

**INSTITUTO INFNET
ESCOLA SUPERIOR DE TECNOLOGIA
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO (LIVE)**



**Disciplina:
Desenvolvimento de Serviços Web e Testes com
Java [24E1_3]**

TP 3

Aluno: Igor de Freitas Monteiro
18/03/2024

TP3

1. Explicar como funcionam API WEB Restful.

- Descreva os conceitos fundamentais de uma API Restful, incluindo recursos, URIs, métodos HTTP, representações de recursos, entre outros.
- Explique como esses conceitos foram aplicados no projeto desenvolvido, destacando como os recursos foram modelados, quais métodos HTTP foram utilizados para cada operação CRUD e como a API segue os princípios Restful.

Uma API Web RESTful é uma interface de programação de aplicações que segue os princípios do estilo arquitetônico REST (Representational State Transfer). Esse estilo arquitetônico foi definido por Roy Fielding em sua tese de doutorado em 2000 e é baseado no protocolo HTTP. O objetivo de uma API RESTful é promover a interoperabilidade entre sistemas na Internet. Vamos detalhar os conceitos fundamentais de uma API RESTful e depois explicar como esses conceitos podem ser aplicados a um projeto.

Conceitos Fundamentais de uma API RESTful

No contexto REST, um recurso é qualquer coisa que possa ser nomeada e que o aplicativo possa querer manipular ou expor aos seus usuários. Exemplos incluem objetos de domínio como usuários, pedidos, produtos, etc. Cada recurso é identificado de forma única através de URIs (Uniform Resource Identifiers).

URIs

URIs são usadas para identificar os recursos dentro do sistema. Uma boa prática em APIs RESTful é usar URIs que fazem sentido para o usuário e que refletem a estrutura dos recursos. Por exemplo, **/users** pode representar todos os usuários, enquanto **/users/123** representa um usuário específico com ID 123.

Métodos HTTP

Os métodos HTTP definem as operações que podem ser realizadas nos recursos. Os principais métodos são:

- **GET**: Solicita a representação de um recurso específico. GETs devem ser idempotentes e seguros, ou seja, não alteram o estado do recurso.
- **POST**: Utilizado para criar um novo recurso.
- **PUT**: Atualiza um recurso existente ou cria um novo recurso na URI especificada.
- **DELETE**: Remove um recurso específico.
- **PATCH**: Aplica atualizações parciais a um recurso.

Recursos podem ser representados em vários formatos, como JSON, XML, HTML, etc. Uma API RESTful tipicamente usa JSON para comunicação devido à sua facilidade de uso e compatibilidade com a maioria das tecnologias web.

Estados e Conexões

Em REST, as interações são stateless, o que significa que cada requisição HTTP deve conter todas as informações necessárias para ser compreendida por si só, sem depender do estado armazenado no servidor.

Aplicação dos Conceitos no Projeto

Modelagem dos Recursos

- Recursos foram definidos para representar entidades do domínio, como **/escola** para acessar escolas e professores para professores.

Métodos HTTP para Operações CRUD

- **GET /escola/listar** para listar todas as escolas, **GET /escola/obter** para obter uma escola específica.
- **POST /escola/incluir** para criar uma nova escola.
- **DELETE /escola/{nome}** para excluir uma escola.
- Cada uma dessas operações respeita os princípios RESTful, usando o método HTTP apropriado para a ação desejada e aplicando as operações ao recurso identificado pela URI.

Princípios RESTful Seguidos

- A API foi projetada para ser **stateless**: cada requisição contém todas as informações necessárias para processá-la.
- As **representações de recursos** (usualmente em JSON) são usadas para comunicar o estado atual dos recursos entre cliente e servidor.
- Uso adequado de **códigos de status HTTP** para indicar o resultado das operações, como **200 OK** para sucesso, **201 Created** para recursos criados, **404 Not Found** para recursos não encontrados, etc.

Parte 2 – TP3

HTTP (Hypertext Transfer Protocol)

O HTTP é um protocolo de camada de aplicação utilizado para transmitir documentos hipermedia, como HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que os pedidos são iniciados pelos recipientes, como um navegador web. O servidor, que armazena recursos como páginas HTML e imagens, responde aos pedidos.

Funcionamento Básico:

1. **Cliente faz uma requisição HTTP**: O cliente (navegador, por exemplo) envia uma mensagem de requisição ao servidor. Esta mensagem inclui um método de requisição (GET, POST, DELETE, etc.), URI e versão do protocolo, seguido por um cabeçalho de mensagem contendo metainformações.
2. **Servidor responde**: O servidor processa o pedido e retorna uma mensagem de resposta para o cliente. A resposta contém um código de status (indicando se o pedido foi bem-sucedido ou não), e, se positivo, o recurso solicitado.

Métodos HTTP na API Desenvolvida:

- **GET**: Utilizado para solicitar dados de um recurso específico. No contexto da API, seria usado para recuperar informações, como a lista de estudantes ou um estudante específico.
- **POST**: Empregado para enviar dados ao servidor para criar um novo recurso. Na API, poderia ser usado para adicionar um novo estudante.

- **DELETE:** Aplicado para excluir um recurso existente. Na API, isso permitiria remover um estudante específico do registro.

HTTPS (HTTP Secure)

O HTTPS é uma extensão do HTTP. Ele é usado para comunicação segura em uma rede de computadores e amplamente utilizado na Internet. No HTTPS, a comunicação é criptografada usando o Protocolo SSL (Secure Sockets Layer) ou TLS (Transport Layer Security).

Principais Características e Funcionamento:

- **Criptografia:** O HTTPS criptografa os dados da sessão, o que significa que os dados são transformados em um código para impedir o acesso não autorizado durante a transmissão.
- **Autenticação:** O HTTPS proporciona uma maneira de verificar a autenticidade do site visitado, protegendo contra ataques man-in-the-middle.
- **Integridade dos Dados:** Garante que os dados transmitidos não sejam alterados ou corrompidos durante a transferência.

Diferenças Entre HTTP e HTTPS:

- **Segurança:** A principal diferença é a camada adicional de segurança que o HTTPS oferece através da criptografia.
- **Porta Padrão:** HTTP usa a porta 80 por padrão, enquanto o HTTPS usa a porta 443.
- **Custo de Performance:** Devido à criptografia, o HTTPS pode ser ligeiramente mais lento que o HTTP. No entanto, as melhorias na tecnologia de computação e otimizações de rede têm minimizado essas diferenças.

Importância do HTTPS e Implementação no Projeto:

O HTTPS é crucial para proteger a comunicação na Internet, especialmente para transações sensíveis, como compras online e operações bancárias. Ele protege contra ataques, garantindo que os dados dos usuários permaneçam privados e seguros.

Para implementar HTTPS em um projeto Spring Boot:

1. **Obter um Certificado SSL:** Pode-se obter um certificado de uma Autoridade Certificadora (AC) ou usar ferramentas como Let's Encrypt para um certificado gratuito.
2. **Configurar o Spring Boot:** Configurar o servidor embutido para usar o certificado SSL, especificando o caminho para o keystore, a senha e o protocolo (TLS) no **application.properties** ou **application.yml**.

Por exemplo, no **application.properties**:

propertiesCopy code

```
server.port=443 server.ssl.key-store=classpath:keystore.p12 server.ssl.key-store-
password=yourpassword server.ssl.keyStoreType=PKCS12 server.ssl.keyAlias=tomcat
```

Implementar HTTPS aumenta a confiabilidade e a segurança da aplicação, protegendo tanto os usuários quanto os desenvolvedores de possíveis vulnerabilidades e ataques cibernéticos.

Parte 3 e 4- TP3

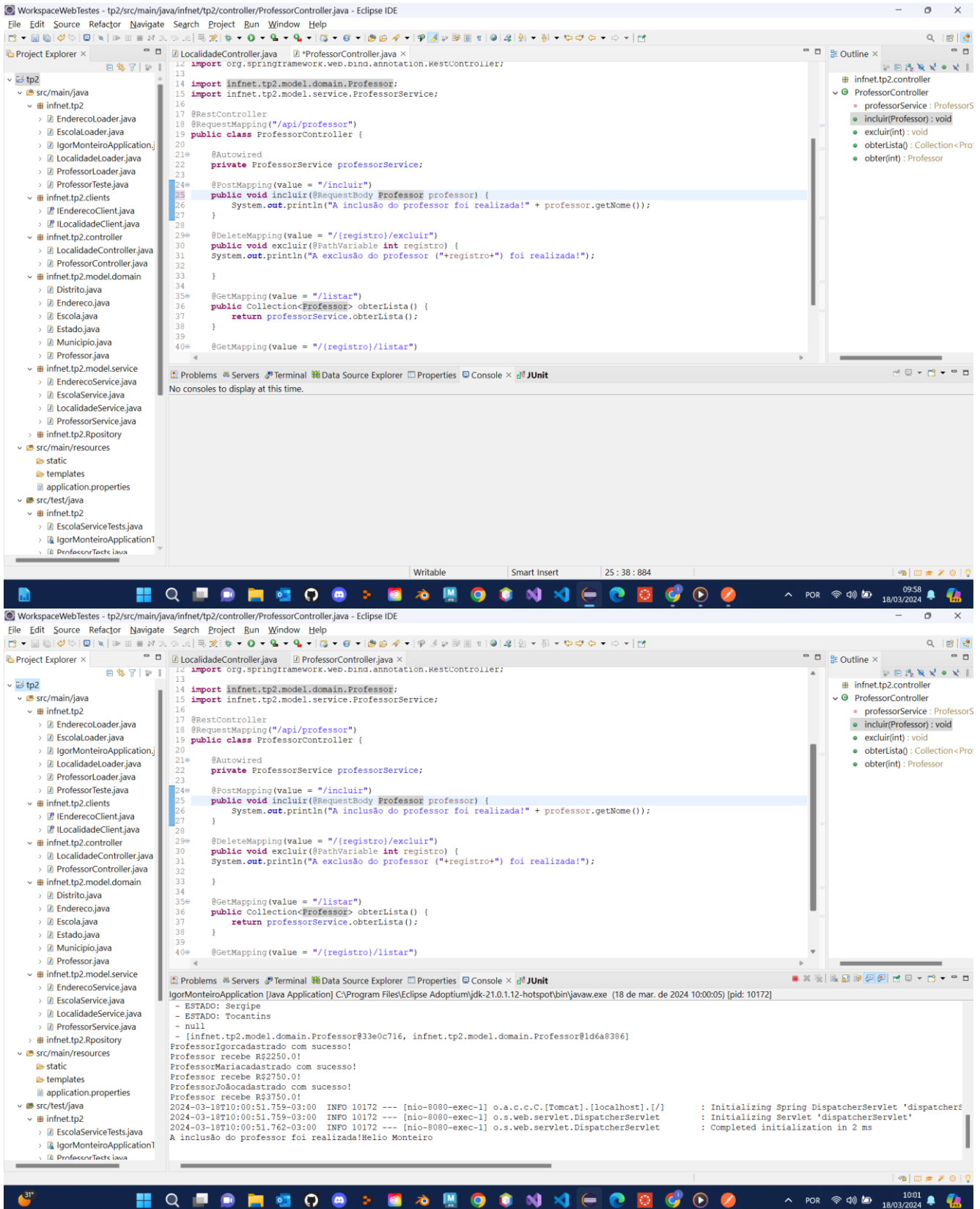
The image displays two screenshots of the Postman API client interface, showing a workspace named 'API Network'.

Top Screenshot: A GET request is configured for the endpoint `http://localhost:8080/api/professor/234/listar`. The response is displayed in the 'Body' tab, showing a JSON object with the following structure:

```
1 {
2   "registro": 234,
3   "nome": "Igor Monteiro",
4   "salario": 777.0,
5   "mestre": false,
6   "endereco": {
7     "cep": "20010020",
8     "logradouro": "Prof Rua São José",
9     "complemento": "Prof 4º andar",
10    "bairro": "Prof Centro",
11    "localidade": "Prof Rio de Janeiro",
12    "uf": "UF",
13    "nome": null
14  }
15 }
```

Bottom Screenshot: A POST request is configured for the endpoint `http://localhost:8080/api/professor/incluir`. The request body is set to 'JSON' and contains a JSON object with the following structure:

```
1 {
2   "registro": 1980,
3   "nome": "Helio Monteiro",
4   "salario": 1000.0,
5   "mestre": false,
6   "endereco": {
7     "cep": "20010020",
8     "logradouro": "Prof Rua São José",
9     "complemento": "Prof 4º andar",
10    "bairro": "Prof Centro",
11    "localidade": "Prof Rio de Janeiro",
12    "uf": "UF",
13    "nome": null
14  }
15 }
```



Home Workspaces API Network Search Postman

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. [Learn more](#)

My Workspace New Import Overview Getting profess GET listar GET obter POST inclui GET excluir No environment

Collections

- Api - Igor Monteiro
 - professor
 - GET listar
 - GET obter
 - POST incluir
 - GET excluir
 - localidade
 - This folder is empty
 - [Add a request](#) to start working.
 - escola
 - This folder is empty
 - [Add a request](#) to start working.

Api - Igor Monteiro / professor / excluir

GET http://localhost:8080/api/professor/123/excluir [Send](#)

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (6) Test Results 405 Method Not Allowed 33 ms 318 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2024-03-18T13:16:24.113+00:00",
3   "status": 405,
4   "error": "Method Not Allowed",
5   "path": "/api/professor/123/excluir"
```

WorkspaceWebTests - tp2/src/main/java/infnet/tp2/controller/ProfessorController.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- tp2
 - src/main/java
 - infnet.tp2
 - EnderecoLoader.java
 - EscolaLoader.java
 - IgorMonteiroApplication.java
 - LocalidadeLoader.java
 - ProfessorLoader.java
 - ProfessorTeste.java
 - infnet.tp2.clients
 - EnderecoClient.java
 - LocalidadeClient.java
 - infnet.tp2.controller
 - LocalidadeController.java
 - ProfessorController.java
 - infnet.tp2.model.domain
 - Distrito.java
 - Endereco.java
 - Escola.java
 - Estado.java
 - Municipio.java
 - Professor.java
 - infnet.tp2.model.service
 - EnderecoService.java
 - EscolaService.java
 - LocalidadeService.java
 - ProfessorService.java
 - infnet.tp2.repository
 - src/main/resources
 - static
 - templates
 - application.properties
 - src/test/java
 - infnet.tp2
 - EscolaServiceTests.java
 - IgorMonteiroApplicationTests.java
 - ProfessorTeste.java

LocalidadeController.java ProfessorController.java

```
import org.springframework.web.bind.annotation.RestController;
import infnet.tp2.model.domain.Professor;
import infnet.tp2.model.service.ProfessorService;

@RestController
@RequestMapping("/api/professor")
public class ProfessorController {

    @Autowired
    private ProfessorService professorService;

    @PostMapping(value = "/incluir")
    public void incluir(@RequestBody Professor professor) {
        System.out.println("A inclusão do professor foi realizada!" + professor.getNome());
    }

    @DeleteMapping(value = "/(registro)/excluir")
    public void excluir(@PathVariable int registro) {
        System.out.println("A exclusão do professor (" + registro + ") foi realizada!");
    }

    @GetMapping(value = "/listar")
    public Collection<Professor> obterLista() {
        return professorService.obterLista();
    }

    @GetMapping(value = "/(registro)/listar")
```

Outline

- infnet.tp2.controller
 - ProfessorController
 - professorService : ProfessorService
 - incluir(Professor) : void
 - excluir(int) : void
 - obterLista() : Collection<Professor>
 - obter(int) : Professor

Problems Servers Terminal Data Source Explorer Properties Console x JUnit

IgorMonteiroApplication [Java Application] C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\javaw.exe (18 de mar. de 2024 10:00:05) [pid: 10172]

Professor recebe R\$2250.0!

ProfessorMariaCadastrado com sucesso!

Professor recebe R\$2750.0!

ProfessorJoãoCadastrado com sucesso!

Professor recebe R\$3750.0!

2024-03-18T10:00:51.759-03:00 INFO 10172 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'

2024-03-18T10:00:51.759-03:00 INFO 10172 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'

2024-03-18T10:00:51.762-03:00 INFO 10172 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms

A inclusão do professor foi realizada! Helio Monteiro

2024-03-18T10:15:40.733-03:00 WARN 10172 --- [nio-8080-exec-5] w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'DELETE' is not supported]

2024-03-18T10:16:24.111-03:00 WARN 10172 --- [nio-8080-exec-8] w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'DELETE' is not supported]

2024-03-18T10:16:52.829-03:00 WARN 10172 --- [nio-8080-exec-10] w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'DELETE' is not supported]

2024-03-18T10:17:08.072-03:00 WARN 10172 --- [nio-8080-exec-2] w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'DELETE' is not supported]

A exclusão do professor (123) foi realizada!

Home Workspaces API Network Search Postman Invite Upgrade

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. Learn more

My Workspace New Import Overview Getting profess GET listar GET obter POST inclui DEL exclui No environment

Collections + - ...

Environments

History

Api - Igor Monteiro

professor

GET listar

GET obter

POST incluir

GET excluir

localidade

This folder is empty
Add a request to start working.

escola

This folder is empty
Add a request to start working.

Api - Igor Monteiro / professor / excluir

DELETE http://localhost:8080/api/professor/123/excluir Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (4) Test Results 200 OK 8 ms 123 B Save as example

Pretty Raw Preview Visualize Text

1

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

Home Workspaces API Network Search Postman Invite Upgrade

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. Learn more

My Workspace New Import GET listar GET obter POST incluir DEL exclui No environment

Collections + - ...

Environments

History

Api - Igor Monteiro

professor

GET listar

GET obter

POST incluir

DEL excluir

localidade

This folder is empty
Add a request to start working.

escola

This folder is empty
Add a request to start working.

Api - Igor Monteiro / professor / listar

GET http://localhost:8080/api/professor/listar Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 6 ms 894 B Save as example

Pretty Raw Preview Visualize JSON

```
31 {
32   "registro": 1980,
33   "nome": "Helio Monteiro",
34   "salario": 1000.0,
35   "mestre": false,
36   "endereco": {
37     "cep": "20010020",
38     "logradouro": "Prof Rua São José",
39     "complemento": "Prof 4º andar",
40     "bairro": "Prof Centro",
41     "localidade": "Prof Rio de Janeiro",
42     "uf": "UF",
43   }
31 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

WorkspaceWebTestes - tp2/src/main/java/infnet/tp2/controller/EscolaController.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- tp2
 - src/main/java
 - infnet/tp2
 - EnderecoLoader.java
 - EscolaLoader.java
 - IgorMonteiroApplication.java
 - LocalidadeLoader.java
 - ProfessorLoader.java
 - ProfessorTeste.java
 - infnet/tp2.clients
 - IEnderecoClient.java
 - ILocalidadeClient.java
 - infnet/tp2.controller
 - EscolaController.java
 - LocalidadeController.java
 - ProfessorController.java
 - infnet/tp2.model.domain
 - Distrito.java
 - Endereco.java
 - Escola.java
 - Estado.java
 - Municipio.java
 - Professor.java
 - infnet/tp2.model.service
 - EnderecoService.java
 - EscolaService.java
 - LocalidadeService.java
 - ProfessorService.java
 - infnet/tp2.repository
 - infnet/tp2.repository
 - src/main/resources
 - static
 - templates
 - application.properties
 - src/test/java
 - infnet/tp2
 - EscolaServiceTests.java
 - IgorMonteiroApplicationTests.java

Outline

- infnet/tp2.controller
 - EscolaController
 - escolaService : EscolaService
 - incluir(Escola) : void
 - excluir(String) : void
 - obterLista() : Collection<Escola>
 - obter(String) : Escola

```

21 public class EscolaController {
22
23     @Autowired
24     private EscolaService escolaService;
25
26     @PostMapping(value = "/incluir")
27     public void incluir(@RequestBody Escola escola) {
28         escolaService.incluir(escola);
29         //System.out.println("A inclusão do professor foi realizada!" + professor.getNome());
30     }
31
32     @DeleteMapping(value = "/{nome}/excluir")
33     public void excluir(@PathVariable String nome) {
34         escolaService.excluir(nome);
35         //System.out.println("A exclusão do professor (" + registro + ") foi realizada!");
36     }
37
38     @GetMapping(value = "/listar")
39     public Collection<Escola> obterLista() {
40         return escolaService.obterLista();
41     }
42
43     @GetMapping(value = "/{nome}/listar")
44     public Escola obter(@PathVariable String nome) {
45         return escolaService.obter(nome);
46     }
47 }
48
49

```

Problems Servers Terminal Data Source Explorer Properties Console x JUnit

No consoles to display at this time.

Postman

Home Workspaces API Network Search Postman Invite Upgrade

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. [Learn more](#)

My Workspace New Import

Collections

- Api - Igor Monteiro
 - professor
 - GET listar
 - GET obter
 - POST incluir
 - DEL excluir
 - escola
 - GET listar
 - GET obter
 - POST incluir
 - DEL excluir
 - localidade

Environments

History

GET obter

Api - Igor Monteiro / professor / obter

GET http://localhost:8080/api/escola/instituto Infnet/listar Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK 27 ms 389 B Save as example


Pretty Raw Preview Visualize JSON

```

1 {
2   "nome": "Instituto Infnet",
3   "email": "instituto@infnet",
4   "professores": null,
5   "endereco": {
6     "cep": "35790-291",
7     "logradouro": "Avenida Soares dos Santos",
8     "complemento": "",
9     "bairro": "Centro",
10    "localidade": "Curvelo",
11    "uf": "MG",
12    "nome": null

```

Hoje



[Postman-win64-Setup.exe](#)
<https://dl.pstmn.io>
[Mostrar numa pasta](#)

×

⌵

Home

Workspaces

API Network

Search Postman

Invite

Upgrade

Learn from experts and join the Postman community at POST/CON 24. Register by March 26 to save 30%. [Learn more](#)

My Workspace

New Import

Overview

Getting started

professor

GET listar

No environment

Collections

Environments

History

API - Igor Monteiro

professor

localidade

escola

GET

http://localhost:8080/api/professor/listar

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key

Value

Description

Bulk Edit

Body

Cookies

Headers (5)

Test Results

200 OK

10 ms

644 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "registro": 0,
4     "nome": null,
5     "salario": 0.0,
6     "mestre": false,
7     "endereco": {
8       "cep": "20041-005",
9       "logradouro": "Rua do Rosário",
10      "complemento": "de 115 ao fim - lado impar",
11      "bairro": "Centro",
12      "localidade": "Rio de Janeiro",
13      "uf": "RJ",
14      "nome": null
15    }
16  },
17  ]
```

Online

Find and replace

Console

Postbot

Runner

Start Proxy

Cookies

Trash

Parte 5 – TP3

```
package infnet.tp2.controller;

import java.util.Collection;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import infnet.tp2.model.domain.Escola;
import infnet.tp2.model.service.EscolaService;

@RestController
@RequestMapping("/api/escola")
public class EscolaController {

    @Autowired
    private EscolaService escolaService;

    @PostMapping(value = "/incluir")
    public void incluir(@RequestBody Escola escola) {
        escolaService.incluir(escola);
    }

    @DeleteMapping(value =("/{nome})/excluir")
    public void excluir(@PathVariable String nome) {
        escolaService.excluir(nome);
    }

    @GetMapping(value = "/listar")
    public Collection<Escola> obterLista() {
        return escolaService.obterLista();
    }

    @GetMapping(value =("/{nome})/listar")
    public Escola obter(@PathVariable String nome) {
        return escolaService.obter(nome);
    }
}
```