



Desenvolvimento de Serviços com Spring Boot

TP1

Luiz Carlos de Souza Ardo vino Ribeiro

Prof:

CPF: 155.647.787-23

Santa Catarina, 04 de agosto de 2025

Questão

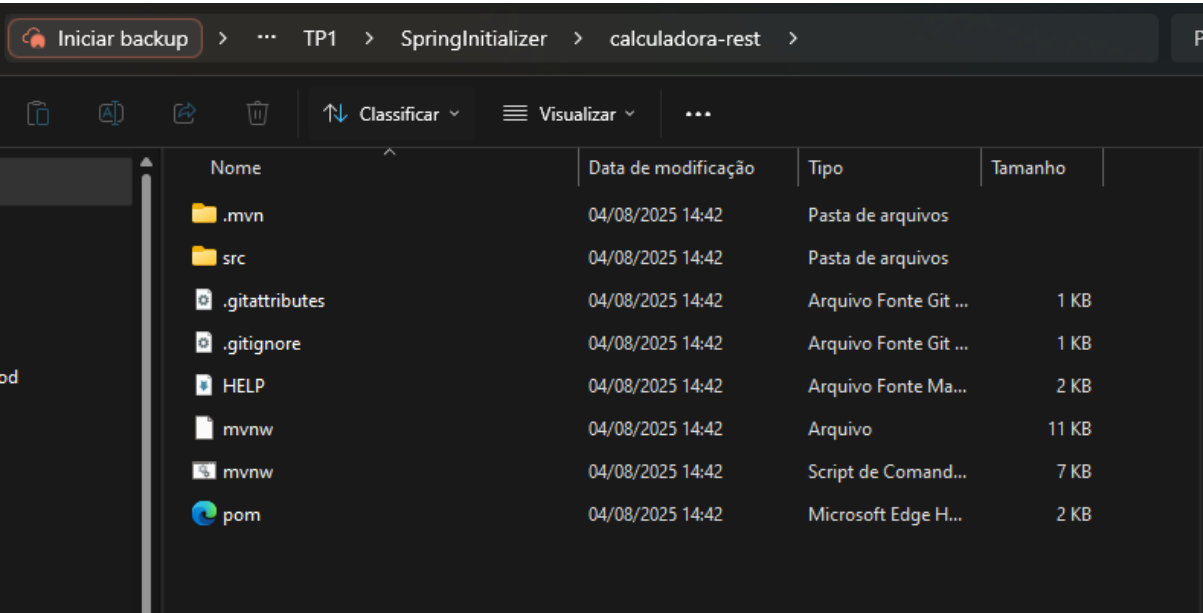
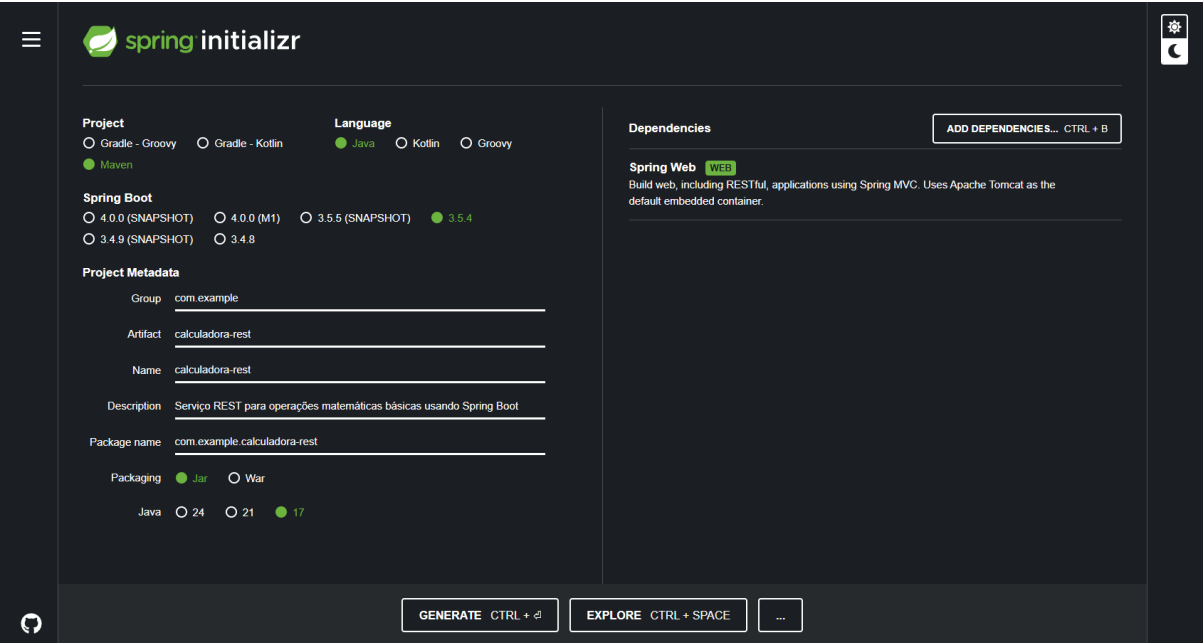
1:

Resposta:

Para a construção do meu projeto, optei por utilizar o Maven como ferramenta de gerenciamento de dependências e build. A escolha se baseia principalmente na ampla adoção do Maven no ecossistema Java e Spring Boot, o que garante grande suporte da comunidade e facilidade para encontrar exemplos e soluções para eventuais problemas. Além disso, o Maven utiliza uma abordagem baseada em convenção, o que simplifica a estruturação de projetos e torna a configuração inicial mais intuitiva para quem está começando. Outra vantagem é sua integração facilitada com as principais IDEs do mercado, como o IntelliJ IDEA e o Eclipse. Embora o Gradle seja reconhecido por sua flexibilidade e performance em projetos mais complexos, o Maven se destaca pela simplicidade, estabilidade e previsibilidade dos builds, tornando-se a escolha mais adequada para projetos acadêmicos e de pequeno a médio porte.

Questão 2:

Resposta:



```
MINGW64:/c:/Users/luiza/Desktop/Desenvolvimento de Servios com Spring Boot/TP1/Te...
$ sdk install springboot

Downloading: springboot 3.5.4

In progress...

##### 100.0%

Installing: springboot 3.5.4
Done installing!

Setting springboot 3.5.4 as default.

luiza@Windows MINGW64 ~/Desktop/Desenvolvimento de Servios com Spring Boot/TP1/
Terminal
$ spring init --dependencies=web --build=maven --java-version=17 --groupId=com.e
xemplo --artifactId=calculadora-rest calculadora-rest
Using service at https://start.spring.io
Project extracted to 'C:\Users\luiza\Desktop\Desenvolvimento de Servios com Spr
ing Boot\TP1\Terminal\calculadora-rest'

luiza@Windows MINGW64 ~/Desktop/Desenvolvimento de Servios com Spring Boot/TP1/
Terminal
$ |

bash.exe[64]:13364 230724[64] 1/1 [+] NUM InpGrp XB2 PRI1 80x25 (3,38) 25V
```

Iniciar backup > ... TP1 > Terminal > calculadora-rest >

	Nome	Data de modificao	Tipo	Tamanho
	.mvn	04/08/2025 15:13	Pasta de arquivos	
	src	04/08/2025 15:13	Pasta de arquivos	
	.gitattributes	04/08/2025 15:13	Arquivo Fonte Git ...	1 KB
	.gitignore	04/08/2025 15:13	Arquivo Fonte Git ...	1 KB
	HELP	04/08/2025 15:13	Arquivo Fonte Ma...	2 KB
	mvnw	04/08/2025 15:13	Arquivo	11 KB
	mvnw	04/08/2025 15:13	Script de Comand...	7 KB
	pom	04/08/2025 15:13	Microsoft Edge H...	2 KB

A principal diferena entre esses dois mtodos est na experincia de uso. O Spring Inicializr via web  ideal para iniciantes ou quando se deseja visualizar todas as opoes

de configuração de forma clara e gráfica, facilitando a seleção das dependências e ajustes iniciais. Já a Spring Boot CLI é mais indicada para desenvolvedores que preferem agilidade e automação, permitindo a criação de projetos diretamente pelo terminal, o que pode ser vantajoso em fluxos de trabalho mais avançados ou scripts de automação. Por isso, cada método é preferível conforme o perfil do desenvolvedor e o contexto do projeto: a interface web prioriza praticidade e visualização, enquanto a CLI proporciona rapidez e integração com ambientes de desenvolvimento automatizados.

Questão

3:

Resposta:

Desde a criação do projeto, adotei práticas eficazes de gerenciamento de dependências utilizando o Maven como ferramenta principal. No momento da geração do projeto pelo Spring Initializr, selecionei apenas as dependências essenciais — no caso, o Spring Web, evitando incluir bibliotecas desnecessárias que poderiam aumentar a complexidade e o tamanho do projeto. Durante o desenvolvimento, mantive o arquivo pom.xml sempre organizado, revisando periodicamente as versões das dependências e removendo aquelas que eventualmente deixaram de ser utilizadas. Também utilizei versões estáveis e oficiais das bibliotecas, reduzindo o risco de incompatibilidades ou vulnerabilidades.

Essa abordagem traz vários benefícios ao ciclo de vida do projeto: mantém o build mais rápido, facilita atualizações e manutenções futuras, reduz conflitos entre versões de bibliotecas e melhora a segurança da aplicação. Além disso, um gerenciamento cuidadoso das dependências contribui para maior previsibilidade do ambiente de desenvolvimento e produção, já que o Maven garante que todos os desenvolvedores usem as mesmas versões das bibliotecas. Dessa forma, a aplicação se mantém enxuta, eficiente e fácil de evoluir.

Questão 4:

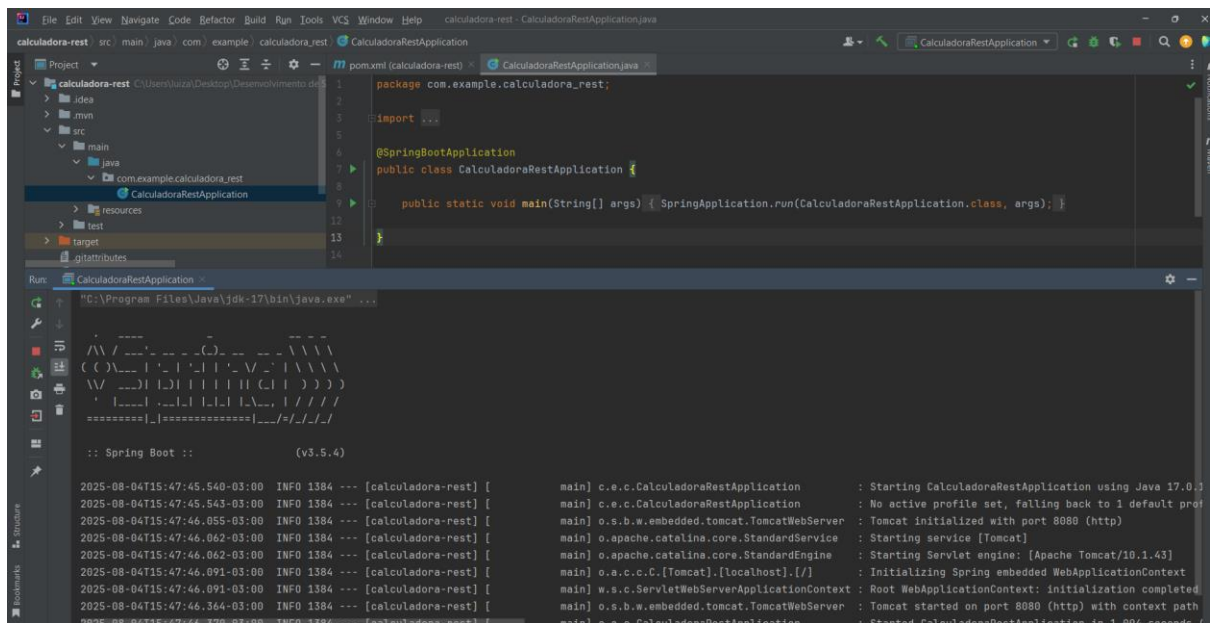
Resposta:

Um dos principais diferenciais do Spring Boot é o recurso de autoconfiguração, que permite reduzir significativamente a quantidade de código repetitivo e configurações manuais necessárias para iniciar e manter um projeto. Ao incluir a dependência spring-boot-starter-web no pom.xml, o próprio Spring Boot realiza automaticamente a configuração de diversos componentes essenciais, como o servidor embutido (Tomcat), o gerenciamento das rotas REST e o tratamento padrão de erros.

No meu projeto, aproveitei essa funcionalidade ao criar a camada de controladores REST simplesmente anotando as classes com `@RestController` e os métodos com `@RequestMapping`, sem necessidade de configurar manualmente arquivos XML, beans ou o próprio servidor de aplicação. Todo o ambiente para expor endpoints HTTP já veio pronto para uso, graças à autoconfiguração. Isso me permitiu focar diretamente na lógica de negócio das operações matemáticas, tornando o desenvolvimento mais ágil, limpo e menos propenso a erros. Essa abordagem também facilita a manutenção do projeto, já que há menos configurações explícitas para revisar e atualizar com o tempo.

Questão 5:

Resposta:

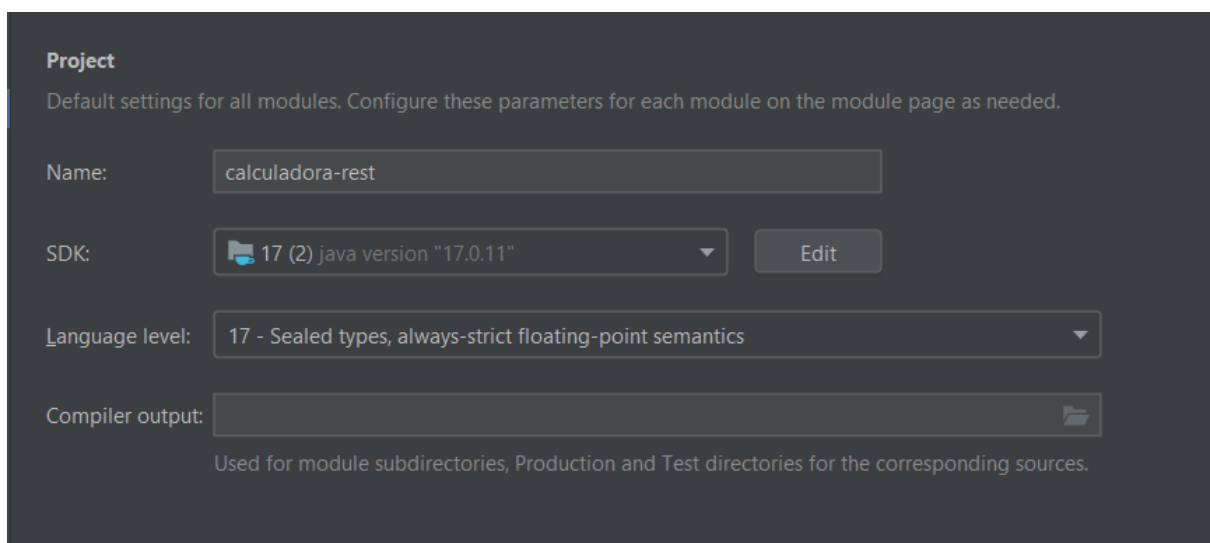


The screenshot shows an IDE with the following components:

- Project Explorer:** Shows the project structure for 'calculadora-rest' with directories like 'src', 'main', 'resources', 'test', 'target', and 'gitattributes'.
- Source Editor:** Displays the code for 'CalculadoraRestApplication.java'. The code is as follows:

```
1 package com.example.calculadora_rest;
2
3 import ...
4
5 @SpringBootApplication
6 public class CalculadoraRestApplication {
7
8     public static void main(String[] args) { SpringApplication.run(CalculadoraRestApplication.class, args); }
9
10 }
```
- Run Console:** Shows the output of the application. It starts with a ASCII art logo and the text ':: Spring Boot :: (v3.5.4)'. The log shows the application starting successfully on port 8080.

```
2025-08-04T15:47:45.540-03:00 INFO 1384 --- [calculadora-rest] [main] c.e.c.CalculadoraRestApplication : Starting CalculadoraRestApplication using Java 17.0.11
2025-08-04T15:47:45.543-03:00 INFO 1384 --- [calculadora-rest] [main] c.e.c.CalculadoraRestApplication : No active profile set, falling back to 1 default profile
2025-08-04T15:47:46.055-03:00 INFO 1384 --- [calculadora-rest] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-08-04T15:47:46.062-03:00 INFO 1384 --- [calculadora-rest] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-08-04T15:47:46.062-03:00 INFO 1384 --- [calculadora-rest] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.43]
2025-08-04T15:47:46.091-03:00 INFO 1384 --- [calculadora-rest] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-08-04T15:47:46.091-03:00 INFO 1384 --- [calculadora-rest] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2025-08-04T15:47:46.364-03:00 INFO 1384 --- [calculadora-rest] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2025-08-04T15:47:46.370-03:00 INFO 1384 --- [calculadora-rest] [main] c.e.c.CalculadoraRestApplication : Started CalculadoraRestApplication in 1.894 seconds
```



The screenshot shows the 'Project' settings dialog in the IDE. It contains the following fields and options:

- Project:** Default settings for all modules. Configure these parameters for each module on the module page as needed.
- Name:** calculadora-rest
- SDK:** 17 (2) java version "17.0.11" (with an 'Edit' button)
- Language level:** 17 - Sealed types, always-strict floating-point semantics
- Compiler output:** (empty field with a folder icon)
- Description:** Used for module subdirectories, Production and Test directories for the corresponding sources.

Questão 6:

Resposta:

```
1 package com.example.calculadora_rest.controller;
2
3 import org.springframework.web.bind.annotation.*;
4
5 no usages
6 @RestController
7 @RequestMapping("/api")
8 public class CalculadoraController {
9
10     no usages
11     @RequestMapping(value = "/adicao", method = {RequestMethod.GET, RequestMethod.POST})
12     public double adicao(@RequestParam double a, @RequestParam double b) {
13         return a + b;
14     }
15
16     no usages
17     @RequestMapping(value = "/subtracao", method = {RequestMethod.GET, RequestMethod.POST})
18     public double subtracao(@RequestParam double a, @RequestParam double b) {
19         return a - b;
20     }
21
22     no usages
23     @RequestMapping(value = "/multiplicacao", method = {RequestMethod.GET, RequestMethod.POST})
24     public double multiplicacao(@RequestParam double a, @RequestParam double b) {
25         return a * b;
26     }
27
28     no usages
29     @RequestMapping(value = "/divisao", method = {RequestMethod.GET, RequestMethod.POST})
30     public double divisao(@RequestParam double a, @RequestParam double b) {
31         if (b == 0) throw new IllegalArgumentException("0 divisor não pode ser zero!");
32     }
33 }
```