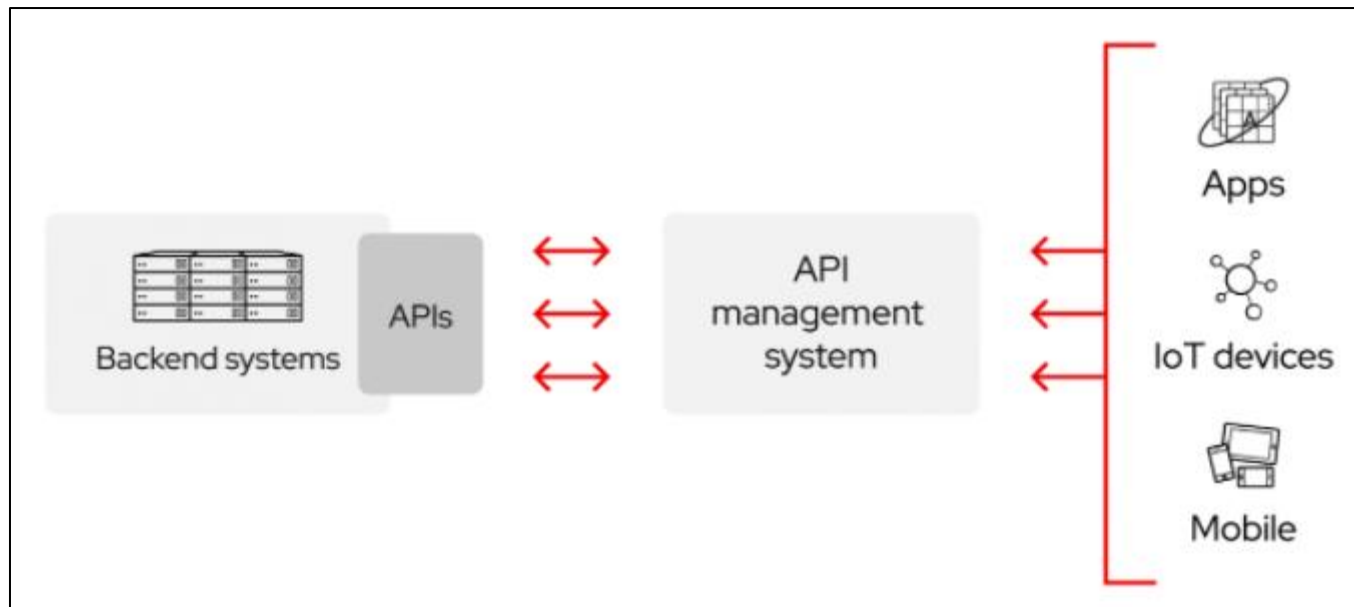
(Interface de Programação de Aplicações)

O que é API?

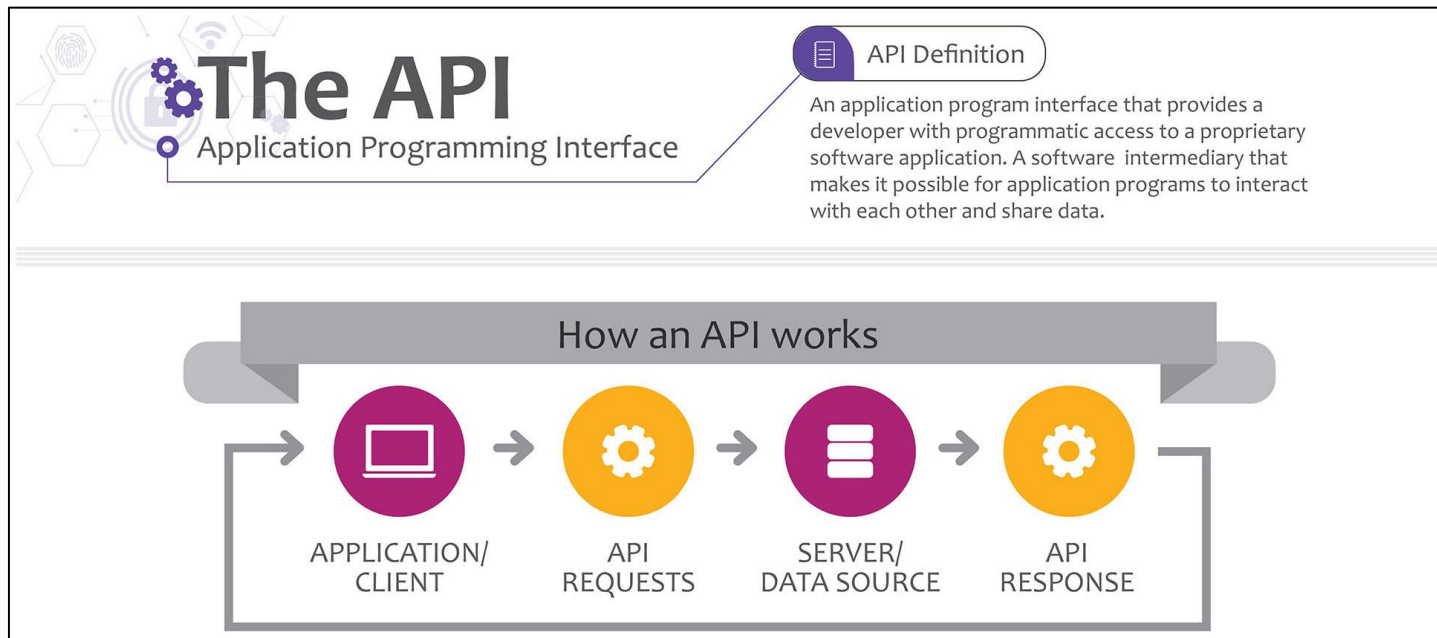
API é um conjunto de definições e protocolos usado no desenvolvimento e na [integração](#) de software de aplicações. API é um acrônimo em inglês que significa interface de programação de aplicações.

Uma API permite que sua solução ou serviço se comunique com outros produtos e serviços sem precisar saber como eles foram implementados. Isso simplifica o desenvolvimento de aplicações, gerando economia de tempo e dinheiro. Ao desenvolver novas ferramentas e soluções (ou ao gerenciar aquelas já existentes), as APIs oferecem a flexibilidade necessária para simplificar o design, a administração e o uso, além de fornecer oportunidades de inovação.

As APIs costumam ser vistas como contratos, com documentações que representam um acordo entre as partes interessadas. Se uma dessas partes enviar uma solicitação remota estruturada de uma forma específica, isso determinará como o software da outra parte responderá.



Fonte: redhat.com



Fonte: JM Baxi Group

REST e RESTful:

Representational State Transfer

(Transferência de Estado Representacional)

REST e RESTful:

Princípios, regras e restrições para a criação de projetos com interfaces bem definidas.

REST:

Princípios, regras e restrições de arquitetura.

RESTful:

Capacidade de aplicar, por um sistema, os princípios REST.

RESTful:

GET <http://domínio.com/alunos>

DELETE <http://domínio.com/alunos/fulano>

POST <http://domínio.com/alunos>
(data {nome: ciclano})

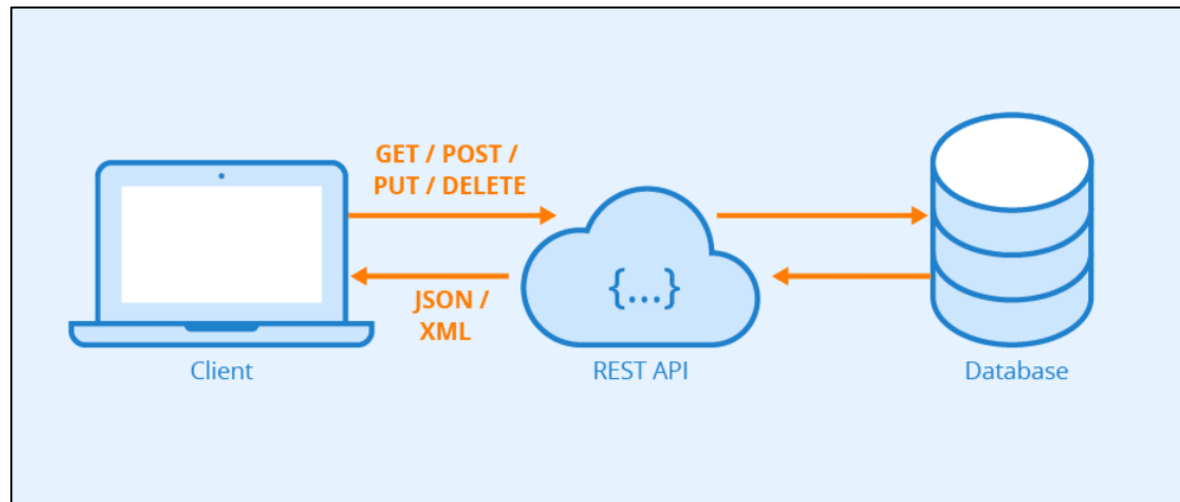
RESTful:

Métodos HTTP:

- GET
- POST
- PUT
- DELETE

Veja mais em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>

RESTFUL:



Fonte: Dev Community

Fundamentação teórica:

- https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- <https://www.baeldung.com/cs/rest-architecture>
- <https://arquiteturadesoftware.online/fundamentos-para-sistemas-com-arquiteturas-rest-capitulo-9-v-1-01/>

Aula 1 – Na prática:

- Acesso à API v1:

<https://579f-138-117-220-184.ngrok.io/livraria/editora>

- Clonar repositório:

```
git clone -b 1_api https://github.com/aopaixao/residencia-2022.1-api-restful.git 1_api
```

Aula 1 – Na prática:

- Criar o banco e as tabelas localmente:

https://github.com/aopaixao/residencia-2022.1-api-restful/blob/main/livraria_v1.sql

- Alterar as configurações do projeto:

`/resources/application.properties`

- Subir o projeto localmente:

`mvn spring-boot:run`

Spring Framework

Spring Framework:



Microservices

Quickly deliver production-grade features with independently evolvable microservices.



Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.



Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.



Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.



Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.



Event Driven

Integrate with your enterprise. React to business events. Act on your streaming data in realtime.



Batch

Automated tasks. Offline processing of data at a time to suit you.

(Maven : Controle de Dependências):

- Gerencia a utilização de bibliotecas (.jar) no projeto
- Automatiza a tarefa, substituindo o download e inserção manual de bibliotecas;
- Configuração de dependências a partir do pom.xml

(Maven : Controle de Dependências):

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  . . . .
```

Spring Boot :: Arquitetura do Projeto RESTful

Entities

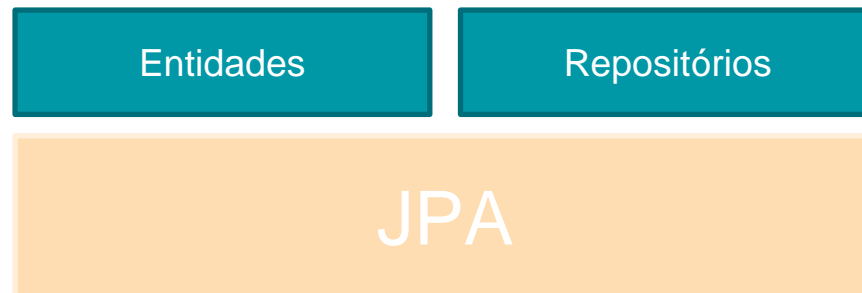
Repositories

Services

Controllers

Spring Boot

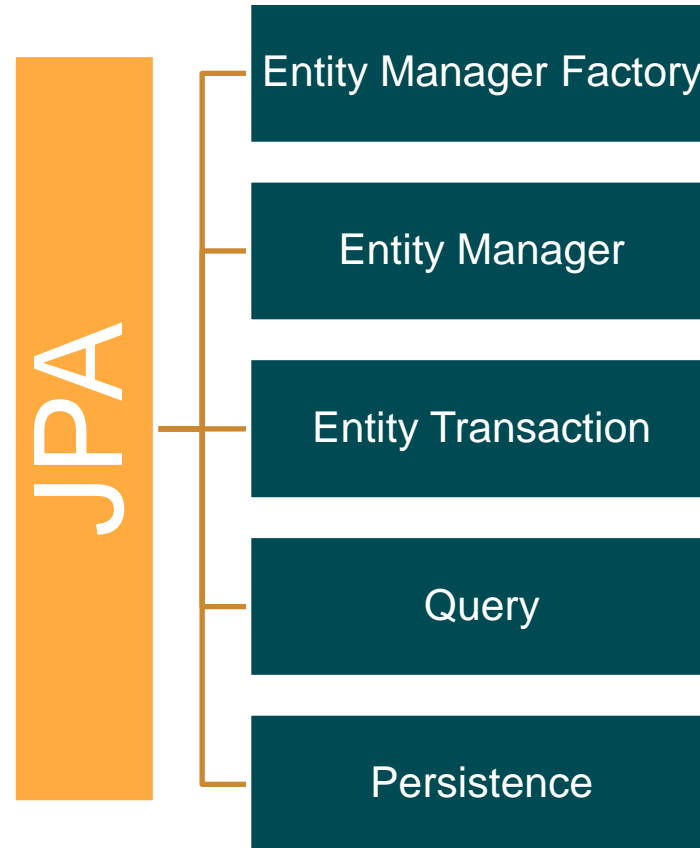
Spring Boot :: Acesso a Dados



Spring Boot :: Acesso a Dados



- Java Persistence API
- Framework baseado em POJOS (objetos Java)
- Mapeamento Objeto-Relacional (ORM)
- Outras funcionalidades



Spring Boot :: Acesso a Dados



- Entidades: @Entity
- Repositórios: Repository Interface -
JpaRepository | PagingAndSortingRepository | CrudRepository

Spring Boot :: Regras/Lógica de Negócio



- Serviços: @Service
- Fazem a interface entre o endpoint (controller) e o repositório (repository)

Spring Boot :: Controller/Endpoint



- Controller: `@RestController`
- Faz a interface com a aplicação externa
- Contem os endereços (URI) dos recursos disponíveis

Spring Framework: Propriedades do Projeto

/src/main/resources/

application.properties

Spring Framework: Propriedades do Projeto

```
#server.port = 8443
server.port = 8000
logging.level.root = INFO
server.servlet.context-path = /alunos
server.session.timeout = 5
spring.jackson.time-zone = America/Sao_Paulo

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto = none
spring.jpa.hibernate.show-sql = true
spring.datasource.url = jdbc:postgresql://localhost:5432/alunos
spring.datasource.username = root
spring.datasource.password = 123456

logging.level.org.hibernate.SQL = INFO
#logging.level.org.hibernate.SQL = DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder = INFO
logging.level.org.hibernate.type = INFO
```

Spring Framework :: Praticando

Iniciando o projeto “Hello World” RESTful com Spring:

<https://spring.io/quickstart>

Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 2.5.1 (SNAPSHOT) ☒ 2.5.0 ☐ 2.4.7 (SNAPSHOT) ☐ 2.4.6
☐ 2.3.12 (SNAPSHOT) ☐ 2.3.11

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 16 ☒ 11 ☐ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web ☒

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Testando:

Acesse o endpoint pelo navegador e pelo Postman

Aula 2:

Criando nossa primeira Entidade

- Criar o projeto “biblioteca” a partir do spring initializr
 - Group: com.residencia.biblioteca
 - Artifact: biblioteca
 - Package: com.residencia.biblioteca
- Incluir as dependências:
 - Spring Web
 - Spring Boot DevTools
 - Spring Security
 - Spring Data JPA
 - PostgreSQL Driver
 - Validation (hibernate validator)

Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 2.5.1 (SNAPSHOT) ☒ 2.5.0 ☐ 2.4.7 (SNAPSHOT) ☐ 2.4.6
☐ 2.3.12 (SNAPSHOT) ☐ 2.3.11

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 16 ☒ 11 ☐ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web **WEB**

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools **DEVELOPER TOOLS**

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Security **SECURITY**

Highly customizable authentication and access-control framework for Spring applications.

Spring Data JPA **SQL**

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

PostgreSQL Driver **SQL**

A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Validation **I/O**

Bean Validation with Hibernate validator.

- Incluir os dados do BD no application.properties

https://github.com/aopaixao/residencia_api_restful/blob/main/biblioteca/app.prop1

- Criar a estrutura do banco:

https://github.com/aopaixao/residencia_api_restful/tree/main/biblioteca

- Criar o package entities
- Criar uma classe para cada tabela
Aluno.java
Editora.java
Emprestimo.java
Livros.java
- Criar os atributos de cada coluna do bd e métodos get e set de cada atributo

- Ler sobre relacionamentos em:

<https://dev.to/jhonifaber/hibernate-onetoone-onetomany-manytoone-and-manytomany-8ba>

<https://ankitkamboj18.medium.com/a-guide-to-jpa-with-hibernate-relationship-mappings-onetoone-onetomany-manytoone-310ce31df3f6>

- Modificar as entidades e criar os respectivos relacionamentos

@ID:

- **Utilizado no Spring para identificar um registro que seja único;**
- **Utilizado nos métodos que retornam UMA instância da Entidade (findById, getOne, etc);**
- **Recuperamos uma instância da entidade em situações onde desejamos realizar operações de Select, Update ou Delete;**
- **Não ter na Entidade uma propriedade anotada com @Id que seja, de fato, única, faz com que seja necessário criar métodos adicionais no Repository para retornar uma instância única da Entidade.**

Aula 3:

Criando os repositórios

- Criar o package repositories
- Criar uma interface para cada Entidade
AlunoRepository.java
EditoraRepository.java
EmprestimoRepository.java
LivrosRepository.java

- Exemplo do AlunoRepository

```
package com.residencia.biblioteca.repositories;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.residencia.biblioteca.entities.Aluno;

@Repository
public interface AlunoRepository extends JpaRepository<Aluno, Integer>{
}
```

Algumas características da Interface Repository

- Possui métodos pré-definidos:
count, delete, deleteAll, existsById, findById, save, findAll, findOne ...
- Permite a paginação da consulta;
- Permite que novos métodos – e também métodos customizados – sejam implementados

Algumas características da Interface Repository

```
public Aluno findByName(String nome);
```

```
public List<Aluno> findAllByName(String nome);
```

```
@Query(value = "FROM Aluno a "  
        + "WHERE LOWER(nome) LIKE LOWER(CONCAT('%', ?1, '%'))")
```

```
public List<Aluno> listByName(String nome);
```

```
@Query(value = "FROM Aluno a "  
        + "WHERE numeroMatriculaAluno = :matricula")
```

```
public List<Aluno> listByMatricula(@Param("matricula") Integer matricula);
```

```
@Query(value = "FROM Aluno a "  
        + "LEFT JOIN a.listEmprestimo e ")
```

```
public List<Aluno> listarAlunosEmprestimos();
```

- Modifique os repositórios criados e adicione (pelo menos) um método personalizado em todos eles

Aula 4:

Criando os serviços

- Criar o package services
- Criar uma classe de serviço para cada Entidade
 - AlunoService.java
 - EditoraService.java
 - EmprestimoService.java
 - LivrosService.java

```
@Service
public class AlunoService {

    @Autowired
    public AlunoRepository alunoRepository;

    public Aluno findById(Integer id) {
        return alunoRepository.findById(id).get();
    }

    public List<Aluno> findAll(Integer pagina, Integer qtdRegistros){
        Pageable page = null;
        List<Aluno> listAluno = null;
        List<Aluno> listAlunoComPaginacao = null;

        if(null != pagina && null != qtdRegistros) {
            page = PageRequest.of(pagina, qtdRegistros);
            listAlunoComPaginacao = alunoRepository.findAll(page).getContent();

            return listAlunoComPaginacao;
        }else {
            listAluno = alunoRepository.findAll();

            return listAluno;
        }
    }

    public List<Aluno> listByMatricula(Integer matricula) {
        return alunoRepository.listByMatricula(matricula);
    }
}
```


- Crie todos os serviços correspondentes aos métodos criados nos Repositórios
- Adicione (pelo menos) um método que utilize paginação

Aula 5:

Criando os controllers

- Criar o package controllers
- Criar uma classe de controller para cada Entidade
AlunoController.java
EditoraController.java
EmprestimoController.java
LivrosController.java

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

    @Autowired
    private AlunoService alunoService;

    @GetMapping("/{id}")
    public ResponseEntity<Aluno> findById(@PathVariable Integer id) {
        HttpHeaders headers = new HttpHeaders();
        return new ResponseEntity<>(alunoService.findById(id), headers, HttpStatus.OK);
    }

    @GetMapping
    public ResponseEntity<List<Aluno>> findAll(@RequestParam(required = false) Integer pagina,
        @RequestParam(required = false) Integer qtdRegistros) throws Exception {

        HttpHeaders headers = new HttpHeaders();
        return new ResponseEntity<>(alunoService.findAll(pagina, qtdRegistros), headers, HttpStatus.OK);
    }

    @GetMapping("/matricula")
    public ResponseEntity<List<Aluno>> listByMatricula(@RequestParam(required = true) Integer matricula) {
        HttpHeaders headers = new HttpHeaders();
        return new ResponseEntity<>(alunoService.listByMatricula(matricula), headers, HttpStatus.OK);
    }

    @GetMapping("/count")
    public Long count() {
        return alunoService.count();
    }
}
```

- Implementar os métodos findById, findAll e Count (todos com o método HTTP GET) relacionados a todas as Entidades
- Testar o funcionamento dos métodos acima no Postman;
- Implementar os métodos save e update (consultar a documentação do Spring)

Aula 6:

Problemas encontrados

- Endpoints retornando entidades aninhadas;
- Ausência de tratamento de erros / exceções;
- Validação de dados;
- Outras?

Entidades aninhadas / Recursividade infinita:

Referências:

- <https://www.baeldung.com/jackson-bidirectional-relationships-and-infinite-recursion>
- <https://www.logicbig.com/tutorials/misc/jackson/json-managed-reference.html>
- <https://www.logicbig.com/tutorials/misc/jackson/json-identity-info-annotation.html>

- **@JsonIgnore**
- **@JsonBackReference / @JsonManagedReference**
- **@JsonIdentityInfo**

- #Sem Tratamento

Get .../turma

```
java.lang.StackOverflowError: null
    at java.base/sun.util.calendar.ZoneInfo.getOffsets(ZoneInfo.java:228) ~[na:na]
    at java.base/java.util.GregorianCalendar.computeFields(GregorianCalendar.java:2301) ~[na:na]
    at java.base/java.util.GregorianCalendar.computeFields(GregorianCalendar.java:2273) ~[na:na]
    at java.base/java.util.Calendar.setTimeInMillis(Calendar.java:1827) ~[na:na]
    at java.base/java.util.Calendar.setTime(Calendar.java:1793) ~[na:na]
    at com.fasterxml.jackson.databind.util.StdDateFormat._format(StdDateFormat.java:455) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.util.StdDateFormat.format(StdDateFormat.java:447) ~[jackson-databind-2.13.3.jar:2.13.3]
    at java.base/java.text.DateFormat.format(DateFormat.java:378) ~[na:na]
    at com.fasterxml.jackson.databind.SerializerProvider.defaultSerializeDateValue(SerializerProvider.java:1199) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.DateTimeSerializerBase._serializeAsString(DateTimeSerializerBase.java:211) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.DateSerializer.serialize(DateSerializer.java:51) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.DateSerializer.serialize(DateSerializer.java:15) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serializeContents(CollectionSerializer.java:145) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serialize(CollectionSerializer.java:107) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serialize(CollectionSerializer.java:25) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
```

- #Sem Tratamento

Get .../instrutor

```
java.lang.StackOverflowError: null
    at com.fasterxml.jackson.databind.util.StdDateFormat._getCalendar(StdDateFormat.java:810) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.util.StdDateFormat._format(StdDateFormat.java:454) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.util.StdDateFormat.format(StdDateFormat.java:447) ~[jackson-databind-2.13.3.jar:2.13.3]
    at java.base/java.text.DateFormat.format(DateFormat.java:378) ~[na:na]
    at com.fasterxml.jackson.databind.SerializerProvider.defaultSerializeDateValue(SerializerProvider.java:1199) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.DateTimeSerializerBase._serializeAsString(DateTimeSerializerBase.java:211) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.DateSerializer.serialize(DateSerializer.java:51) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.DateSerializer.serialize(DateSerializer.java:15) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serializeContents(CollectionSerializer.java:145) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serialize(CollectionSerializer.java:107) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serialize(CollectionSerializer.java:25) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serializeContents(CollectionSerializer.java:145) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serialize(CollectionSerializer.java:107) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.CollectionSerializer.serialize(CollectionSerializer.java:25) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanPropertyWriter.serializeAsField(BeanPropertyWriter.java:728) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.std.BeanSerializerBase.serializeFields(BeanSerializerBase.java:774) ~[jackson-databind-2.13.3.jar:2.13.3]
    at com.fasterxml.jackson.databind.ser.BeanSerializer.serialize(BeanSerializer.java:178) ~[jackson-databind-2.13.3.jar:2.13.3]
```

- @JsonIgnore

```
@Entity
@Table(name = "instrutor")
public class Instrutor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_instrutor")
    private Integer idInstrutor;

    @Column(name = "rg")
    private Integer rgInstrutor;

    @Column(name = "nome")
    private String nomeInstrutor;

    @Column(name = "nascimento")
    private Date dataNascimento;

    @Column(name = "titulacao")
    private Integer titulacaoInstrutor;

    @OneToMany(mappedBy = "instrutor")
    @JsonIgnore
    private List<Turma> turmaList;

    //Get's
    //Set's
}
```

```
@Entity
@Table(name = "turma")
public class Turma {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_turma")
    private Integer idTurma;

    @Column(name = "horario")
    private Date horarioTurma;

    @Column(name = "duracao")
    private Integer duracaoTurma;

    @Column(name = "data_inicio")
    private Date dataInicio;

    @Column(name = "data_fim")
    private Date dataFim;

    @ManyToOne
    @JoinColumn(name = "id_instrutor", referencedColumnName = "id_instrutor")
    private Instrutor instrutor;

    //Get's
    //Set's
}
```

Get .../turma

```
[
  {
    "idTurma": 2,
    "horarioTurma": "2022-05-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-05-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00",
    "instrutor": {
      "idInstrutor": 1,
      "rgInstrutor": 123456789,
      "nomeInstrutor": "Alexandre",
      "dataNascimento": "2000-05-20T21:00:00.000+00:00",
      "titulacaoInstrutor": 1
    }
  },
  {
    "idTurma": 3,
    "horarioTurma": "2022-06-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-06-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00",
    "instrutor": {
      "idInstrutor": 1,
      "rgInstrutor": 123456789,
      "nomeInstrutor": "Alexandre",
      "dataNascimento": "2000-05-20T21:00:00.000+00:00",
      "titulacaoInstrutor": 1
    }
  }
]
```

Get .../instrutor

```
[
  {
    "idInstrutor": 1,
    "rgInstrutor": 123456789,
    "nomeInstrutor": "Alexandre",
    "dataNascimento": "2000-05-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 1
  },
  {
    "idInstrutor": 3,
    "rgInstrutor": 456789,
    "nomeInstrutor": "João",
    "dataNascimento": "2020-06-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 3
  }
]
```

- @JsonIgnore

```
@Entity
@Table(name = "instrutor")
public class Instrutor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_instrutor")
    private Integer idInstrutor;

    @Column(name = "rg")
    private Integer rgInstrutor;

    @Column(name = "nome")
    private String nomeInstrutor;

    @Column(name = "nascimento")
    private Date dataNascimento;

    @Column(name = "titulacao")
    private Integer titulacaoInstrutor;

    @OneToMany(mappedBy = "instrutor")
    private List<Turma> turmaList;

    //Get's
    //Set's
}
```

```
@Entity
@Table(name = "turma")
public class Turma {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_turma")
    private Integer idTurma;

    @Column(name = "horario")
    private Date horarioTurma;

    @Column(name = "duracao")
    private Integer duracaoTurma;

    @Column(name = "data_inicio")
    private Date dataInicio;

    @Column(name = "data_fim")
    private Date dataFim;

    @ManyToOne
    @JsonIgnore
    @JoinColumn(name = "id_instrutor", referencedColumnName = "id_instrutor")
    private Instrutor instrutor;

    //Get's
    //Set's
}
```

Get .../turma

```
[
  {
    "idTurma": 2,
    "horarioTurma": "2022-05-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-05-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00"
  },
  {
    "idTurma": 3,
    "horarioTurma": "2022-06-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-06-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00"
  }
]
```

Get .../instrutor

```
[
  {
    "idInstrutor": 1,
    "rgInstrutor": 123456789,
    "nomeInstrutor": "Alexandre",
    "dataNascimento": "2000-05-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 1,
    "turmaList": [
      {
        "idTurma": 2,
        "horarioTurma": "2022-05-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-05-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00"
      },
      {
        "idTurma": 3,
        "horarioTurma": "2022-06-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-06-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00"
      }
    ]
  },
  {
    "idInstrutor": 3,
    "rgInstrutor": 456789,
    "nomeInstrutor": "João",
    "dataNascimento": "2020-06-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 3,
    "turmaList": []
  }
]
```


• @JsonBackReference / @JsonManagedReference

```
@Entity
@Table(name = "instrutor")
public class Instrutor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_instrutor")
    private Integer idInstrutor;

    @Column(name = "rg")
    private Integer rgInstrutor;

    @Column(name = "nome")
    private String nomeInstrutor;

    @Column(name = "nascimento")
    private Date dataNascimento;

    @Column(name = "titulacao")
    private Integer titulacaoInstrutor;

    @OneToMany(mappedBy = "instrutor")
    @JsonManagedReference
    private List<Turma> turmaList;

    //Get's
    //Set's
}
```

```
@Entity
@Table(name = "turma")
public class Turma {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_turma")
    private Integer idTurma;

    @Column(name = "horario")
    private Date horarioTurma;

    @Column(name = "duracao")
    private Integer duracaoTurma;

    @Column(name = "data_inicio")
    private Date dataInicio;

    @Column(name = "data_fim")
    private Date dataFim;

    @ManyToOne
    @JsonBackReference
    @JoinColumn(name = "id_instrutor", referencedColumnName = "id_instrutor")
    private Instrutor instrutor;

    //Get's
    //Set's
}
```

Get .../turma

```
[
  {
    "idTurma": 2,
    "horarioTurma": "2022-05-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-05-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00"
  },
  {
    "idTurma": 3,
    "horarioTurma": "2022-06-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-06-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00"
  }
]
```

Get .../instrutor

```
[
  {
    "idInstrutor": 1,
    "rgInstrutor": 123456789,
    "nomeInstrutor": "Alexandre",
    "dataNascimento": "2000-05-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 1,
    "turmaList": [
      {
        "idTurma": 2,
        "horarioTurma": "2022-05-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-05-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00"
      },
      {
        "idTurma": 3,
        "horarioTurma": "2022-06-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-06-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00"
      }
    ]
  },
  {
    "idInstrutor": 3,
    "rgInstrutor": 456789,
    "nomeInstrutor": "João",
    "dataNascimento": "2020-06-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 3,
    "turmaList": []
  }
]
```

• @JsonBackReference / @JsonManagedReference

```
@Entity
@Table(name = "instrutor")
public class Instrutor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_instrutor")
    private Integer idInstrutor;

    @Column(name = "rg")
    private Integer rgInstrutor;

    @Column(name = "nome")
    private String nomeInstrutor;

    @Column(name = "nascimento")
    private Date dataNascimento;

    @Column(name = "titulacao")
    private Integer titulacaoInstrutor;

    @OneToMany(mappedBy = "instrutor")
    private List<Turma> turmaList;

    //Get's
    //Set's
}
```

```
@Entity
@Table(name = "turma")
public class Turma {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_turma")
    private Integer idTurma;

    @Column(name = "horario")
    private Date horarioTurma;

    @Column(name = "duracao")
    private Integer duracaoTurma;

    @Column(name = "data_inicio")
    private Date dataInicio;

    @Column(name = "data_fim")
    private Date dataFim;

    @ManyToOne
    @JsonBackReference
    @JoinColumn(name = "id_instrutor", referencedColumnName = "id_instrutor")
    private Instrutor instrutor;

    //Get's
    //Set's
}
```

Get .../turma

```
[
  {
    "idTurma": 2,
    "horarioTurma": "2022-05-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-05-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00"
  },
  {
    "idTurma": 3,
    "horarioTurma": "2022-06-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-06-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00"
  }
]
```

Get .../instrutor

```
[
  {
    "idInstrutor": 1,
    "rgInstrutor": 123456789,
    "nomeInstrutor": "Alexandre",
    "dataNascimento": "2000-05-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 1,
    "turmaList": [
      {
        "idTurma": 2,
        "horarioTurma": "2022-05-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-05-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00"
      },
      {
        "idTurma": 3,
        "horarioTurma": "2022-06-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-06-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00"
      }
    ]
  },
  {
    "idInstrutor": 3,
    "rgInstrutor": 456789,
    "nomeInstrutor": "João",
    "dataNascimento": "2020-06-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 3,
    "turmaList": []
  }
]
```

- @JsonIdentityInfo

```
@Entity
@Table(name = "instrutor")
@JsonIdentityInfo(
    generator = ObjectIdGenerators.PropertyGenerator.class,
    property = "idInstrutor")
public class Instrutor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_instrutor")
    private Integer idInstrutor;

    @Column(name = "rg")
    private Integer rgInstrutor;

    @Column(name = "nome")
    private String nomeInstrutor;

    @Column(name = "nascimento")
    private Date dataNascimento;

    @Column(name = "titulacao")
    private Integer titulacaoInstrutor;

    @OneToMany(mappedBy = "instrutor")
    private List<Turma> turmaList;

    //Get's
    //Set's
}
```

```
@Entity
@Table(name = "turma")
@JsonIdentityInfo(
    generator = ObjectIdGenerators.PropertyGenerator.class,
    property = "idTurma")
public class Turma {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_turma")
    private Integer idTurma;

    @Column(name = "horario")
    private Date horarioTurma;

    @Column(name = "duracao")
    private Integer duracaoTurma;

    @Column(name = "data_inicio")
    private Date dataInicio;

    @Column(name = "data_fim")
    private Date dataFim;

    @ManyToOne
    @JoinColumn(name = "id_instrutor", referencedColumnName = "id_instrutor")
    private Instrutor instrutor;

    //Get's
    //Set's
}
```

Get .../turma

```
[
  {
    "idTurma": 2,
    "horarioTurma": "2022-05-19T15:00:00.000+00:00",
    "duracaoTurma": 1,
    "dataInicio": "2022-05-01T15:00:00.000+00:00",
    "dataFim": "2023-01-01T02:59:59.000+00:00",
    "instrutor": {
      "idInstrutor": 1,
      "rgInstrutor": 123456789,
      "nomeInstrutor": "Alexandre",
      "dataNascimento": "2000-05-20T21:00:00.000+00:00",
      "titulacaoInstrutor": 1,
      "turmaList": [
        2,
        {
          "idTurma": 3,
          "horarioTurma": "2022-06-19T15:00:00.000+00:00",
          "duracaoTurma": 1,
          "dataInicio": "2022-06-01T15:00:00.000+00:00",
          "dataFim": "2023-01-01T02:59:59.000+00:00",
          "instrutor": 1
        }
      ]
    }
  },
  3
]
```

Get .../instrutor

```
[
  {
    "idInstrutor": 1,
    "rgInstrutor": 123456789,
    "nomeInstrutor": "Alexandre",
    "dataNascimento": "2000-05-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 1,
    "turmaList": [
      {
        "idTurma": 2,
        "horarioTurma": "2022-05-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-05-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00",
        "instrutor": 1
      },
      {
        "idTurma": 3,
        "horarioTurma": "2022-06-19T15:00:00.000+00:00",
        "duracaoTurma": 1,
        "dataInicio": "2022-06-01T15:00:00.000+00:00",
        "dataFim": "2023-01-01T02:59:59.000+00:00",
        "instrutor": 1
      }
    ]
  },
  {
    "idInstrutor": 3,
    "rgInstrutor": 456789,
    "nomeInstrutor": "João",
    "dataNascimento": "2020-06-20T21:00:00.000+00:00",
    "titulacaoInstrutor": 3,
    "turmaList": []
  }
]
```

Tratamento de Exceções:

Referências:

- <https://reflectoring.io/spring-boot-exception-handling/>
- <https://www.baeldung.com/exception-handling-for-rest-with-spring>
- <https://www.bezkoder.com/spring-boot-restcontrolleradvice/>

- **@RestControllerAdvice / CustomExceptionHandler**

Os erros são tratados de forma Global através da anotação
@RestControllerAdvice.

Permite o tratamento de erros específicos, como “Not Found”, como de erros não conhecidos ou não tratados, definindo um erro “default”.

- **.../exception**
- **.../exception/CustomExceptionHandler.java**
- **.../exception/NoSuchElementException.java**
- **.../exception/ErrorResponse.java**

- **.../exception/CustomExceptionHandler.java**

```
@RestControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler {

    @ExceptionHandler(Exception.class)
    public final ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest request) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        HttpStatus httpStatus = HttpStatus.INTERNAL_SERVER_ERROR;
        ErrorResponse error = new ErrorResponse(httpStatus.value(), "Erro Interno no Servidor", details);

        return new ResponseEntity<>(error, HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler(NoSuchElementException.class)
    public final ResponseEntity<Object> handleUserNotFoundException(NoSuchElementException ex, WebRequest request) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        HttpStatus httpStatus = HttpStatus.NOT_FOUND;
        ErrorResponse error = new ErrorResponse(httpStatus.value(), "Registro Não Encontrado", details);
        return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
    }

    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
                                                                    HttpHeaders headers, HttpStatus status, WebRequest request) {
        List<String> details = new ArrayList<>();
        for (ObjectError error : ex.getBindingResult().getAllErrors()) {
            details.add(error.getDefaultMessage());
        }
        HttpStatus httpStatus = HttpStatus.BAD_REQUEST;
        ErrorResponse error = new ErrorResponse(httpStatus.value(), "Falha na Validação dos Dados da Requisição", details);
        return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST);
    }
}
```

- **.../exception/NoSuchElementException.java**

```
public class NoSuchElementException extends RuntimeException {  
    public NoSuchElementException(String message) {  
        super(message);  
    }  
}
```

- **.../exception/ErrorResponse.java**

```
@JsonInclude(JsonInclude.Include.NON_NULL)
public class ErrorResponse {
    private final int status;
    private final String message;
    private List<String> details;

    public ErrorResponse(int status, String message) {
        super();
        this.status = status;
        this.message = message;
    }

    public ErrorResponse(int status, String message, List<String> details) {
        super();
        this.status = status;
        this.message = message;
        this.details = details;
    }

    public int getStatus() {
        return status;
    }

    public String getMessage() {
        return message;
    }

    public List<String> getDetails() {
        return details;
    }

    public void setDetails(List<String> details) {
        this.details = details;
    }
}
```

- **@ExceptionHandler**

Os erros são tratados de forma local (no Controller) através da anotação @ExceptionHandler.

Permite o tratamento de erros específicos, sendo necessário mapear cada um deles.

- **.../controller/InstrutorController.java**

```
@DeleteMapping("/{id}")
public ResponseEntity<String> deleteInstrutor(@PathVariable Integer id) {
    if(null == instrutorService.findInstrutorById(id))
        throw new InstrutorNotFoundException("Não foi encontrado Instrutor com o id " + id);

    instrutorService.deleteInstrutor(id);
    return new ResponseEntity<>("", HttpStatus.OK);
}

@ExceptionHandler(InstrutorNotFoundException.class)
public ResponseEntity<String> handleNoInstrutorFoundException(InstrutorNotFoundException exception) {
    return ResponseEntity
        .status(HttpStatus.BAD_REQUEST)
        .body(exception.getErrMsg());
}
```

- **.../exception/InstrutorNotFoundException.java**

```
public class InstrutorNotFoundException extends RuntimeException {  
  
    private static final long serialVersionUID = 1L;  
  
    private String errCode;  
    private String errMsg;  
  
    public String getErrCode() {  
        return errCode;  
    }  
  
    public void setErrCode(String errCode) {  
        this.errCode = errCode;  
    }  
  
    public String getErrMsg() {  
        return errMsg;  
    }  
  
    public void setErrMsg(String errMsg) {  
        this.errMsg = errMsg;  
    }  
  
    public InstrutorNotFoundException(String errCode, String errMsg) {  
        this.errCode = errCode;  
        this.errMsg = errMsg;  
    }  
  
    public InstrutorNotFoundException(String errMsg) {  
        this.errMsg = errMsg;  
    }  
}
```


Exercícios / Atividade I:

Tipo: individual

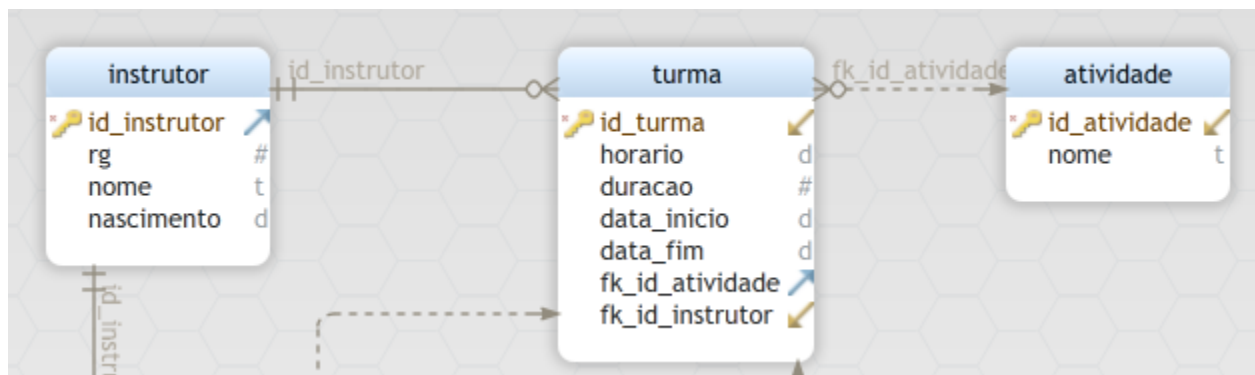
O que será verificado:

- a) Entendimento geral dos conceitos necessários para criação de uma API Rest;**
- b) Mapeamento Objeto Relacional;**
- c) Tratamento de Exceções;**
- d) Utilização de VO/DTO;**

Datas de entrega:

- 24/05/2022 - itens a), b) , c)
- 25/05/2022 – a), b), c) e d)

1. Crie no banco de dados a tabela atividade – incluindo seu relacionamento com a tabela turma;



- 2. Mapeie, na API, a tabela atividade e seu relacionamento com a tabela turma;**
- 3. Criar a entidade, repositório, serviço e controller de Atividade;**
- 4. Crie o tratamento de exceção para Turma (analise os recursos/endpoints disponibilizados no controller para identificar as exceções existentes), Instrutor e Atividade;**

***API de referência :**

<https://github.com/aopaixao/residencia-20221-api-academia>

Próximos conteúdos:

- **VO / DTO**
- **Tratamento de Exceções - Validação**
- **Swagger**
- **Consumo de API's externas;**
- **Upload de Arquivos**
- **Envio de e-mails**

- **VO / DTO**
- VO – Value Object
- DTO – Data Transfer Object

- **DTO**

É um padrão de projetos que serve para o transporte de dados entre diferentes componentes de um sistema, diferentes instâncias ou processos de um sistema distribuído ou diferentes sistemas via serialização. A ideia consiste basicamente em agrupar um conjunto de atributos numa classe simples de forma a otimizar a comunicação.

- **Motivação**

Normalmente, numa API REST escrita em Java/Spring, os dados são transferidos através de instâncias de uma entidade (e de suas entidades relacionadas, quando for o caso).

Utilizar, no lugar das Entidades, VO's/DTO's garante otimização de desempenho e também maior flexibilidade e controle sobre os dados transitados.

- **Como utilizar:**

1. Crie um novo package no projeto chamado dto;
2. Crie, dentro desse novo package, uma classe Java com o sufixo “DTO” para cada entidade existente no projeto;
3. Nessas classes, defina os mesmos atributos e relacionamentos existentes nas Entidades;
4. Nos DTOs não use nenhuma anotação, como @id, @column, @OneToMany, entre outras;
5. Nos DTOs os relacionamentos devem ser estabelecidos entre DTO's, sem misturá-los com as Entidades;
6. Nos controllers e services, troque as Entidades pelos DTO's

- **Exemplo:**

```
import java.util.Date;
import java.util.List;

public class InstrutorDTO {
    private Integer idInstrutor;
    private Integer rgInstrutor;
    private String nomeInstrutor;
    private Date dataNascimento;
    private Integer titulacaoInstrutor;
    private List<TurmaDTO> turmaDTOList;

    //Get's
    //Set's
}
```

```
@RestController
@RequestMapping("/instrutor")
public class InstrutorController {
    @Autowired
    InstrutorService instrutorService;

    @GetMapping
    public ResponseEntity<List<InstrutorDTO>> findAllInstrutor() {
        List<InstrutorDTO> instrutorDTOList = instrutorService.findAllInstrutor();
        return new ResponseEntity<>(instrutorDTOList, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<InstrutorDTO> findInstrutorById(@PathVariable Integer id) {
        InstrutorDTO instrutorDTO = instrutorService.findInstrutorById(id);
        if(null == instrutorDTO)
            return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
        else
            return new ResponseEntity<>(instrutorDTO, HttpStatus.OK);
    }

    @PostMapping
    public ResponseEntity<InstrutorDTO> saveInstrutor(@RequestBody InstrutorDTO instrutorDTO) {
        InstrutorDTO novoInstrutorDTO = instrutorService.saveInstrutor(instrutorDTO);
        return new ResponseEntity<>(novoInstrutorDTO, HttpStatus.CREATED);
    }
}
```

```
@Service
public class InstrutorService {
    @Autowired
    InstrutorRepository instrutorRepository;

    public List<InstrutorDTO> findAllInstrutor(){
        List<Instrutor> instrutorList = instrutorRepository.findAll();
        List<InstrutorDTO> instrutorDTOList = new ArrayList<>();
        for(Instrutor instrutor : instrutorList){
            InstrutorDTO instrutorDTO = new InstrutorDTO();
            instrutorDTO = convertEntidadeToDTO(instrutor);
            instrutorDTOList.add(instrutorDTO);
        }
        return instrutorDTOList;
    }

    public InstrutorDTO findInstrutorById(Integer id) {
        Instrutor instrutor = instrutorRepository.findById(id).isPresent() ?
            instrutorRepository.findById(id).get() : null;
        InstrutorDTO instrutorDTO = null;
        if(null != instrutor){
            instrutorDTO = convertEntidadeToDTO(instrutor);
        }
        return instrutorDTO;
    }

    public InstrutorDTO saveInstrutor(InstrutorDTO instrutorDTO) {
        Instrutor instrutor = convertDTOToEntidade(instrutorDTO);
        Instrutor novoInstrutor = instrutorRepository.save(instrutor);
        InstrutorDTO instrutorDTO = convertEntidadeToDTO(novoInstrutor);
        return instrutorDTO;
    }

    private Instrutor convertDTOToEntidade(InstrutorDTO instrutorDTO) {
        Instrutor instrutor = new Instrutor();
        instrutor.setIdInstrutor(instrutorDTO.getIdInstrutor());
        instrutor.setRgInstrutor(instrutorDTO.getRgInstrutor());
        instrutor.setNomeInstrutor(instrutorDTO.getNomeInstrutor());
        instrutor.setDataNascimento(instrutorDTO.getDataNascimento());
        instrutor.setTitulacaoInstrutor(instrutorDTO.getTitulacaoInstrutor());

        return instrutor;
    }

    private InstrutorDTO convertEntidadeToDTO(Instrutor instrutor) {
        InstrutorDTO instrutorDTO = new InstrutorDTO();
        instrutorDTO.setIdInstrutor(instrutor.getIdInstrutor());
        instrutorDTO.setRgInstrutor(instrutor.getRgInstrutor());
        instrutorDTO.setNomeInstrutor(instrutor.getNomeInstrutor());
        instrutorDTO.setDataNascimento(instrutor.getDataNascimento());
        instrutorDTO.setTitulacaoInstrutor(instrutor.getTitulacaoInstrutor());

        return instrutor;
    }
}
```

Tratamento de Exceções

-> Validação:

Referências:

- <https://www.baeldung.com/spring-boot-bean-validation>
- <https://reflectoring.io/bean-validation-with-spring-boot/>
- <https://docs.jboss.org/hibernate/beanvalidation/spec/2.0/api/javax/validation/constraints/package-summary.html>
- <https://www.baeldung.com/spring-validate-requestparam-pathvariable>

- **Objetivo:**

Validar os dados enviados para a API.

- **Como usar:**

- 1. Incluir a dependência “spring-boot-starter-validation” no pom.xml;**
- 2. Incluir a anotação @Valid no controller, antes do @RequestBody;**
 - 2.1. Incluir as anotações de validação no(s) atributo(s) da Entidade.**
- 3. Incluir a anotação @Validated na classe Controller;**
 - 3.1. Incluir as anotações de validação nos parâmetros do método, no Controller**

- **Como usar:**

- 1. Incluir a dependência “spring-boot-starter-validation” no pom.xml;**
- 2. Incluir a anotação @Valid no controller, antes do @RequestBody;**
 - 2.1. Incluir as anotações de validação no(s) atributo(s) da Entidade.**
- 3. Incluir a anotação @Validated na classe Controller;**
 - 3.1. Incluir as anotações de validação nos parâmetros do método, no Controller**
- 4. [Opcional]Customizar as mensagens de erro na classe @RestControllerAdvice;**

- **Como usar:**

1. **Incluir a dependência “spring-boot-starter-validation” no pom.xml**

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

- **Como usar:**

- 2. Incluir a anotação @Valid no controller, antes do @RequestBody**

- 2.1. Incluir as anotações de validação no(s) atributo(s) da Entidade**

```
...  
@Max  
@Min  
@NotBlank  
@NotEmpty  
@NotNull  
@Null  
@Pattern  
@Size  
...
```

```
@Entity
@Table(name = "fornecedor")
public class Fornecedor {

    //...

    @Column(name = "cnpj")
    @NotEmpty(message = "O número do CNPJ não pode ficar em branco.")
    private String cnpj;

    //...

}
```

```
@RestController
@RequestMapping("/fornecedor")
@Validated
public class FornecedorController {

    //...

    @PostMapping
    public ResponseEntity<Fornecedor> saveFornecedorCompleto(
        @Valid @RequestBody Fornecedor fornecedor) {
        Fornecedor novoFornecedor = fornecedorService.saveFornecedor(fornecedor);
        return new ResponseEntity<>(novoFornecedor, HttpStatus.CREATED);
    }

    //...

}
```

3. Incluir a anotação @Validated na classe Controller;

3.1. Incluir as anotações de validação nos parâmetros do método, no Controller

```
@RestController
@RequestMapping("/fornecedor")
@Validated
public class FornecedorController {
    // ...
    @GetMapping("/cnpj/{cnpj}")
    public ResponseEntity<CadastroEmpresaReceitaDTO> consultarDadosPorCnpj(
        @PathVariable
        @Pattern(regexp="^[0-9]{14}", message="O CNPJ deve conter apenas números, com 14 dígitos.")
        String cnpj) {
        CadastroEmpresaReceitaDTO cadEmpresaDTO = fornecedorService.consultarDadosPorCnpj(cnpj);
        if (null == cadEmpresaDTO)
            throw new NoSuchElementException("Não foram encontrados dados para o CNPJ informado");
        else
            return new ResponseEntity<>(cadEmpresaDTO, HttpStatus.OK);
    }

    @GetMapping("/cnpj/query")
    public ResponseEntity<CadastroEmpresaReceitaDTO> queryConsultarDadosPorCnpj(
        @RequestParam
        @Pattern(regexp="^[0-9]{14}", message="O CNPJ deve conter apenas números, com 14 dígitos.")
        String cnpj) {
        CadastroEmpresaReceitaDTO cadEmpresaDTO = fornecedorService.consultarDadosPorCnpj(cnpj);
        if (null == cadEmpresaDTO)
            throw new NoSuchElementException("Não foram encontrados dados para o CNPJ informado");
        else
            return new ResponseEntity<>(cadEmpresaDTO, HttpStatus.OK);
    }
    //...
}
```

4. [Opcional] Customizar as mensagens de erro na classe @RestControllerAdvice;

```
@RestControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler {
    //...
    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
        HttpHeaders headers, HttpStatus status, WebRequest request) {
        List<String> details = new ArrayList<>();
        for (ObjectError error : ex.getBindingResult().getAllErrors()) {
            details.add(error.getDefaultMessage());
        }
        HttpStatus httpStatus = HttpStatus.BAD_REQUEST;
        ErrorResponse error = new ErrorResponse(httpStatus.value(),
            "Falha na Validação dos Dados da Requisição", details);
        return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler({ConstraintViolationException.class})
    protected ResponseEntity<Object> handleConstraintViolationException(ConstraintViolationException ex) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        ErrorResponse error = new ErrorResponse(HttpStatus.BAD_REQUEST.value(),
            "Falha na Validação dos Dados da Requisição", details);
        return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST);
    }
    //...
}
```

- **Algumas Particularidades:**

1. **Parâmetros do tipo Integer possuem um tratamento interno diferente, uma vez que sua ausência (quando o valor não é enviado na requisição e o parâmetro foi anotado para ser validado como `@NotBlank`) dispara uma exceção específica, a `"MissingServletRequestParameterException"`**
 - 1.1. **Nesse caso, deve-se acrescentar um método para tratamento de tal exceção na classe `@RestControllerAdvice`**

Algumas Particularidades

```
@RestControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler {
    //...
    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
        HttpHeaders headers, HttpStatus status, WebRequest request) {
        List<String> details = new ArrayList<>();
        for(ObjectError error : ex.getBindingResult().getAllErrors()) {
            details.add(error.getDefaultMessage());
        }
        HttpStatus httpStatus = HttpStatus.BAD_REQUEST;
        ErrorResponse error = new ErrorResponse(httpStatus.value(),
            "Falha na Validação dos Dados da Requisição", details);
        return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler({ConstraintViolationException.class})
    protected ResponseEntity<Object> handleConstraintViolationException(ConstraintViolationException ex) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        ErrorResponse error = new ErrorResponse(HttpStatus.BAD_REQUEST.value(),
            "Falha na Validação dos Dados da Requisição", details);
        return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST);
    }

    @Override
    protected ResponseEntity<Object> handleMissingServletRequestParameter(
        MissingServletRequestParameterException ex, HttpHeaders headers,
        HttpStatus status, WebRequest request) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        details.add("Parâmetro não processado: " + ex.getParameterName());
        ErrorResponse error = new ErrorResponse(HttpStatus.BAD_REQUEST.value(),
            "Falha na Validação dos Dados da Requisição", details);
        return new ResponseEntity<>(error, HttpStatus.BAD_REQUEST);
    }
    //...
}
```

- Testes – Parâmetros do tipo String:

```
@GetMapping("/query")
public ResponseEntity<Produto> findByIdQuery(
    @RequestParam
    @NotBlank(message = "O sku deve ser preenchido.")
    String sku) {
    return new ResponseEntity<>(null, HttpStatus.CONTINUE);
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** host:8080/comercio/produto/query?sku=
- Action:** Send
- Status:** 400 Bad Request
- Time:** 52.1 ms
- Timestamp:** 1 9 Minutes Ago
- Body:** Auth, Query, Header, Docs
- Preview:** Header (4), Cookie, Timeline
- Response Body (JSON):**

```
1 {
2   "status": 400,
3   "message": "Falha na Validação dos Dados da
Requisição",
4   "details": [
5     "findByIdQuery.sku: O sku deve ser preenchido."
6   ]
7 }
```


- Testes – Parâmetros do tipo Integer:

```
@GetMapping("/request")
public ResponseEntity<Produto> findByIdRequest(
    @RequestParam
    @NotBlank(message = "O id deve ser preenchido.")
    Integer id){

    return new ResponseEntity<>(null, HttpStatus.CONTINUE);
}
```

GET host:8080/comercio/produto/request?id= Send 400 Bad Request 12.1 ms 10 Minutes Ago

Body Auth Query Header Docs Preview Header 4 Cookie Timeline



```
1 {
2   "status": 400,
3   "message": "Falha na Validação dos Dados da
  Requisição",
4   "details": [
5     "Required request parameter 'id' for method
  parameter type Integer is present but converted to
  null",
6     "Parâmetro não processado: id"
7   ]
8 }
```

Exercício (Avaliação II):

1. Configure seu projeto para utilizar a validação;
2. Visite as páginas de referência para conhecer mais sobre as diferentes validações existentes;
3. Escolha, ao menos, três validações e as aplique nos controllers Turma, Instrutor e Atividade;
4. Teste as validações através do Insomnia.

Swagger

-> Documentação de APIs:

Atenção: problemas de compatibilidade com o spring >= 2.6.0

Swagger

-> SpringFox:

Atenção: problemas de compatibilidade com o spring >= 2.6.0

Referências:

<https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>

Alguns exemplos:

<https://reverb.com/swagger#/articles>

<https://vocadb.net/swagger/index.html>

<https://docs.internetofthings.ibmcloud.com/apis/swagger/v0002/org-admin.html>

<https://developers.themoviedb.org/3/account/get-account-details>

- **Swagger :: Documentação (e testes) da API**

1. Incluir dependências no pom.xml e anotação na classe Application
2. Incluir a classe de configuração
3. Acessar a documentação pela URL
<http://localhost:8080/CONTEXTTO/v2/api-docs>
4. Acessar a interface gráfica pelo endereço:
<http://localhost:8080/CONTEXTTO/swagger-ui/index.html>

1. Incluir dependências no pom.xml e anotação na classe Application

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>3.0.0</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

1.1. Incluir anotação @EnableWebMvc na classe Application

```
@EnableWebMvc
@SpringBootApplication
public class AcademiaApplication {

    public static void main(String[] args) {
        SpringApplication.run(AcademiaApplication.class, args);
    }

}
```

2. Incluir a classe de configuração (no package “config”)

```
@Configuration
public class SpringFoxConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.any())
            .paths(PathSelectors.any())
            .build();
    }
}
```

3. Acessar a documentação pela URL
<http://localhost:8080/CONTEXTO/v2/api-docs>
4. Acessar a interface gráfica pelo endereço:
<http://localhost:8080/CONTEXTO/swagger-ui/index.html>

Swagger

-> Springdoc:

Referências:

<https://springdoc.org/>

- **Swagger :: Documentação (e testes) da API**
 1. Incluir dependência no pom.xml
 2. Acessar a documentação pela URL
<http://localhost:8080/CONTEXTO/Swagger-ui.html>

1. Incluir dependência no pom.xml

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-ui</artifactId>
  <version>1.6.8</version>
</dependency>
```

Exercício (Avaliação II):

1. Configure as informações do cabeçalho da documentação da API;
2. Inclua informações adicionais em pelo menos dois métodos nos endpoints dos controllers Instrutor, Turma e Atividade. Exemplos: @ApiResponse, @Parameter, entre outras;

Consumo de APIs externas:

- **Consumir API's externas**
 1. Utilizar RestTemplate
 2. Mapear o retorno da API para um VO/DTO

- Consumir API's externas

```
public ReceitaWsDadosCnpjDTO consultarDadosPorCnpj(String cnpj) {  
    RestTemplate restTemplate = new RestTemplate();  
    String uri = "https://www.receitaws.com.br/v1/cnpj/{cnnpj}";  
    Map<String, String> params = new HashMap<String, String>();  
    params.put("cnnpj", cnnpj);  
  
    ReceitaWsDadosCnpjDTO receitaWsDadosCnpjVO =  
        restTemplate.getForObject(uri, ReceitaWsDadosCnpjDTO.class, params);  
  
    return receitaWsDadosCnpjVO;  
}
```

```
public class ReceitaWsDadosCnpjDTO {  
    private String tipo;  
    private String nome;  
    private String uf;  
    private String telefone;  
    private String email;  
    private String fantasia;  
    private String situacao;  
    private String bairro;  
    private String logradouro;  
    private String numero;  
    private String complemento;  
    private String cep;  
    private String municipio;  
    private String abertura;  
    //Gets e Sets  
}
```

Upload de Arquivos:

1. Incluir no `application.properties` o endereço do diretório onde os arquivos serão armazenados;
2. Inserir no `application.properties` os limites de tamanho dos arquivos a serem carregados;
3. Criar um Controller com os recursos de manipulação de arquivos que serão disponibilizados (upload, delete, etc);
4. Criar um Service específico para manipular os arquivos;

5. Criar, no serviço, um método para converter a string recebida, contendo os dados da entidade, numa instância da entidade;
6. Definir a regra de negócio para armazenamento do nome da imagem no banco de dados via Entidade(ou DTO);

1. Incluir no application.properties o endereço do diretório onde os arquivos serão armazenados;
2. Inserir no application.properties os limites de tamanho dos arquivos a serem carregados;

```
pasta.arquivos.imagem = D:\\_workspace\\residencia_software\\arquivos_imagem  
spring.servlet.multipart.max-file-size = 10MB  
spring.servlet.multipart.max-request-size = 10MB
```

3. Criar um Controller com os recursos de manipulação de arquivos que serão disponibilizados (upload, delete, etc);

```
@PostMapping(value = "/produto-com-foto", consumes = { MediaType.APPLICATION_JSON_VALUE,  
    MediaType.MULTIPART_FORM_DATA_VALUE })  
public ResponseEntity<Produto> saveProdutoComFoto(@RequestPart("produto") String produto,  
    @RequestPart("file") MultipartFile file) {  
  
    Produto novoProduto = produtoService.saveProdutoComFoto(produto, file);  
    if(null == novoProduto)  
        return new ResponseEntity<>(null, HttpStatus.BAD_REQUEST);  
    else  
        return new ResponseEntity<>(novoProduto, HttpStatus.CREATED);  
}
```

4. Criar um Service específico para manipular os arquivos;

```
@Service
public class ArquivoService {

    private Path fileStorageLocation;

    @Value("${pasta.arquivos.imagem}")
    private String dirArquivosImagem;

    private void createDirectory() {
        this.fileStorageLocation = Paths.get(dirArquivosImagem)
            .toAbsolutePath().normalize();
        try {
            Files.createDirectories(this.fileStorageLocation);
        } catch (Exception ex) {
            throw new
                ArquivosException("Não foi possível criar o diretório para armazenar o arquivo.", ex);
        }
    }

    public String storeFile(MultipartFile file, String fileName) {
        createDirectory();

        try {
            if (fileName.contains("..")) {
                throw new
                    ArquivosException("Nome de arquivo inválido! " + fileName);
            }
            // Copia/Sobrescrita do arquivo na pasta de destino
            Path targetLocation = this.fileStorageLocation.resolve(fileName);
            Files.copy(file.getInputStream(), targetLocation, StandardCopyOption.REPLACE_EXISTING);

            return fileName;
        } catch (IOException ex) {
            throw new
                ArquivosException("Ocorreu um erro e não foi possível armazenar o arquivo " + fileName, ex);
        }
    }
}
```

5. Criar, no serviço, um método para converter a string recebida, contendo os dados da entidade, numa instância da entidade;

```
private Produto convertProdutoFromStringJson(String produtoJson) {  
    Produto produto = new Produto();  
  
    try {  
        ObjectMapper objectMapper = new ObjectMapper();  
        produto = objectMapper.readValue(produtoJson, Produto.class);  
    } catch (IOException err) {  
        System.out.printf("Ocorreu um erro ao tentar converter a string" +  
            "json para um instância da entidade Produto", err.toString());  
    }  
  
    return produto;  
}
```

6. Definir a regra de negócio para armazenamento do nome da imagem no banco de dados via Entidade(ou DTO);

```
public Produto saveProdutoComFoto(@RequestPart("produto") String produto,
    @RequestPart("file") MultipartFile file) {
    Produto produtoFromJson = convertProdutoFromStringJson(produto);
    Produto novoProduto = produtoRepository.save(produtoFromJson);

    // Limpeza no nome do arquivo
    String fileName = StringUtils.cleanPath(file.getOriginalFilename());

    //Concatena o id do produto ao nome do arquivo
    fileName = novoProduto.getIdProduto().toString() + "_" + fileName;

    //Seta no produto recém criado o nome da imagem
    novoProduto.setImagemProduto(fileName);



    //Armazena a foto no diretorio
    arquivoService.storeFile(file, fileName);

    //Atualiza o produto recém criado, agora com o nome da imagem
    return produtoRepository.save(novoProduto);
}
```

Testando o Upload de Arquivos:

1. No Insomnia, selecionar o método POST. Na aba “Body” selecionar Multipart;
2. Incluir nos campos de chave-valor:
 - a) O nome do parâmetro (anotado com `@RequestPart`) definido no Controller que receberá a String contendo os dados da Entidade e a string Json com os dados da Entidade;
 - b) O nome do parâmetro, definido no Controller, que receberá o MultipartFile contendo a imagem e em “File” alterar o tipo para “File”.
 - c) Por fim, selecionar o arquivo a ser enviado.

Multipart ² ▾ Auth ▾ Query Header ¹ Docs

≡	produto	 159 bytes	▾	✓	🗑
≡	file	 bermuda.webp	▾	✓	🗑
⚙	New name	New value			

Edit produto

```
{
  "sku": "BM1",
  "nomeProduto": "Bermuda Azul",
  "imagemProduto": "",
  "fornecedor": {
    "idFornecedor": 1
  },
  "categoria": {
    "idCategoria": 1
  }
}
```


Envio de E-mails:

Referências:

<https://www.baeldung.com/spring-email>

<https://mailtrap.io/blog/spring-send-email/>

<https://zetcode.com/springboot/email/>

1. Incluir a dependência starter-mail no pom.xml;
2. Incluir no application.properties as propriedades do spring para configuração do envio de e-mails e de seus respectivos valores;
3. Criar um serviço específico com a função de enviar e-mails;

1. Incluir a dependência starter-mail no pom.xml;

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

2. Incluir no application.properties as propriedades do spring para configuração do envio de e-mails e de seus respectivos valores;

```
#Propriedades de e-mail
spring.mail.host=XXXXXXXXXX
spring.mail.port=587
spring.mail.username=XXXXXXXXXXXXXX
spring.mail.password=XXXXXXXXXXXXXX
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.ssl.enable=false
spring.mail.test-connection=true
```

3. Criar um serviço específico com a função de enviar e-mails;

```
@Service
public class EmailService {
    @Autowired
    public JavaMailSender emailSender;

    @Value("${spring.mail.username}")
    private String emailRemetente;

    @Value("${spring.mail.host}")
    private String emailServerHost;

    @Value("${spring.mail.port}")
    private Integer emailServerPort;

    @Value("${spring.mail.username}")
    private String emailServerUserName;

    @Value("${spring.mail.password}")
    private String emailServerPassword;

    @Value("${mail.from}")
    private String mailFrom;

    //Metodos
}
```

```
@Service
public class EmailService {
    public JavaMailSender javaMailSender() {
        JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
        Properties prop = new Properties();
        mailSender.setHost(emailServerHost);
        mailSender.setPort(emailServerPort);
        mailSender.setUsername(emailServerUserName);
        mailSender.setPassword(emailServerPassword);
        prop.put("mail.smtp.auth", true);
        prop.put("mail.smtp.starttls.enable", true);
        mailSender.setJavaMailProperties(prop);
        return mailSender;
    }

    public EmailService(JavaMailSender javaMailSender) {
        this.emailSender = javaMailSender;
    }

    public void sendTextMail(String toEmail, String subject, String message) {
        var mailMessage = new SimpleMailMessage();
        mailMessage.setTo(toEmail);
        mailMessage.setSubject(subject);
        mailMessage.setText(message);
        mailMessage.setFrom(mailFrom);
        emailSender.send(mailMessage);
    }

    public void sendHtmlMail(String toEmail, String subject, String message) throws Exception {
        this.emailSender = javaMailSender();
        MimeMessage mimeMessage = emailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, true);
        try {
            helper.setFrom(mailFrom);
            helper.setTo(toEmail);
            helper.setSubject(subject);
            StringBuilder sBuilder = new StringBuilder();
            sBuilder.append("<html>\r\n");
            sBuilder.append("    <body>\r\n");
            sBuilder.append("        <div align=\"center\">\r\n");
            sBuilder.append("            E-MAIL NO FORMATO HTML\r\n");
            sBuilder.append("        </div>\r\n");
            sBuilder.append("        <br/>\r\n");
            sBuilder.append("        <center>\r\n");
            sBuilder.append(message);
            sBuilder.append("        </center>\r\n");
            sBuilder.append("    </body>\r\n");
            sBuilder.append("</html>");
            helper.setText(sBuilder.toString(), true);
            emailSender.send(mimeMessage);
        } catch (Exception e) {
            throw new Exception("Erro ao enviar o email - " + e.getMessage());
        }
    }
}
```

Exercício:

1. **Crie um serviço de envio de e-mail na API comércio;**
2. **Configure as propriedades da API para utilizar um e-mail real do Gmail ou Outlook;**
3. **Deverá ser enviado um e-mail, no formato HTML, sempre que um novo produto for cadastrado;**
4. **Teste o envio e o recebimento do e-mail na conta configurada acima.**

Dicas úteis:

1. @CreationTimestamp / @UpdateTimestamp

<https://thorben-janssen.com/persist-creation-update-timestamps-hibernate/>

2. Cors

```
@Configuration
public class ConfigurationCors {

    @SuppressWarnings({ "rawtypes", "unchecked" })
    @Bean
    public FilterRegistrationBean corsFilter() {
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        CorsConfiguration config = new CorsConfiguration().applyPermitDefaultValues();
        config.addAllowedMethod(HttpMethod.PUT);
        config.addAllowedMethod(HttpMethod.DELETE);
        source.registerCorsConfiguration("/**", config);
        FilterRegistrationBean bean = new FilterRegistrationBean(new CorsFilter(source));
        bean.setOrder(0);
        return bean;
    }
}
```

1. @CreationTimestamp / @UpdateTimestamp

<https://thorben-janssen.com/persist-creation-update-timestamps-hibernate/>

2. Cors

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

<https://spring.io/blog/2015/06/08/cors-support-in-spring-framework>

https://github.com/aopaixao/residencia_api_restful/tree/main/configuration

3. Insertable e Updatable no Mapeamento da Entidade

```
@ManyToOne
@JsonBackReference
@JoinColumn(name = "orderid", referencedColumnName = "orderid", insertable = false, updatable = false)
private Orders orders;

@Column(name = "orderid")
private Integer orderId;
```

4. Query e Native Query

```
Class ClienteEnderecoVO{  
  
    private String nomeCliente;  
    private Integer codigoCliente;  
    private String logradouro;  
    private Integer logradouroNumero;  
    private String logradouroComplemento;  
    private String logradouroBairro;  
    private String logradouroCidade;  
    private String logradouroUf;  
  
}
```

```
@Entity  
@Table(name = "cliente")  
public class Cliente{  
  
    @Id  
    @Column(name="codigo_cliente")  
    private Integer id;  
  
    @Column(name="nome_cliente")  
    private String nome;  
  
}
```

```
Class ClienteEnderecoVO{
```

```
    private String nomeCliente;  
    private Integer codigoCliente;  
    private String logradouro;  
    private Integer logradouroNumero;  
    private String logradouroComplemento;  
    private String logradouroBairro;  
    private String logradouroCidade;  
    private String logradouroUf;
```

```
}
```

```
@Entity
```

```
@Table(name = "cliente")
```

```
public class Cliente{
```

```
    @Id
```

```
    @Column(name="codigo_cliente")
```

```
    private Integer id;
```

```
    @Column(name="nome_cliente")
```

```
    private String nome;
```

```
}
```

```
public interface ClienteRepository extends JpaRepository<Cliente,Integer> {
```

```
    @Query(value= "SELECT" +
```

```
        "c.nome AS nomeCliente, c.id_cliente AS codigoCliente, " +
```

```
        "e.logradouro, e.logradouroNumero, e.logradouroComplemento, " +
```

```
        "e.logradouroBairro, e.logradouroCidade, e.logradouroUf " +
```

```
        "FROM cliente c INNER JOIN endereco e " +
```

```
        "ON (c.codigoCliente = e.codigoCliente) " +
```

```
        "WHERE c.codigoCliente = :codigoCliente ", nativeQuery = true)
```

```
    ClienteEnderecoVO listarEnderecoCliente(@Param("codigoCliente") TypedParameterValue id);
```

```
    @Query(value= "SELECT nome, id From Cliente Where id = ?1")
```

```
    Cliente listCodigoAndNome(Integer codigoCliente)
```

```
}
```