Curso de Jogos Digitais

Disciplina de Tecnologias Web

Aula 16

Implementação do Back-end

Professor: André Flores dos Santos

Santa Maria – RS 2025

Mas, o que é Spring Boot?

O **Spring Boot é um framework Java open source** que tem como objetivo facilitar o processo de criação e configuração em aplicações Java. Consequentemente, ele traz mais agilidade para o processo de desenvolvimento, uma vez que devs conseguem **reduzir o tempo gasto com as configurações iniciais**.

Com o Spring Boot conseguimos abstrair e facilitar a configuração de, por exemplo:

- Servidores;
- Gerenciamento de dependências;
- Configurações de bibliotecas;
- Métricas & health checks (testes);
- Entre outros!

Como o Spring Boot funciona?

Para realizar todo esse processo o Spring Boot utiliza um conceito chamado **convenção sobre configuração**.

Mas o que isso significa? Significa que é uma ferramenta que decide para você a melhor forma de se fazer algo. É o que chamamos de ferramenta opinativa, ela toma as decisões no nosso lugar baseado em convenções, aplicando configurações padrões e facilitando o trabalho.

No entanto ela **não é inflexível** e ainda permite uma configuração diferente da *default* caso o usuário assim deseje.

Por exemplo, você pode alterar para que ele utilize o Jetty como servidor ao invés do Tomcat que é a configuração padrão.

Uma das maiores vantagens que o Spring Boot trouxe ao desenvolvimento é que toda essa configuração não necessita mais ser realizada pelos temidos XMLs, embora ele ainda suporte esse tipo de configuração. A maior parte da **configuração pode ser feita de forma programática** via anotações.

O Spring Boot é composto por vários módulos que ajudam nesse processo. Alguns deles são:

Spring Boot

É o módulo principal que ajuda na configuração e integração dos outros módulos.

Spring Boot Starters

Os starters são dependências que agrupam outras dependências com um propósito em comum. Dessa forma, somente uma configuração é realizada no seu gerenciador de dependências.
Por exemplo, o spring-boot-starter-amqp, é um starter que permite a construção de soluções de mensageria baseadas em AMQP e RabbitMQ.
Ao realizar a configuração no meu gerenciador de dependência se define somente o starter:

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

O Spring Boot é composto por vários módulos que ajudam nesse processo. Alguns deles são:

•**Spring Boot Starter Web**: Auxilia na construção de aplicações web trazendo já disponíveis para uso Spring MVC, Rest e o Tomcat como servidor.

•**Spring Boot Starter Test**: Contém a maioria das dependências necessárias para realizar testes da sua aplicação: Junit, AssertJ, Hamcrest, Mockito, entre outros

•**Spring Boot Starter Data JPA**: Facilita a construção da nossa camada de persistência, ajudando na abstração do nosso banco de dados provendo uma série de facilidades para criação de repositories, escrita de queries, entre outros.
Como podem ver, reduzem o número de dependências adicionadas, deixando meu arquivo muito mais limpo.

Spring Boot Autoconfigure

Como dito anteriormente o **Spring Boot trabalha de forma opinativa, tomando decisões para você**.

Mas baseado em que? Essas decisões padrões são baseadas através do conteúdo do seu *classpath*.

O *Autoconfigure* é responsável por ler este conteúdo e realizar as configurações necessárias para que a aplicação funcione. É ele quem gerencia todo o processo de configuração da aplicação.

Spring Boot Actuator

O Spring Boot Actuator é uma ferramenta que permite monitorar e gerenciar as aplicações implantadas. Dentre os recursos disponibilizados temos:

•**Métricas**: Obtém e disponibiliza diversos dados da nossa aplicação, como por exemplo, espaço em disco, memória, tempo de resposta etc.

•**Logging**: Facilita o acesso ao arquivo de log da aplicação por meio de um *endpoint* específico.

•**Heath Checks**: (verificações de saúde) são endpoints que permitem monitorar o estado de uma aplicação em tempo real. Elas verificam se vários aspectos da aplicação estão funcionando corretamente, como conectividade com o banco de dados, disponibilidade de serviços externos, uso de memória, e mais. Esses checks são úteis para garantir que a aplicação está em um estado saudável e pronta para responder a solicitações.

•**Informações da Aplicação**: Permite a disponibilização de informações da aplicação. Por exemplo, versão, informações do git etc.

Spring Boot Test

O Spring Boot Test contém funcionalidades úteis e anotações que facilitam e ajudam a testar sua aplicação.

Spring Boot Devtools

Spring Boot Devtools é um conjunto de funcionalidades que ajuda o trabalho de qualquer dev. Como, por exemplo, restart automático da aplicação quando ocorre alguma mudança no código.

Spring Tool Suite

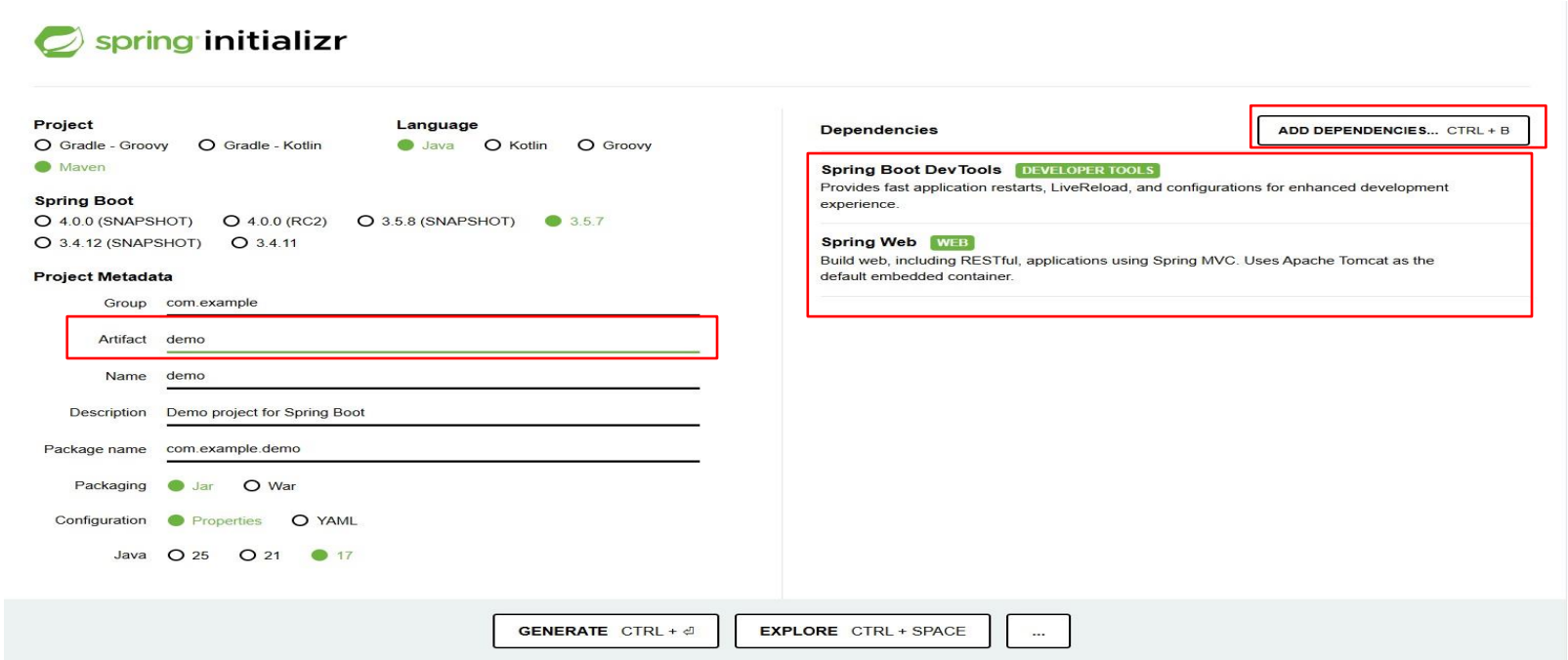
O Spring nos fornece uma IDE totalmente customizada para o desenvolvimento de aplicações do ecossistema spring: o **Spring Tool Suite (STS)**.

O STS é uma IDE baseada em Eclipse que já vem com algumas funcionalidades facilitadoras para projetos Spring.

download no site oficial.
Além da IDE baseada em Eclipse, o STS já está disponível como plugin para VSCode.

Spring Initializr

E para facilitar a criação de aplicações utilizando outras IDEs a Spring disponibilizou o **Spring Initializr.**
Permite a criação de projetos Sprint Boot de forma facilitada.



Através dele definimos nome do projeto, pacotes, dependências (starters do spring e outros projetos), linguagem (Java, Groovy ou Kotlin). Uma vez definido é só clicar no botão *Generate* e o projeto será criado, gerando um zip pronto para ser importado na IDE de sua preferência.
Escolha essas opções e clique em 'Generate' para baixar o pacote do projeto pronto.

Site do SpringBoot:

https://spring.io/projects

https://spring.io/

https://spring.io/quickstart

Exercício treino:

Hello world springboot, vamos praticar o nosso primeiro exemplo!

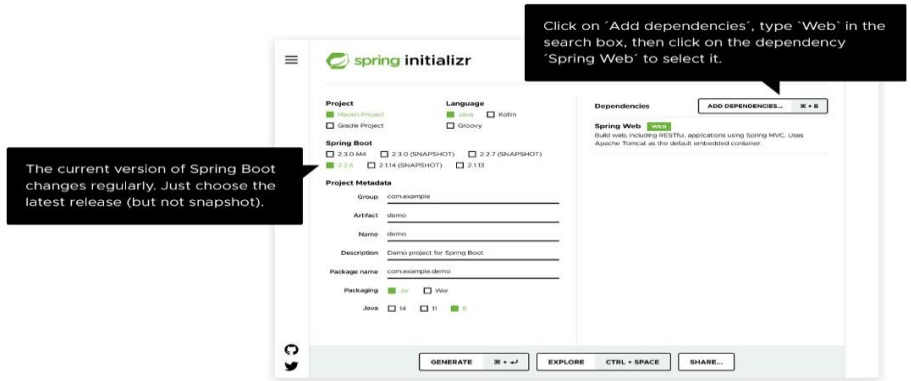
Gere o projeto inicial em: https://start.spring.io/

Acesse as informações em: https://spring.io/quickstart e preencha o resto do código que precisa para imprimir uma mensagem Hello World! (próximo slide)

Atenção! Quando for rodar programas com SpringBoot que utiliza o Apache Tomcat como servidorweb desabilitar o servidor apache do pacote xampp se estiver ativado para não dar conflitos na porta 8080.

Etapa 1: iniciar um novo projeto Spring Boot

Usarstart.spring.iopara criar um projeto "web". Na caixa de diálogo "Dependencies", procure e adicione a dependência "web" conforme mostrado na captura de tela. Clique no botão "Gerar", baixe o zip e descompacte-o em uma pasta no seu computador.



Etapa 2: adicione seu código

Abra o projeto em seu IDE e localize o `DemoApplication.java` arquivo na `src/main/java/com/example/demo` pasta.
Agora altere o conteúdo do arquivo adicionando o método extra e as anotações mostradas no código abaixo.
Você pode copiar e colar o código ou apenas digitá-lo.

CÓPIA DE

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {

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

   @GetMapping("/hello")
   public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
      return String.format("Hello %s!", name);
   }
}
```

O hello() método que adicionamos foi projetado para receber um parâmetro String chamado name e, em seguida, combinar esse parâmetro com a palavra "Hello“ no código. Isso significa que, se você definir seu nome “Amy” na solicitação, a resposta será “Hello Amy”. A @RestController anotação informa ao Spring que este código descreve um endpoint que deve ser disponibilizado na web. O @GetMapping(“/hello”) diz ao Spring para usar nosso hello() método para responder a solicitações que são enviadas para o http://localhost:8080/hello endereço. Finalmente, @RequestParam está dizendo ao Spring para esperar um name valor na solicitação, mas se não estiver lá, ele usará a palavra “World” por padrão.

Passo 3: Experimente

Vamos construir e executar o programa. Abra uma linha de comando (ou terminal) e navegue até a pasta onde você tem os arquivos do projeto. Podemos construir e executar o aplicativo emitindo o seguinte comando:

Mac OS/Linux:

```
./mvnw spring-boot:run
```

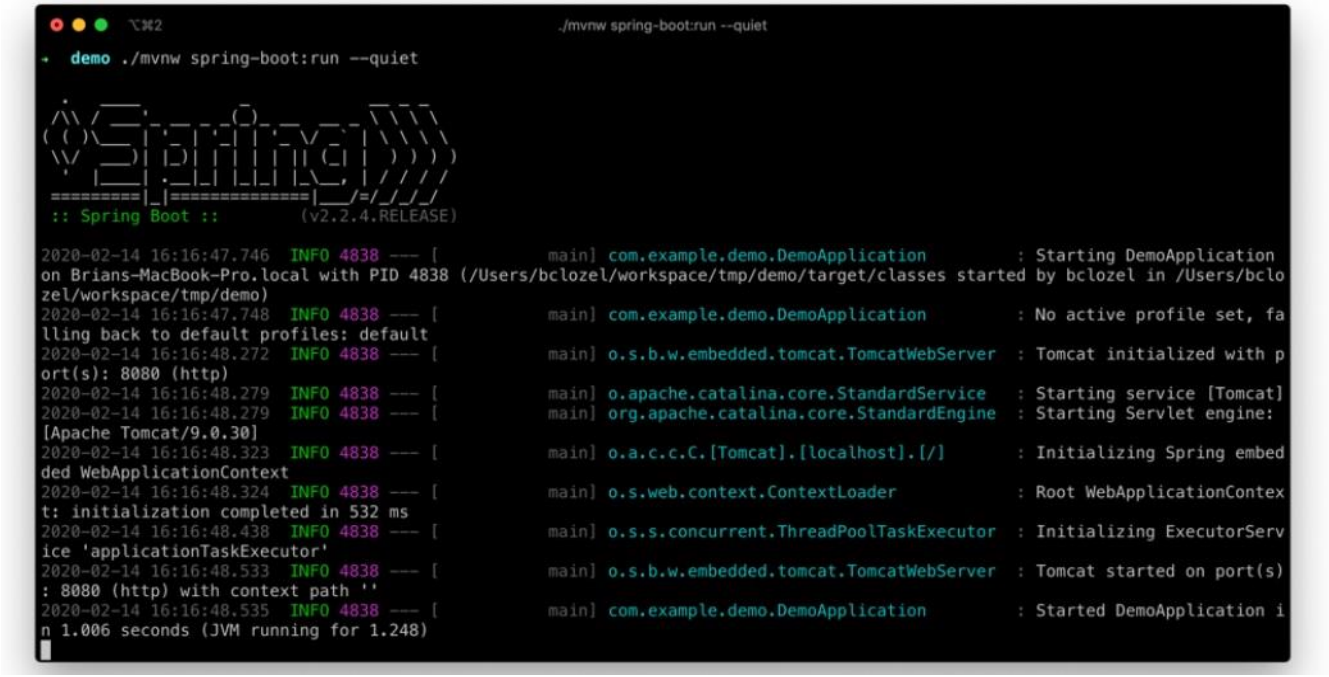
Janelas:

```
mvnw spring-boot:run
```

Obs: Podemos executar o aplicativo direto no Eclipse dando um run (botão de executar), a mesma tela do próximo slide irá aparecer no console mostrando se deu tudo certo. Porém sempre devemos rodar em um local ou em outro, pois se rodar os dois ao mesmo tempo irá dar problemas nas requisições.

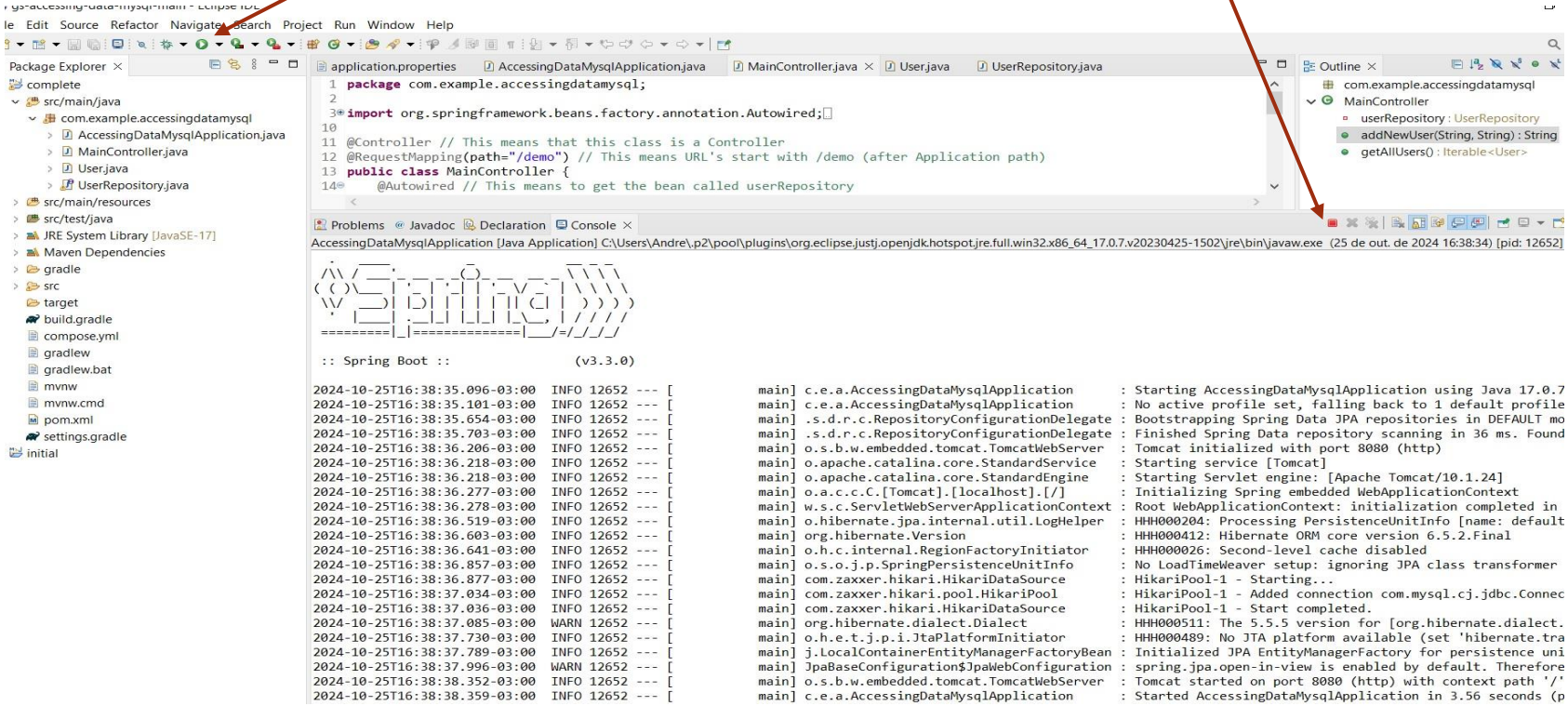
É mais fácil executar pela interface da IDE escolhida

Você deve ver uma saída muito semelhante a esta:



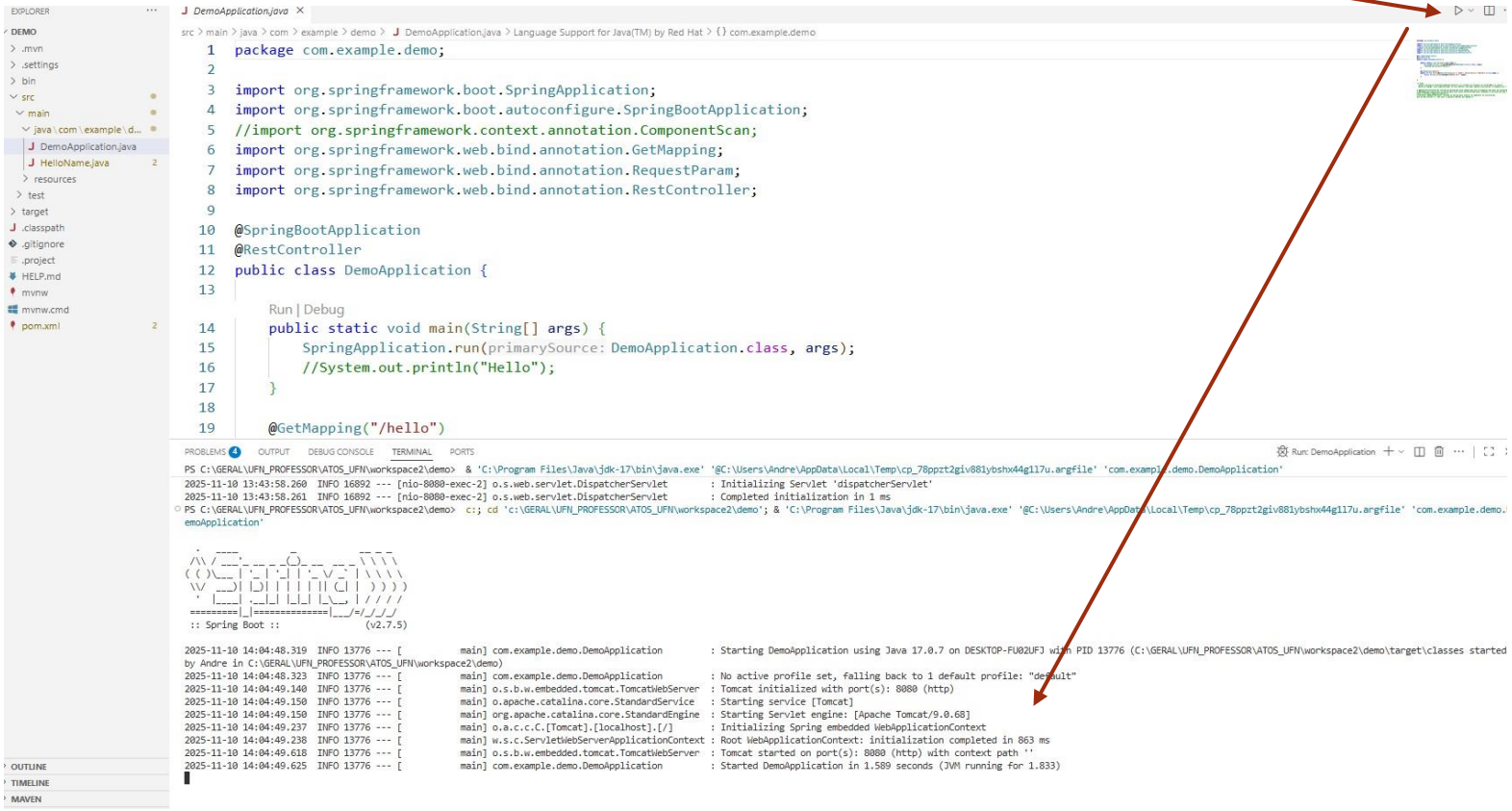
As últimas linhas aqui nos dizem que a spring começou. O servidor Apache Tomcat incorporado do Spring Boot está agindo como um servidor web e está escutando solicitações na localhost porta 8080.

Podemos executar diretamente utilizando o Eclipse, Intellij ou VScode.
Clicar em cima do Projeto: Run as Java Application. Para parar o projeto clicar em ‘stop’.



Se não houver nenhum erro é acessar os ‘end points’ de alguma forma

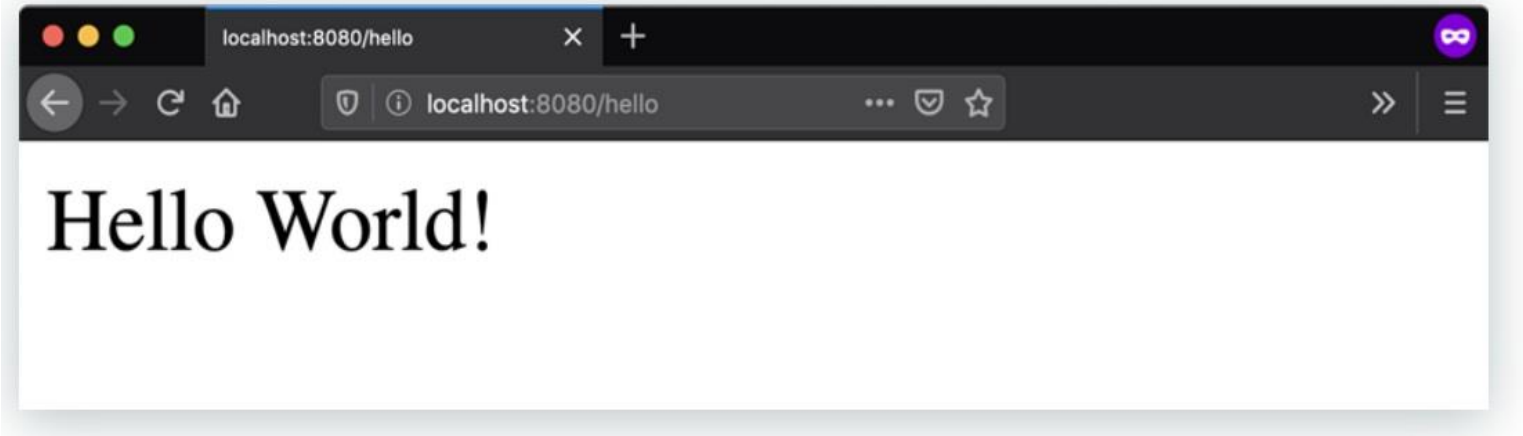
No Vscode instalando a biblioteca SpringBoot Extension Pack. Após executar



Se não houver nenhum erro é só acessar os ‘end points’ de alguma forma

Após executar o projeto na IDE que você escolheu, podemos testar.

Abra seu navegador e na barra de endereços na parte superior, digite http://localhost:8080/hello. Você deve obter uma boa resposta amigável como esta:

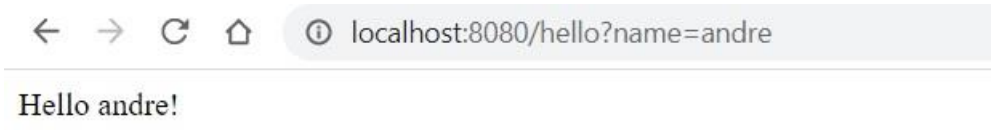


Agora vamos mandar um nome para trocar pela palavra padrão ‘world’

```
import org.springframework.boot.SpringApplication;

@SpringBootApplication
@RestController
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
        //System.out.println("Hello");
    }

    @GetMapping("/hello")
    public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
        return String.format("Hello %s!", name);
    }

}
```

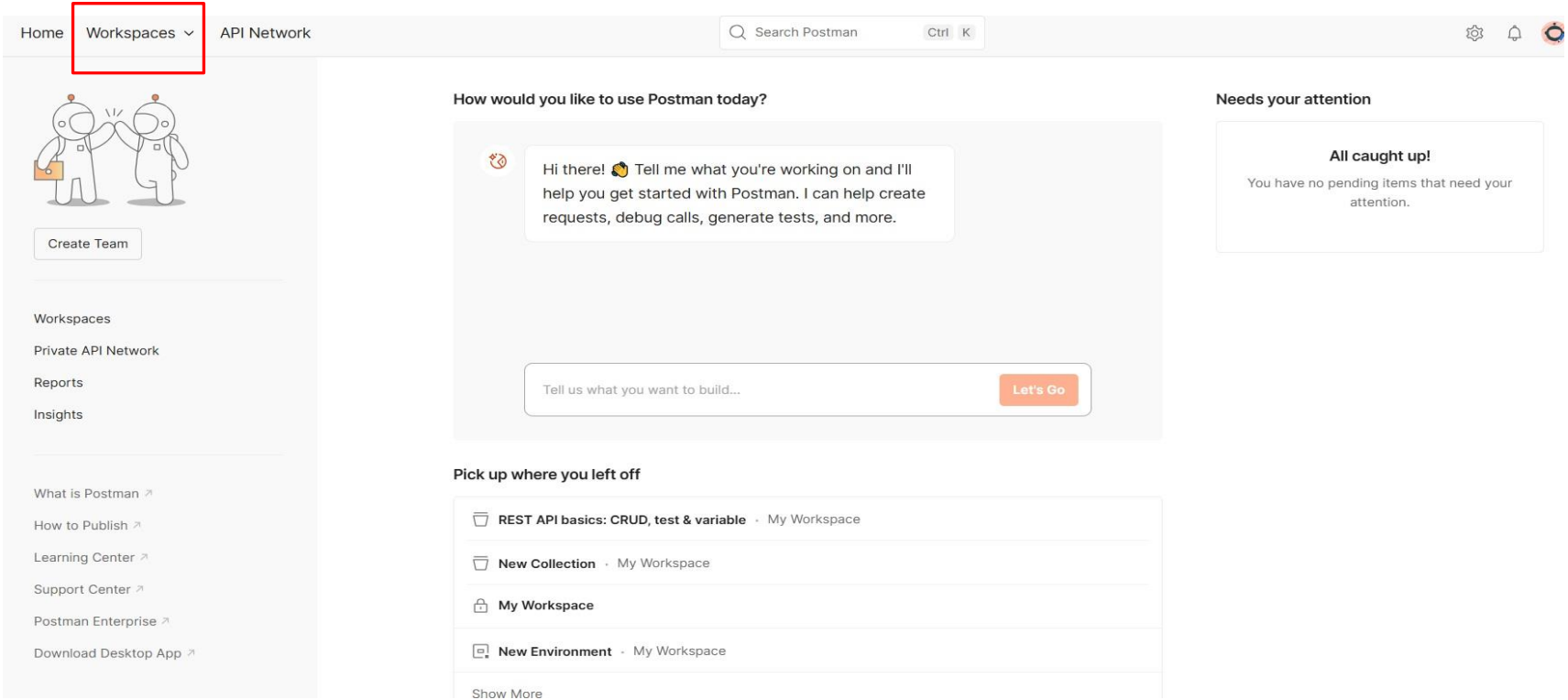
Digitar http://localhost:8080/hello?name=Andre

A resposta foi:

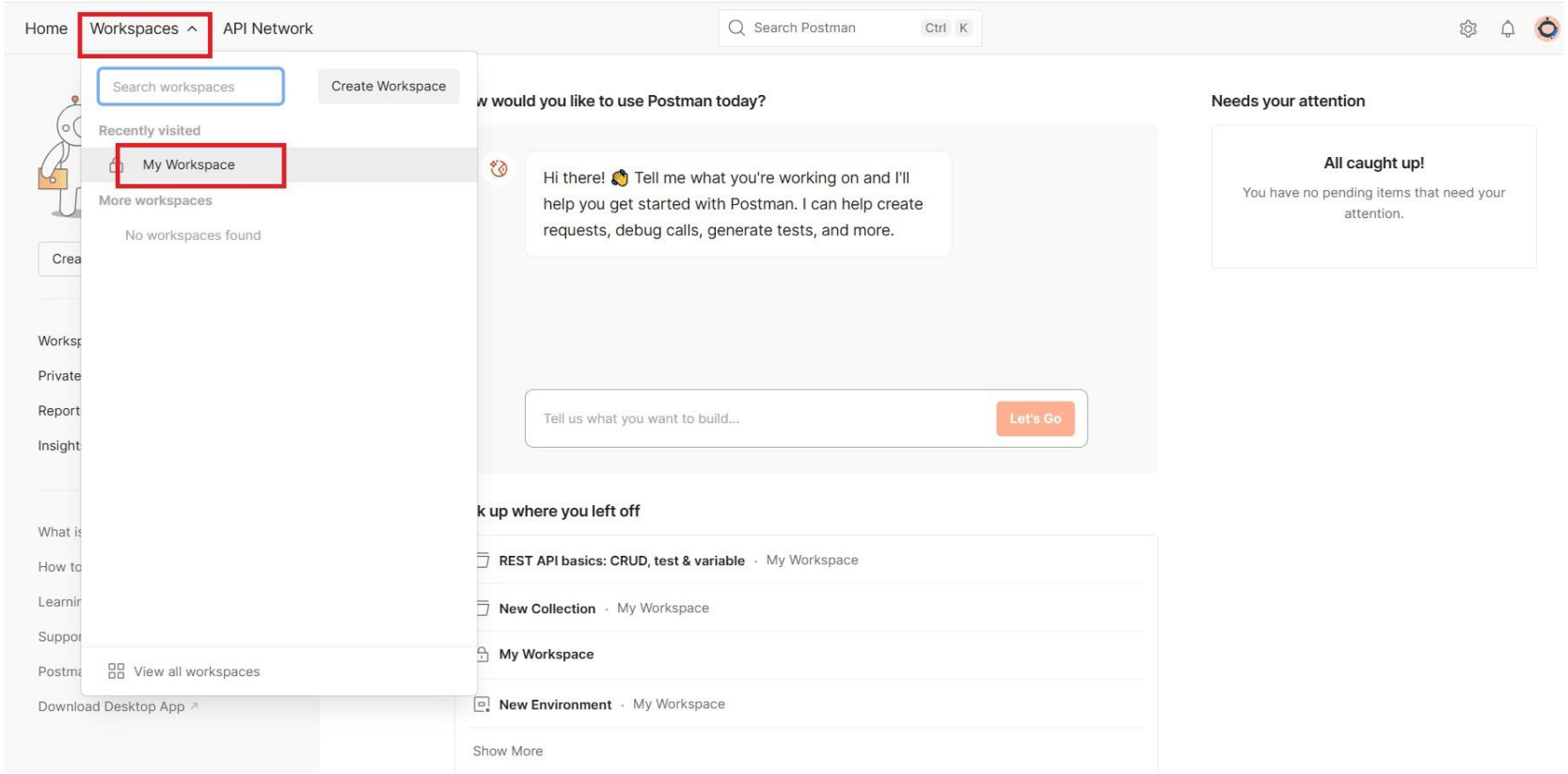


Digitar http://localhost:8080/hello?name=Andre

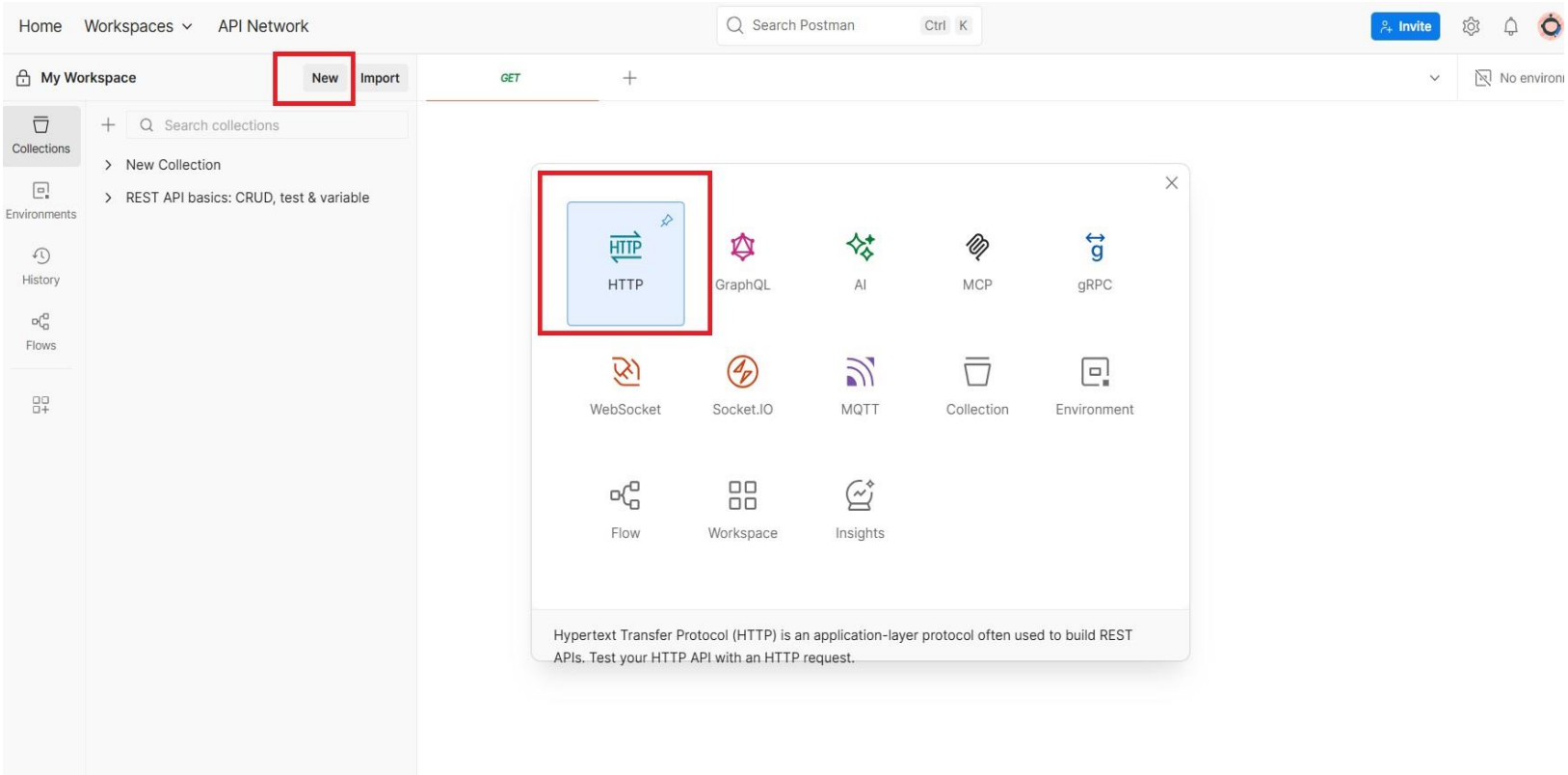
Vamos usar a ferramenta Postman: https://web.postman.co/. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints.



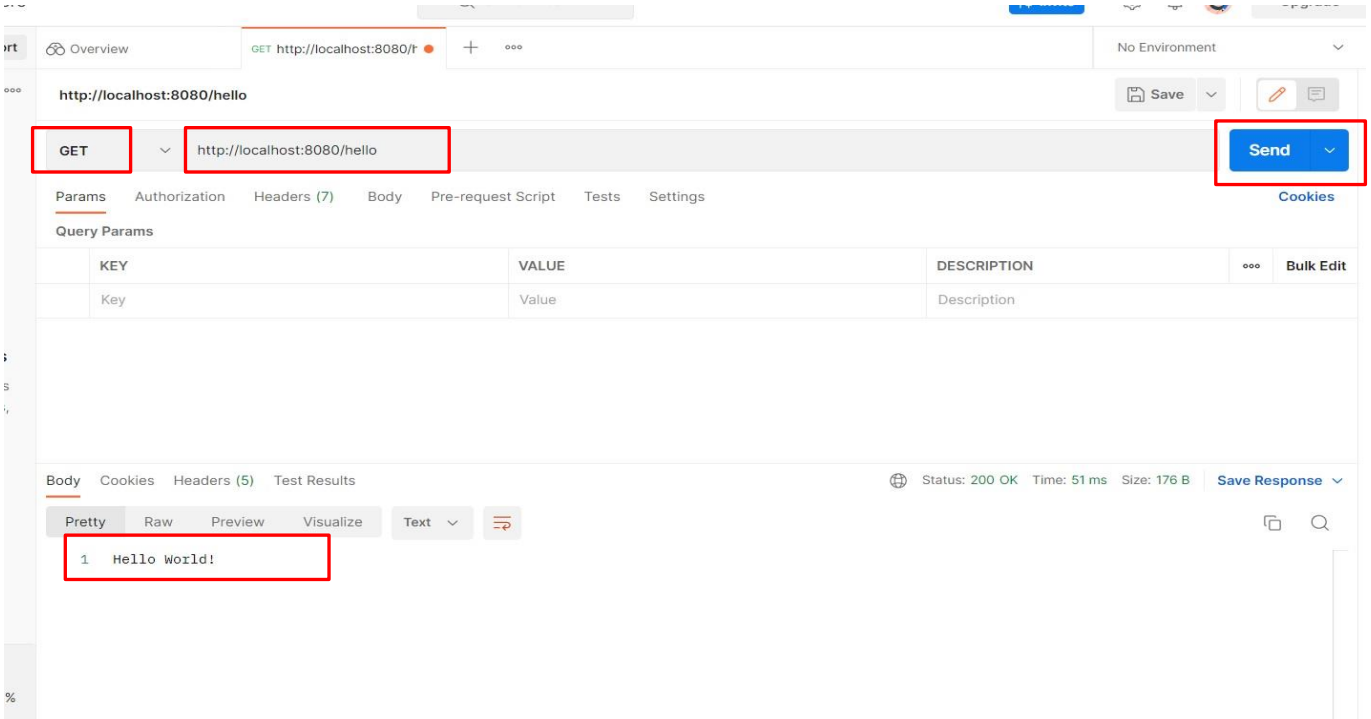
Vamos usar a ferramenta Postman: https://web.postman.co/. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints.



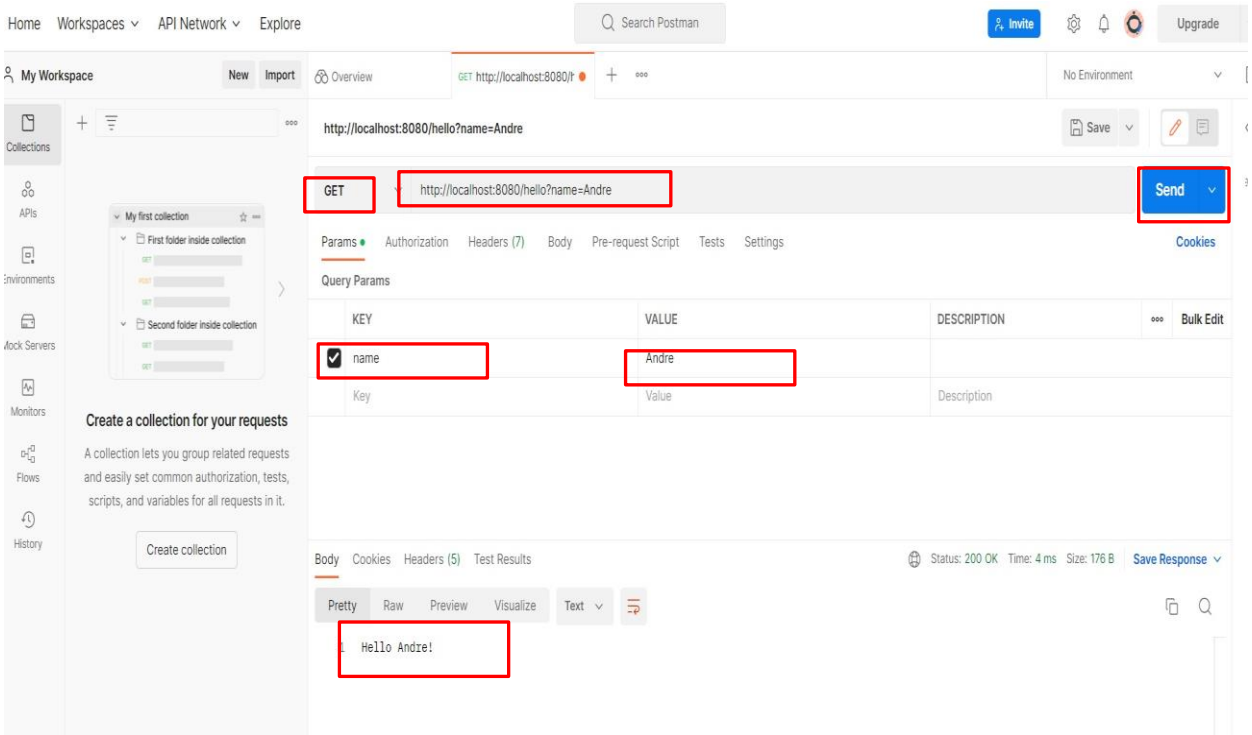
Vamos usar a ferramenta Postman: https://web.postman.co/. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints.



Vamos usar a ferramenta Postman: https://web.postman.co/. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints. Se for utilizar via web e solicitar para baixar o Postman Agent, instale ele para poder usar. Ou faça download do Postman Desktop que será executado direto na máquina, sem problemas.



Testando o envio do nome ‘Andre’ ou o desejado junto.



Métodos HTTP no Spring Boot

1. O que são métodos HTTP

•Quando um cliente (navegador, Postman, app) fala com o servidor, ele **usa um "método"** para dizer **o que quer fazer**.

•Os mais usados são:

- **GET** → pedir informações (ler dados)
- **POST** → enviar informações (criar ou enviar dados)
- **PUT** → atualizar algo
- **DELETE** → excluir algo

2. GET — Leitura de dados

```
@GetMapping("/hello")
public String hello(@RequestParam String name) {
    return "Hello, " + name;
}
```

O método @GetMapping:

Usa **GET** → leitura.

Espera os dados **na URL**, por exemplo:

2. GET — Leitura de dados

```
@GetMapping("/hello")
public String hello(@RequestParam String name) {
    return "Hello, " + name;
}
```

O método @GetMapping:

Usa **GET** → leitura.

Espera os dados **na URL**, por exemplo:

http://localhost:8080/hello?name=Andre

O navegador e o Postman enviam os parâmetros como query string.

⚠ Os dados aparecem na URL, por isso não é seguro para senhas ou informações sensíveis.

3. POST — Envio de dados

```
@PostMapping("/hello")
public String hello(@RequestBody Map<String, String> body) {
    return "Hello, " + body.get("name");
}
```

O método `@PostMapping`:

- Usa **POST** → envio.
- Os dados vão **no corpo (body)** da requisição.
- No Postman: **Body** → **raw** → **JSON**
- `{ "name": "Andre" }`
- Mais seguro (dados não aparecem na URL).
- Usado para **criar ou enviar** dados ao servidor.

Métodos HTTP no Spring Boot

Ação	Método	Envio	Segurança	Uso típico
Ler dados	GET	Parâmetros na URL (?name=...)	Aparece na URL	Consultas
Enviar dados	POST	Corpo (JSON, form)	Mais seguro	Envio/criação

5. No Postman

- Se o controller usa `@GetMapping` → selecione **GET** e use **Params**.
- Se usa `@PostMapping` → selecione **POST** e coloque o **JSON no Body**.
- O método HTTP precisa **bater com a anotação** do código.

Resumo final

- `@GetMapping` → consulta, dados simples, URL visível.
- `@PostMapping` → envio, dados estruturados, corpo da requisição.
- É o **Spring Boot** (anotação) que define **qual tipo de requisição** seu endpoint espera.

Tarefa para casa, testar todo esse processo novamente!!

Agora vamos para a parte II.

Referências

HOGAN, Brian P. HTML 5 e CSS3: Desenvolva hoje com o padrão de amanhã . Rio de Janeiro (RJ): Ciência Moderna Ltda., 2012 282 p. ISBN 978-85-399-0260-6

Fábio Flatschart. HTML 5 - Embarque Imediato, 2011. (Biblioteca Digital)

Deitel, Paul J.; Deitel, Harvey M.. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores, 2008. (Biblioteca Digital)

SILVA, Maurício Samy. HTML 5: a linguagem de marcação que revolucionou a web. 2. ed. São Paulo, SP: Novatec, 2014. 335 p. ISBN 978-85-7522-403-8.

Denilson Bonatti. Desenvolvimento de Jogos em HTML5, 2014. (Biblioteca Digital)

W3SCHOOL. The world's largest web development site. Acessado em 2018. Disponível em: http://www.w3schools.com/.

W3C. World Wide Web Consortium. Acessado em 2018. Disponível em: http://www.w3.org

MORRISON, Michael. Use a cabeça JavaScript. Rio de Janeiro (RJ): Alta Books, 2008. 606 p.

https://br.freepik.com/fotos-vetores-gratis/desenvolvimento-web

Material do Professor Fabrício Tonetto Londero, 2023.

Obrigado pela atenção!!

Email: **andre.flores@ufn.edu.br**

Santa Maria – RS
2025