Engenharia de Software

11/8/2025

Samuel Hermany

INSTITUTO INFNET

**TP1**

Desenvolvimento de Serviços com SpringBoot

Professor(a): Flávio da Silva Neves

# Link GitHub

<https://github.com/faculdade-infnet/V-1-Spring_Boot/tree/main/TP1>

# Escolha da Ferramenta de Construção:

Para projetos acadêmicos e pequenos, Maven é mais simples e direto

# Iniciação do Projeto:

Eu escolhi o Spring Initializr pois ele permite criar rapidamente um projeto Spring Boot configurado com as dependências desejadas, pois eu prefiro algo mais visual e sequencial ao invés da linha de comando.

# 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:

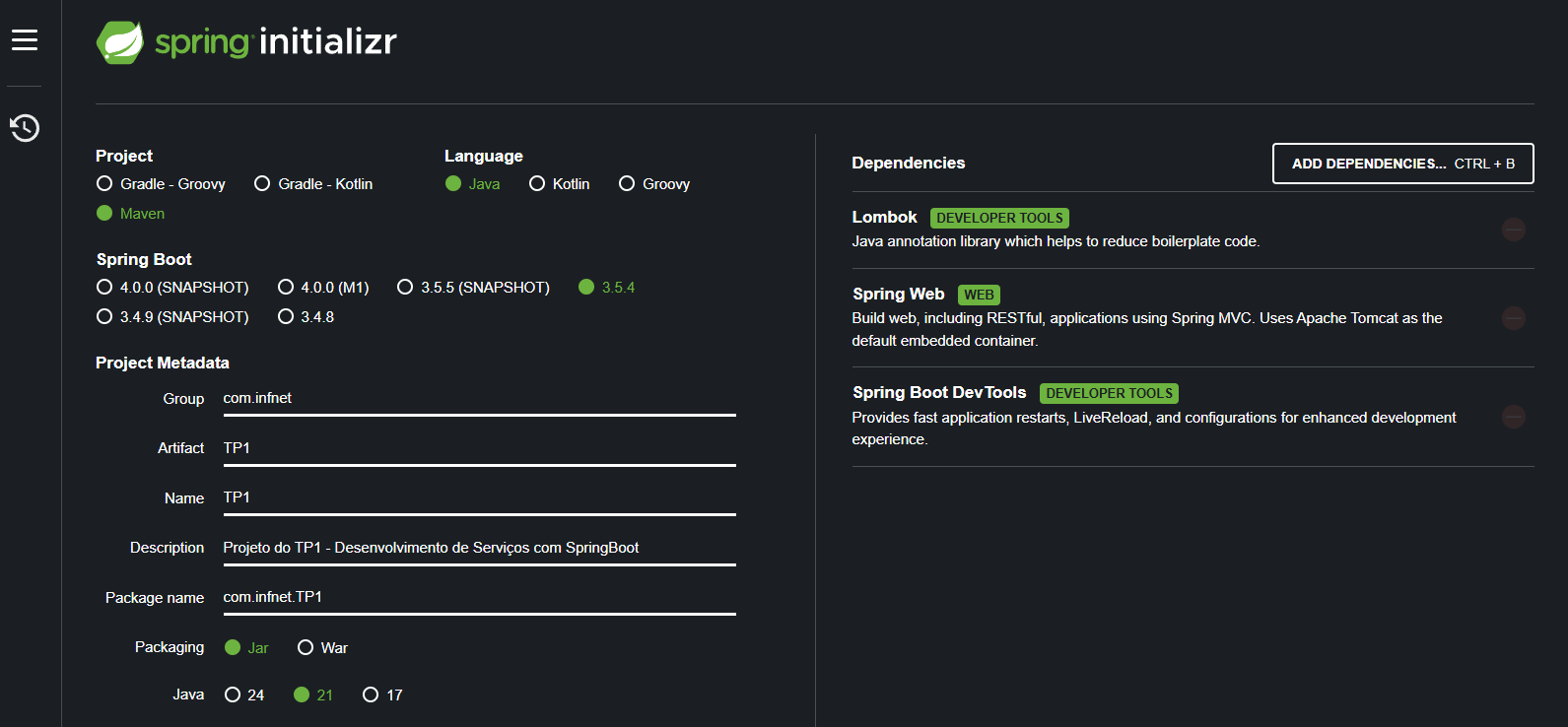
Spring Boot configura automaticamente o contexto da aplicação para o Spring Web, via dependência spring-boot-starter-web, evitando necessidade de configuração manual de servidores, roteamento, e beans.

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.

# Configuração da IDE:

Utilizei IntelliJ IDEA. Com as seguintes configurações:

* Importação do projeto Maven.
* Ativação do plugin Lombok (instalado no IntelliJ).
* Configuração do JDK 21 para o projeto.
* Configuração do Spring Boot run/debug configurations para executar o projeto com facilidade.



# Desenvolvimento de Serviços REST

Organização das Rotas

## Adição

* GET /adicao?number1=5&number2=3 → Recebe parâmetros via URL.
* POST /adicao → Recebe um JSON no corpo da requisição no formato:

json

{

"number1": 5,

"number2": 3

}

## Subtração

* GET /subtracao?number1=5&number2=3
* POST /subtracao → JSON no corpo.

## Multiplicação

* GET /multiplicacao?number1=5&number2=3
* POST /multiplicacao → JSON no corpo.

## Divisão

* GET /divisao?number1=5&number2=3
* Retorna erro 400 caso number2 seja 0.
* POST /divisao → JSON no corpo, também com validação para divisão por zero.

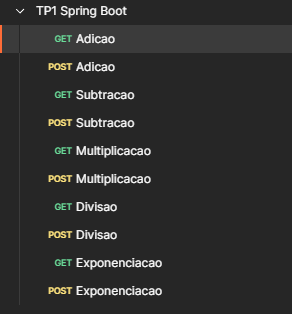
## Exponenciação

* GET /exponenciacao?number1=2&number2=3
* POST /exponenciacao → JSON no corpo.

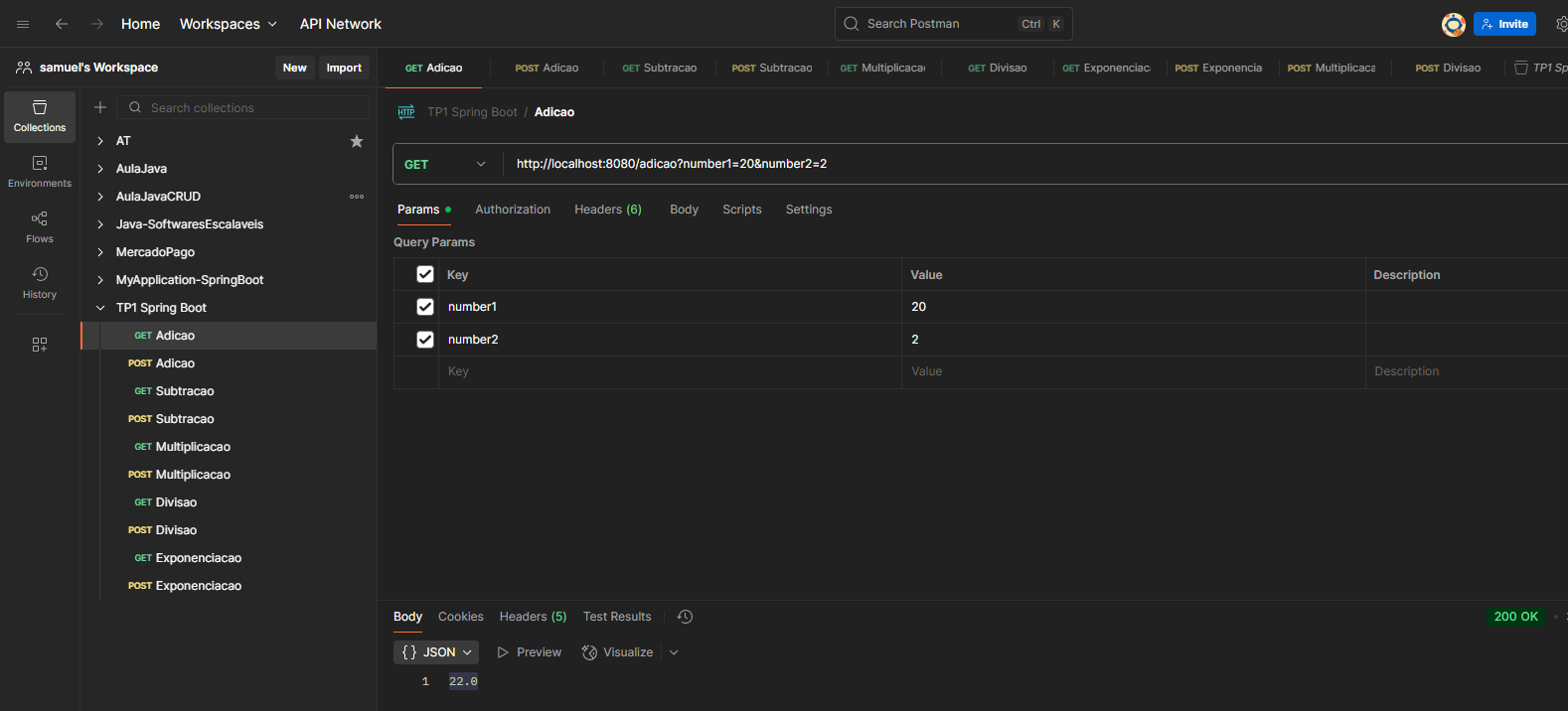
## Padrão Adotado

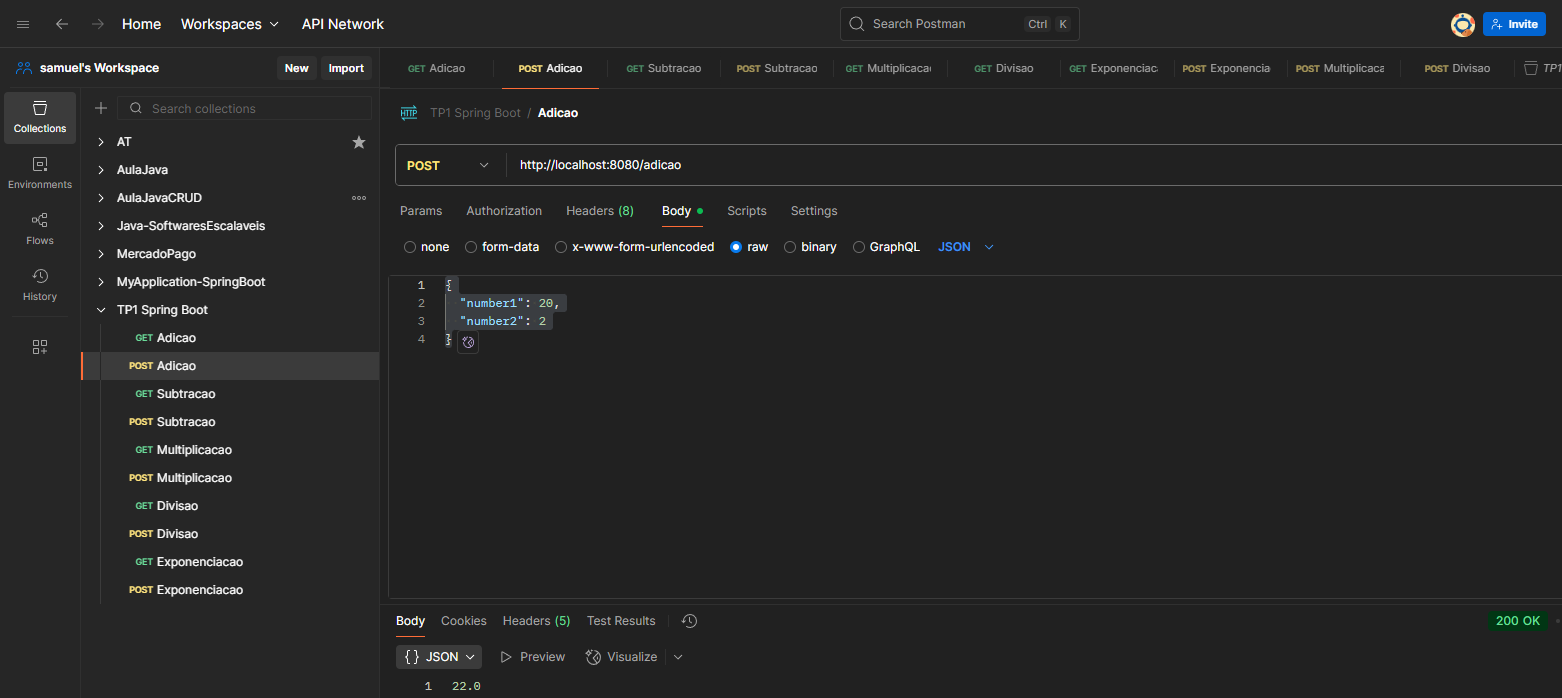
* GET: usado para testes rápidos via query string, ideal para chamadas diretas no navegador ou no Postman sem corpo na requisição.
* POST: usado para enviar os números no corpo da requisição em formato JSON, permitindo melhor integração com sistemas que enviam dados mais complexos.
* Validação: na operação de divisão, foi usado ResponseEntity para retornar mensagens de erro adequadas.
* Modelo de Dados: a classe CalculationModel centraliza os atributos number1 e number2, facilitando a manutenção e o reuso.

## Testes dos Endpoints com POSTMAN

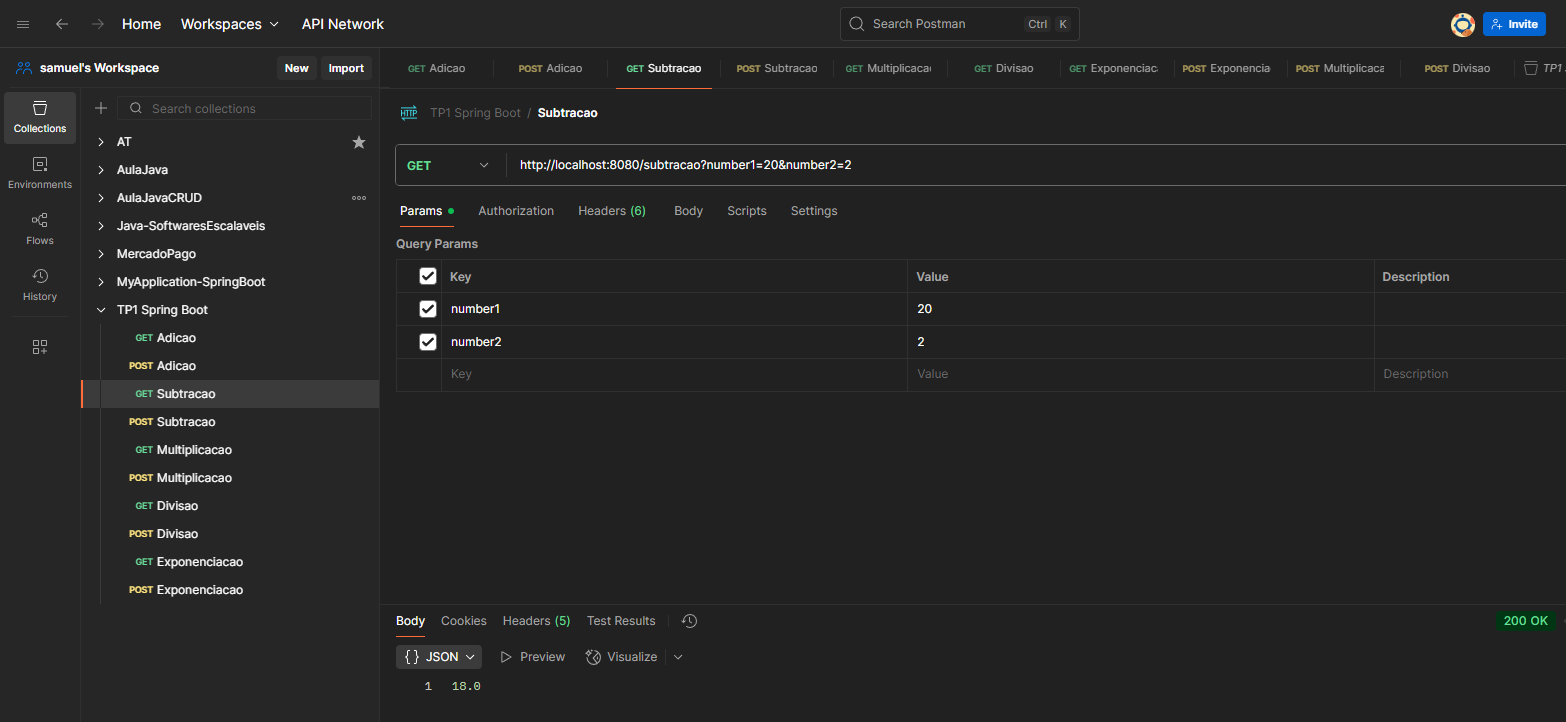


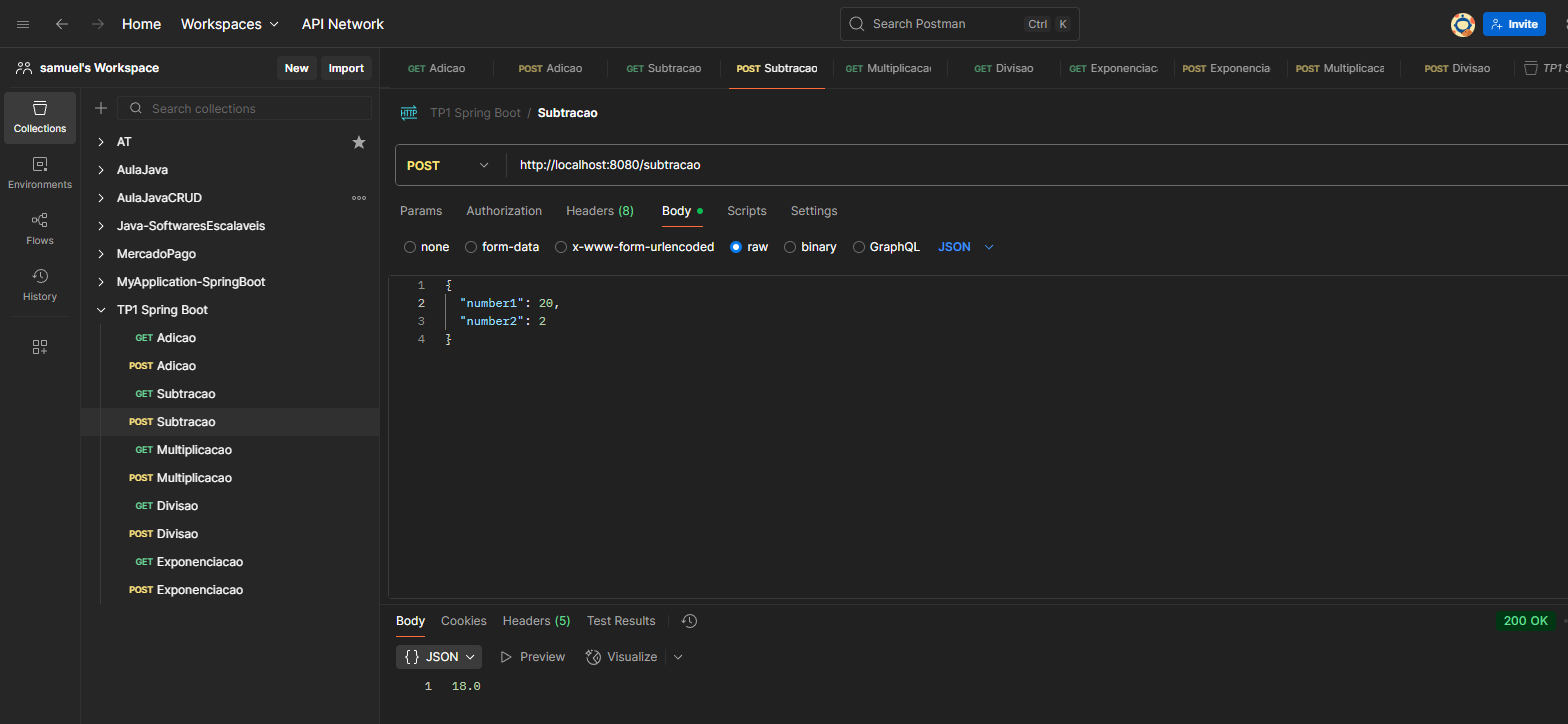
### Adição – GET / POST



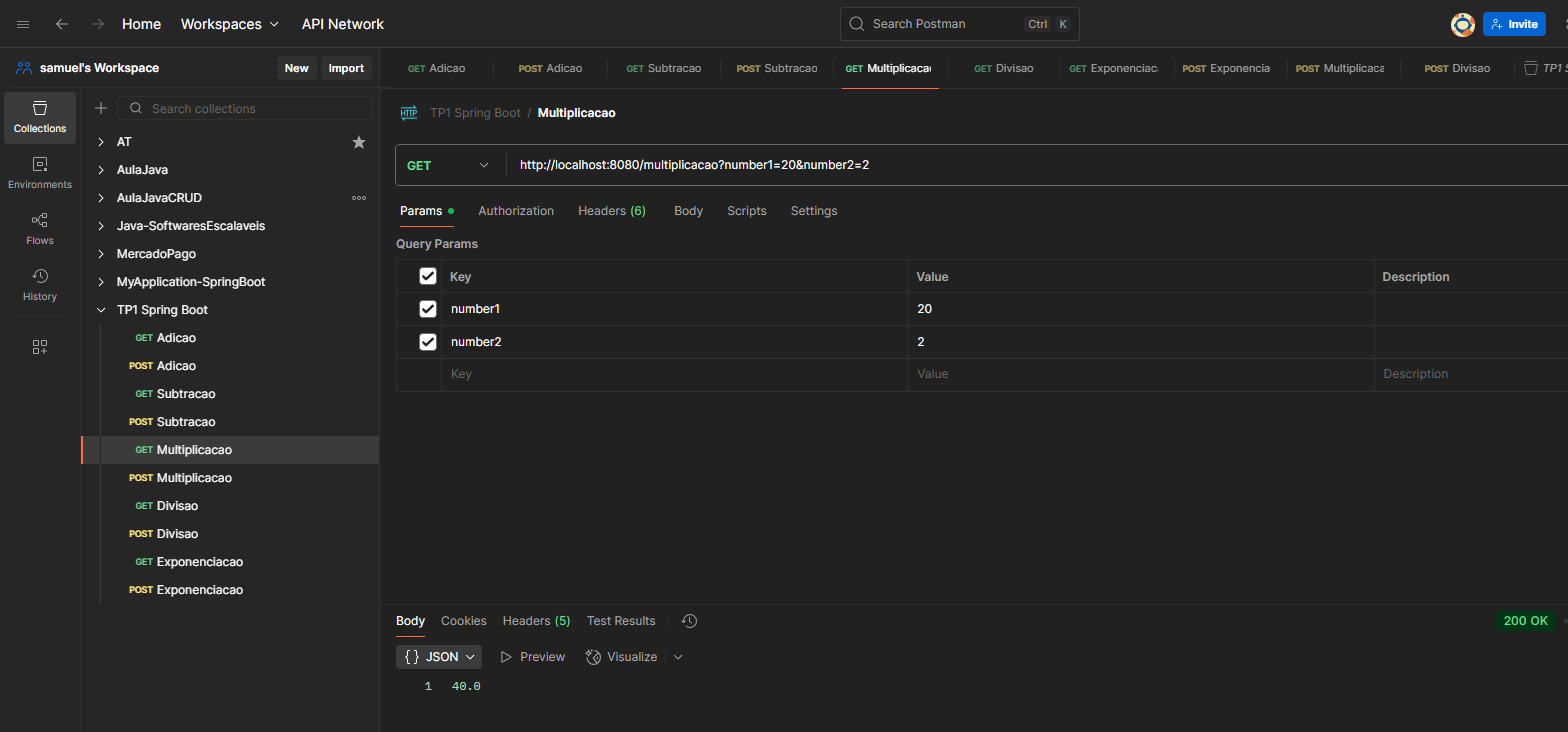


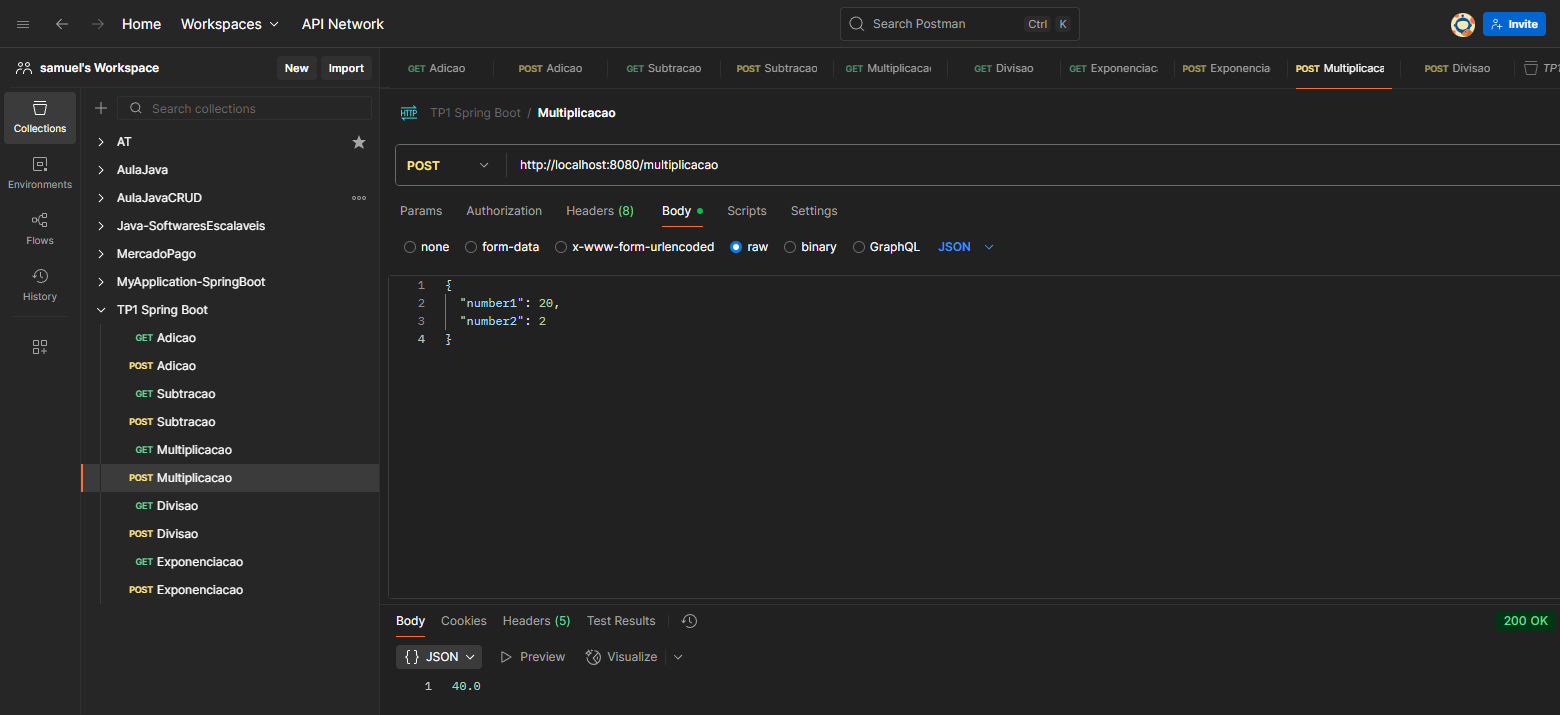
### Subtração – GET / POST



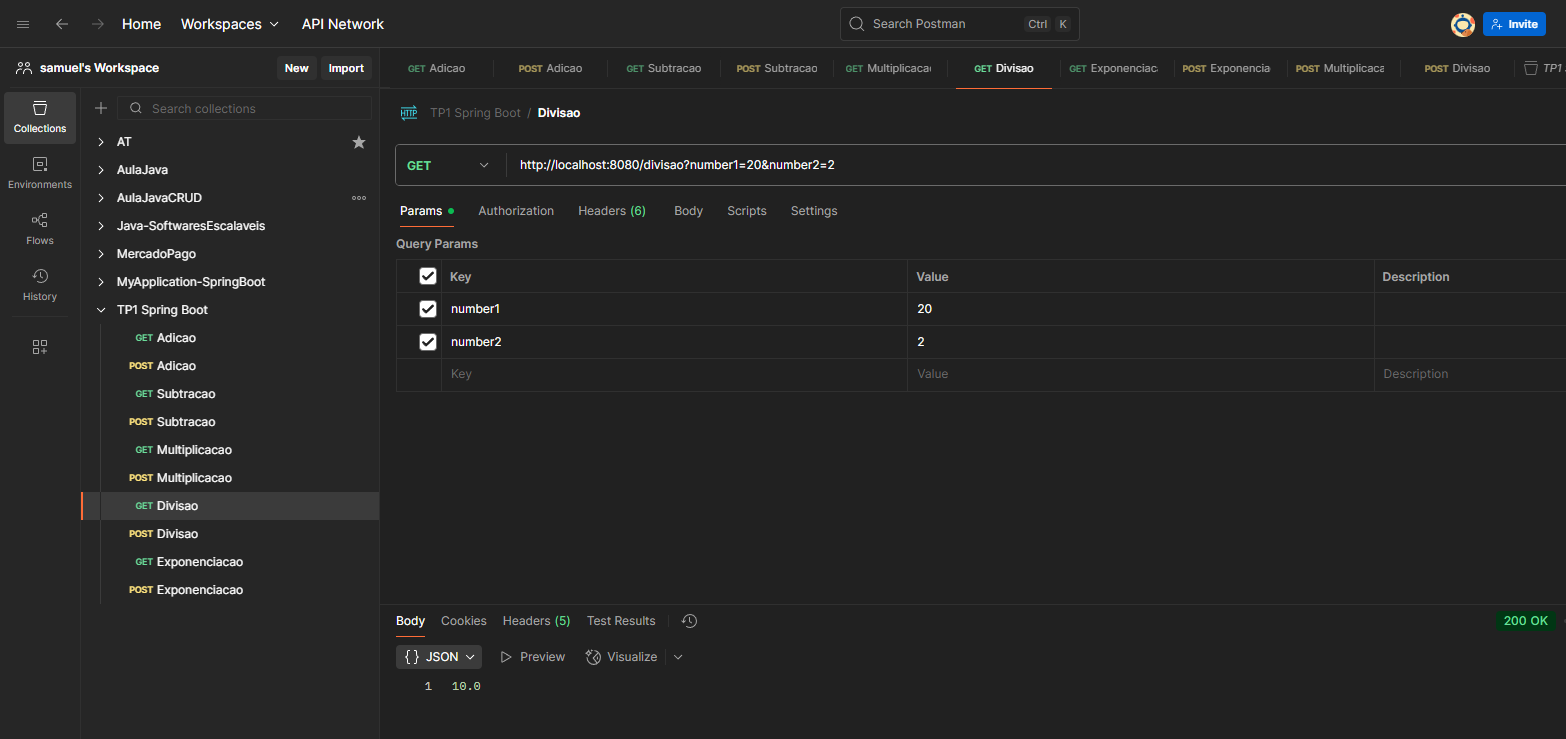


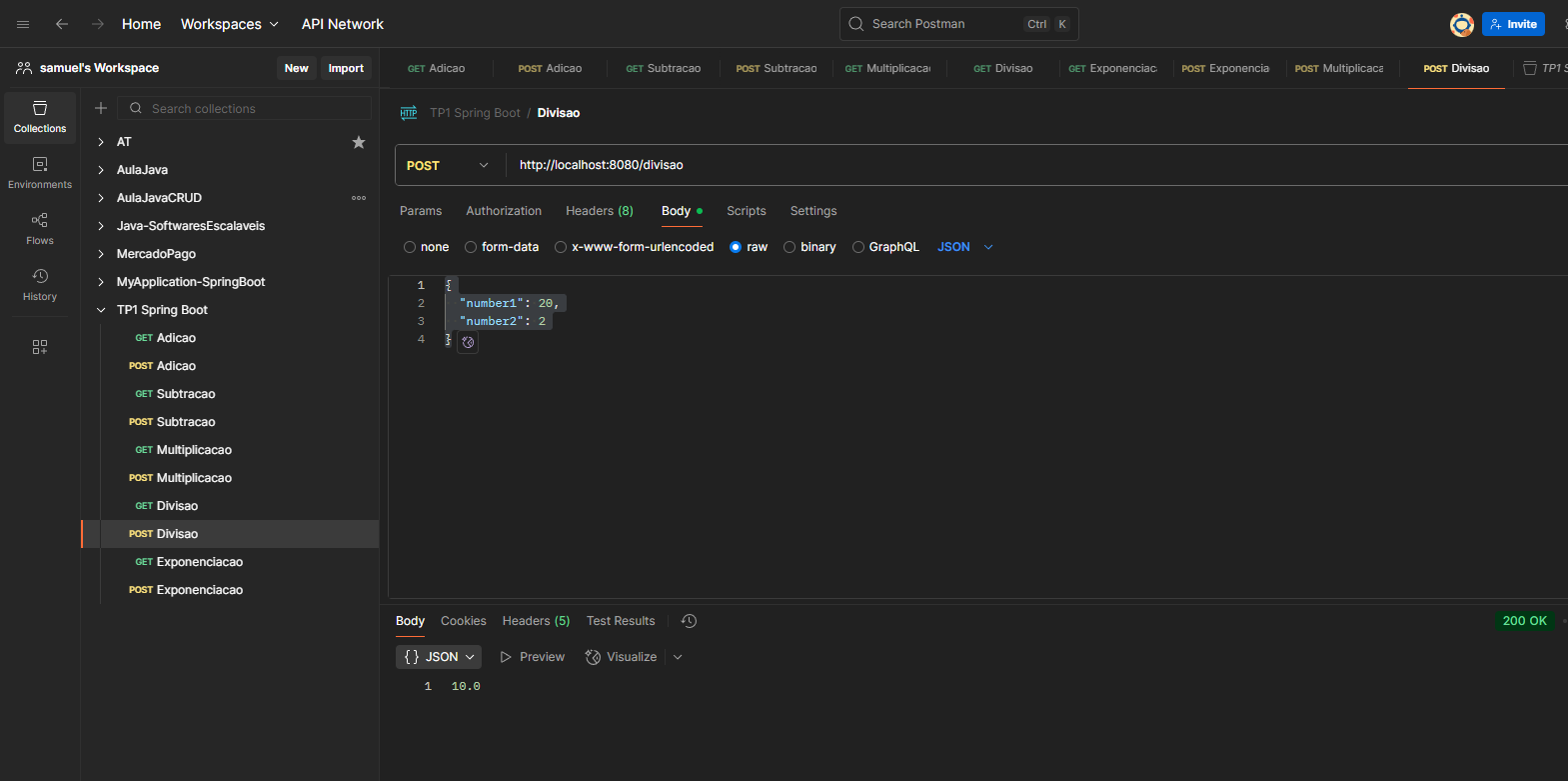
### Multiplicação – GET / POST





### Divisão – GET / POST





### Exponenciação – GET / POST

