Engenharia de Software

25/8/2025

Samuel Hermany

INSTITUTO INFNET

**TP2**

Desenvolvimento de Serviços com SpringBoot

Professor(a): Flávio da Silva Neves

# Link GitHub

<https://github.com/faculdade-infnet/V-1-Spring_Boot/tree/main/TP2>

# Versão do Java

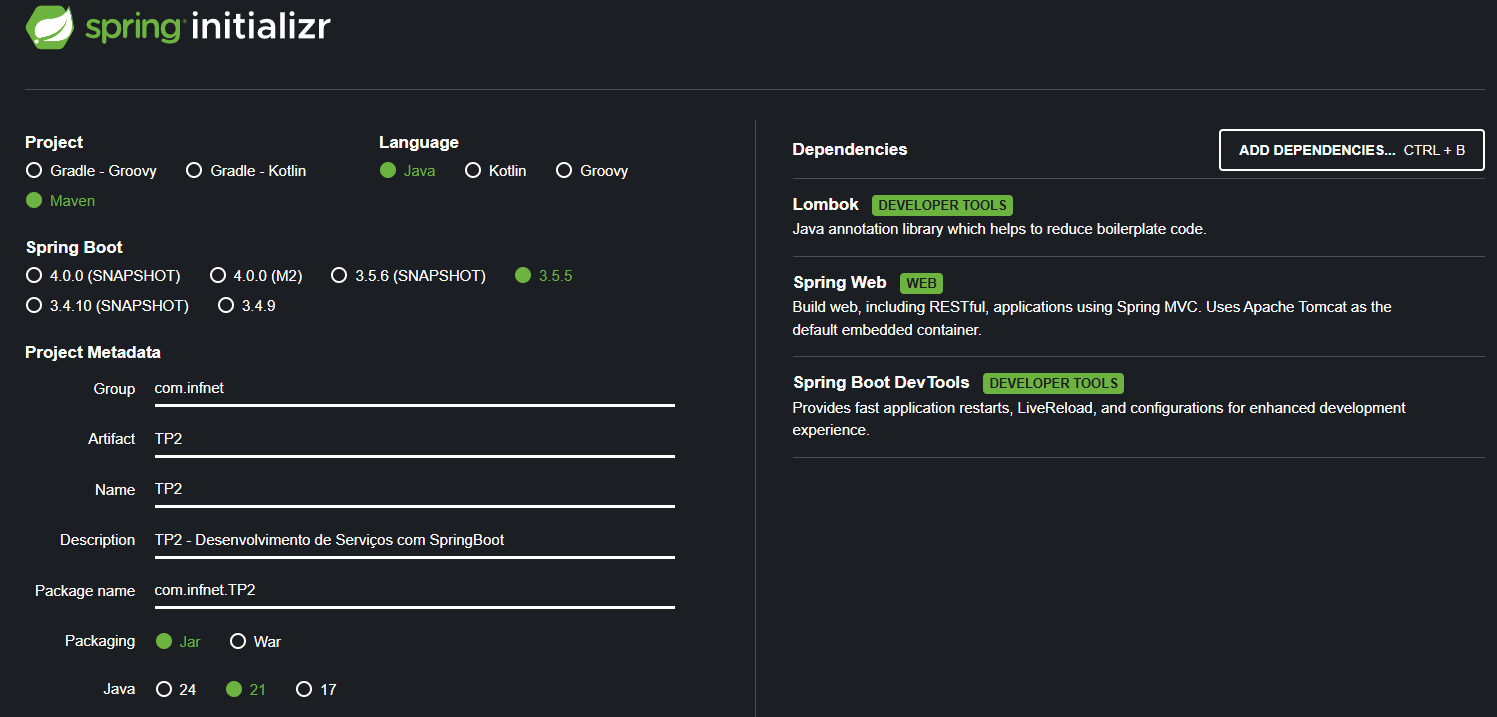
## Versão Java

Escolhida a versão Java 21, por ser a versão LTS (Long-Term Support) mais recente, ou seja, ela recebe suporte estendido de longo prazo, com atualizações de segurança e correções por vários anos, garantindo estabilidade e confiabilidade para aplicações em produção.

## Versão Spring Boot

Escolhida a versão Spring Boot 3.5.5, que é versão estável mais recente (GA – General Availability), tendo as últimas correções de bugs e melhoria, além de ser totalmente compatível com o Java 21 e com isso proporcionando um ambiente moderno e robusto para desenvolvimento de aplicações de grande porte.

# Iniciação do Projeto



# Gerenciamento de Dependências:

Utilizei o arquivo pom.xml do Maven para declarar todas as dependências necessárias.

Os principais benefícios da minha escolha:

* Controle claro das bibliotecas usadas.
* Facilidade para adicionar/remover dependências.

# Utilização de Autoconfiguração:

Spring Boot configura automaticamente o contexto da aplicação para o Spring Web, via

dependência spring-boot-starter-web, evitando necessidade de configuração manual de

servidores, roteamento, e beans.

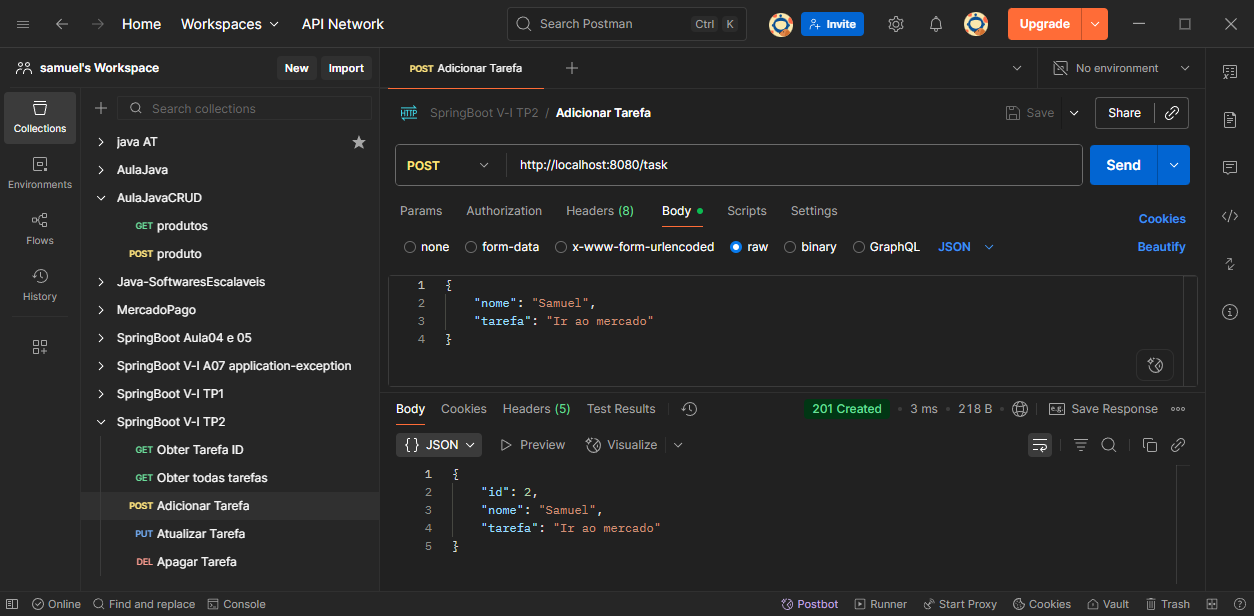
Utilizei em conjunto o @SpringBootApplication que inclui @EnableAutoConfiguration, no início

onde da aplicação, resultando em um código mais limpo, menos repetição de código e foco no

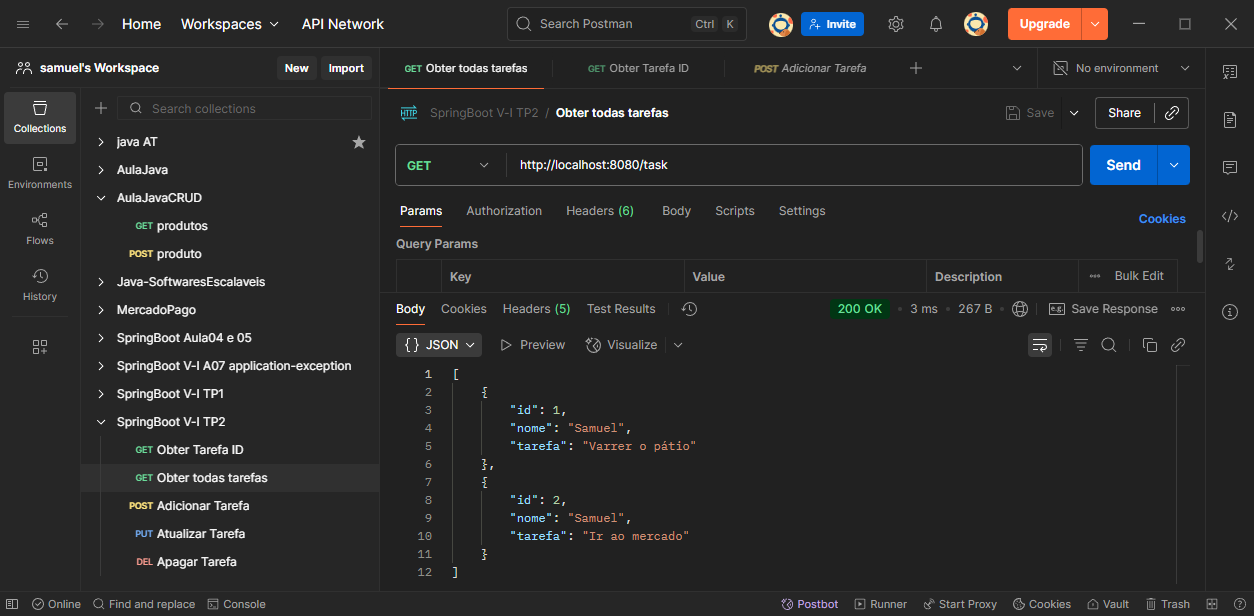
negócio.

# Validação dos códigos de status HTTP de acordo com as especificações REST:

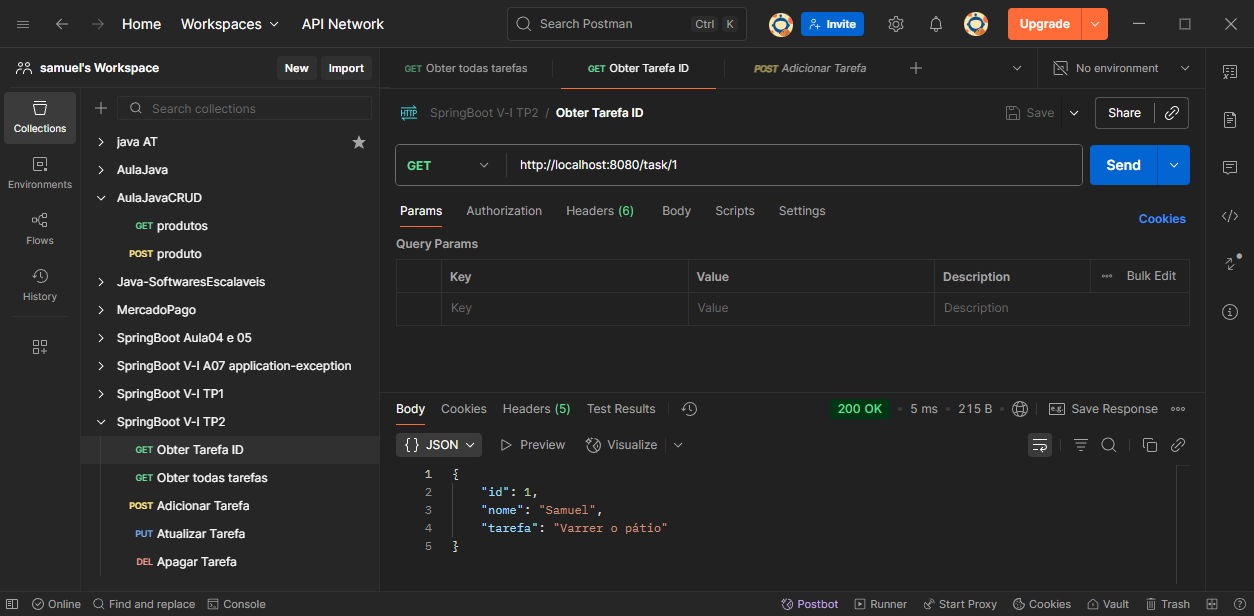
## POST – Adicionar Tarefa



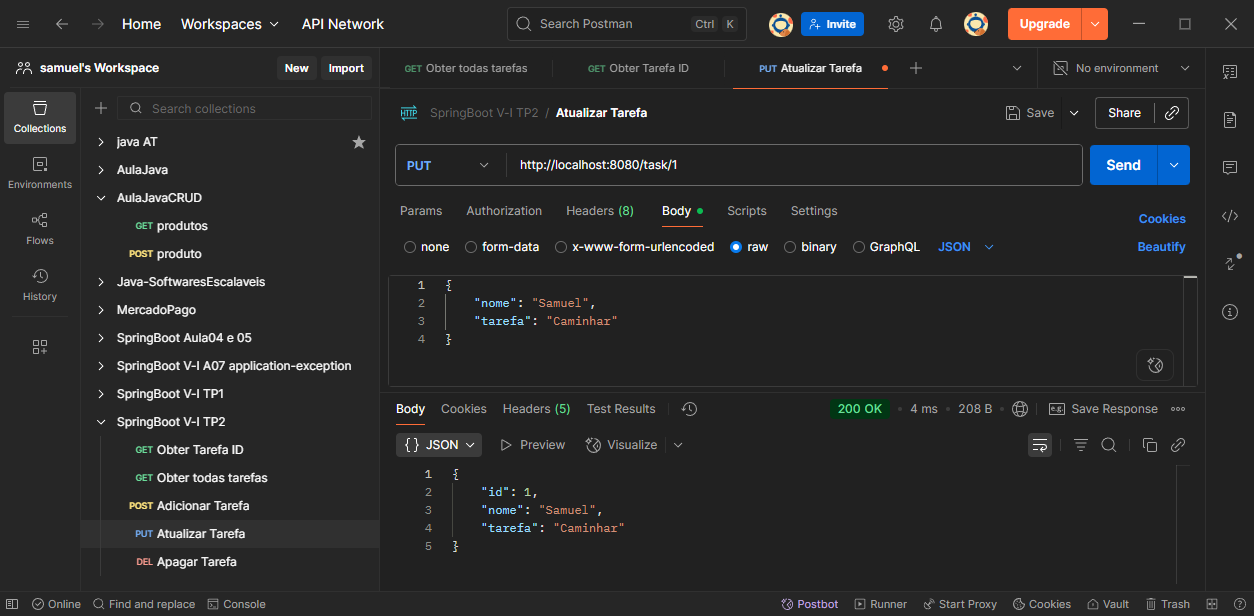
## GET – Obter todas as tarefas



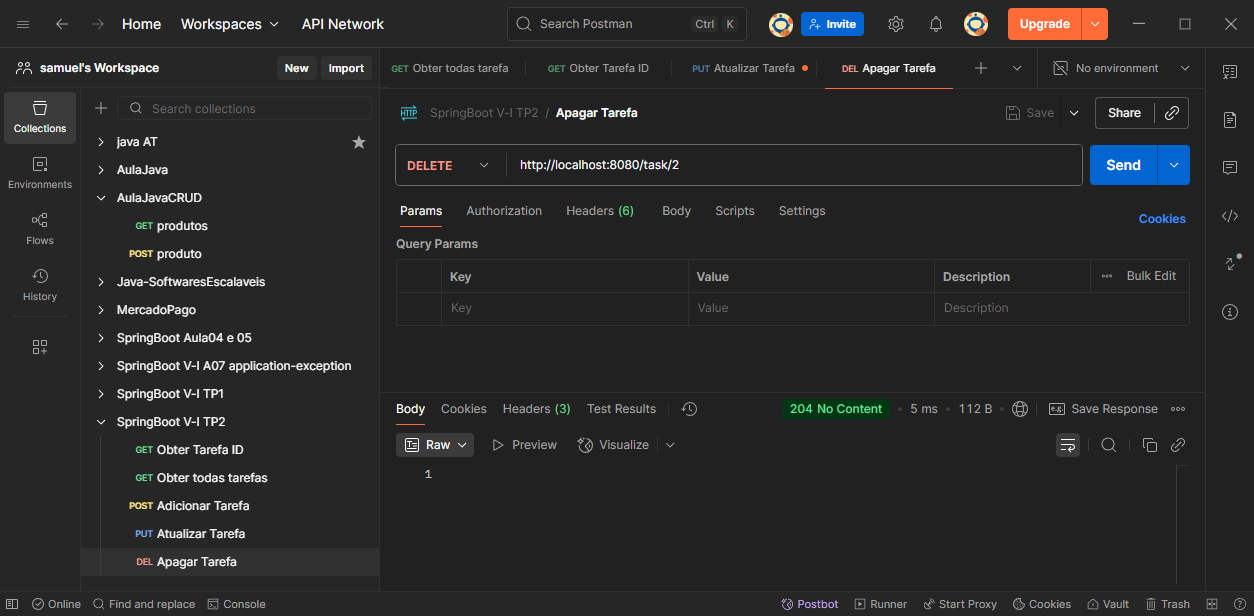
## GET – Obter tarefa por ID



## PUT – Atualizar Tarefa



## DELETE – Deletar tarefa



# Desenvolvimento de Serviços REST:

package com.infnet.TP2.api;  
  
import com.infnet.TP2.facade.TaskFacade;  
import com.infnet.TP2.models.TaskModels;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.MediaType;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@Controller  
@RequestMapping(value = "/task", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
public class TaskAPI {  
  
 // Ativa a injeção de dependência automática  
 @Autowired  
 private TaskFacade taskFacade;  
  
 // Adiciona uma tarefa  
 @PostMapping  
 @ResponseBody  
 public ResponseEntity<TaskModels> create(@RequestBody TaskModels taskModels) {  
 TaskModels created = taskFacade.create(taskModels);  
 return ResponseEntity.*status*(HttpStatus.*CREATED*).body(created); // 201 Created  
 }  
  
 // Obtém todas as tarefas  
 @GetMapping  
 @ResponseBody  
 public ResponseEntity<List<TaskModels>> getAll(){  
 List<TaskModels> tasks = taskFacade.getAll();  
 return ResponseEntity.*ok*(tasks); // 200 OK  
 }  
  
 // Obtém uma tarefa por id  
 @GetMapping("/{taskId}")  
 @ResponseBody  
 public ResponseEntity<TaskModels> getById(@PathVariable Long taskId) {  
 TaskModels task = taskFacade.getById(taskId);  
 if (task == null) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).build(); // 404 Not Found  
 }  
 return ResponseEntity.*ok*(task); // 200 OK  
 }  
  
 // Atualiza uma tarefa por id  
 @PutMapping("/{taskId}")  
 @ResponseBody  
 public ResponseEntity<TaskModels> update(@PathVariable("taskId") Long taskId, @RequestBody TaskModels taskModels) {  
 TaskModels existing = taskFacade.getById(taskId);  
 if (existing == null) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).build(); // 404 Not Found  
 }  
 taskModels.setId(taskId);  
 TaskModels updated = taskFacade.update(taskModels, taskId);  
 return ResponseEntity.*ok*(updated); // 200 OK  
 }  
  
 // Deleta uma tarefa por id  
 @DeleteMapping("/{taskId}")  
 @ResponseBody  
 public ResponseEntity<Void> delete(@PathVariable("taskId") Long taskId) {  
 TaskModels existing = taskFacade.getById(taskId);  
 if (existing == null) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).build(); // 404 Not Found  
 }  
 taskFacade.delete(taskId);  
 return ResponseEntity.*noContent*().build(); // 204 No Content  
 }  
}