



Spring Boot

Fabício Tonetto Londero

Spring Boot

- É um framework que surgiu para facilitar a criação de aplicações Java
- Fornece tudo que você precisa para adotar a linguagem Java em um ambiente corporativo



IO EXECUTION



Spring XD



Boot



Grails

IO FOUNDATION



Integration



Batch



Big Data



Web



Relational

DATA



Non-Relational

CORE



Framework



Security



Groovy



Reactor



Spring Security

- Spring Security é uma estrutura Java que fornece autenticação, autorização e outros recursos de segurança para aplicativos corporativos.



Spring batch

- Spring Batch é uma estrutura de código aberto para processamento em lote .
- É uma solução leve e abrangente projetada para permitir o desenvolvimento de aplicações em lote robustas.



Criando um projeto

1º passo

Acessar o site oficial
para criarmos um
projeto Spring:

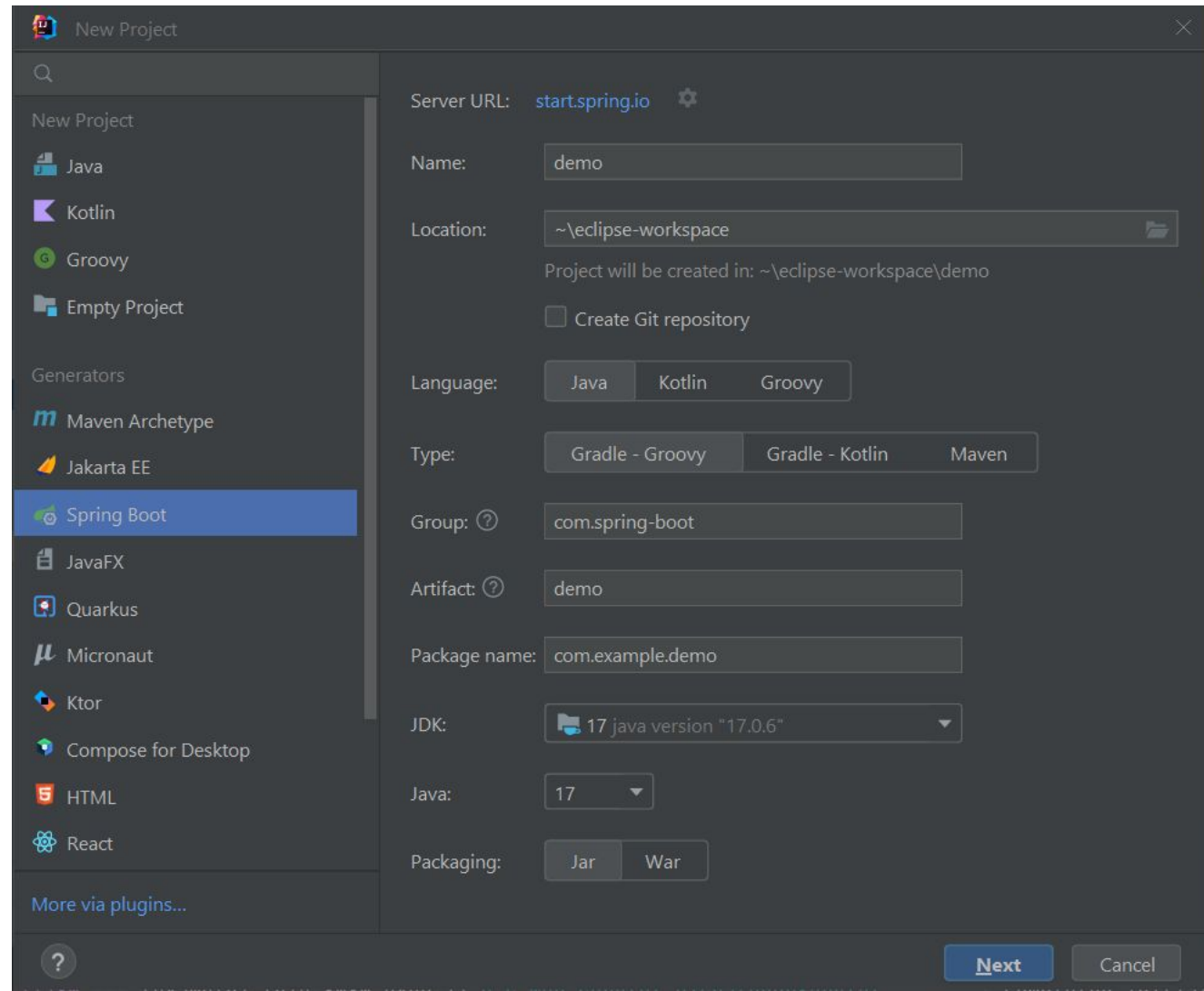
<https://start.spring.io/>

Configurando o Projeto

De forma análoga, é possível configurar o nosso projeto Spring Boot, dentro do próprio IntelliJ.

Basta acessar:

File->New->Project->Spring Boot



Configurando o Projeto

Precisamos configurar o nome projeto, pacotes e nossas dependências. As dependências utilizadas nesse exemplo são:

- Spring Web

Disponibiliza diversos recursos para a nossa controller. Podemos dizer que é uma evolução de um servlet Java.

- Thymeleaf

É um framework que disponibiliza recursos avançados para integração de templates HTML com Spring Boot.

**Project**

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy
☒ Maven

Language**Spring Boot**

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M1) ☐ 3.3.3 (SNAPSHOT) ☒ 3.3.2
☐ 3.2.9 (SNAPSHOT) ☐ 3.2.8

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 22 ☐ 21 ☒ 17

Dependencies[ADD DEPENDENCIES... CTRL + B](#)**Spring Web** WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.


Thymeleaf TEMPLATE ENGINES


A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.


[GENERATE CTRL + ↵](#)[EXPLORE CTRL + SPACE](#)[SHARE...](#)


Explore - (CTRL + ESPAÇO)


demo.zip


 .gitignore


 .mvn

 HELP.md

 mvnw

 mvnw.cmd

 pom.xml

 src

1 <?xml version="1.0" encoding="UTF-8"?>

2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

3 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

4 <modelVersion>4.0.0</modelVersion>

5 <parent>

6 <groupId>org.springframework.boot</groupId>

7 <artifactId>spring-boot-starter-parent</artifactId>

8 <version>3.3.2</version>

9 <relativePath/> <!-- Lookup parent from repository -->

10 </parent>

11 <groupId>com.spring-boot.example</groupId>

12 <artifactId>demo</artifactId>

13 <version>0.0.1-SNAPSHOT</version>

14 <name>spring-example</name>

15 <description>Primeiro Projeto com Spring Boot</description>

16 <url/>

17 <licenses>

18 <license/>

19 </licenses>

20 <developers>

21 <developer/>

22 </developers>

23 <scm>

24 <connection/>

25 <developerConnection/>

26 <tag/>

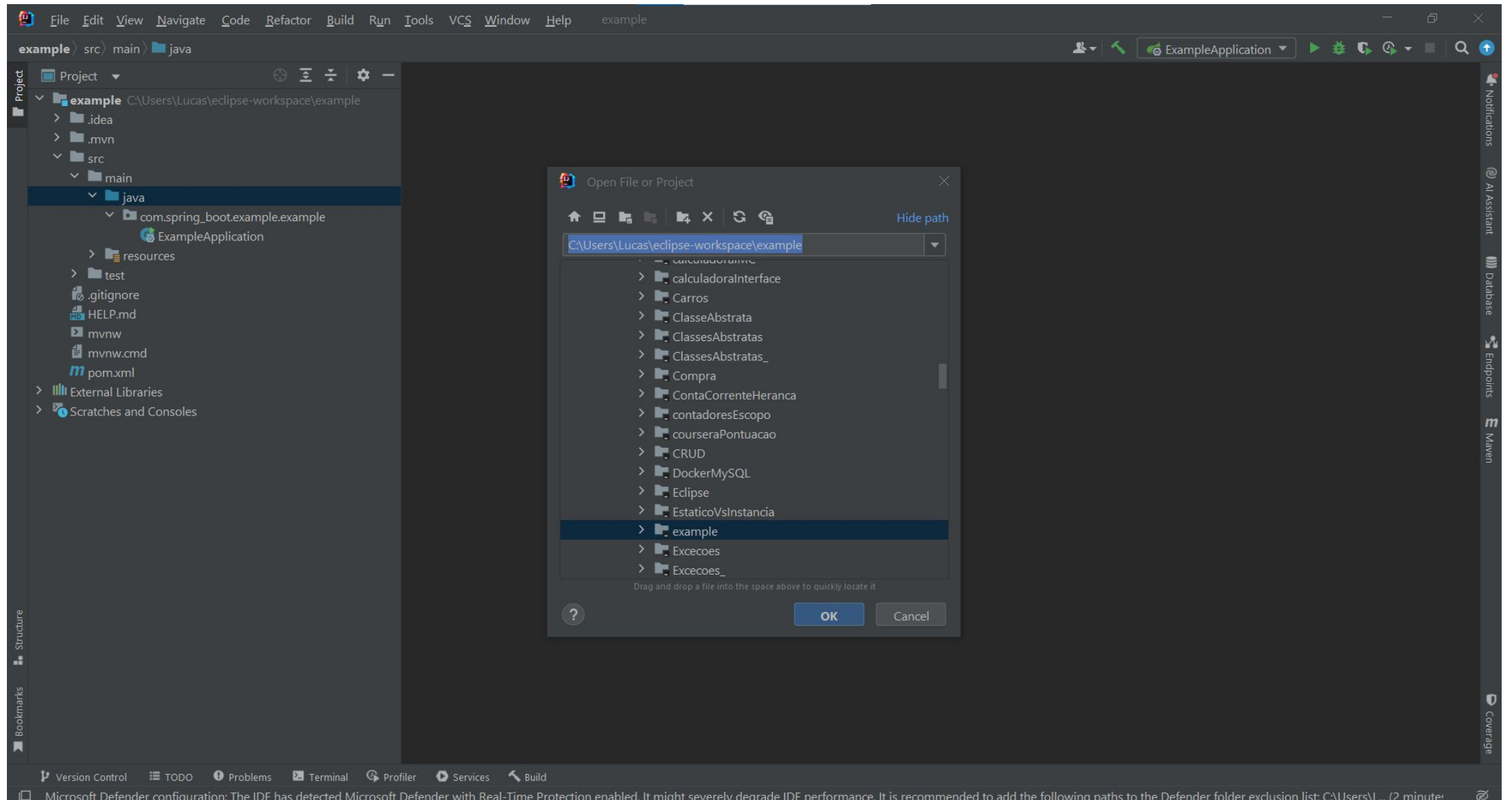
27 <url/>

28 </scm>

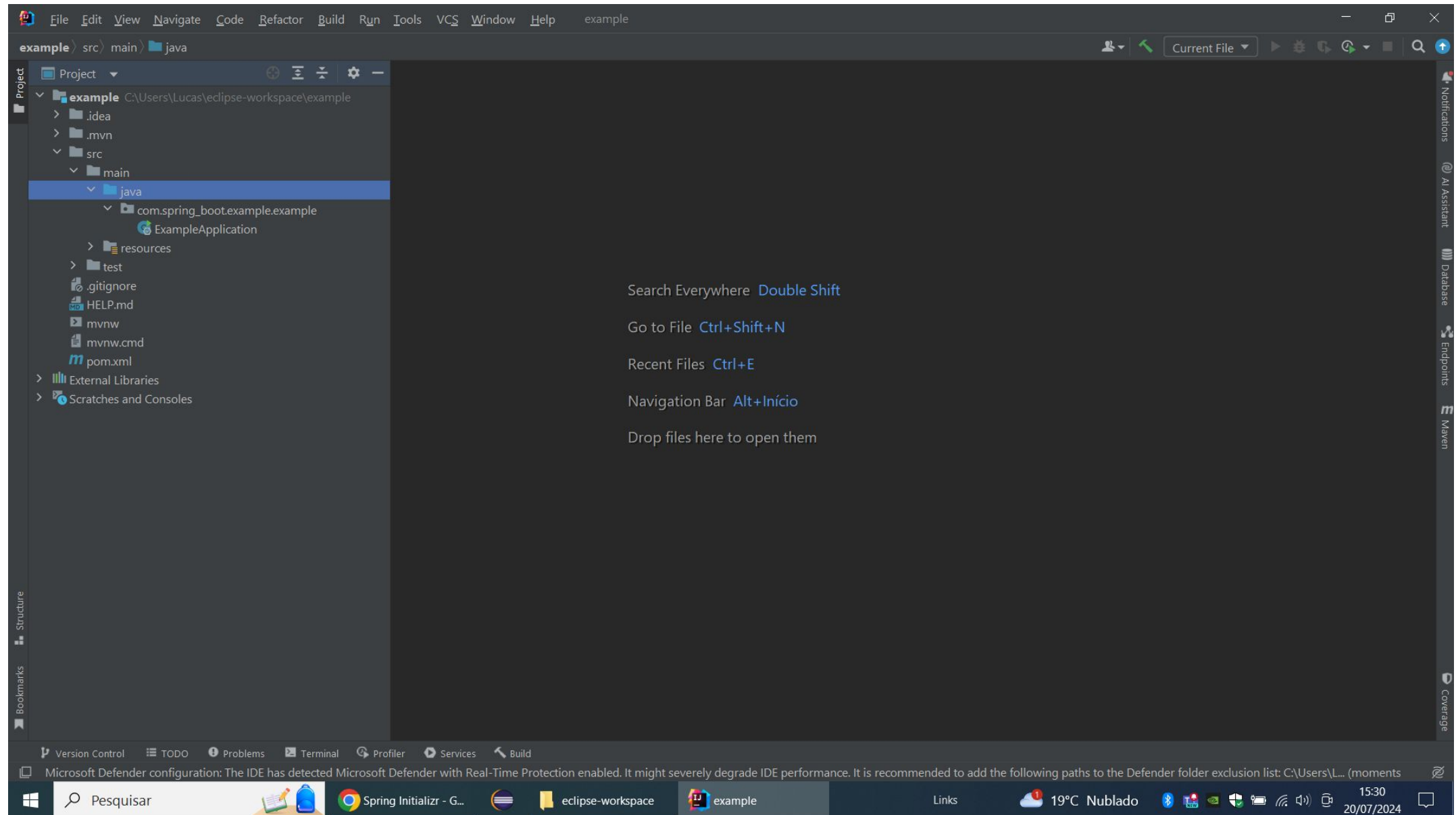
DOWNLOAD COPY

DOWNLOAD CTRL + ↵ CLOSE ESC

Extrair e Abrir o Projeto



Importando o Projeto Pré Configurado



Importando o Projeto Pré Configurado

Criar um pacote de nome Controller.

Anotar a classe controller com @Controller.

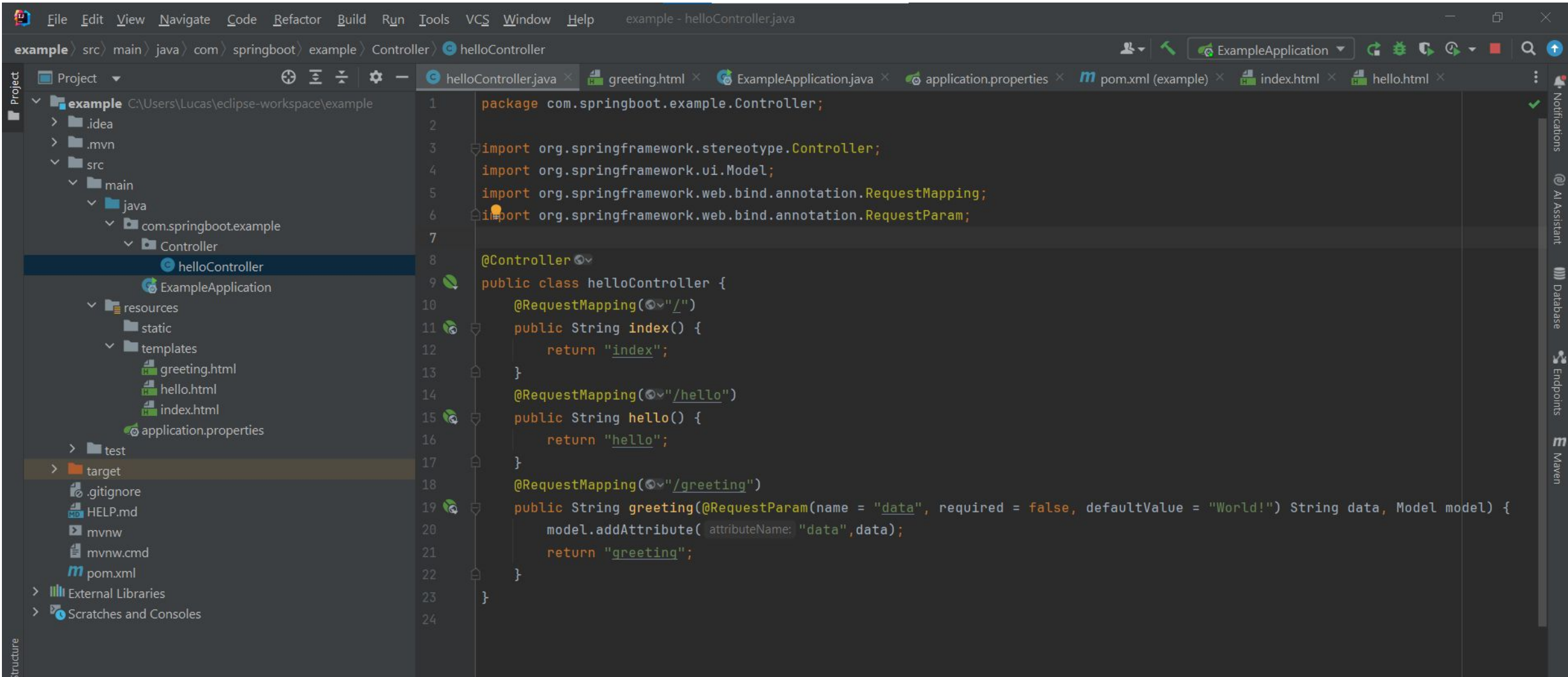
Assim indicamos para o Spring Boot, que essa é uma classe que atenderá a requisições HTTP, também através de anotações.

Nota-se também que o método retorna uma String, cujo valor corresponde a *view* que desejamos mostrar.

Ex.:

```
@Controller
public class helloController {
    @RequestMapping ("/")
    public String index() {
        return "index";
    }
}
```

Criando um Controller

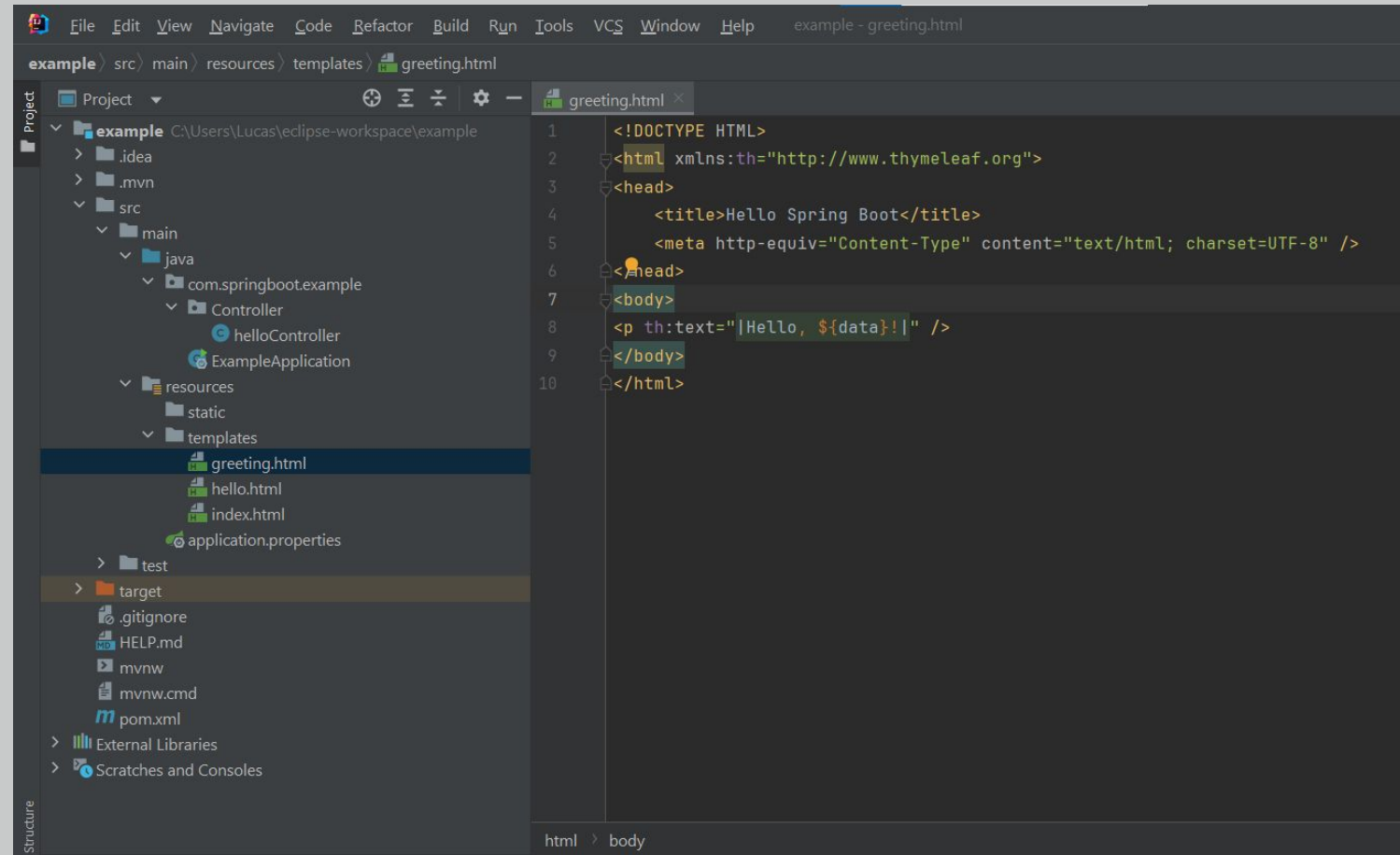


The screenshot shows an IDE window with the following components:

- Project Explorer (Left):** Displays the project structure. The path `example > src > main > java > com.springboot.example > Controller` is selected, and the `helloController` file is highlighted.
- Editor (Center):** Shows the code for `helloController.java`. The code includes package declarations, imports for Spring Framework annotations and classes, and the implementation of the `helloController` class with three methods: `index()`, `hello()`, and `greeting()`.
- Run and Debug Console (Top Right):** Shows the current configuration as `ExampleApplication`.
- Toolbars (Right):** Includes icons for Notifications, AI Assistant, Database, Endpoints, and Maven.

```
1 package com.springboot.example.Controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RequestParam;
7
8 @Controller
9 public class helloController {
10     @RequestMapping("/")
11     public String index() {
12         return "index";
13     }
14     @RequestMapping("/hello")
15     public String hello() {
16         return "hello";
17     }
18     @RequestMapping("/greeting")
19     public String greeting(@RequestParam(name = "data", required = false, defaultValue = "World!") String data, Model model) {
20         model.addAttribute("data", data);
21         return "greeting";
22     }
23 }
24
```

- Na pasta Resources/templates, crie o arquivos:
 - index.html
 - hello.html
 - greeting.html
- Crie o código html básico da página
- Teste as rotas no navegador



Código greeting.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Hello Spring Boot</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p th:text="|Hello, ${data}!|" />
</body>
</html>
```

Executando

- Em um navegador, abra “localhost:8080”
- A controller atenderá às requisições mapeada em nossa aplicação:
 - /
 - /hello
 - /greeting

O mapeamento “greeting” está configurado para receber um parâmetro de nome data:

Ex.:

`http://localhost:8080/greeting?data=John`

Exercício

Crie uma aplicação com Spring Boot e Thymeleaf, que receba o nome de uma pessoa como parâmetro e o exiba em uma página (user.html).

- “Olá *nome*! Seja bem-vindo!”

Caso não seja informado nenhum nome, deverá ser exibida a página (error.html) com a seguinte mensagem:

- “Erro ao exibir informações de Usuário.”

Adicionalmente crie uma página (new.html) e um formulário para submeter o nome para a controller.

Dicas

O Thymeleaf se difere um pouco, em relação ao que foi trabalhado com Servlets.

Esse formulário submete ao endpoint “/user” com o método “post” o parâmetro name.

Pesquise mais sobre na documentação do [Thymeleaf](http://www.thymeleaf.org)

Ex.:

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="UTF-8">

    <title>Submeter um novo usuário</title>

</head>

<body>

<h1>Novo usuário</h1>

<form action="#" th:action="@{/user}" th:method="post">

    <input type="text" th:name="name">

    <input type="submit" value="Adicionar">

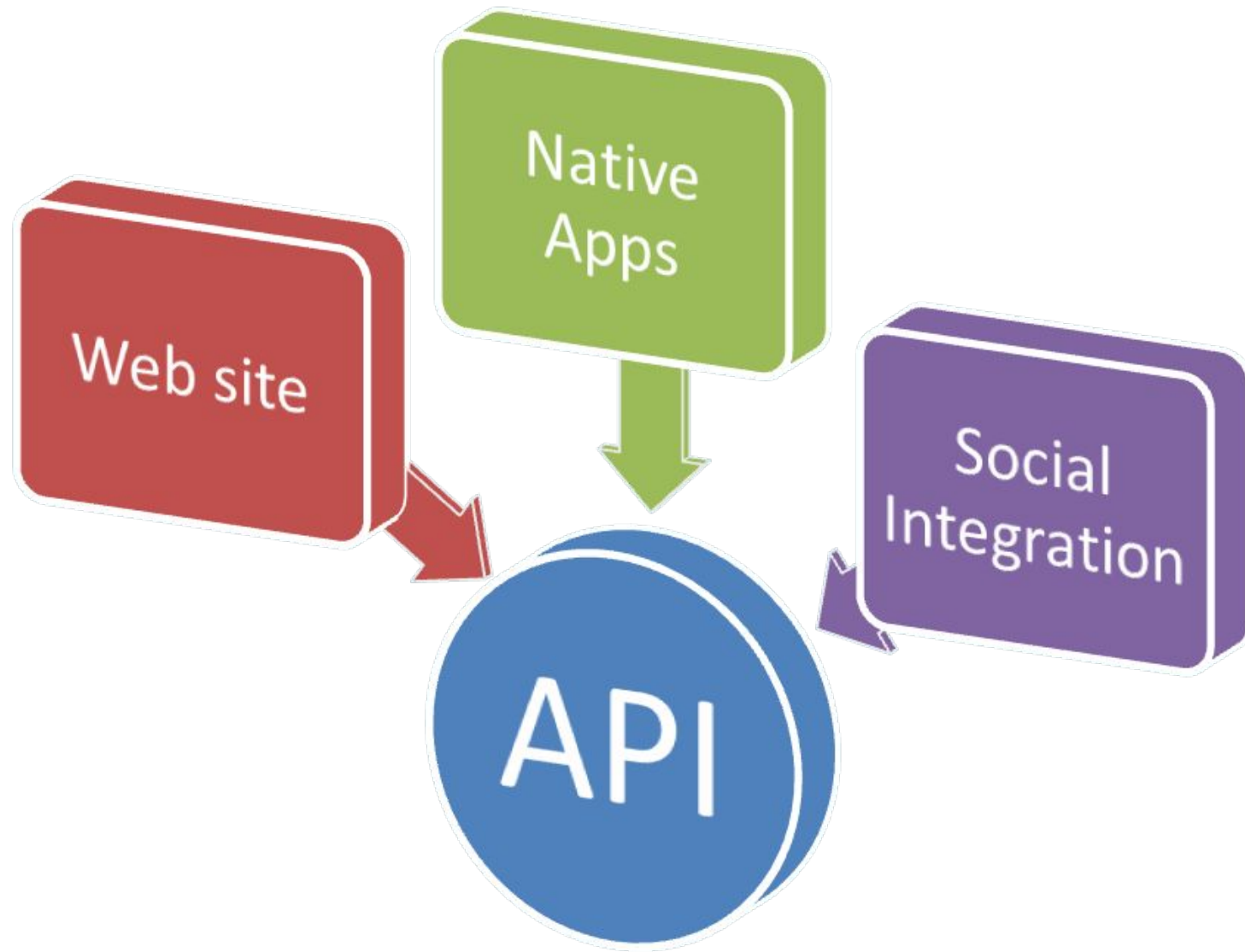
</form>

</body>

</html>
```


API, REST e RESTful

- Por vezes, diferentes aplicativos em diferentes plataformas precisam acessar os mesmos conjuntos de dados
 - Uma alternativa para a não reimplementação para diferentes plataformas, é a utilização de APIs
- Uma API é um conjunto de definições e protocolos para construir e integrar software de aplicativo.
- Podemos pensar em uma API como um mediador entre os usuários ou clientes e os recursos ou serviços da web que desejam obter
- *Application Programming Interface* (Em português, significa Interface de Programação de Aplicações)



API

- Desta forma, entendemos que as APIs permitem uma interoperabilidade entre aplicações.
- Em outras palavras, a comunicação entre aplicações e entre os usuários.
- Também pode aparecer com o nome WebServices

Representações

- Existem atualmente, três formas de representação
 - XML
 - JSON
 - YAML

Representação XML

```
1 <endereco>
2   <rua>
3     Rua Recife
4   </rua>
5   <cidade>
6     Paulo Afonso
7   </cidade>
8 </endereco>
```

Representação JSON

```
1 { endereco:
2   {
3     rua: Rua Recife,
4     cidade: Paulo Afonso
5   }
6 }
```

Representação YAML

```
1 endereco:
2 rua: rua Recife
3 cidade: Paulo Afonso
```


REST

- API REST é uma forma de implementação de APIs que seguem algumas restrições arquitetônicas.
- REST significa **Representational State Transfer**. Em português, **Transferência de Estado Representacional**
- Uma API transfere uma representação do estado do recurso para o solicitante ou terminal. Essas informações, ou representação, são fornecidas em um dos vários formatos via HTTP (json, xml...)
- cabeçalhos e parâmetros também são importantes nos métodos HTTP de uma solicitação HTTP da API REST
- Existem cabeçalhos de solicitação e cabeçalhos de resposta, cada um com suas próprias informações de conexão HTTP e códigos de status.

RESTful



É apenas uma API que implementa todas as “regras” de uma API REST.



sistemas que utilizam os princípios REST são chamados de RESTful

RESTful - critérios

Uma arquitetura cliente-servidor composta de clientes, servidores e recursos, com solicitações gerenciadas por meio de **HTTP**.

Comunicação ***Stateless*** entre cliente-servidor, ou seja, as informações do cliente não são armazenadas nas solicitações GET e cada pedido é separado e desconectado.

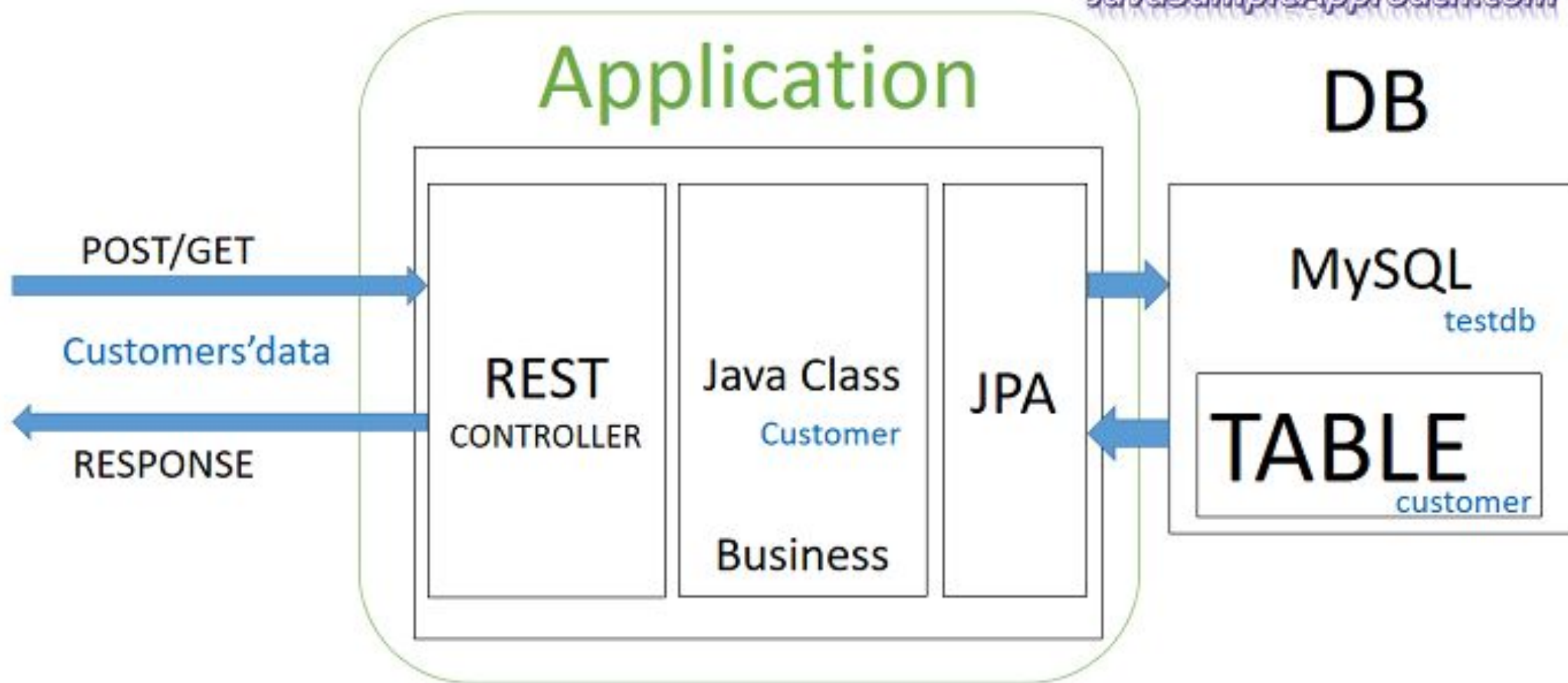
Dados armazenáveis em cache que otimizam as interações cliente-servidor.

informações transferidas em um formato padrão

os recursos solicitados são identificáveis e separados das representações enviadas ao cliente, permitindo que o cliente manipule as informações e recursos

RESTful - critérios

- Informação disponível ao cliente, de como acessar os demais recursos(links) da API
- Um sistema em camadas que organiza cada tipo de servidor (os responsáveis pela segurança, balanceamento de carga, etc.) envolvia a recuperação das informações solicitadas em hierarquias, invisíveis para o cliente.
- Code-on-demand (opcional): a capacidade de enviar código executável do servidor para o cliente quando solicitado, estendendo a funcionalidade do cliente.



HTTP

- O Hypertext Transfer Protocol, sigla HTTP (em português Protocolo de Transferência de Hipertexto) é um protocolo de comunicação.
- é a base para a comunicação de dados da World Wide Web
- O protocolo HTTP faz a comunicação entre o cliente e o servidor por meio de mensagens.
 - O cliente envia uma mensagem de requisição de um recurso e o servidor envia uma mensagem de resposta ao cliente com a solicitação.
- Uma mensagem, tanto de requisição quanto de resposta, é composta por uma linha inicial, nenhuma ou mais linhas de cabeçalhos, uma linha em branco obrigatória finalizando o cabeçalho e por fim o corpo da mensagem, opcional em determinados casos.

Cabeçalho

O cabeçalho da mensagem (header) é utilizado para transmitir informações adicionais entre o cliente e o servidor.

Existem quatro tipos de cabeçalhos que poderão ser incluídos na mensagem os quais são:
general-header, request-header, response-header e entity-header.

Corpo

- . Em uma mensagem de resposta, o corpo da mensagem é o recurso que foi requisitado pelo cliente, ou ainda uma mensagem de erro, caso este recurso não seja possível.

| Exemplo | Descrição |
|-------------------------------|---|
| text/plain | Arquivo no formato texto (ASCII) |
| text/html | Arquivo no formato HTML , utilizado como padrão para documentos Web |
| Image/gif | Imagem com o formato GIF |
| Image/jpeg | Imagem com o formato JPEG |
| application/zip | Arquivo compactado |
| application/json | Arquivo no formato JSON |
| application/xml (ou text/xml) | Arquivo no formato XML |

Métodos HTTP

- O protocolo HTTP define oito métodos que indicam a ação a ser realizada no recurso especificado. Também são conhecidos como Verbos HTTP.
 - GET
 - HEAD
 - POST
 - PUT
 - DELETE
 - TRACE
 - OPTIONS
 - CONNECT

Métodos HTTP em API RESTFul

- Os métodos HTTP mais comuns e utilizados são:
 - POST, GET, PUT, PATCH e DELETE
 - Eles correspondem às operações de criação, leitura, atualização e exclusão (ou CRUD), respectivamente.

| HTTP Verb | CRUD | Entire Collection (e.g. /customers) | Specific Item (e.g. /customers/{id}) |
|-----------|----------------|--|--|
| POST | Create | 201 (Created), 'Location' header with link to /customers/{id} containing new ID. | 404 (Not Found), 409 (Conflict) if resource already exists.. |
| GET | Read | 200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists. | 200 (OK), single customer. 404 (Not Found), if ID not found or invalid. |
| PUT | Update/Replace | 405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection. | 200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid. |
| PATCH | Update/Modify | 405 (Method Not Allowed), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid. |
| DELETE | Delete | 405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable. | 200 (OK). 404 (Not Found), if ID not found or invalid. |

Referências

- <https://becode.com.br/o-que-e-api-rest-e-restful/>
- <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- <https://www.toptal.com/spring/spring-boot-oauth2-jwt-rest-protection>
- <https://spring.io/guides/#getting-started-guides>
- <https://www.toptal.com/spring/spring-boot-oauth2-jwt-rest-protection>
- <https://betterprogramming.pub/secure-a-spring-boot-rest-api-with-json-web-token-reference-to-angular-integration-e57a25806c50>
- <https://www.restapitutorial.com/lessons/httpmethods.html#:~:text=The%20primary%20or%20most%2Dcommonly,but%20are%20utilized%20less%20frequently.>