

Java Advanced

Prof. Dr. Marcel Stefan Wagner

Aula 14 – REST, *Spring Boot* e *Postman*

FIAP

Tópicos Abordados

- 1 Introdução
- 2 REST e RESTful
- 3 Conceitos REST
- 4 SOAP x REST
- 5 REST com NetBeans
- 6 Programação
- 7 *Framework Spring*
- 8 *Postman*
- 9 Temas para a Próxima Aula
- 10 Referências Bibliográficas

Introdução ao REST

REST

- REST (*Representational State Transfer*) é uma arquitetura de *software* que impõe condições sobre como uma API (*Application Program Interface*) deve funcionar.
- A REST foi criada inicialmente como uma diretriz para gerenciar a comunicação em uma rede complexa como a Internet. Você pode usar a arquitetura baseada em REST para possibilitar uma comunicação confiável e de alta performance em escala.
- Você pode implementá-la e modificá-la facilmente, trazendo visibilidade e portabilidade entre plataformas para qualquer sistema de API.

Introdução

REST

- Os desenvolvedores de API podem projetar APIs usando várias arquiteturas diferentes.
- As APIs que seguem o estilo de arquitetura REST são chamadas de APIs REST.
- Os serviços da *Web* que implementam a arquitetura REST são chamados de serviços da *Web* RESTful.
- O termo API RESTful geralmente se refere a APIs da *Web* RESTful. No entanto, você pode usar os termos API REST e API RESTful de forma intercambiável.

RESTful

- A API RESTful é uma interface que dois sistemas de computador usam para trocar informações de forma segura pela Internet.
- A maioria das aplicações de negócios precisa se comunicar com outras aplicações internas e de terceiros para executar várias tarefas. Por exemplo, para gerar contracheques mensais, seu sistema interno de contas precisa compartilhar dados com o sistema bancário de seu cliente a fim de automatizar o faturamento e se comunicar com uma aplicação interna de planilha de horas.
- As APIs RESTful suportam essa troca de informações porque seguem padrões de comunicação de *software* seguros, confiáveis e eficientes.

REST

- Serviços REST possuem uma gramática simples, facilmente compreensível.
- Facilmente escalável.
 - Basta adicionar um novo *Web service*.
- Conceitos de REST:
 - *Resources*;
 - *Representations*;
 - *Operations*;
 - *Hypertext*;
 - *Statelessness*.

Conceitos REST

- Resources
- Todos os recursos recebem um identificador.
- Representados por URLs (*Uniform Resource Locators*).
- Exemplo:
 - <http://obeautifulcode.com/game/robots/four-hand-george>
 - <https://tranquilotech.negocio.site/>

Conceitos REST

- Representations
- Sistemas RESTful permitem que os Clientes perguntem em uma forma que eles possam entender.
 - Exemplo de: HTTP *Header*

```
GET /pages/archive HTTP/1.1  
Host: obeautifulcode.com  
Accept: text/html
```
- *text/html* representa o **MIME type**.

Conceitos REST

- Operations
- REST define 4 operações padrão que são invocadas por HTTP (*Hypertext Transfer Protocol*):
 - GET: recupera algum recurso.
 - PUT: cria um recurso ou atualiza um existente.
 - POST: cria um recurso, mas deixa o *Server* decidir a URL.
 - DELETE: exclui o recurso.

Conceitos REST

- Hypertext
- No REST, o estado de um aplicativo é transferido e descoberto por meio de mensagens de hipertexto.
- O Cliente REST tem menos necessidade de saber como interagir com qualquer serviço.
- Por meio de hipertexto, é possível definir o que os navegadores *Web* devem fazer.

Conceitos REST

- Statelessness
- O REST estabelece que o Servidor não mantém nenhuma informação sobre o estado da sessão do Cliente.
 - A requisição que o Cliente faz deve conter toda a informação necessária para entendê-la.
 - O Cliente é responsável por mandar informações do estado para o Servidor sempre que necessário.

SOAP vs REST

SOAP x REST

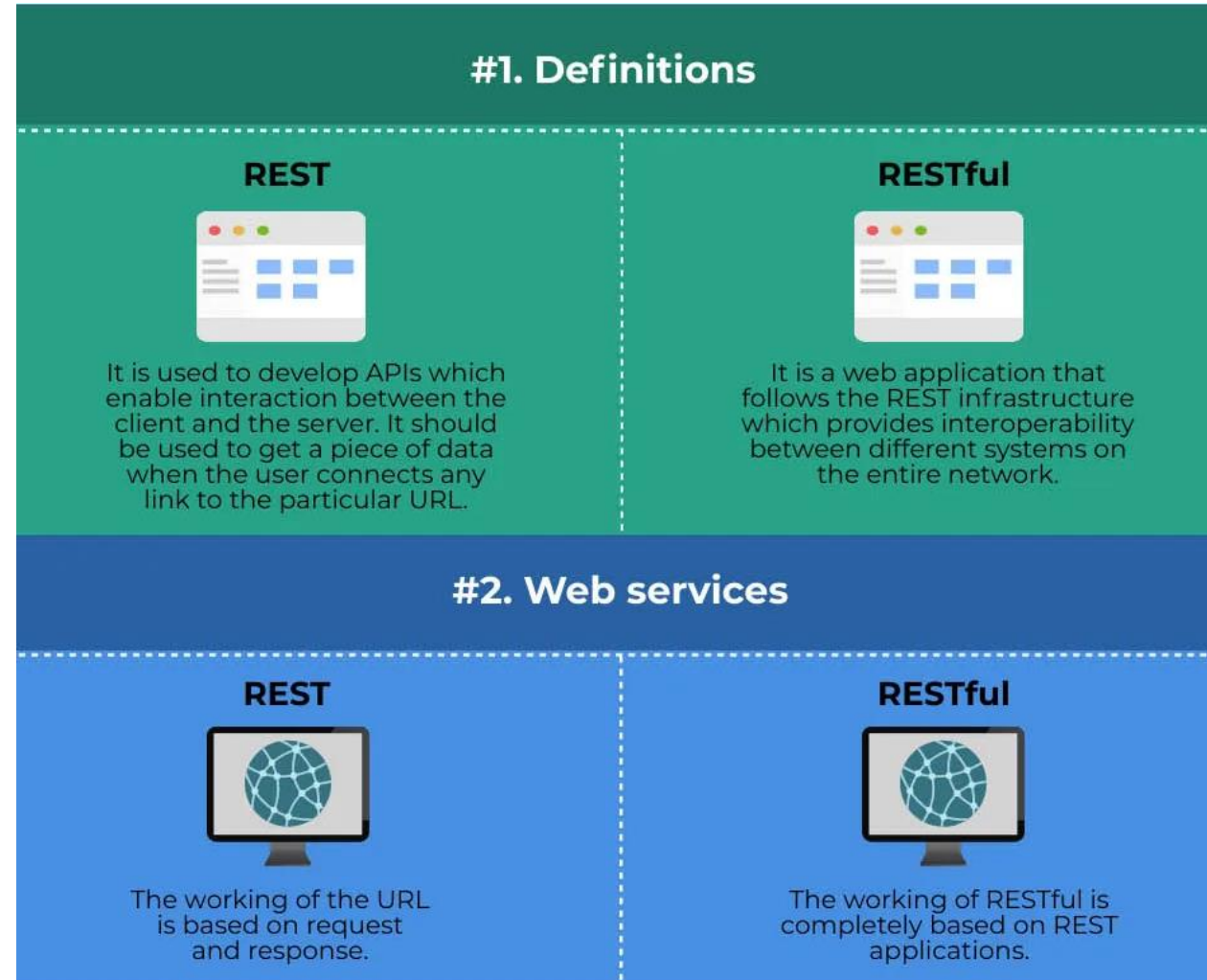
- SOAP (*Simple Object Access Protocol*) é considerado um protocolo, enquanto que o REST é um estilo arquitetural.
- REST é muito mais simples que SOAP. Criar *Clients*, desenvolver APIs e documentar é muito mais fácil e simples em REST.
- REST permite diferentes formatos para dados (JSON, XML, entre outros), enquanto SOAP permite somente XML. JSON por exemplo, é enxuto e tem processamento rápido.

SOAP x REST

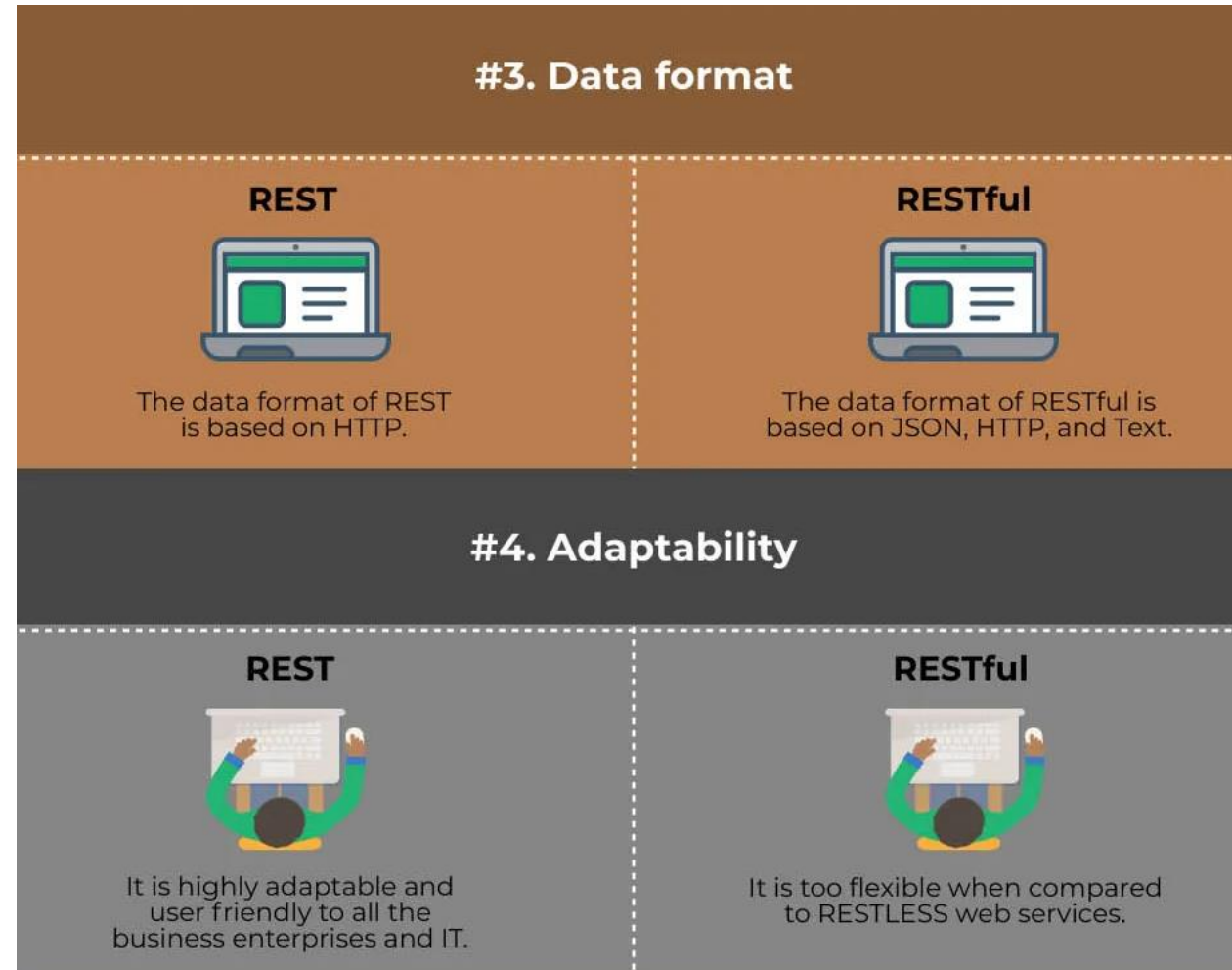
- REST tem uma melhor performance e escalabilidade, além de poder ser cacheado (uso de memória *cache*), enquanto que o SOAP não pode ser cacheado.
- Uma das vantagens do SOAP é o suporte à *WS-Security* (*Web Services Security*) que adiciona camadas de segurança extras além das suportadas através de HTTP, SSL (*Secure Socket Layer*), etc. Isso lhe dá algumas vantagens em determinadas finalidades no mundo “Enterprise”.
- O SOAP suporta transações ACID (*Atomicity, Consistency, Isolation and Durability*) – mesmo conceito encontrado em transações de banco de dados. O REST também suporta.

REST vs RESTful

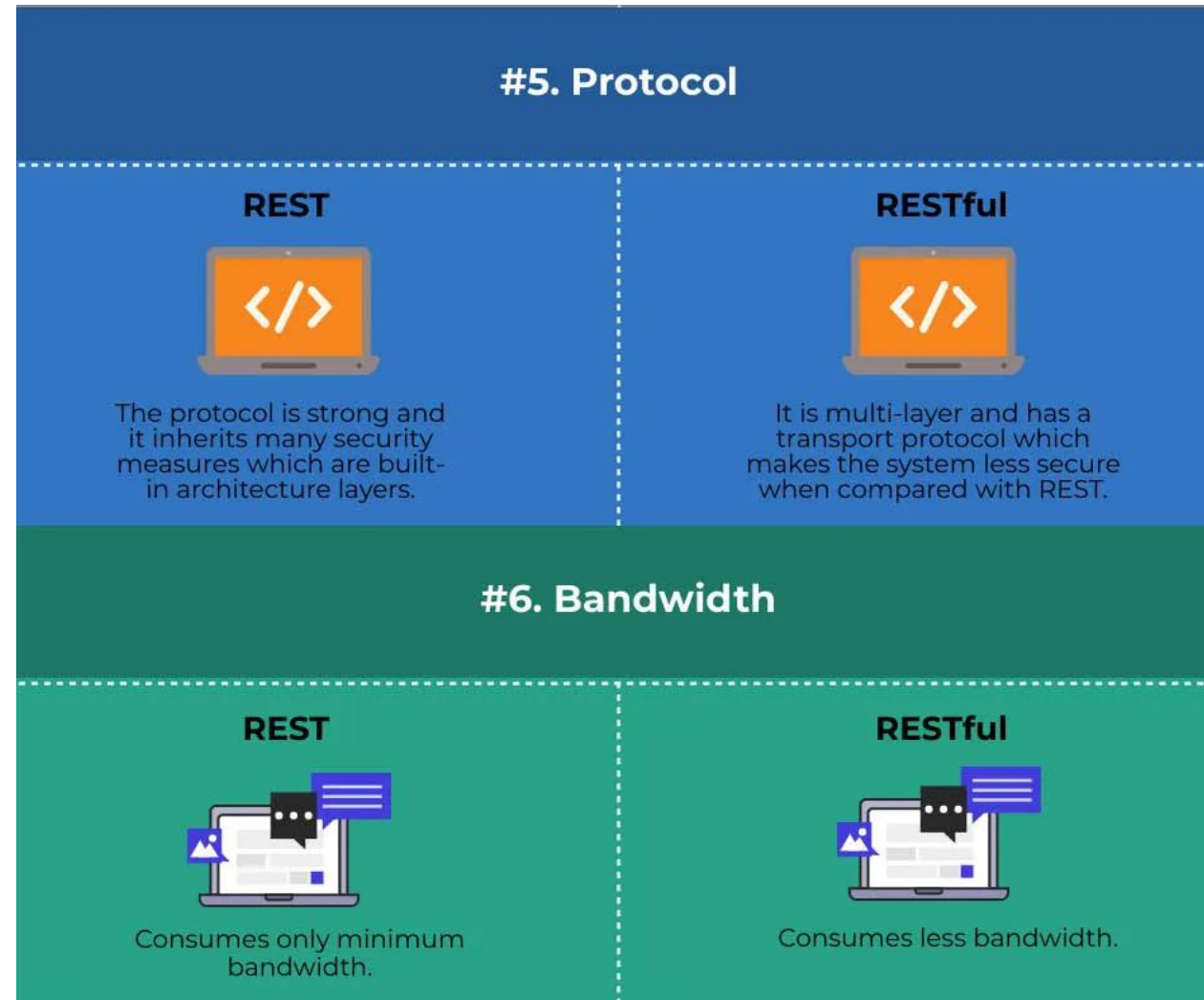
REST vs RESTful



REST vs RESTful



REST vs RESTful



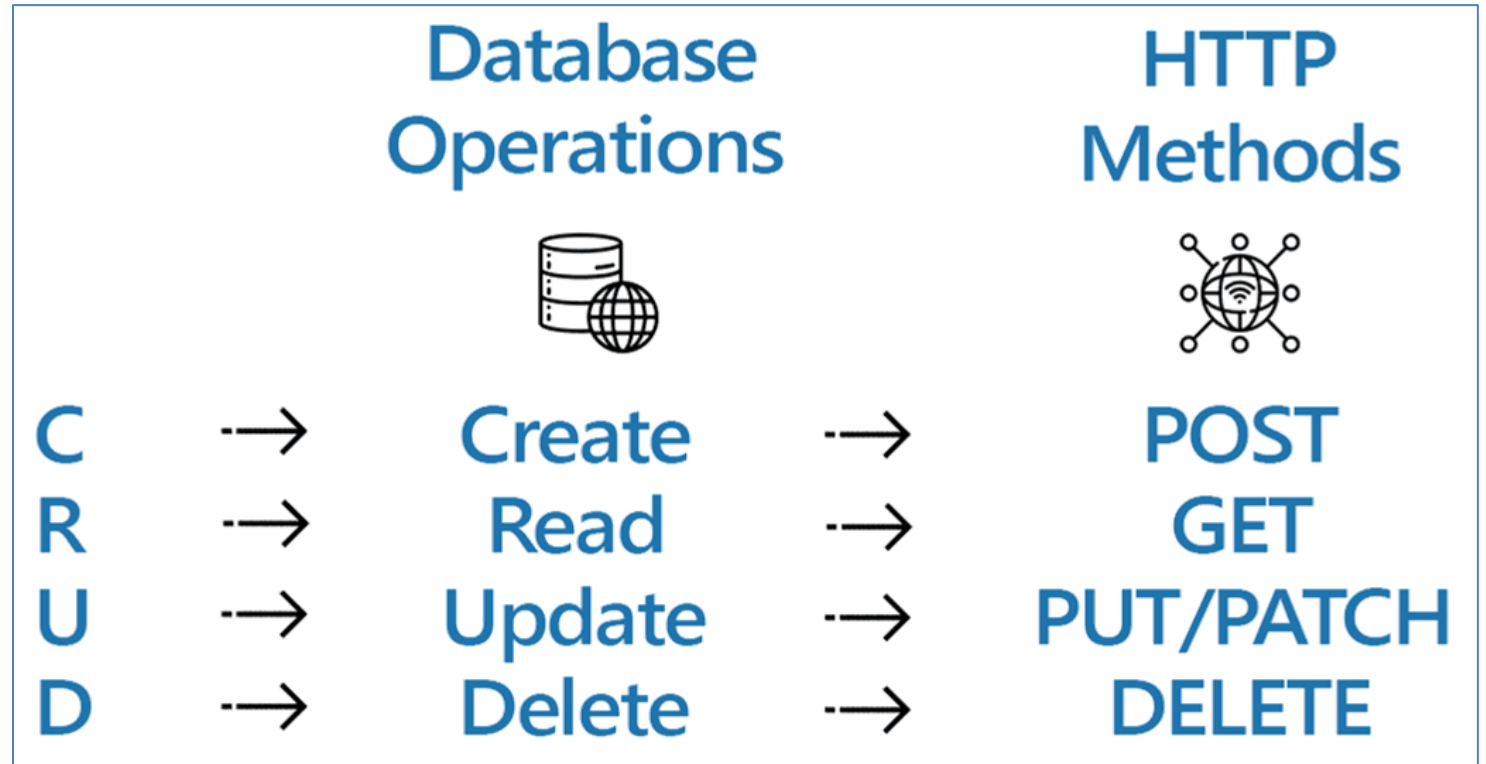
CRUD

(Conceito básico)

CRUD

• CRUD é uma sigla que designa quatro operações que são realizadas em bancos de dados relacionais:

- Criação (*Create*);
- Leitura (*Read*);
- Atualização (*Update*);
- Remoção (*Delete*).



REST com NetBeans

REST com NetBeans – *Framework Spring*

- Comece acessando o *Framework* a seguir:
 - <https://start.spring.io/>
 - Este site vai ajudar a gerar o projeto inicial que será importado para o NetBeans.



Programação

REST com NetBeans

• Configure o projeto:

C:\ Prompt de Comando

```
Microsoft Windows [versão 10.0.19045.2965]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\MSWagner>java --version
java 20 2023-03-21
Java(TM) SE Runtime Environment (build 20+36-2344)
Java HotSpot(TM) 64-Bit Server VM (build 20+36-2344, mixed mode, sharing)

C:\Users\MSWagner>
```

OU

C:\ Prompt de Comando

```
Microsoft Windows [versão 10.0.19045.4170]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\MSWagner>java -version
openjdk version "1.8.0_302"
OpenJDK Runtime Environment (Temurin)(build 1.8.0_302-b08)
OpenJDK 64-Bit Server VM (Temurin)(build 25.302-b08, mixed mode)

C:\Users\MSWagner>
```

Project

☐ Gradle - Groovy

☐ Gradle - Kotlin ☒ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.1.1 (SNAPSHOT) ☒ 3.1.0 ☐ 3.0.8 (SNAPSHOT) ☐ 3.0.7

☐ 2.7.13 (SNAPSHOT) ☐ 2.7.12

Project Metadata

Group net.codejava

Artifact hellospringboot

Name hellospringboot

Description My first Spring Boot app

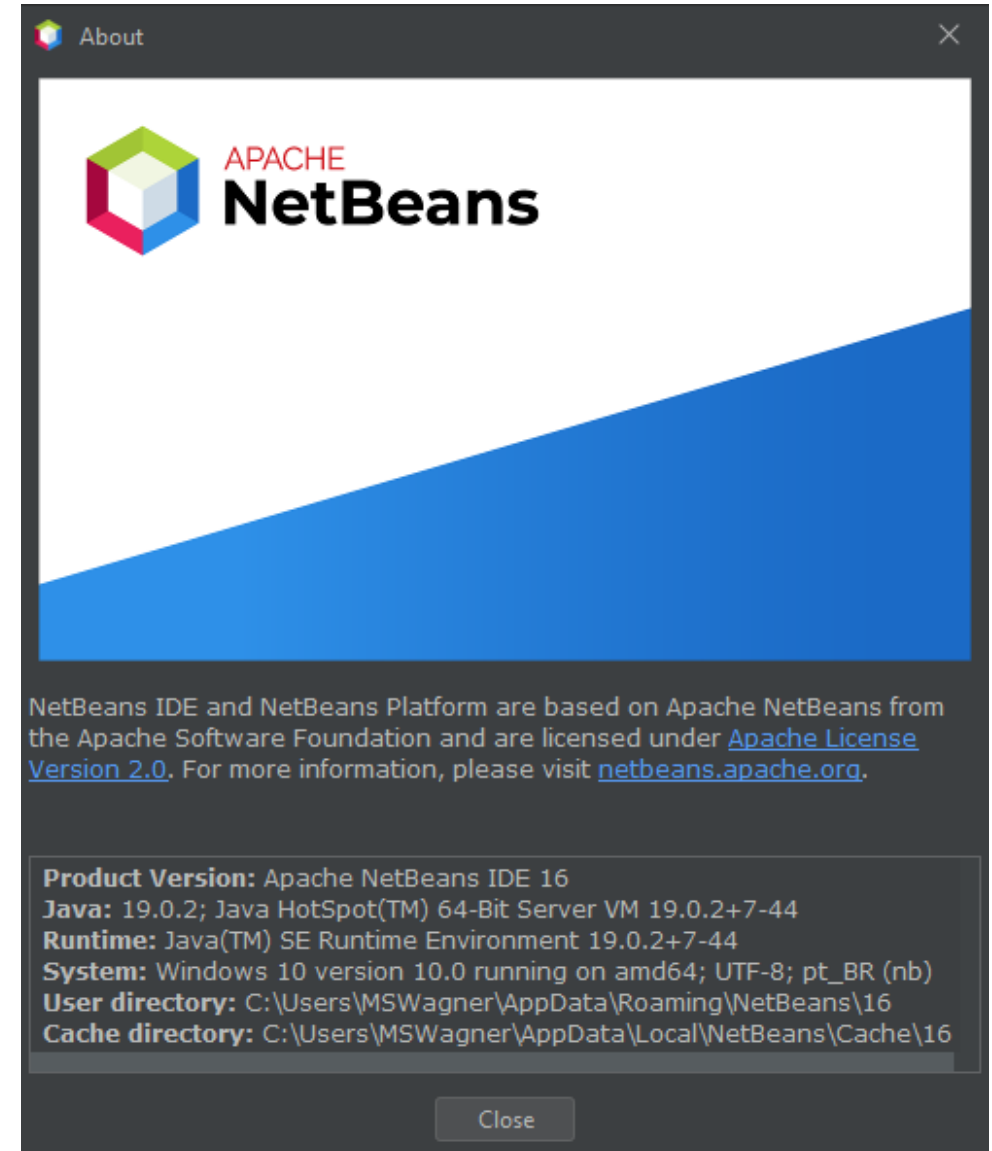
Package name net.codejava.hellospringboot

Packaging ☒ Jar ☐ War

Java ☒ 20 ☐ 17 ☐ 11 ☐ 8

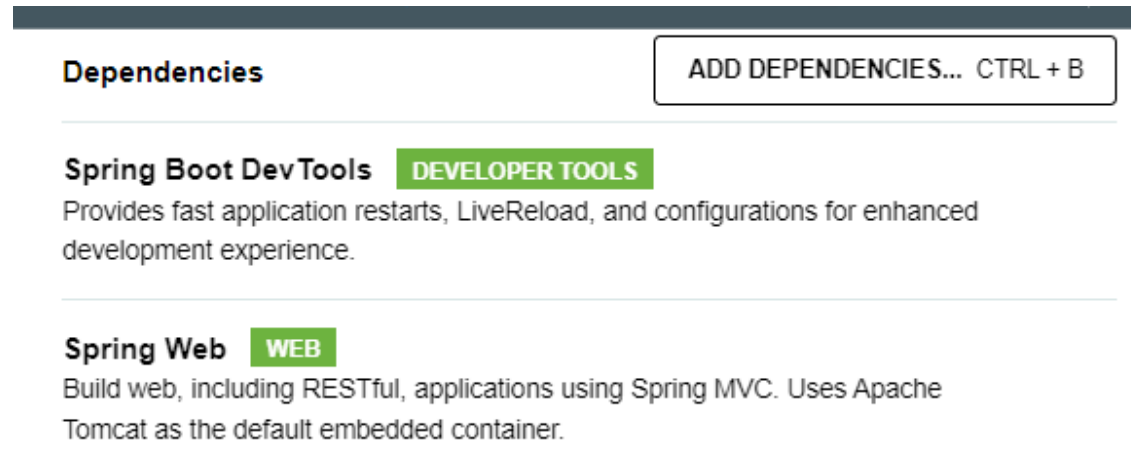
REST com NetBeans

- Escolha a versão do Java correspondente à sua máquina.
 - Para fazer isso no NetBeans, clique em Sobre (*About*).

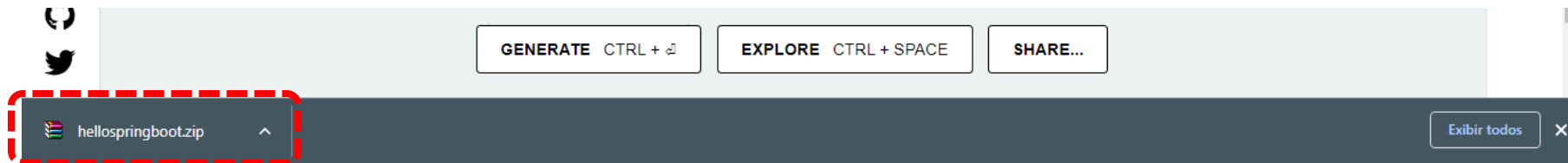


REST com NetBeans

- Adicione a dependência do *Spring Boot* para o projeto.



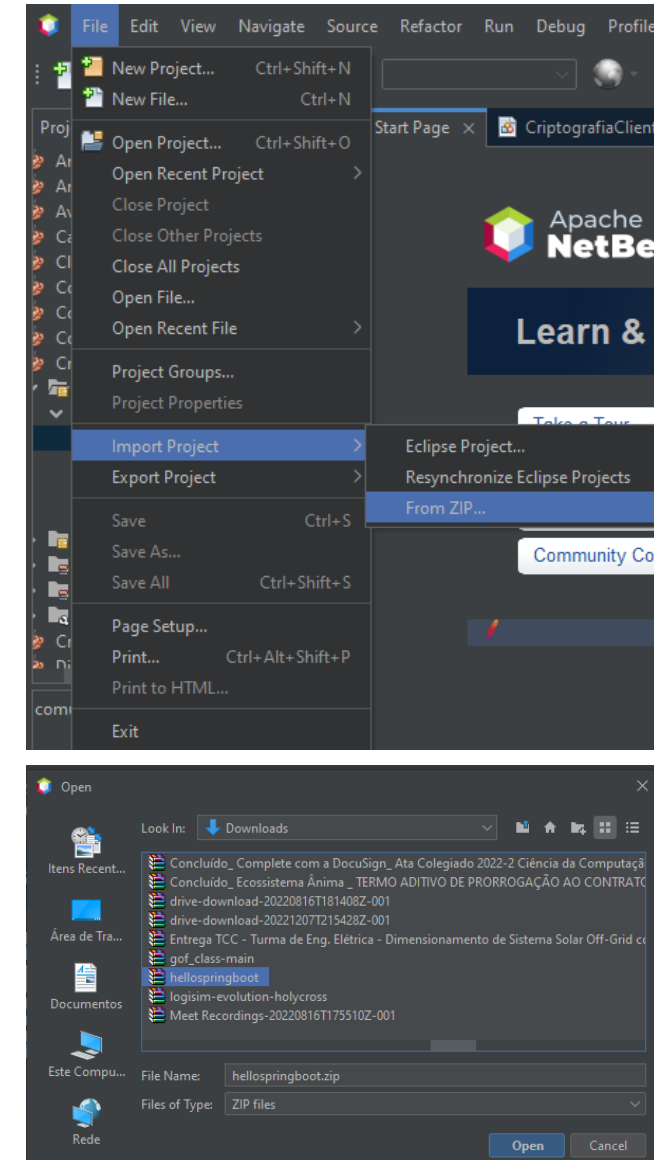
- Quando terminar tudo, clique em *Generate* e salve o arquivo.



Programação

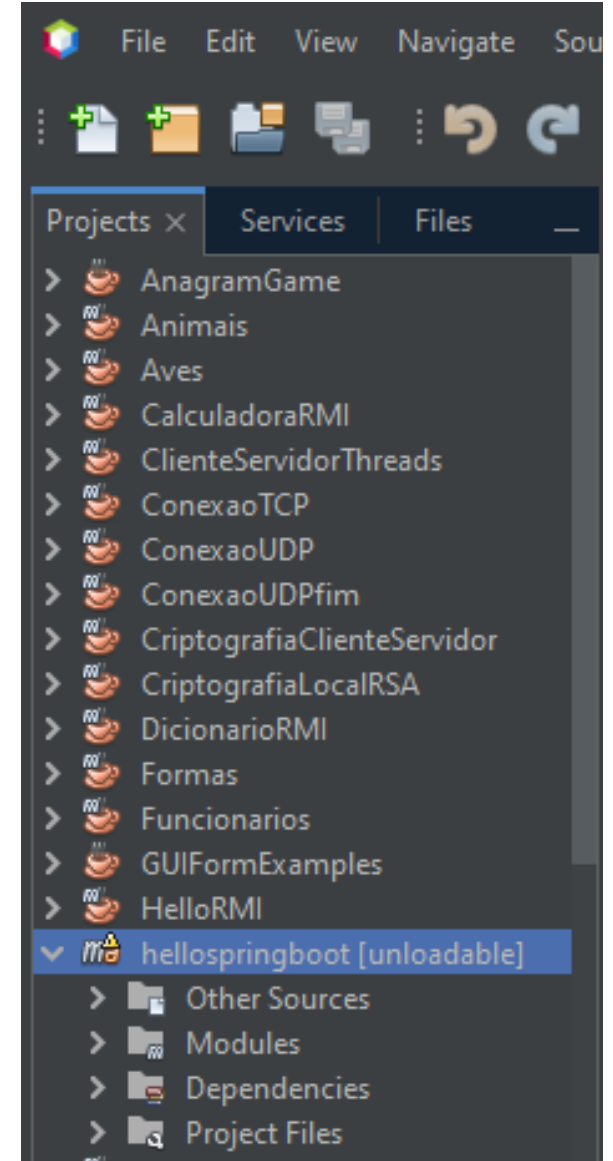
REST com NetBeans

- Para importar o arquivo gerado no NetBeans, selecione:
 - Arquivo → Importar Projeto → ZIP.
- Se do lado do nome do projeto, a palavra *unloadable* for mostrada, houve um problema na importação do projeto.
 - Clique com o lado direito do *mouse* no projeto e em Propriedades.
 - Clique no botão de resolver problemas no lado direito.
 - Espere o *Maven* baixar os módulos necessários.



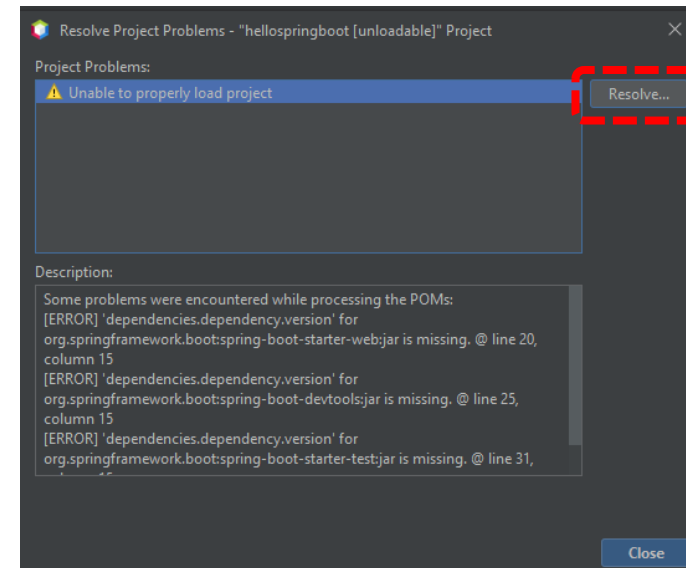
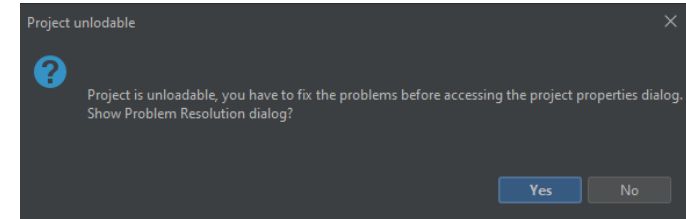
REST com NetBeans

- Para importar o arquivo gerado no NetBeans, selecione:
 - Arquivo → Importar Projeto → ZIP.
- Se do lado do nome do projeto, a palavra *unloadable* for mostrada, houve um problema na importação do projeto.
 - Clique com o lado direito do *mouse* no projeto e em Propriedades.
 - Clique no botão de resolver problemas no lado direito.
 - Espere o *Maven* baixar os módulos necessários.



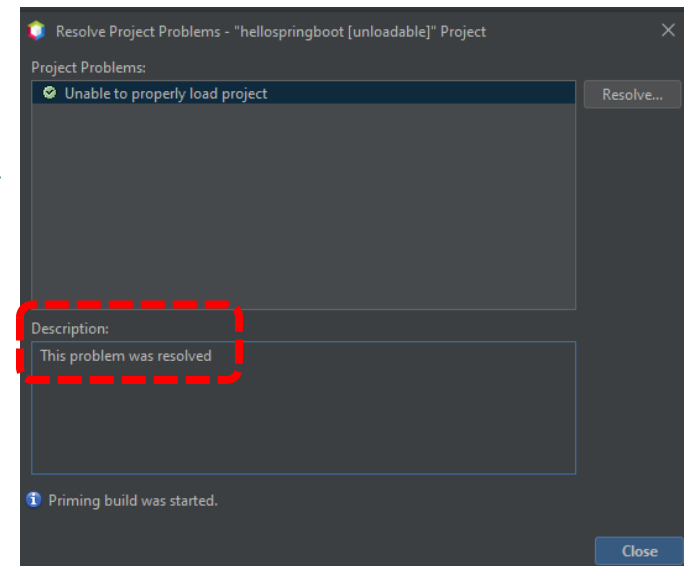
REST com NetBeans

- Para importar o arquivo gerado no NetBeans, selecione:
 - Arquivo → Importar Projeto → ZIP.
- Se do lado do nome do projeto, a palavra *unloadable* for mostrada, houve um problema na importação do projeto.
 - Clique com o lado direito do *mouse* no projeto e em Propriedades.
 - Clique no botão de resolver problemas no lado direito.
 - Espere o *Maven* baixar os módulos necessários.



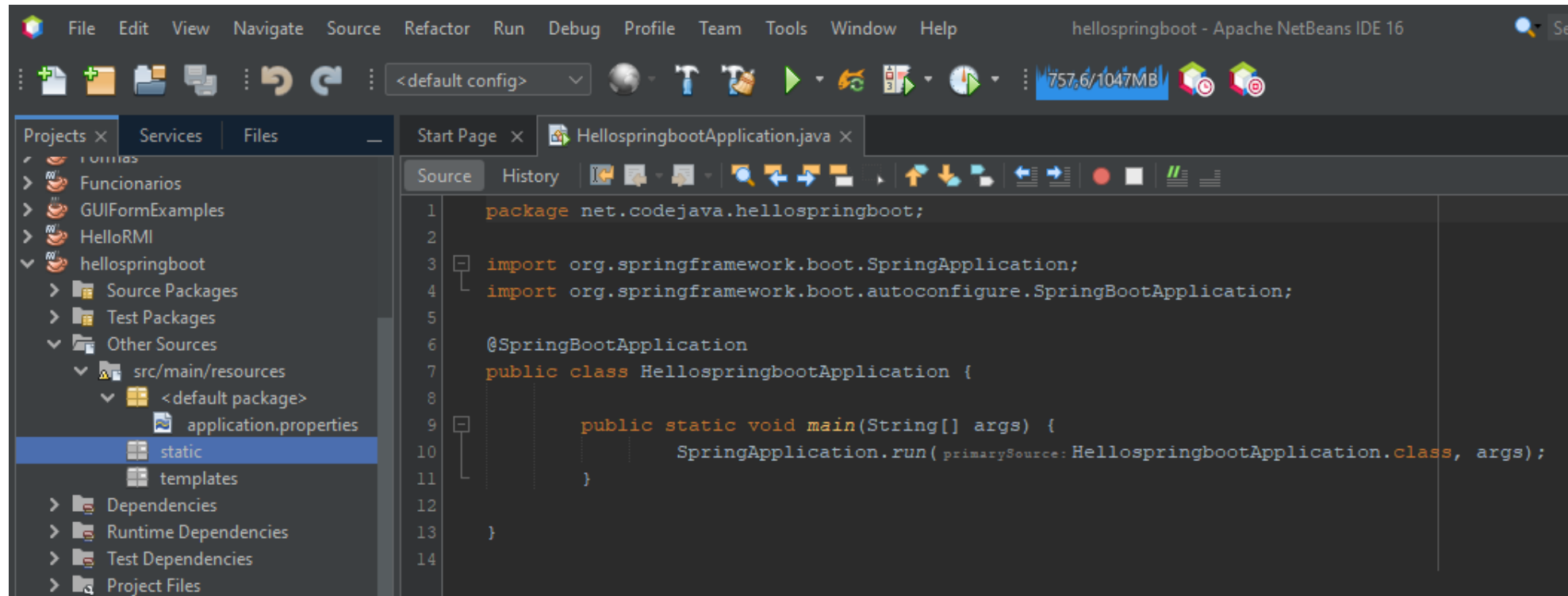
REST com NetBeans

- Para importar o arquivo gerado no NetBeans, selecione:
 - Arquivo → Importar Projeto → ZIP.
- Se do lado do nome do projeto, a palavra *unloadable* for mostrada, houve um problema na importação do projeto.
 - Clique com o lado direito do *mouse* no projeto e em Propriedades.
 - Clique no botão de resolver problemas no lado direito.
 - Espere o *Maven* baixar os módulos necessários.



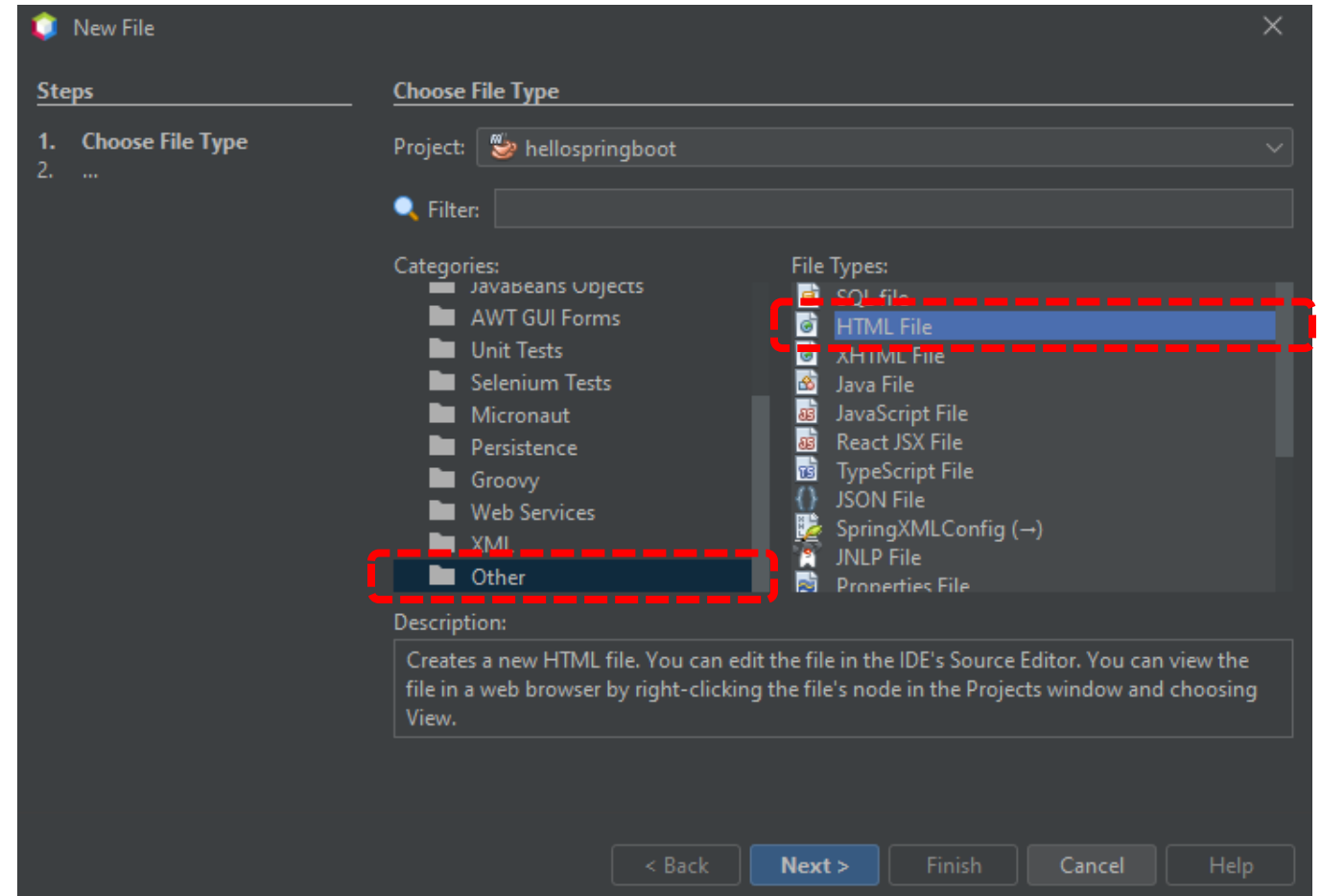
REST com NetBeans

- Vamos fazer um primeiro exemplo utilizando uma página HTML.
- Na pasta *static*, em *Other Sources*, crie um arquivo chamado *index.html*.



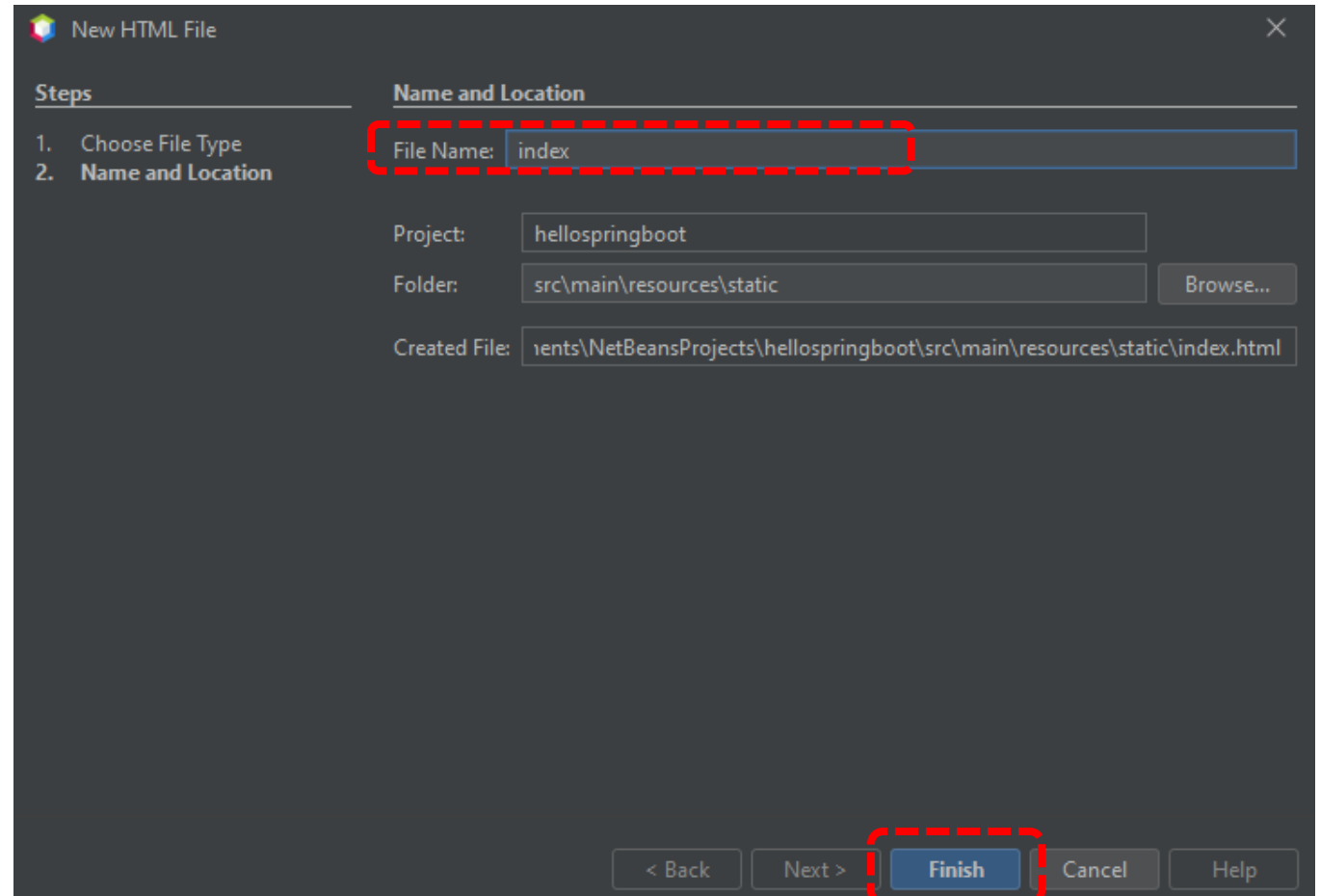
REST com NetBeans

- Vamos fazer um primeiro exemplo utilizando uma página HTML.
- Na pasta *static*, em *Other Sources*, crie um arquivo chamado *index.html*.



REST com NetBeans

- Vamos fazer um primeiro exemplo utilizando uma página HTML.
- Na pasta *static*, em *Other Sources*, crie um arquivo chamado *index.html*.



REST com NetBeans

- Vamos fazer um primeiro exemplo utilizando uma página HTML.
- Na pasta *static*, em *Other Resources*, crie um arquivo chamado *index.html*.

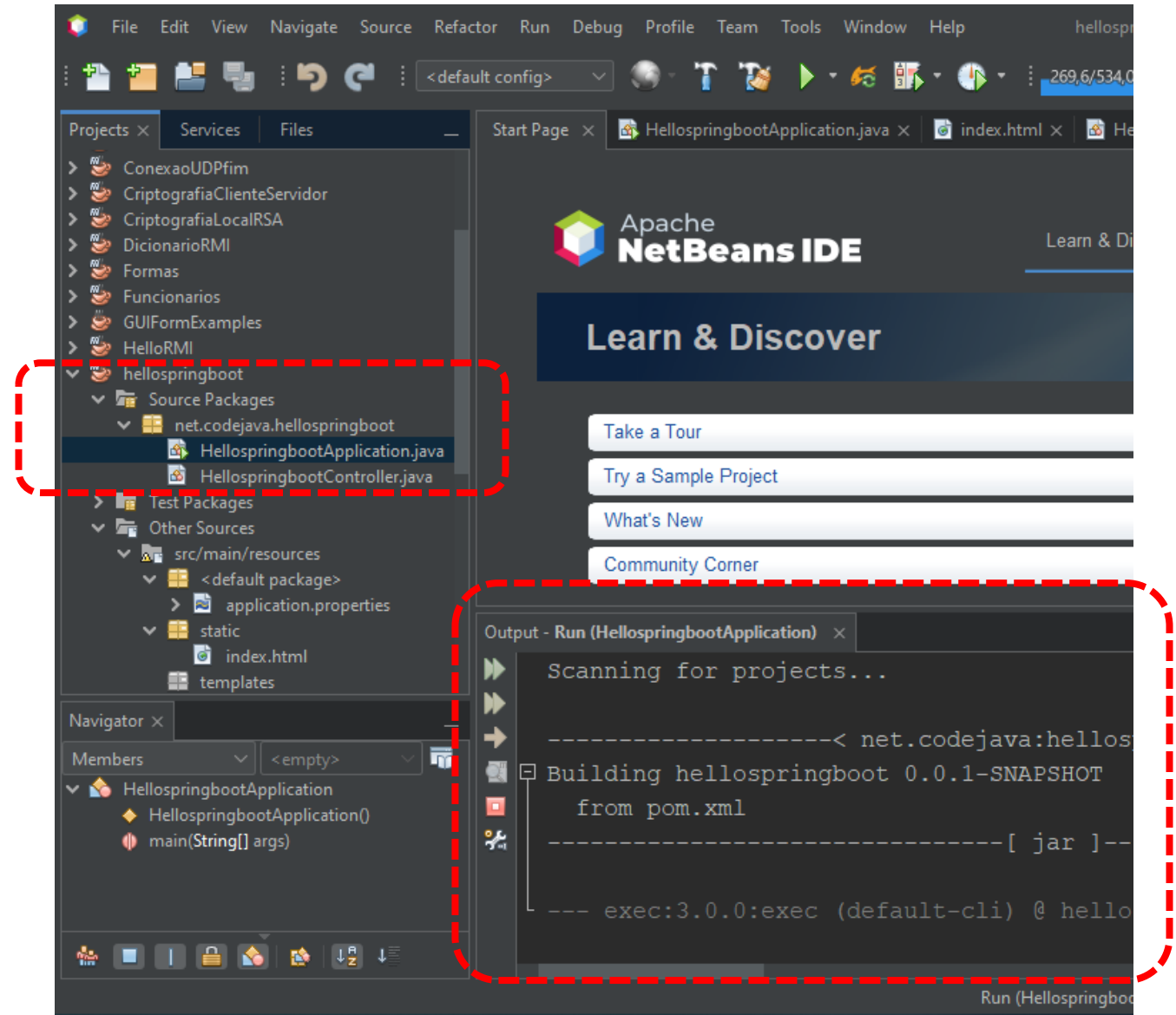
```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div>TODO write content</div>
  </body>
</html>
```

Programação

REST com NetBeans

- A aplicação é executada por padrão na porta 8080, então podemos acessá-la pelo navegador:

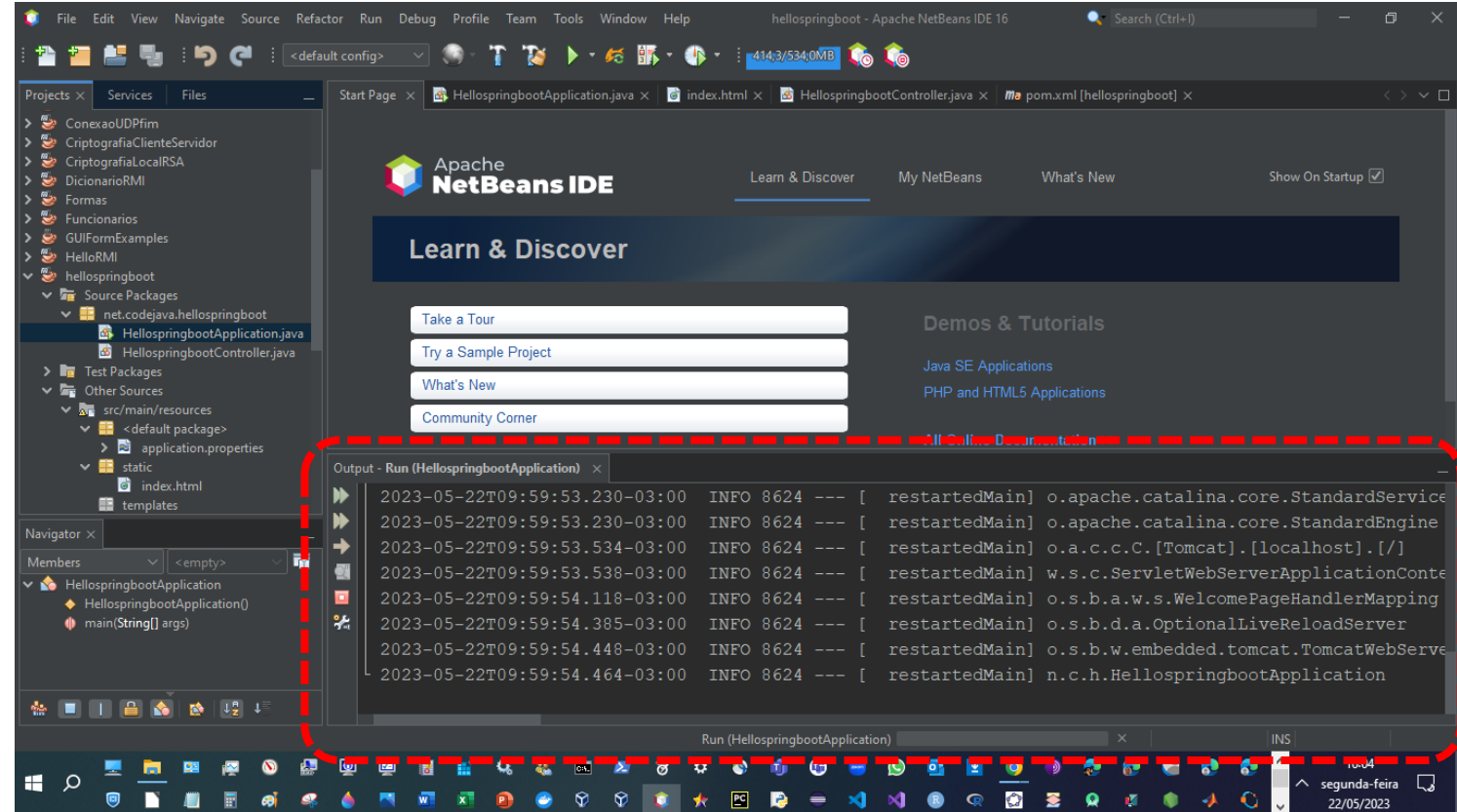
- *localhost:8080* ← Antes de qualquer coisa, inicialize o Servidor no NetBeans. Clique com o botão direito do *mouse* e selecione *run file*.



Programação

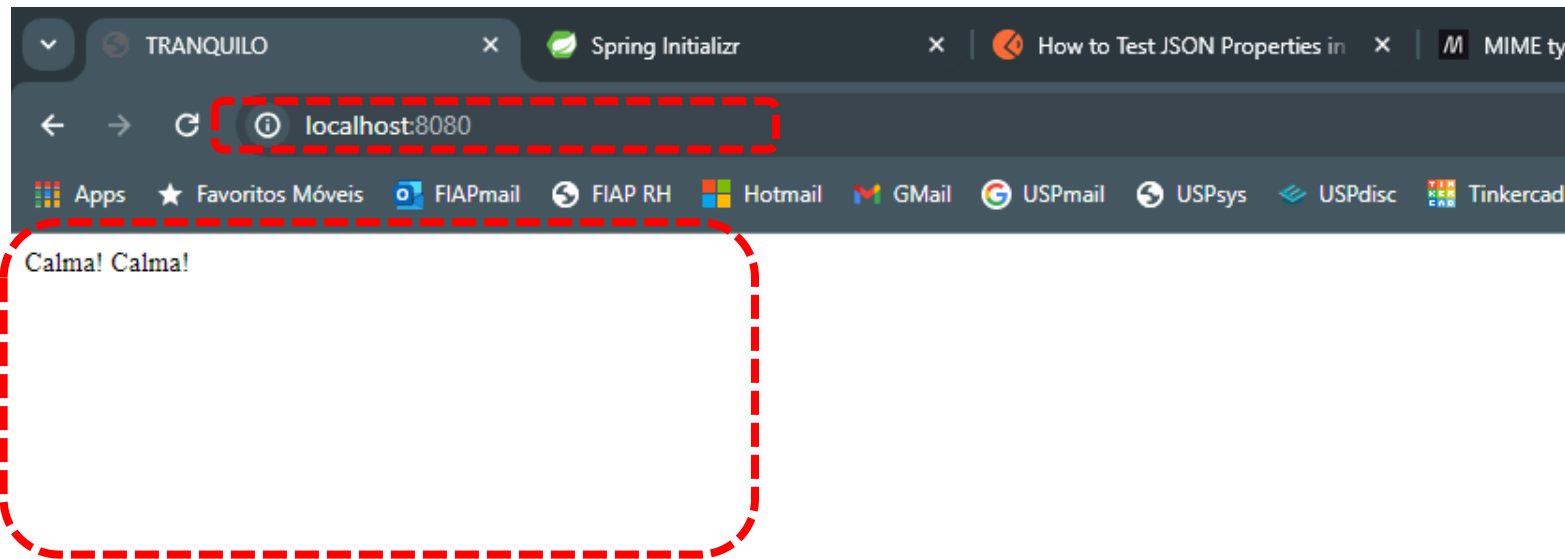
REST com NetBeans

- A aplicação é executada por padrão na porta 8080, então podemos acessá-la pelo navegador:
 - Se estiver tudo funcionando corretamente, a tela ao lado será visualizada.



REST com NetBeans

- A aplicação é executada por padrão na porta 8080, então podemos acessá-la pelo navegador:
 - *localhost:8080*



REST com NetBeans

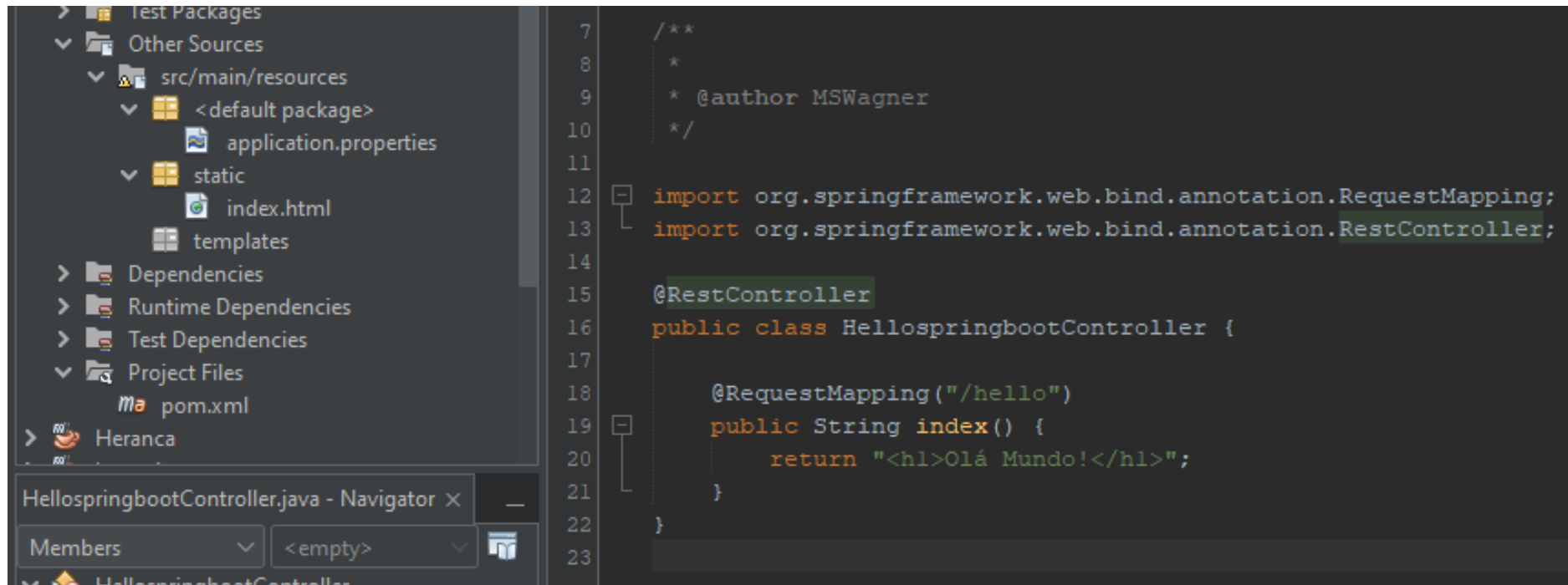
- Agora, no mesmo pacote do arquivo *HelloSpringBootApplication.java*, adicione um arquivo:

HelloSpringbootController.java.

- Vamos criar neste momento, um *endpoint* chamado “*hello*”.

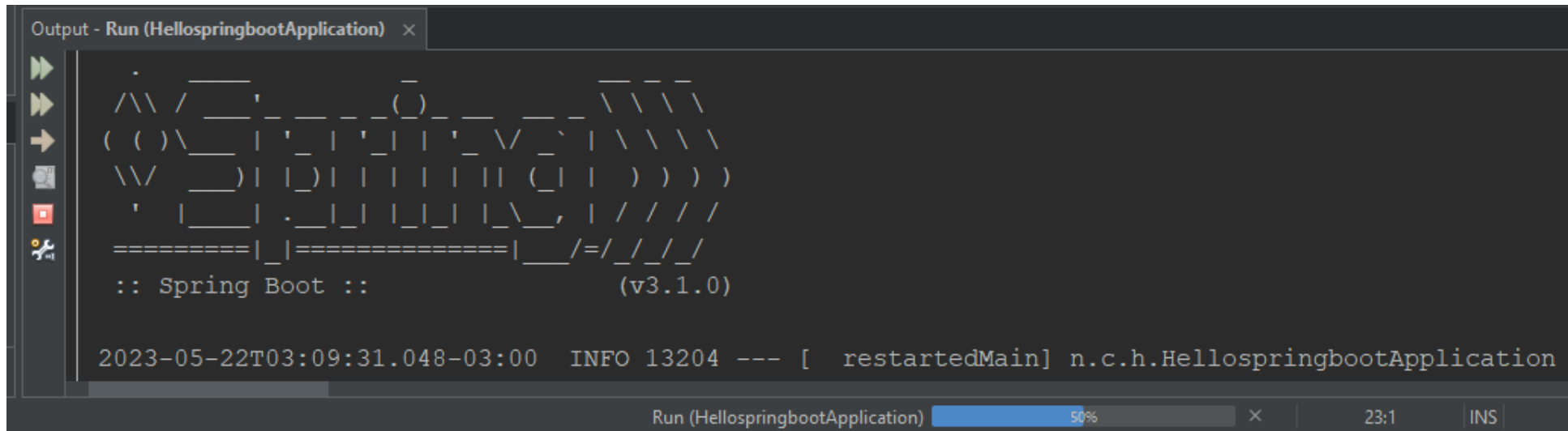
REST com NetBeans

- Adicionaremos os *endpoints* no arquivo *Controller*.
 - Adicione a anotação *RestController* no topo da classe.



REST com NetBeans

- Adicionaremos os *endpoints* no arquivo *Controller*.
 - Adicione a anotação *RestController* no topo da classe.
- Consumimos a API usando o endereço:
 - *localhost:8080/hello*



The screenshot shows the 'Output - Run (HelloSpringBootApplication)' window in NetBeans. It displays the ASCII art logo for Spring Boot (v3.1.0) and a log message: '2023-05-22T03:09:31.048-03:00 INFO 13204 --- [restartedMain] n.c.h.HelloSpringBootApplication'. The status bar at the bottom indicates the application is running at 50% progress.

```
Output - Run (HelloSpringBootApplication) x
>>>
>>>
>
[Spring Boot logo]
:: Spring Boot ::                (v3.1.0)

2023-05-22T03:09:31.048-03:00 INFO 13204 --- [ restartedMain] n.c.h.HelloSpringBootApplication

Run (HelloSpringBootApplication) 50% x 23:1 INS
```


REST com NetBeans

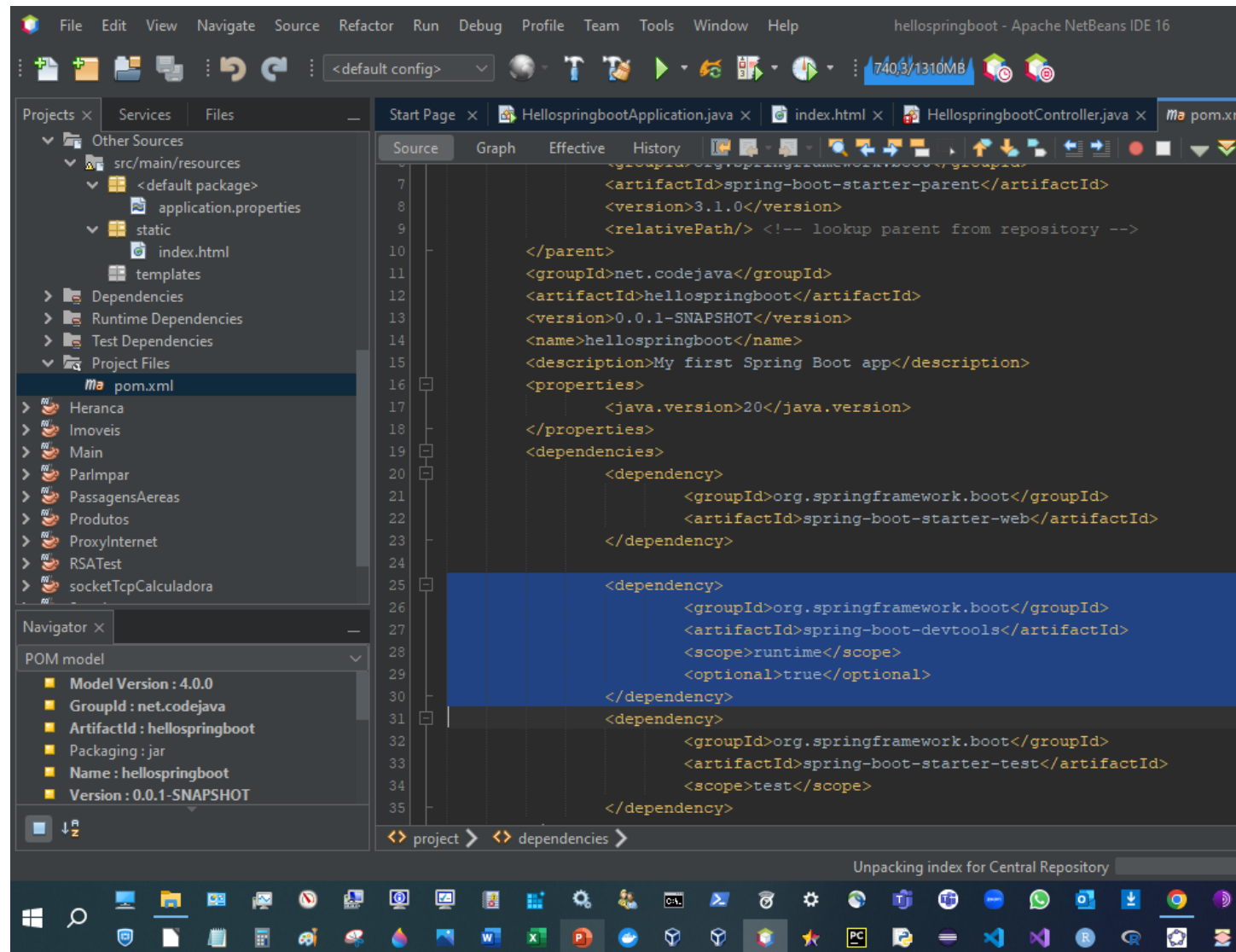
- Se mudarmos algo, é necessário reiniciar a aplicação.
 - Para não ser necessário reiniciá-la a cada mudança, adicionamos a dependência *devtools* em *pom.xml*.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
```

Programação

REST com NetBeans

- Se mudarmos algo, é necessário reiniciar a aplicação.
- Para não ser necessário reiniciá-la a cada mudança, adicionamos a dependência *devtools* em *pom.xml*.
- No meu caso, não foi necessário, pois já constava a dependência.



REST com NetBeans

- Anotação ***PathVariable***:
 - Podemos utilizá-la para trabalhar com variáveis de *template*.
 - São passadas como parâmetros de método.

```
@RequestMapping("/hello")
public String index() {
    return "<h1>Olá Mundo!</h1>";
}

@RequestMapping("/cadastro/{nome}")
public String dizernome(@PathVariable String nome) {
    return "Olá, meu nome é " + nome;
}
```

- Base de acesso: *localhost:8080/cadastro/%7Bnome%7D*
- Para visualizar: *localhost:8080/cadastro/Marcel*

REST com NetBeans

- Anotação ***RequestParam***:
 - Podemos utilizá-la para extrair parâmetros da *query* da requisição realizada.

```
@RequestMapping("/cadastro/{nome}")
public String dizernome(@PathVariable String nome) {
    return "Olá, meu nome é " + nome;
}

@RequestMapping("/info")
public String apresentar(@RequestParam("nome") String nome, @RequestParam("idade") int idade) {
    return "<h1>Olá pessoal, meu nome é " + nome + " e eu tenho " + idade + " anos</h1>";
}
```

- Base de acesso: *localhost:8080/info?nome=<nome>&idade=<idade>*
- Para visualizar: *localhost:8080/info?nome=Marcel&idade=30*

REST com NetBeans

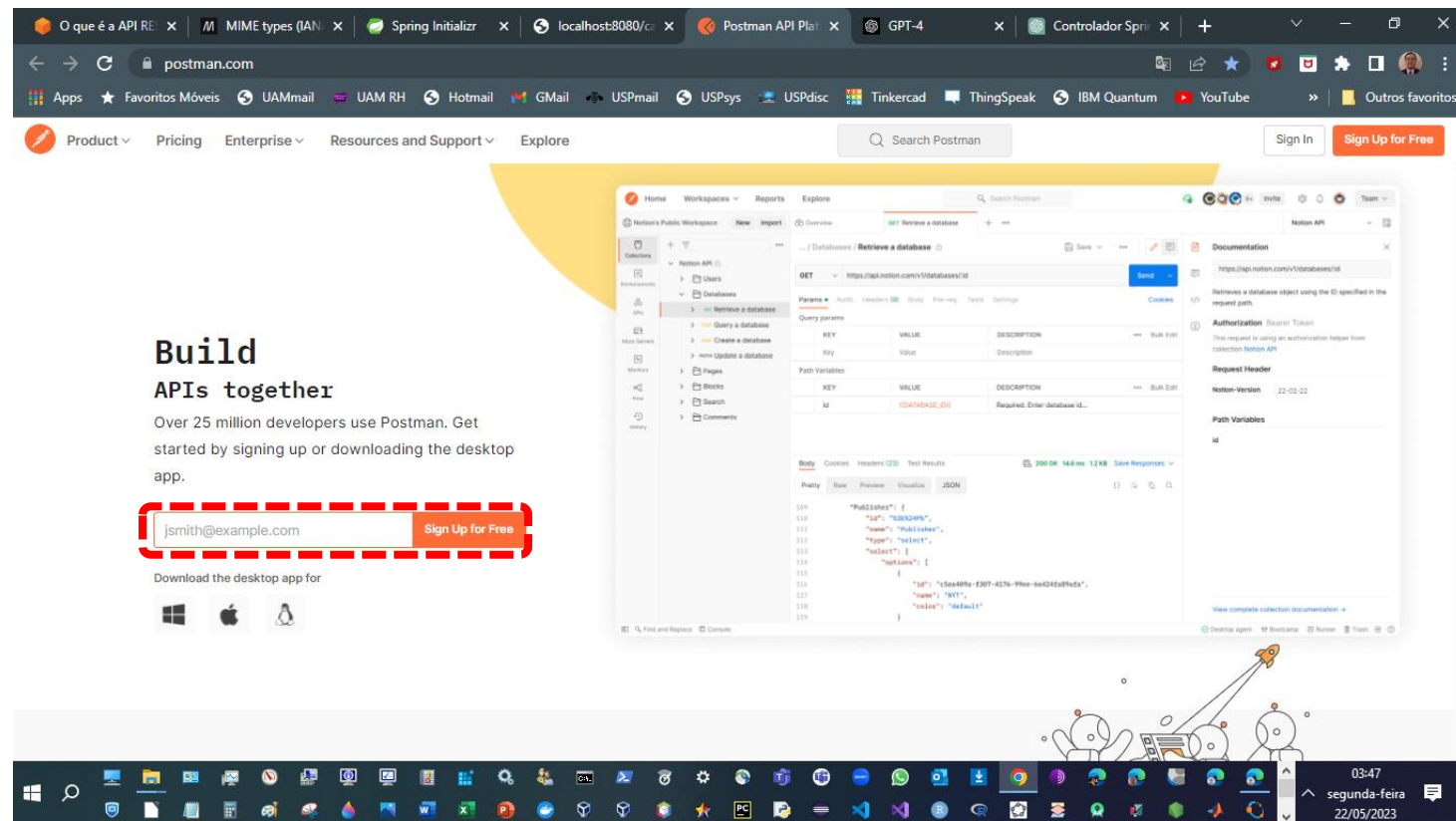
- Até agora, só fizemos requisições HTTP do tipo GET (*RequestMapping*).
 - Verbo padrão HTTP.
 - Utilizado com mais frequência.

REST com NetBeans

- Até agora, só fizemos requisições HTTP do tipo GET (*RequestMapping*).
 - Verbo padrão HTTP.
 - Utilizado com mais frequência.
- Podemos utilizar outros verbos HTTP como o POST.
- Vamos utilizar uma aplicação para nos ajudar a simular as requisições de outros tipos além de GET.

REST com NetBeans

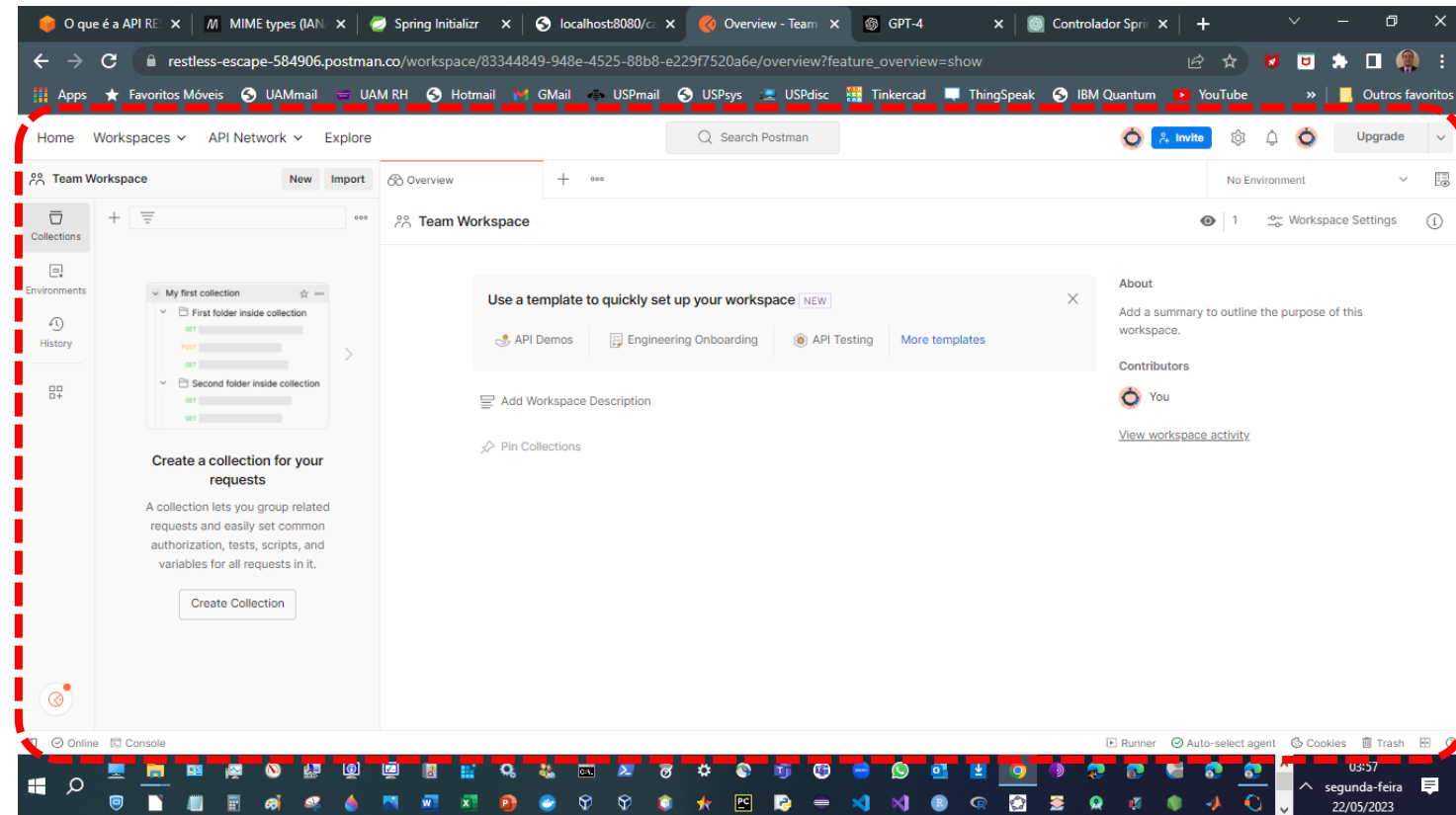
- Postman
- Disponível em:
 - <https://www.postman.com/>
- Ferramenta utilizada para testar APIs.



Programação

REST com NetBeans

- Postman
- Disponível em:
 - <https://www.postman.com/>
- Ferramenta utilizada para testar APIs.



REST com NetBeans

- Vamos criar um *endpoint* para requisições POST.
 - Utilizamos a anotação *PostMapping* para atender requisições POST.

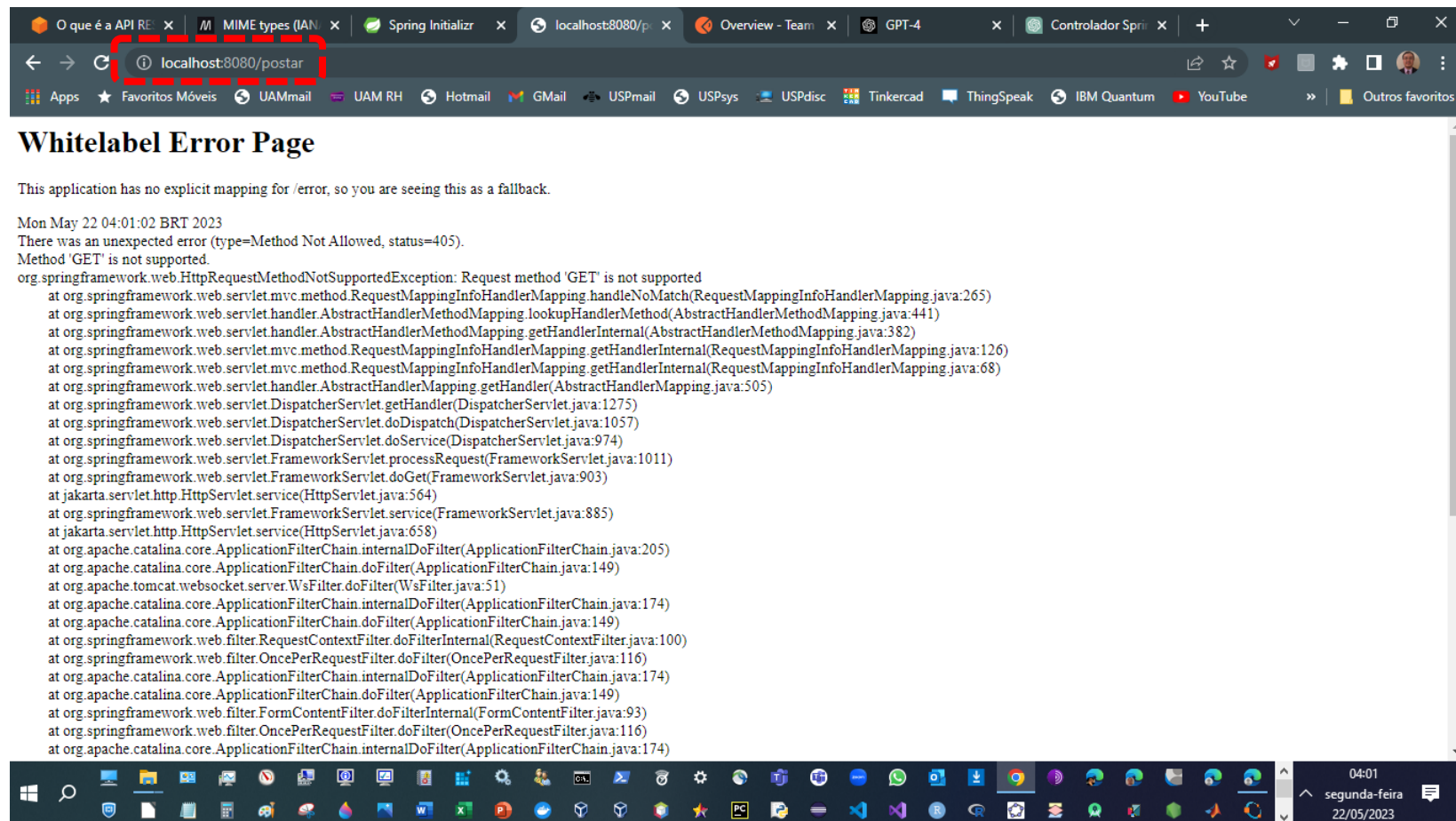
```
@RequestMapping("/info")
public String apresentar(@RequestParam("nome") String nome, @RequestParam("idade") int idade) {
    return "<h1>Olá pessoal, meu nome é " + nome + " e eu tenho " + idade + " anos</h1>";
}

@PostMapping("/postar")
public String postar() {
    return "Objeto postado com sucesso.";
}
```

- No *Postman*, verifique a resposta da URL para requisições POST:
 - *localhost:8080/postar*

REST com NetBeans

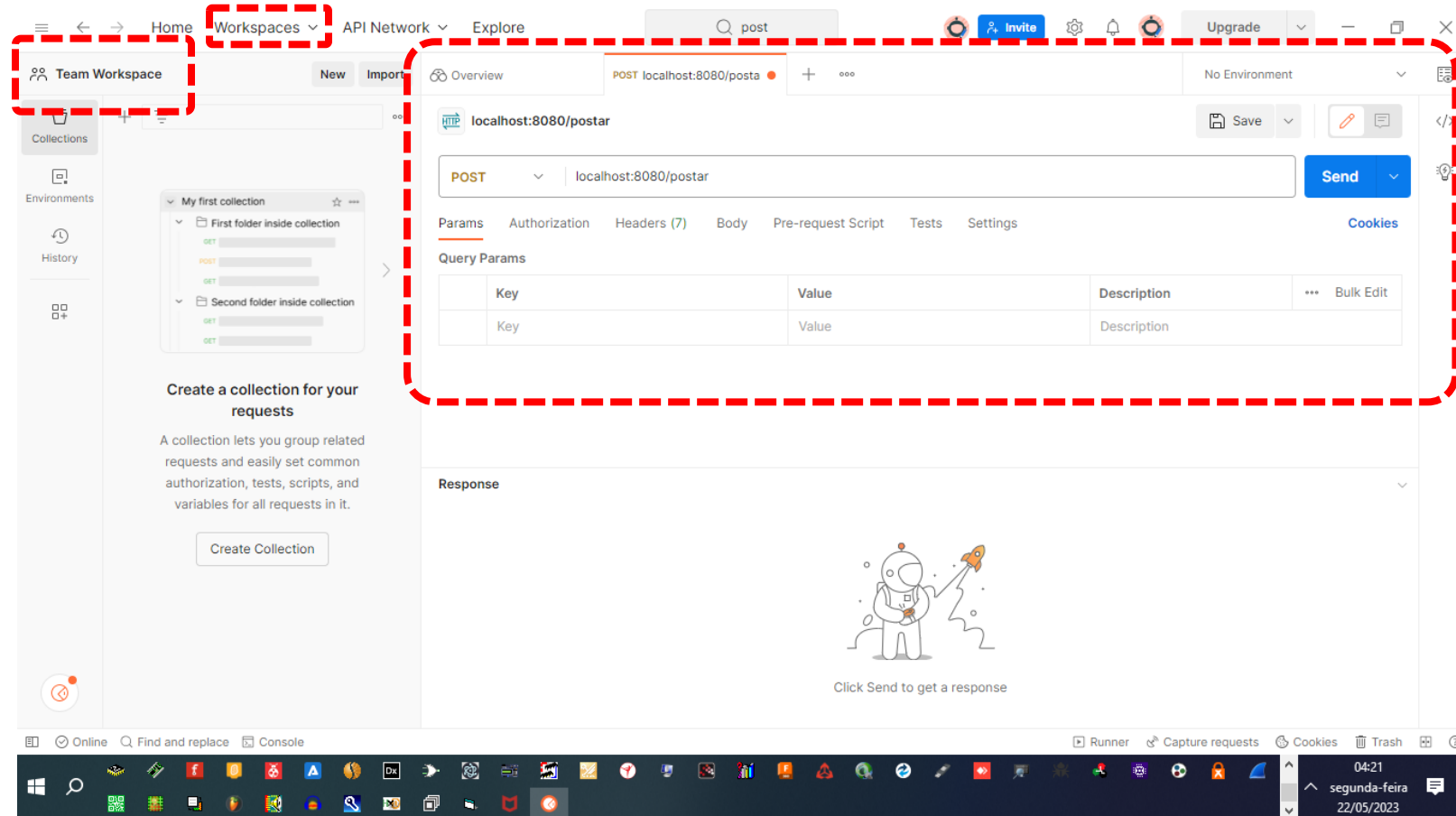
- Se for pelo navegador dá erro, pois trabalha apenas com o GET e, desta forma, não atende as requisições POST.



Programação

REST com NetBeans

- No aplicativo POSTMAN, vá até a parte superior e selecione o método utilizado, clicando no (+):



REST com NetBeans

- Resposta recebida em POST:

```
@PostMapping("/postar")
public String postar() {
    return "Objeto postado com sucesso.";
}
```

The screenshot shows the NetBeans REST Client interface. At the top, the URL bar shows 'localhost:8080/postar' with a dropdown menu set to 'POST'. The 'Send' button is visible. Below the URL bar, the 'Params' tab is selected, showing a table for 'Query Params' with columns 'Key', 'Value', and 'Description'. The 'Body' tab is also visible, showing the response body 'Objeto postado com sucesso.' in a text editor. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 28 ms', and 'Size: 191 B'.

Key	Value	Description
Key	Value	Description

Key	Value	Description
Key	Value	Description

Objeto postado com sucesso.

Status: 200 OK Time: 28 ms Size: 191 B

REST com IntelliJ

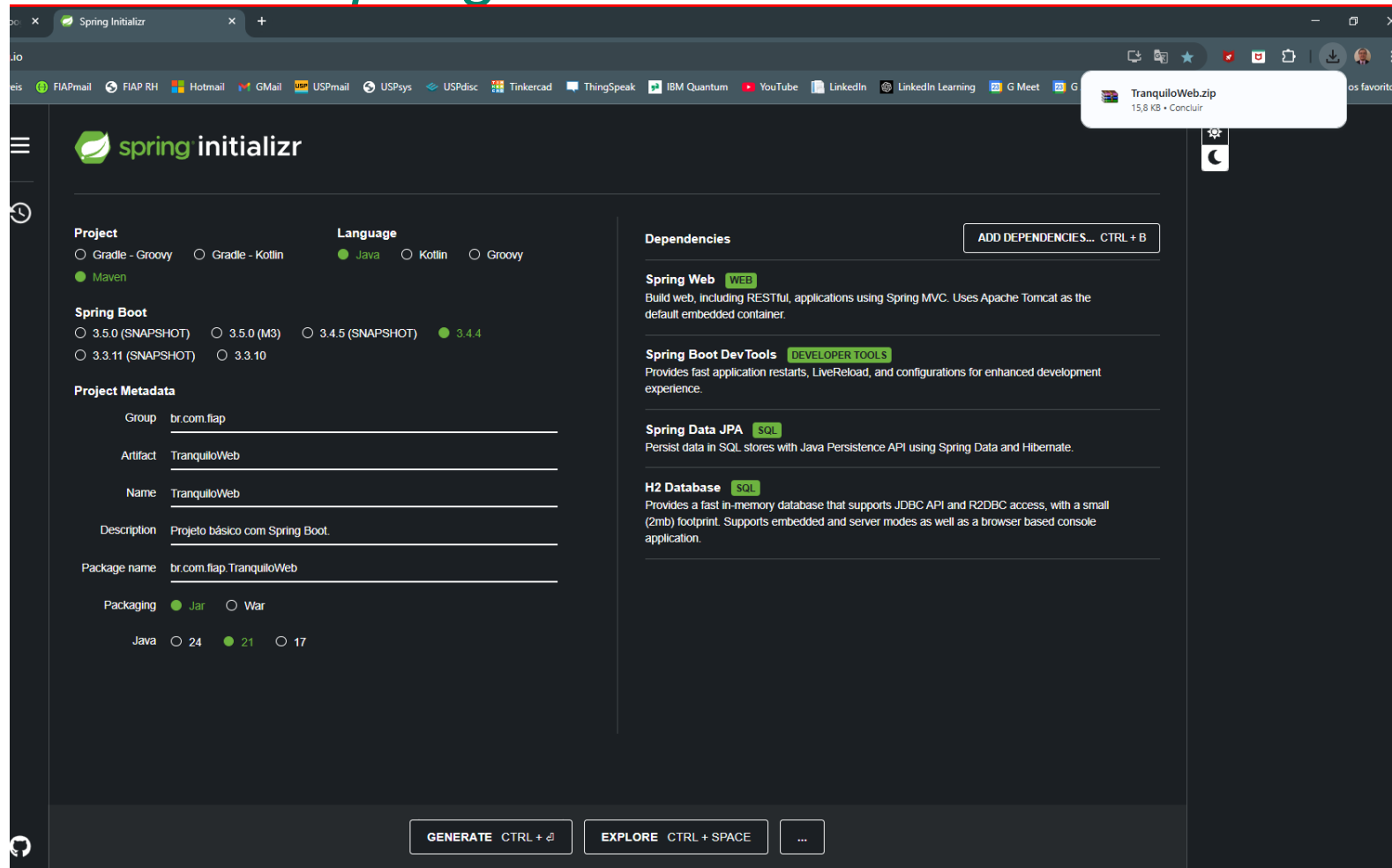
REST com IntelliJ – *Spring Initializr*

The screenshot displays the Spring Initializr web interface, which is used to generate a Spring project. The interface is divided into several sections:

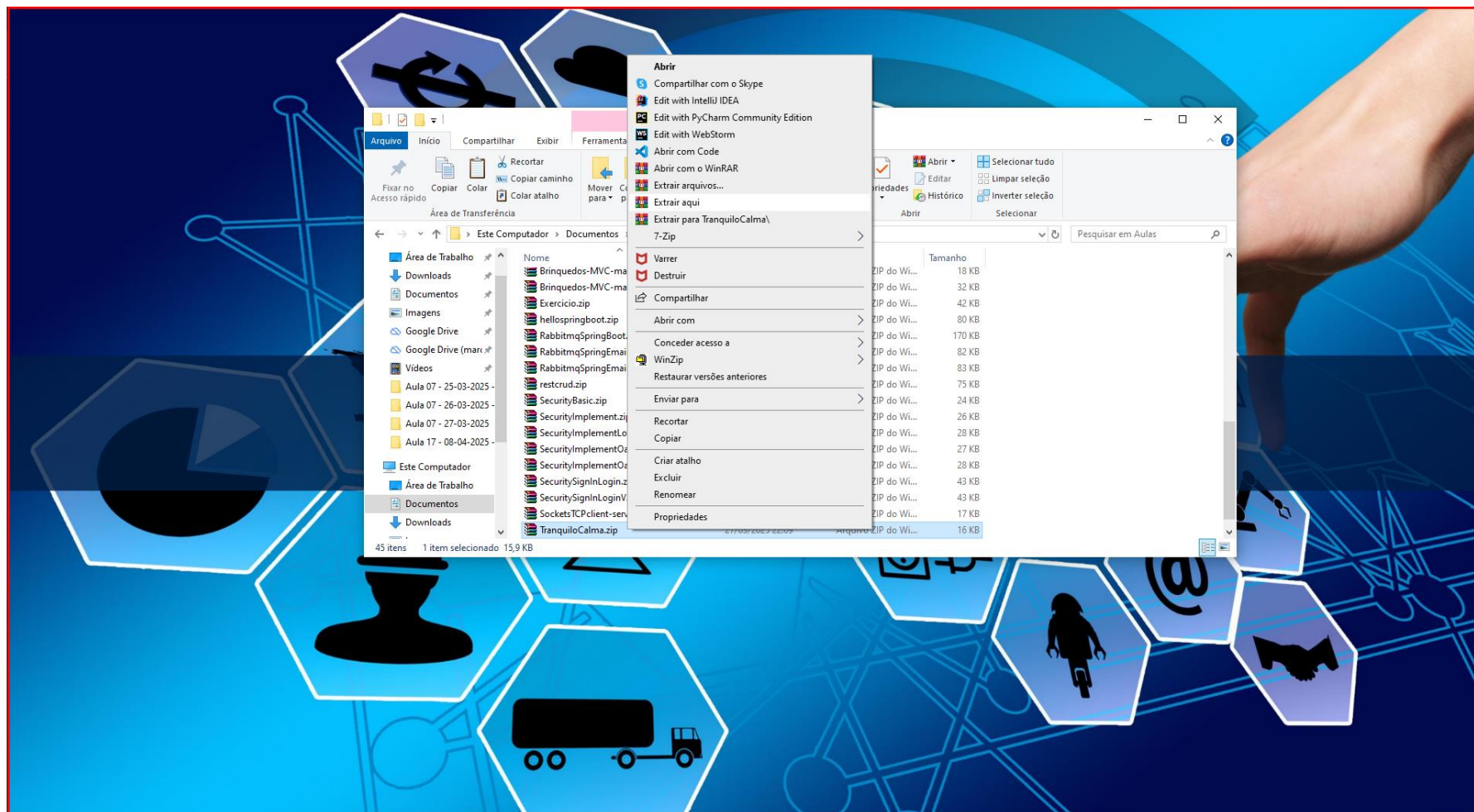
- Project:** Includes radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, **Java** (selected), **Kotlin**, and **Groovy**. There is also a **Maven** option.
- Spring Boot:** Includes radio buttons for **3.5.0 (SNAPSHOT)**, **3.5.0 (M3)**, **3.4.5 (SNAPSHOT)**, **3.4.4** (selected), and **3.3.11 (SNAPSHOT)**. There is also a **3.3.10** option.
- Project Metadata:** Includes input fields for **Group** (br.com.fiap), **Artifact** (TranquiloWeb), **Name** (TranquiloWeb), **Description** (Projeto básico com Spring Boot), and **Package name** (br.com.fiap.TranquiloWeb).
- Packaging:** Includes radio buttons for **Jar** (selected) and **War**.
- Language:** Includes radio buttons for **Java** (selected), **24**, **21** (selected), and **17**.
- Dependencies:** Includes a button **ADD DEPENDENCIES... CTRL + B** and a list of dependencies:
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - Spring Data JPA** (SQL): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
 - H2 Database** (SQL): Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

At the bottom, there are buttons for **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and a **...** button.

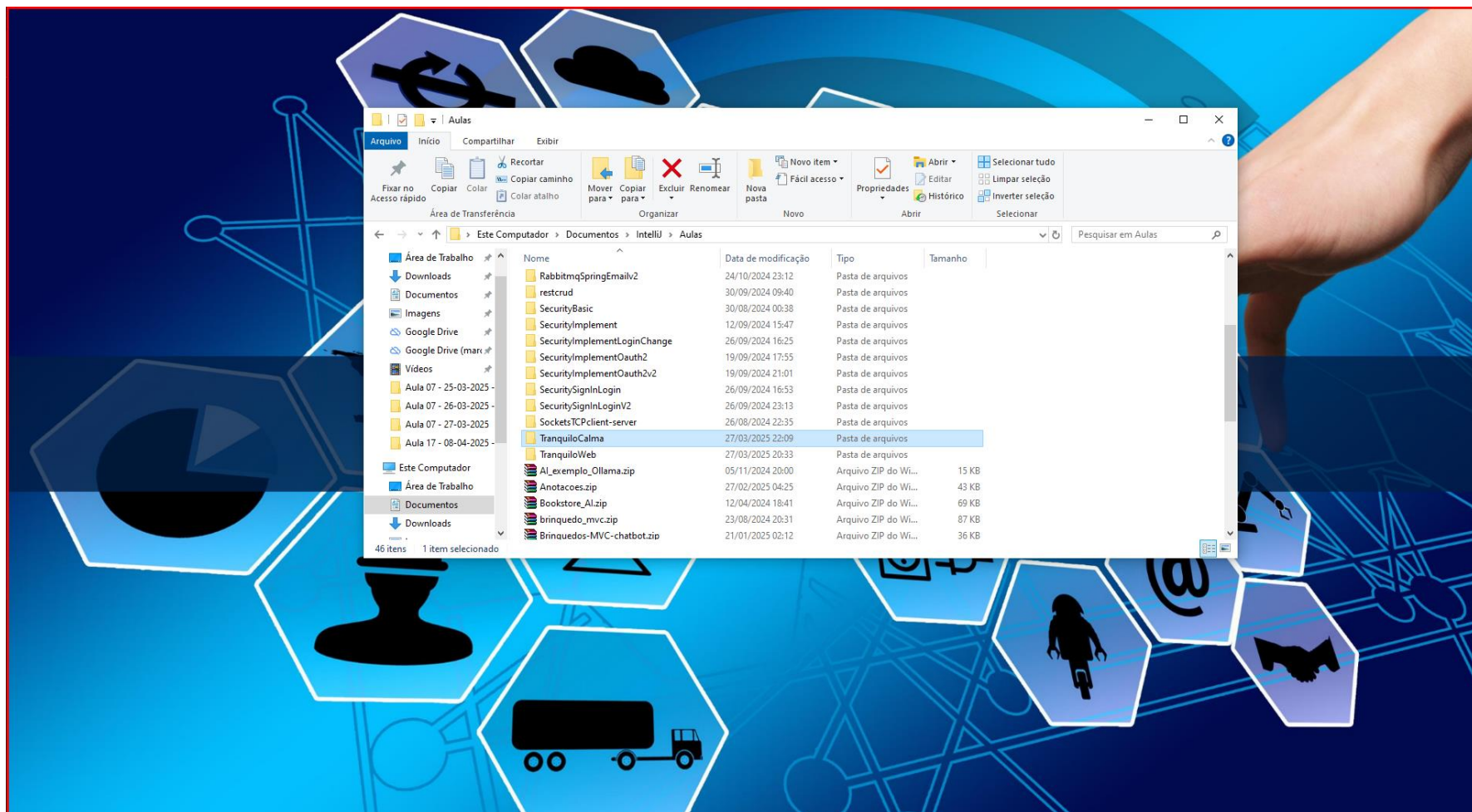
REST com IntelliJ – *Spring Initializr*



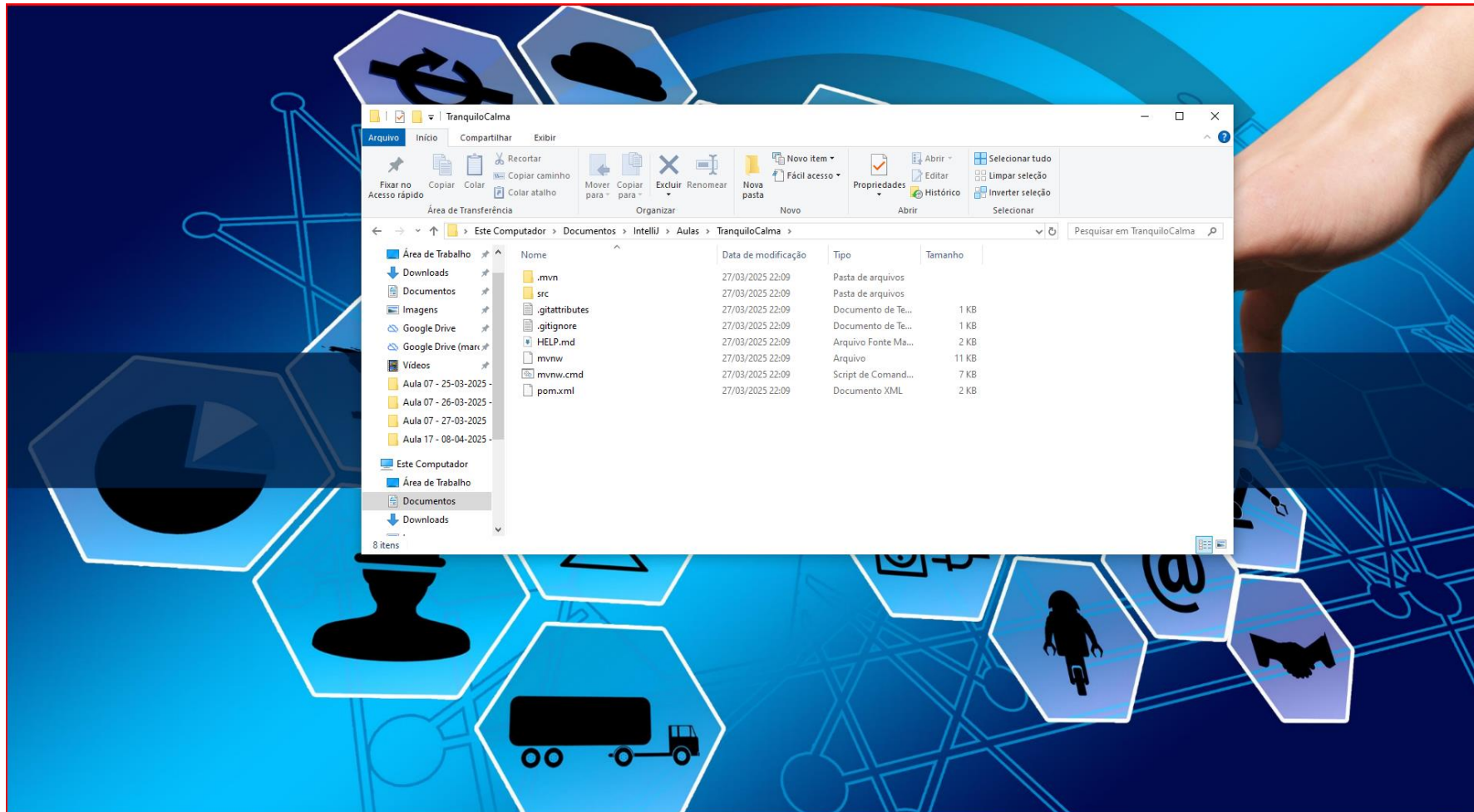
REST com IntelliJ – *Spring Initializr*



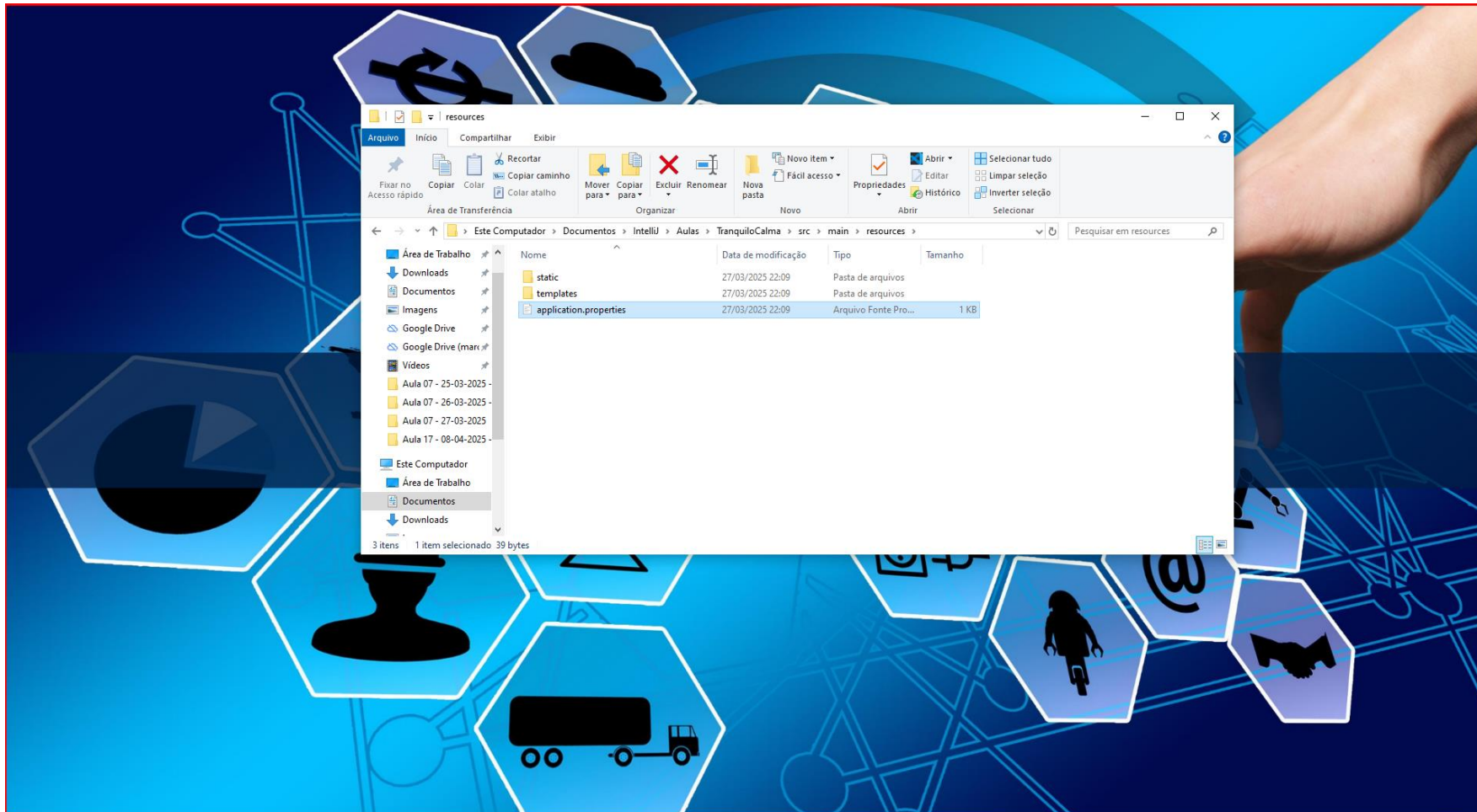
REST com IntelliJ – *Spring Initializr*



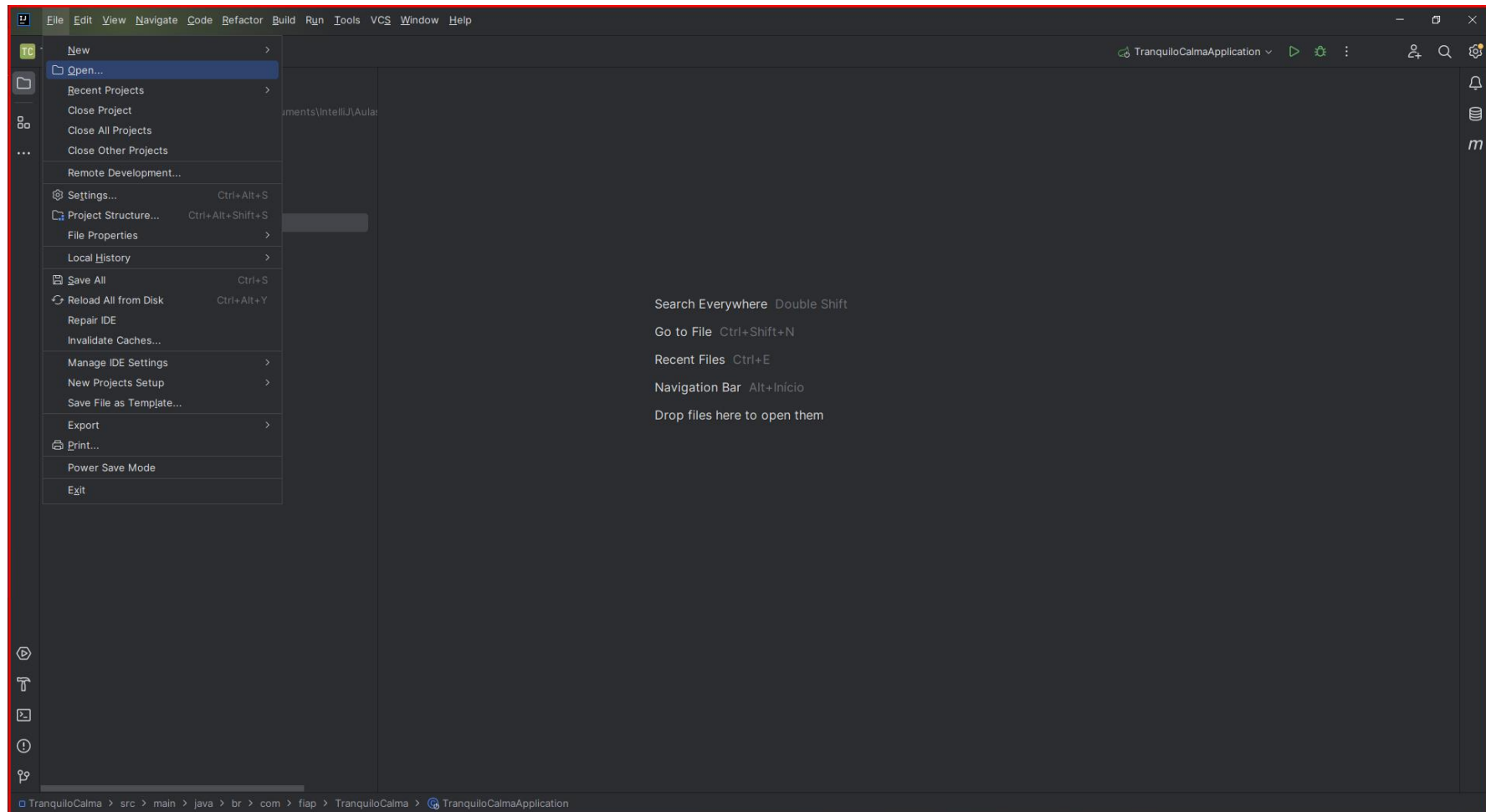
REST com IntelliJ – *Spring Initializr*



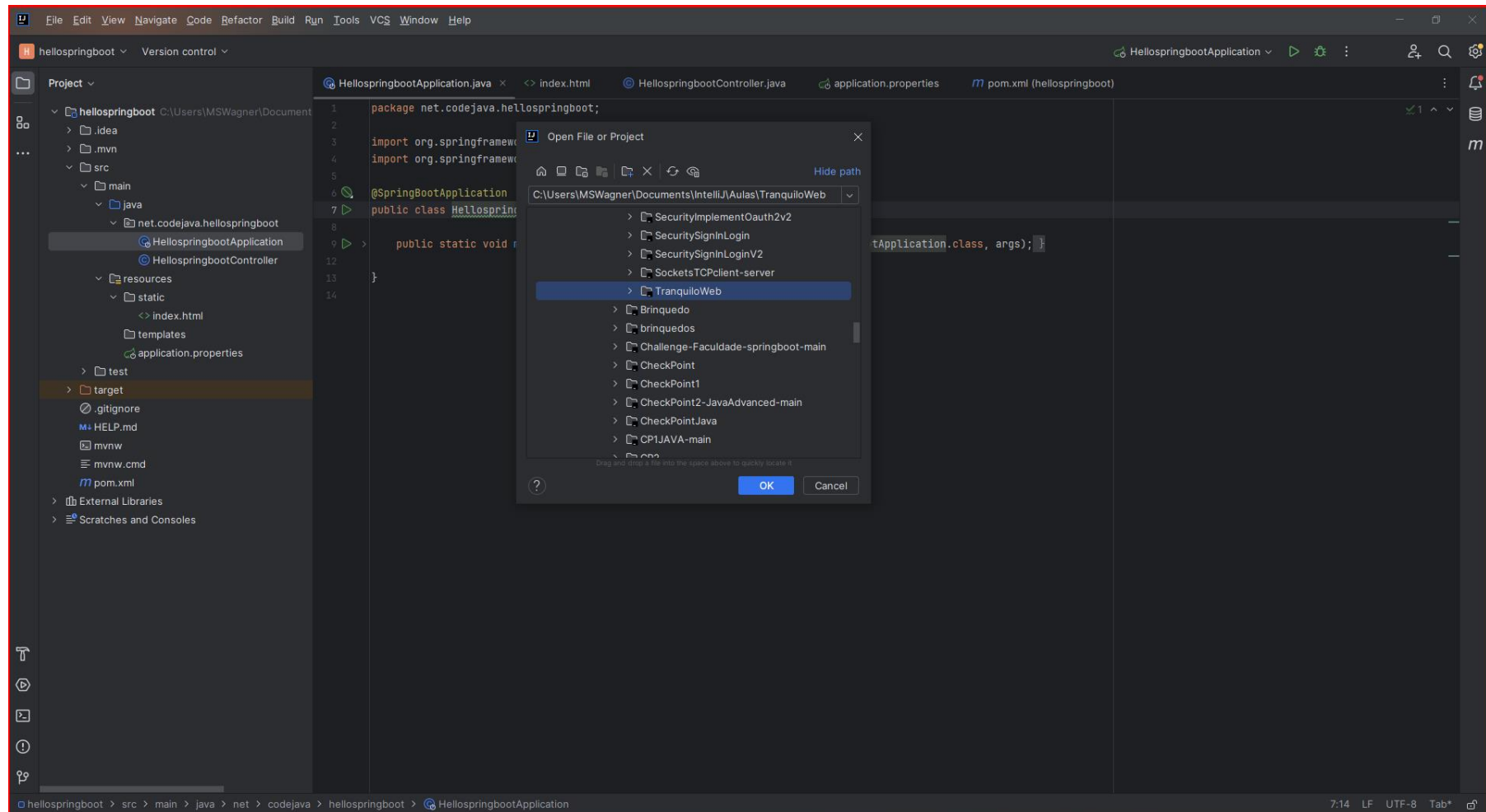
REST com IntelliJ – *Spring Initializr*



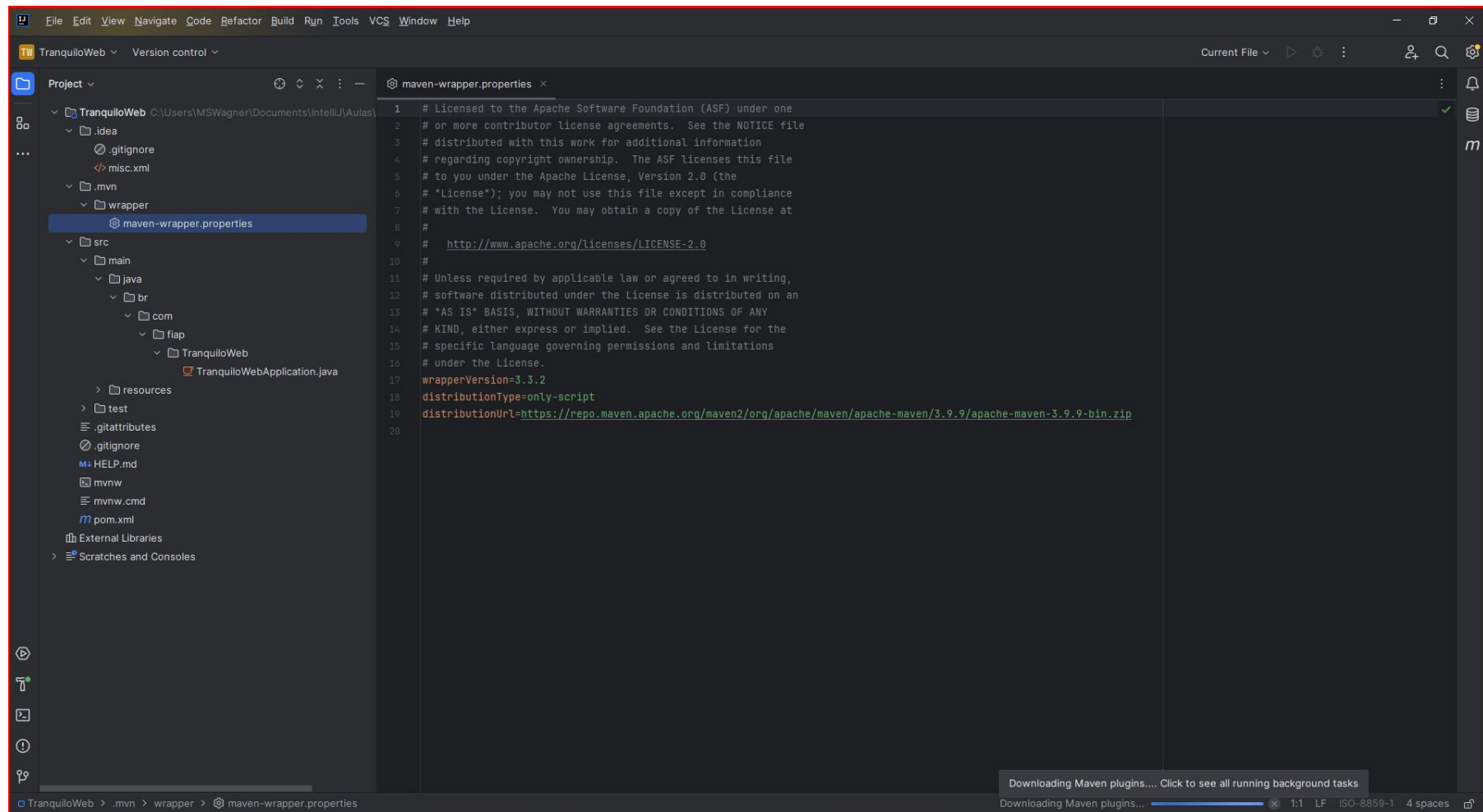
REST com IntelliJ – *Spring Initializr*



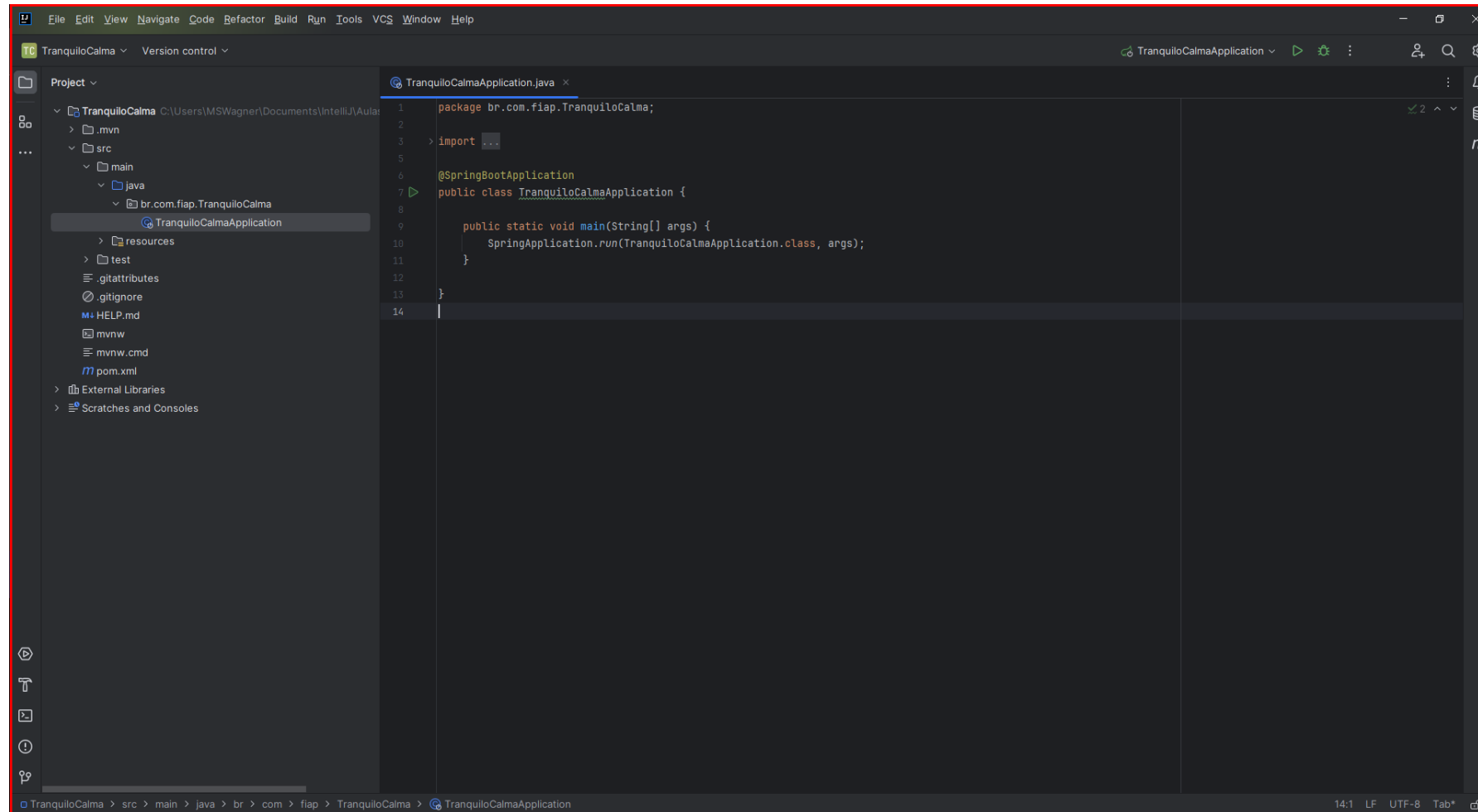
REST com IntelliJ



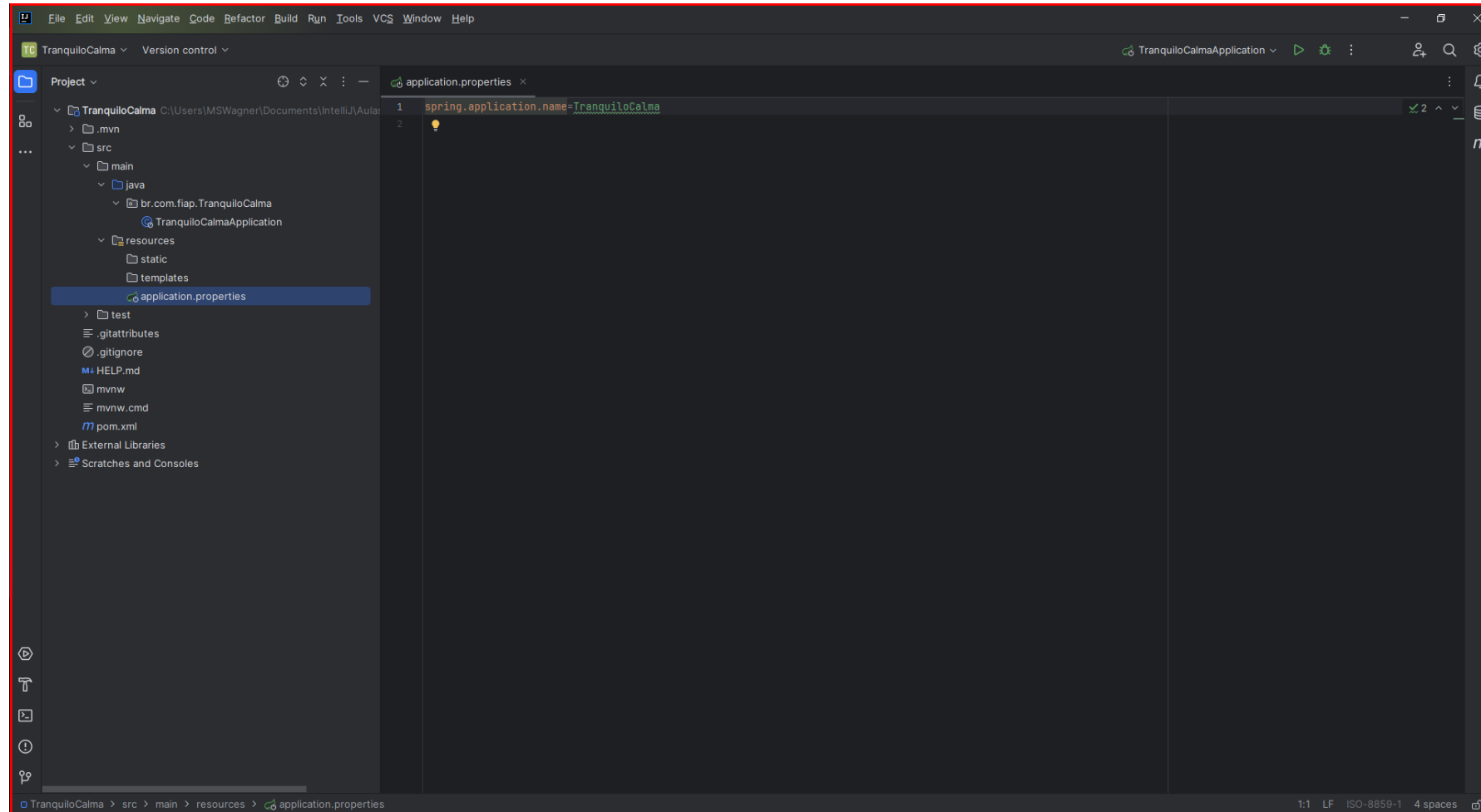
REST com IntelliJ



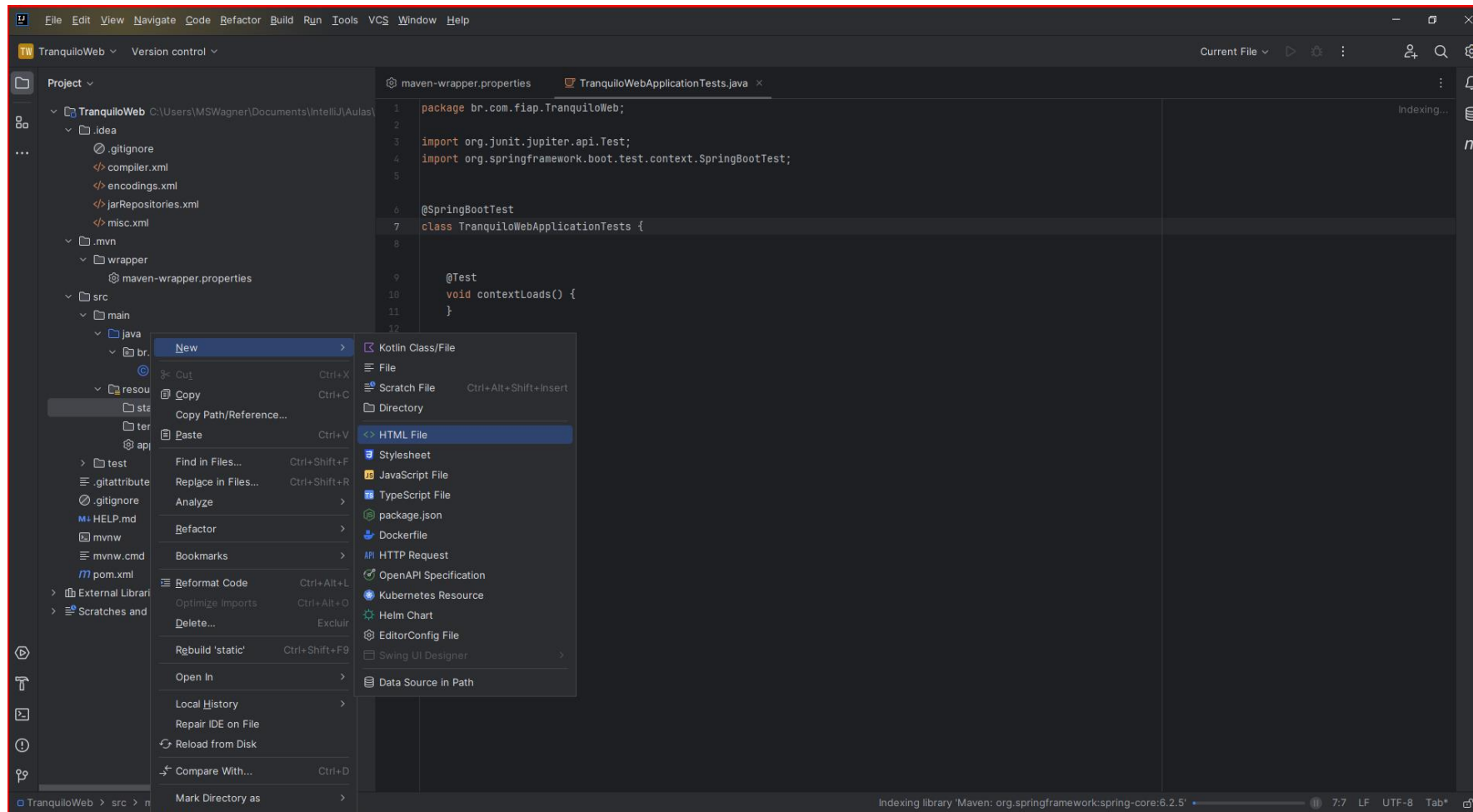
REST com IntelliJ



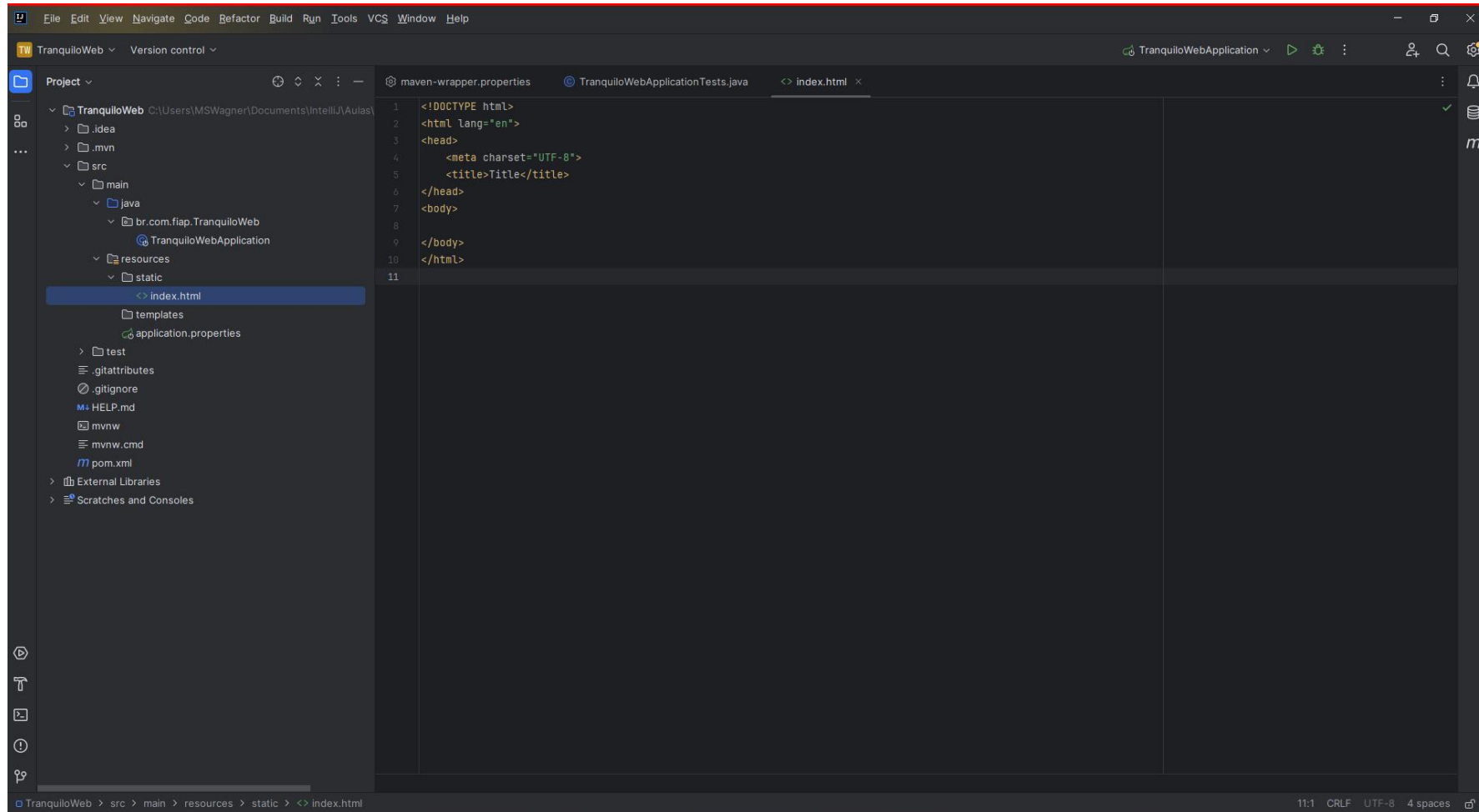
REST com IntelliJ



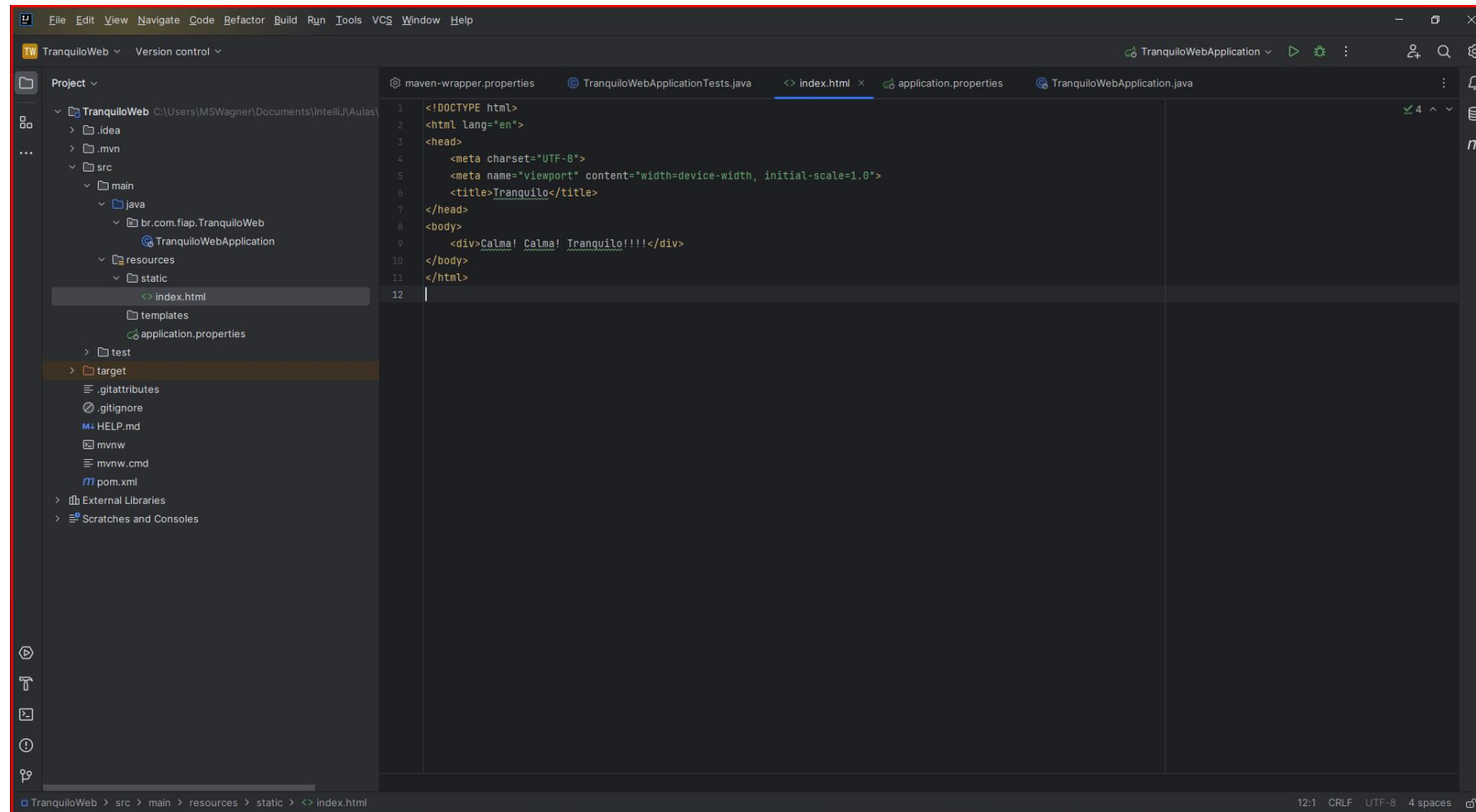
REST com IntelliJ



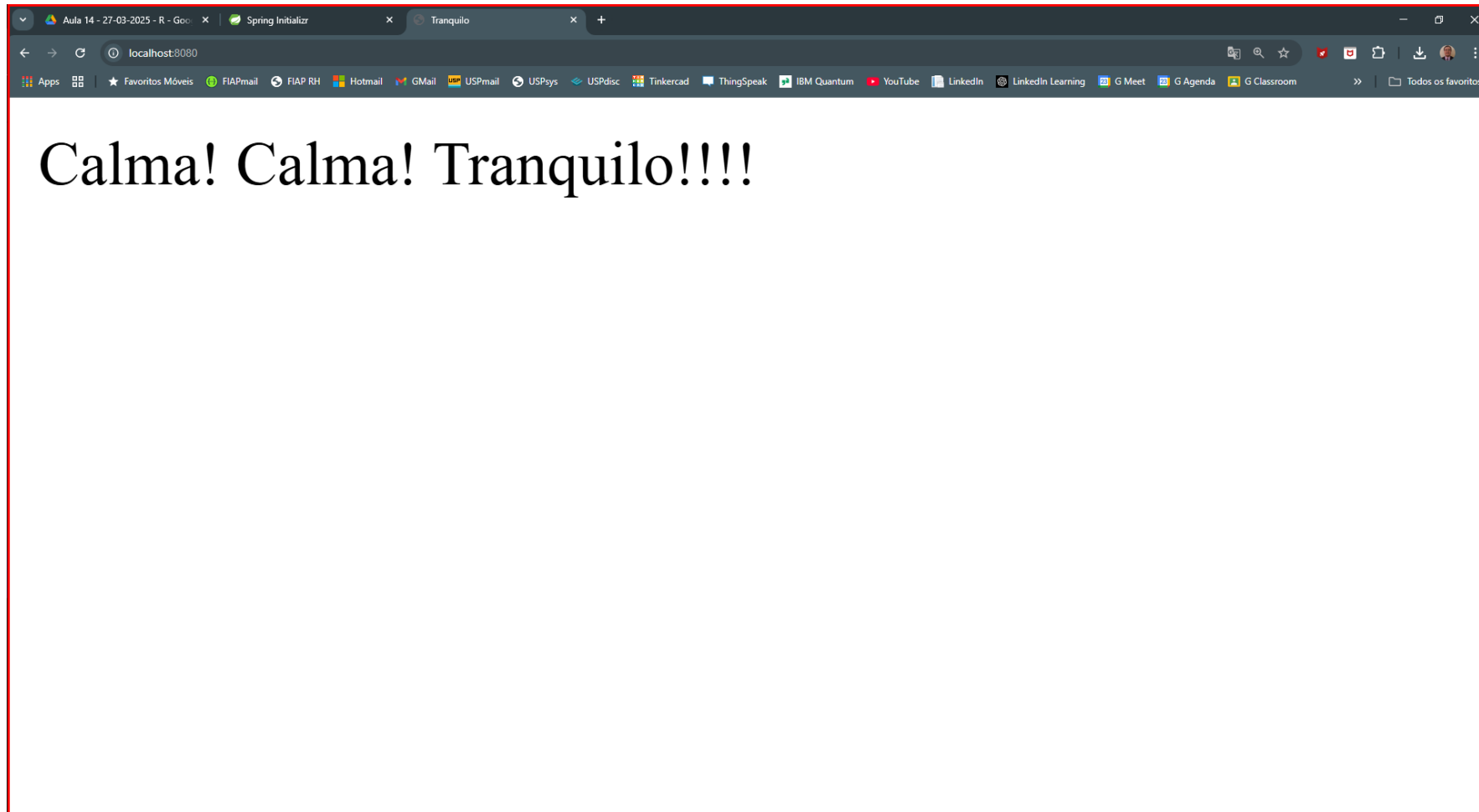
REST com IntelliJ



REST com IntelliJ



REST com IntelliJ



Exercício de REST

REST

- Exercício:

Criar uma API REST que faça o cadastro e consulta de nomes de usuários.

- Cadastro de usuários;
- Consulta de usuários;
- A princípio, sem uma conexão direta com Banco de Dados como MySQL, MongoDB, ou outro, fazendo uma interface simples com armazenamento básico.



Referências

George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. **Sistemas Distribuídos: Conceitos e Projeto**. Bookman Editora, 5 edition, 2013.

Harvey M Deitel, Paul J Deitel, David R Choffnes, et al. **Sistemas Operacionais**. Pearson/Prentice Hall, 3 edition, 2005.

Maarten Van Steen and A Tanenbaum. **Sistemas Distribuídos: Princípios e Paradigmas**. Pearson/Prentice Hall, 2 edition, 2007.

Harvey M Deitel and Paul J Deitel. **Java, como programar**. Ed. Pearson/Prentice Hall, 8 edition, 2010.

StackOverflow. Disponível em: <<https://pt.stackoverflow.com/>>. Acesso em: abril de 2023.

GAMMA, Erich et al. **Elements of Reusable Object-Oriented Software**. Design Patterns, 1995.

COOPER, James William. **Java design patterns: a tutorial**. 2000.

GUERRA, Eduardo. **Design Patterns com Java: Projeto orientado a objetos guiado por padrões**. Editora Casa do Código, 2014.

.....

Obrigado!

Agradecimento pela parceria e elaboração de materiais aos professores:
Prof. Me. Gustavo Torres Custódio
Prof. Thiago Yamamoto

.....

Contato: profmarcel.wagner@fiap.com.br

Cursos:

Tecnologia em Análise e Desenvolvimento de Sistemas (TDS)

Tecnologia em Defesa Cibernética (TDC)

Engenharia de Software (ES)

