



## Java para WEB

### Bootcamp Banco Pan – Desenvolvedor Júnior

Alcides Wenner

2021

## **Java para WEB**

### **Bootcamp Banco Pan – Desenvolvedor Júnior**

**Alcides Wenner Ferreira Bastos**

© Copyright do Instituto de Gestão e Tecnologia da Informação.

Todos os direitos reservados.

## Sumário

---

<b>Capítulo 1. Introdução</b>	<b>4</b>
Introdução ao Spring Boot	4
Instalação do Ambiente de Desenvolvimento	5
Variáveis de Ambiente	5
IDE SPRING TOOLS4	6
Postman	8
Instalação do MySQL Workbench	9
GIT E GIT HUB	9
Deplor da Aplicação no Heroku	12
 <b>Capítulo 2. Anotações do Spring Boot</b>	 <b>14</b>
Anotação Lombok – Menos código e o mesmo resultado	18
 <b>Capítulo 3. Banco de dados Mysql no projeto Spring</b>	 <b>20</b>
Criando a entidade JPA	21
Criando os repositórios	21
Criando os controladores	23
 <b>Capítulo 4. Arquitetura do projeto – Padrão de camadas</b>	 <b>25</b>
Exemplo prático do padrão de camadas em um projeto	26
O que são classes	
DTO.....	28

<b>Referências.....</b>	<b>32</b>
-------------------------	-----------

## Capítulo 1. Introdução

---

Este capítulo dá uma introdução ao módulo de Java Web e descreve como instalar as ferramentas de desenvolvimento do framework Spring Boot. Além disso alguns conceitos importantes serão abordados.

### Introdução ao Spring Boot

---

O Spring Boot é um framework bastante usado em aplicações web, e o objetivo da criação desse framework era substituir a versão Enterprise do JAVA EE visando facilitar a produtividade no desenvolvimento web.

O Spring favorece a criação de projetos em um curto período de tempo, em virtude do recurso de Autoconfiguração, possibilitando a declaração de dependências no arquivo *pom* e a sua própria configuração automática no projeto.

Entre as suas principais características, pode-se citar:

- Autoconfiguração
- Segurança personalizada
- Reduz o tempo gasto no desenvolvimento
- Aumenta a eficiência na entrega dos projetos
- Biblioteca imensa de plug-ins
- Servidores embutidos na aplicação
- Conexão simplificadas aos Bancos de Dados.

Nesta apostila, destacaremos alguns conceitos e detalhes fundamentais do framework Spring Boot, tanto na teoria como na pratica. Para iniciar o desenvolvimento de API, são necessárias a configuração e a instalação do ambiente de programação, isso será explanado nos próximos capítulos.

## Instalação do Ambiente de Desenvolvimento

---

Neste módulo, será necessário a instalação de algumas ferramentas importantes, são elas:

- JDK JAVA versão 11
- IDE Spring Tools4
- VSCODE (Opcional)
- POSTMAN (Testar as requisições)
- MySQL Workbench
- GIT
- GIT HUB (CRIAR UMA CONTA)
- HEROKU
- GOOGLE CHROME

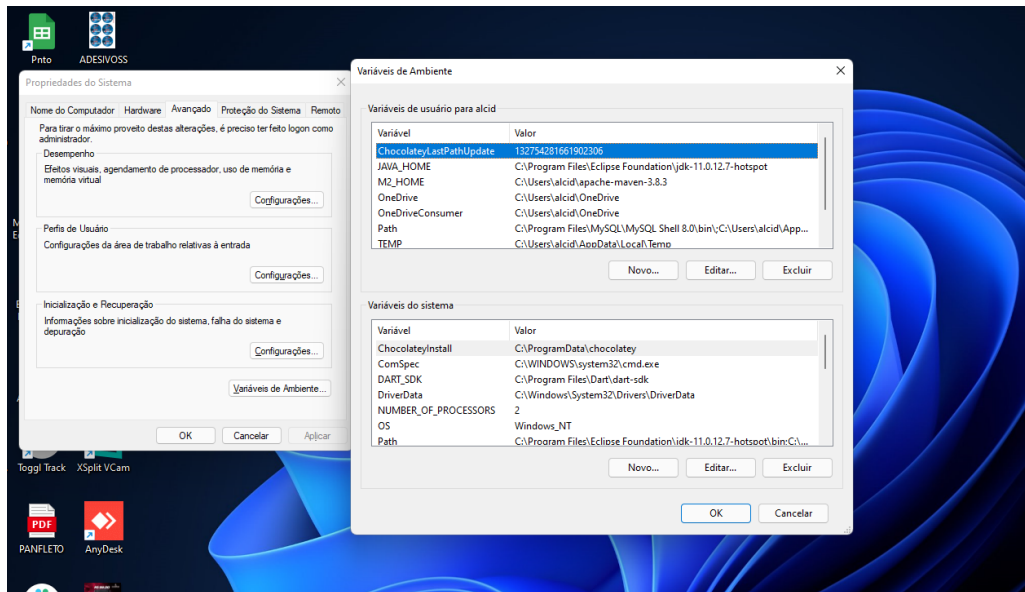
Instale essas ferramentas e crie as contas necessárias para ter acesso aos recursos de desenvolvimento, como por exemplo, a conta do GIT HUB.

## Variáveis de Ambiente

---

Após a instalação da versão 11 do JAVA configure a variável de ambiente JAVA no Windows.

- Digite no Windows [Variáveis de Ambiente] e clique.
- Em variáveis de ambiente de usuário, clique em novo, crie uma variável com o nome JAVA\_HOME
- Cole o endereço do JDK versão 11 no campo valor da variável.



Para verificar a versão do java instalada na sua máquina digite no prompt de comando (CMD), o seguinte comando:

```
java -version
```

## IDE SPRING TOOLS4

Além do JDK, é também interessante instalar a IDE de desenvolvimento; nesse módulo instalaremos o SpringTools4.

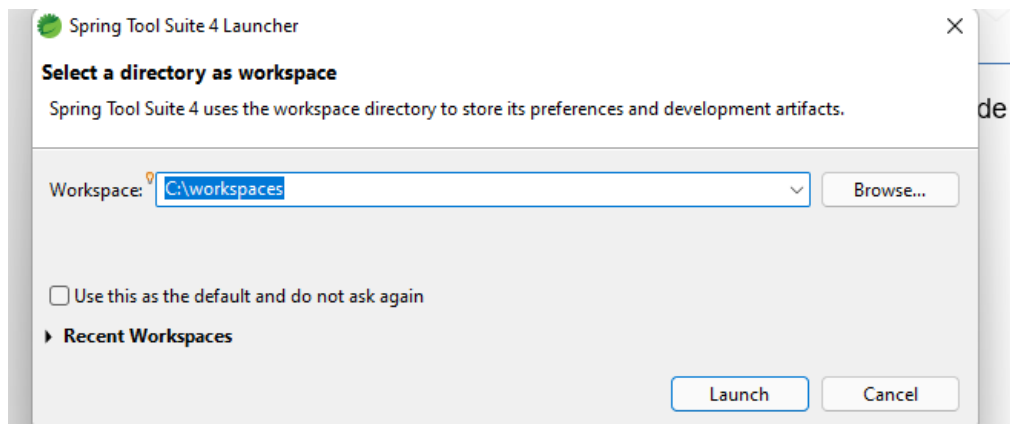
Ele pode ser baixado através da URL abaixo:

<https://spring.io/tools>

Clique no botão referente ao seu sistema operacional.

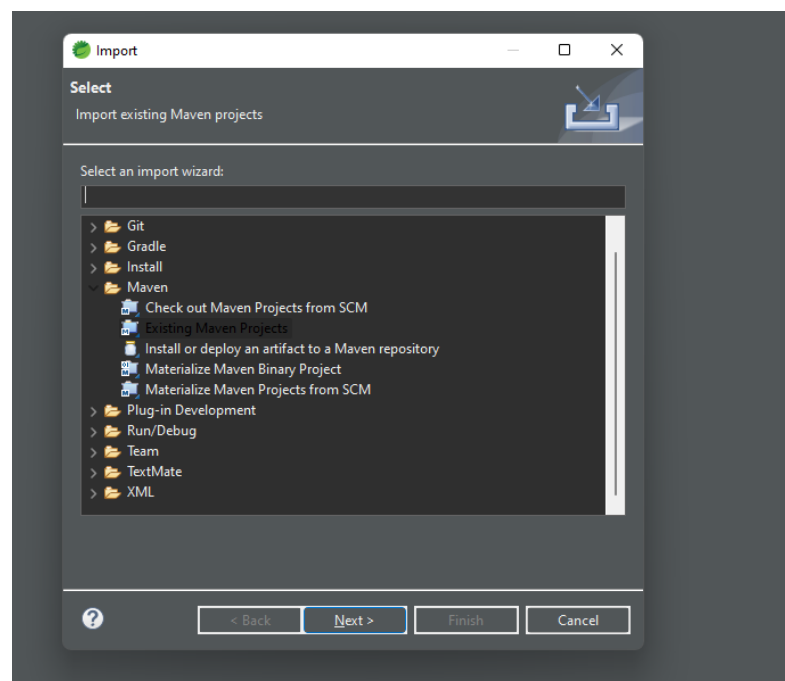
Após ter baixado e instalado, crie uma pasta no diretório C:\ com o nome workspaces, que servirá como uma área de trabalho para guardar seus projetos spring boot, como no exemplo:

Windows	20/10/2021 16:37	Pasta de arquivos
workspaces	18/10/2021 10:46	Pasta de arquivos



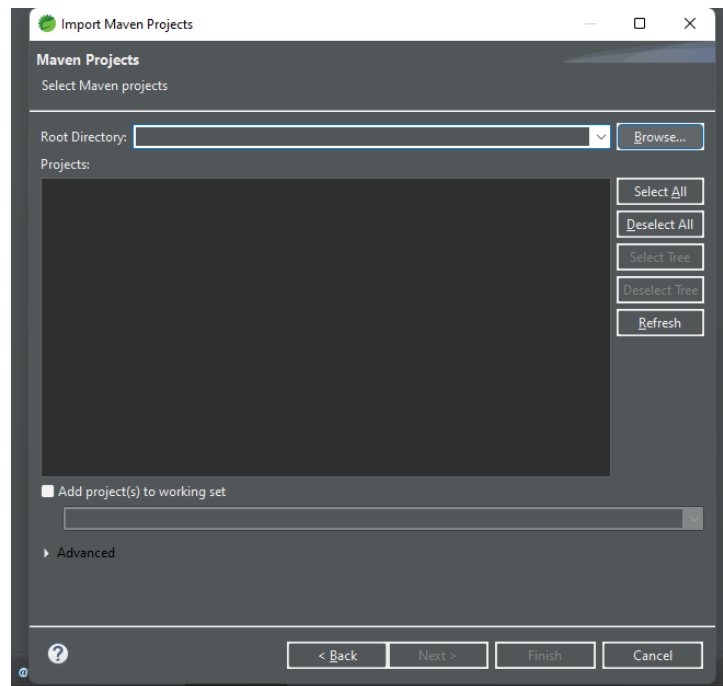
Certifique-se de selecionar o diretório workspaces, como na imagem acima. Depois que a IDE já estiver aberta, clique em File, e procure a opção IMPORT, para importar seu projeto para IDE, lembre-se de descompactar o arquivo zip antes de importar.

Clique em projeto Maven existentes (Existing Maven Projects).





Escolha o local do seu projeto, clicando em browser.



## Postman

Para testar as requisições que serão criadas no projeto, crie uma conta no Postman e instale seu executável. Com essa ferramenta, poderemos ver se as nossas requisições estão funcionando, e além do mais, ela possui inúmeros recursos importantes, entre eles, os métodos GET, POST, DELETE, PUT e etc.

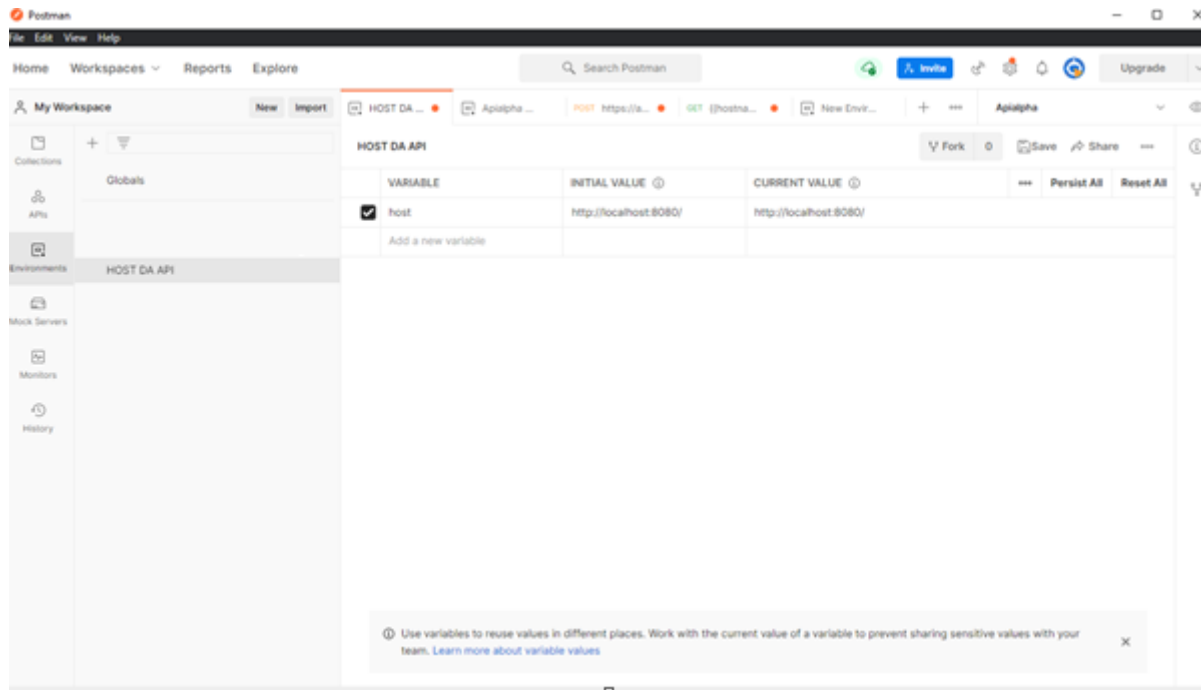
Para criar uma conta, clique no link abaixo:

<https://www.postman.com/>

Depois de criada a conta, instale o programa para desktop, através do link:

<https://www.postman.com/downloads/>

Quando estiver tudo pronto e instalado, abra o Postman na versão Desktop e configure a variável host de acesso, como no exemplo abaixo:



## Instalação do MySQL Workbench

Então para executar as consultas SQL, devemos instalar um administrador de banco de dados, e o que iremos usar é o MySql Workbench.

Para instalar clique no link abaixo, e selecione a versão do seu sistema operacional:

<https://dev.mysql.com/downloads/windows/installer/8.0.html>

Observação: No site vai aparecer apenas a versão 32 bits, não se preocupe, ela também irá funcionar na versão 64 bits.

## GIT E GIT HUB

O versionamento do nosso projeto é o processo pelo qual vamos atribuir um nome único ou uma numeração única para indicar o estado da aplicação, e para isso vamos baixar e instalar o Git, através do link abaixo:

<https://git-scm.com/downloads>

Logo após a sua instalação, se você aluno não tiver uma conta no GitHub, crie uma por meio do link:

[https://github.com/signup?ref\\_cta=Sign+up&ref\\_loc=header+logged+out&ref\\_page=%2F&source=header-home](https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home)

Alguns comandos Git serão importantes no desenvolvimento do projeto, entre eles podemos destacar:

### 1. Git Init

O comando git init é geralmente o primeiro comando executado em qualquer novo projeto que ainda não seja um repositório Git (também comumente chamado de repo). Ele pode ser usado para converter uma pasta existente sem versão em um repositório Git. Além disso, você pode usá-lo para inicializar um repositório vazio.

### 2. Clone Git

O comando git clone é usado para baixar o código-fonte de um repositório remoto (como GitHub, Bitbucket ou GitLab). Usando o comando Git Clone:

**\$ git clone <https://url-of-the-repository>**

Quando você clona um repo, o código é baixado automaticamente para sua máquina local.

### 3. Git Branch

Branch é uma das funcionalidades mais importantes do Git. Isso permite que as equipes trabalhem na mesma base de código em paralelo. Ele permite que as equipes criem fluxos de trabalho Git e tornem seus fluxos de trabalho mais eficientes. Use o comando Git Branch para criar um novo branch:

**\$ git branch <branch-name>**

#### **4. Git Checkout**

Depois de criar um branch, você terá que alternar para ele com outro comando. É aí que entra o comando Git Checkout: Usando o comando Git Checkout

**\$ git checkout <branch-name>**

O nome do branch que você mencionou no comando, mudará automaticamente.

#### **5. Git Add**

Cada vez que você criar um novo arquivo, excluí-lo ou fazer uma alteração, você terá que dizer ao Git para rastreá-lo e adicioná-lo à área de teste. Caso contrário, os arquivos nos quais você fez alterações não seriam adicionados quando você tentar enviar suas alterações. Usando o comando Git Add:

**\$ git add <file-name>**

Será adicionado apenas um único arquivo ao seu próximo commit. Se você quiser adicionar todos os arquivos aos quais as alterações foram feitas, você pode usar o comando:

**\$ git add .**

Ponto no final do comando significa que todos os arquivos e diretórios serão adicionados.

#### **6. Git Commit**

Pense no comando Git Commit como um ponto de verificação em seu processo de desenvolvimento. É comumente usado para salvar suas alterações. Talvez depois de concluir um item de trabalho específico atribuído a você em sua ferramenta ágil.

## 7. Git Push

Para disponibilizar todas as alterações confirmadas para seus companheiros de equipe, você terá que enviá-las para a origem remota. Use o Git Push Command:

**\$ git push <remote> <branch-name>**

## 8. Git Pull

Claro, você também gostaria de receber as últimas atualizações dos colegas de equipe! O comando git pull permite que você busque todas as mudanças que seus colegas de equipe enviaram e as mescle automaticamente em seu repositório local. Use o comando Git Pull:

**\$ git pull <remote>**

## 9. Status do Git

Quando você está se sentindo um pouco perdido com o que aconteceu em seu repo (sim, pode acontecer), o comando Git Status pode lhe dizer todas as informações que você precisa saber. Eu o comando Git Status:

**\$ git status**

## Deploy da aplicação no Heroku

---

Depois que nossa aplicação estiver finalizada, vamos subir o código para Heroku, que é uma plataforma em nuvem que serve para hospedar o backend. Como exemplo, teremos o código que desenvolveremos nesse curso, com ele será possível acessar o código java em uma aplicação frontend que fará o uso dos dados no formato json. Para instalar o Heroku na sua máquina clique no link abaixo:

<https://devcenter.heroku.com/articles/heroku-cli>

Escolha a versão do seu sistema operacional e clique em baixar. Depois que a aplicação estiver instalada abra o cmd e digite o seguinte comando:

**heroku login**

Automaticamente você será redirecionado para o site para fazer o login, lembre-se de fazer o seu cadastro no site.

O modelo Spring Boot de implantação de aplicativos independentes é uma ótima opção para o Heroku. Você pode usar o Maven ou o Gradle para implantar um aplicativo Spring no Heroku, mas para este guia, assumiremos que você está usando o Maven instalado em sua máquina.

Conectando-se a um banco de dados, você pode anexar um banco de dados PostgreSQL ao seu aplicativo executando o seguinte comando da CLI:

**heroku addons:create heroku-postgresql**

Se você preferir usar MySQL ou outro fornecedor de banco de dados, verifique o Mercado de complementos para ver o que está disponível.

O Heroku oferece uma ampla gama de recursos para aplicativos Spring. Você pode provisionar add-ons que introduzem serviços de nuvem de terceiros, como persistência, registro, monitoramento e muito mais. O add-on marketplace tem

um grande número de armazenamentos de dados, de provedores Redis e MongoDB a Postgres e MySQL.

## Capítulo 2. Anotações do Spring Boot

As anotações são funcionalidades que estão presentes no framework, que visa facilitar o desenvolvimento, e todas elas são precedidas pelo caractere @, como exemplo: @controller.

Vamos listar aqui as principais anotações do spring boot, observe abaixo:

- @RestController é uma anotação mais nova, adicionada na versão 4.0 do Spring, ele serve basicamente para informar que a classe em questão será uma controladora, que suporta serviços RESTFULL.

Exemplo:

```
@RestController
@RequestMapping(value = "/usuarios")
public class UsuarioController {
```

- @RequestMapping é uma anotação criada para definir o path ou caminho da requisição, o exemplo podemos ver acima. A partir dela pode acessar a url no navegador ou no postman.
- @Autowired fornece um controle onde a injeção de dependência deve ocorrer, ou seja, ela faz uma comunicação entre os beans.

Exemplo:

```
@Autowired
private SaleService service;
```

- @GetMapping serve para mapear as solicitações GET.



Exemplo:

```
@GetMapping(value = "/usuarios-by-cargo")
public ResponseEntity<List<Usuario>> listausuarios() {
```

- @PostMapping é usado para lidar com o método POST.

Exemplo:

```
@PostMapping("add-user")
public void addUser(@RequestBody Usuario user){
```

- @Entity é uma anotação do JPA, que é utilizada para representar que a classe será uma entidade, ou seja, a partir dela o JPA poderá fazer uma ligação e comunicação em uma tabela da base de dados e ficará mais fácil o mapeamento objeto relacional.

Exemplo:

```
@Entity
@Table(name = "tb_usuarios")
public class Usuario {
```

- @Table é uma anotação do JPA, que serve para declarar o nome da tabela no banco de dados.

Exemplo:

```
@Table(name = "tb_usuarios")
public class Usuario {
```

Result Grid		Filter Rows:	
	Tables_in_teste		
▶	tb_usuarios		

- @Id é uma anotação do JPA obrigatória, que serve para informar que determinado campo será a chave primária da tabela.

Exemplo:

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

- @GeneratedValue é uma anotação do JPA que tem como objetivo gerar um identificador automático que é gerenciado pela persistência de dados, e ele pode ser classificado em tipos, veja abaixo:

Exemplo:

- (a) GenerationType.AUTO - gera um valor automático e padrão do banco de dados;
- (b) GenerationType.SEQUENCE - os valores serão gerados a partir de uma sequência lógica;

(c) GenerationType.IDENTITY - para cada registro inserido no banco de dados, esse tipo de gerador cria um identificador em ordem crescente.

- @Column é uma anotação do JPA opcional, que serve para informar os detalhes da coluna. Você não precisa necessariamente colocar essa anotação no atributo para defini-la como coluna no banco de dados, o spring já sabe que o atributo é uma coluna, porém caso opte em utilizar, ela será necessária para informar o tamanho, nome e etc.

Exemplo:

```
@Column
private LocalDate date;
```

- @JoinColumn é uma anotação do JPA que indica o lado mais forte do relacionamento, ou seja, ela é uma chave estrangeira.

Exemplo:

```
@ManyToOne
@JoinColumn(name = "vendedor_id")
private Seller seller;
```

- @ManyToOne é uma anotação do JPA que informa que no relacionamento existem muitos registros relacionados a uma entidade, relacionamento de muitos para um.

Exemplo:

**Muitas vendas podem estar relacionadas a um só vendedor.**

- @Query é uma anotação do JPA que faz o uso do JPQL para método de escrita.
- @Transactional Descreve um atributo de transação em um método individual ou em uma classe e garante que todo processo deva ser executado em sua totalidade. Exemplo: Em uma transação bancária, ela só será 100% concluída depois que todos os processos tiverem ocorrido com sucesso, podemos citar a queda de internet, se no meio da transação acontecer algo do tipo, a transação não será efetuada.

### Anotação Lombok – Menos código e o mesmo resultado

O Lombok é uma biblioteca que tem como objetivo aumentar a produtividade do desenvolvedor, que a partir do uso de anotações específicas, “ensinamos ao compilador java” qual código precisa ser gerado automaticamente.

Para que o Lombok fique disponível no seu projeto é necessário que ele seja adicionado nas dependências do projeto no arquivo pom.xml.

Listaremos as principais anotações do Lombok:

- @Getter e @Setter - com essas anotações o Lombok cria todos os getters e setters dos atributos da classe.
- @NoArgsConstructor – com essa anotação será criada um construtor sem parâmetros.
- @AllArgsConstructor - cria todos os argumentos nos parâmetros do construtor.
- @ToString - retorna o nome da classe e os campos adicionados em seu método.
- @Data gera todos os getters, setters e toString da classe.

```
10 @NoArgsConstructor
11 @AllArgsConstructor
12 @Getter
13 @Builder
14 public class CustomerDTO {
15     private UUID id;
16     private String name;
17 }
18
```

### Capítulo 3. Banco de dados Mysql no projeto Spring

---

Levando em conta que o MySQL já está instalado na máquina e devidamente configurado, vamos ao próximo passo da configuração do banco no projeto.

1. Adicione as dependências MySQL no pom.xml

Para que o MySQL fique disponível no projeto, precisamos adicionar as dependências ***spring-boot-starter-data-jpa*** e ***mysql-connector-java***.

2. Configurar o arquivo application.properties com os dados do banco de dados MySQL, sendo eles:

```
spring.datasource.url=jdbc:mysql://localhost:3306/vendas
spring.datasource.username=root
spring.datasource.password=1234
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

3. A propriedade spring.jpa.hibernate.ddl-auto é para inicialização do banco de dados.

O arquivo pom.xml ficará da seguinte forma:

```
<!--mysql-->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
```

*Lembre-se de adicionar no site Inicializar a dependência do MySQL.*

## Criando a entidade JPA

---

Para criar uma entidade é necessário o uso das anotações que foram estudadas nos capítulos anteriores. Para começar vamos criar uma entidade Vendedor, ela terá a aparência abaixo:

```
@Entity
@Table(name = "tb_vendedor")
public class Seller {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
```

O Hibernate irá converter essa entidade para tabela no MySQL.

## Criando os repositórios

---

O repositório nos ajudará a executar a operação CRUD normal no banco de dados. Usaremos a interface do repositório Spring que gera o mapeamento necessário para nós e nos ajuda a evitar a escrita de código usual.

```
public interface VendedorRepository extends JpaRepository<Vendedor, Long> {
```

Esta é uma interface e estamos estendendo o JpaRepository. O Spring criará automaticamente um bean para esta entrada (nome do bean como userRepository) e fornecerá implementações para as operações CRUD.

Os repositórios Java são comumente conhecidos como repositórios baseados em JPA, usados sob a estrutura Spring Boot. Os repositórios definem um novo método elegante e atual de armazenamento, atualização e extração de dados armazenados de aplicativos JAVA no backend. Todas as operações CRUD (Criar, ler, atualizar e excluir) podem ser implementadas com a ajuda de uma interface de repositório.

JPA é uma abreviatura de JAVA Persistence API (Application program interface). Como o nome sugere, JPA ajuda a persistir os objetos java em bancos de dados relacionais. Existem duas maneiras de fazer isso:

- As tabelas relacionais são mapeadas para subsistemas usando classes de mapas;
- API EntityManager.

Uma vez que a interface JPA REPOSITORY é criada, funções como “save ()”, “count ()”, “info ()”, “findAll ()”, “sort ()” e outras são usadas para realizar a consulta de dados ou manipulação de dados necessária . Precisamos ter um banco de dados configurado para inserir, atualizar ou excluir os valores das tabelas abaixo por meio do aplicativo java.

O uso de repositórios facilita o manuseio de dados de bancos de dados. Os repositórios JPA são muito úteis, pois fornecem uma plataforma genérica para criar consultas na linguagem JAVA, mas podem ser usados com qualquer banco de dados abaixo.

Ele reduz o número de linhas de código, fornecendo funções elegantes para realizar as tarefas apoiadas por bibliotecas. Ele reduz o uso de código “padrão”, melhorando assim a aparência e a velocidade de execução.

Para definir métodos de acesso mais específicos, Spring JPA suporta algumas opções:



- fornecer a consulta JPQL real usando a anotação @Query;
- use o suporte mais avançado de Especificação e Query no Spring Data;
- definir consultas personalizadas por meio de consultas nomeadas JPA.

## Criando os controladores

O controlador será importante para interagir com a classe de serviços ou de repositórios, a partir daqui será possível criar os endpoints para serem testados via postman.

Exemplo:

```
@RestController
@RequestMapping(value = "/vendedor")
public class SellerController {
    @Autowired
    private VendedorService service;

    @GetMapping
    public ResponseEntity<List<VendedorDTO>>findAll(){
        List<VendedorDTO>list=service.findAll();
        return ResponseEntity.ok(list);
    }
}
```

As classes do controlador no Spring são anotadas pelo @Controller ou pela @RestController. Eles marcam as classes do controlador como um manipulador de

solicitação para permitir que o Spring o reconheça como um serviço RESTful durante o tempo de execução.

Essencialmente, `@RestController` estende os recursos das anotações `@Controller` e `@ResponseBody`. Além do fato de que `@RestController` existe para permitir que os controladores Spring sejam uma linha mais curtos, não há grandes diferenças entre as duas anotações.

A função principal de ambas as anotações é permitir que uma classe seja reconhecida como um componente gerenciado por Spring e permitir o tratamento de solicitações HTTP usando a API REST.

A `@Controller` é uma especialização da `@Component` de estereótipo genérico, que permite que uma classe seja reconhecida como um componente gerenciado por Spring.

A `@Controller` estende o caso de uso, `@Component` marca a classe anotada como uma camada de negócios ou apresentação. Quando uma solicitação é feita, isso informará ao `DispatcherServlet` para incluir a classe do controlador na varredura de métodos mapeados pela `@RequestMapping`.

## Capítulo 4. Arquitetura do projeto

### – Padrão de camadas

---

O padrão em camadas é o mais utilizado dos padrões de arquitetura de software. Muitos desenvolvedores o usam, sem realmente saber seu nome. O objetivo é dividir seu código em “camadas”, onde cada camada ou parte tem uma certa responsabilidade e fornece um serviço a uma camada superior.

Não existe uma quantidade necessária de camadas para serem criadas, mas estas são as que você verá com mais frequência:

- Apresentação ou camada de IU
- Camada de aplicação
- Camada de negócios ou domínio
- Persistência ou camada de acesso a dados
- Camada de banco de dados

A ideia é que o usuário inicie um trecho de código na camada de apresentação executando alguma ação (por exemplo, clicando em um botão). A camada de apresentação então chama a camada subjacente, ou seja, a camada de aplicação. Em seguida, vamos para a camada de negócios e, finalmente, a camada de persistência armazena tudo no banco de dados. Portanto, as camadas superiores dependem e fazem chamadas para as camadas inferiores.

Você verá variações disso, dependendo da complexidade dos aplicativos. Alguns aplicativos podem omitir a camada de aplicativo, enquanto outros adicionam uma camada de cache.

#### **Responsabilidade da Camada**

Conforme vimos, cada camada tem sua própria responsabilidade. A camada de apresentação tem como objetivo cuidar do design gráfico do aplicativo, bem como qualquer código para lidar com a interação do usuário.

A camada de negócios é onde você coloca os modelos e a lógica, que são específicos para o problema de negócios que está tentando resolver.

A camada de aplicativo fica entre a camada de apresentação e a camada de negócios. Por um lado, fornece uma abstração para que a camada de apresentação não precise conhecer a camada de negócios. Em teoria, você poderia alterar a pilha de tecnologia da camada de apresentação sem alterar nada mais em seu aplicativo (por exemplo, alterar de WinForms para WPF). Por outro lado, a camada de aplicativo fornece um local para colocar certa lógica de coordenação que não se encaixa na camada de negócios ou de apresentação.

Finalmente, a camada de persistência contém o código para acessar a camada de banco de dados. A camada de banco de dados é a tecnologia de banco de dados subjacente (por exemplo, SQL Server, MongoDB). A camada de persistência é o conjunto de código para manipular o banco de dados: instruções SQL, detalhes de conexão, etc.

### **Vantagens**

A maioria dos desenvolvedores está familiarizada com esse padrão. Ele fornece uma maneira fácil de escrever um aplicativo bem organizado e testável.

### **Desvantagens**

Isso tende a levar a aplicativos monolíticos que são difíceis de separar depois.

Os desenvolvedores frequentemente se pegam escrevendo muito código para passar pelas diferentes camadas, sem agregar nenhum valor a essas

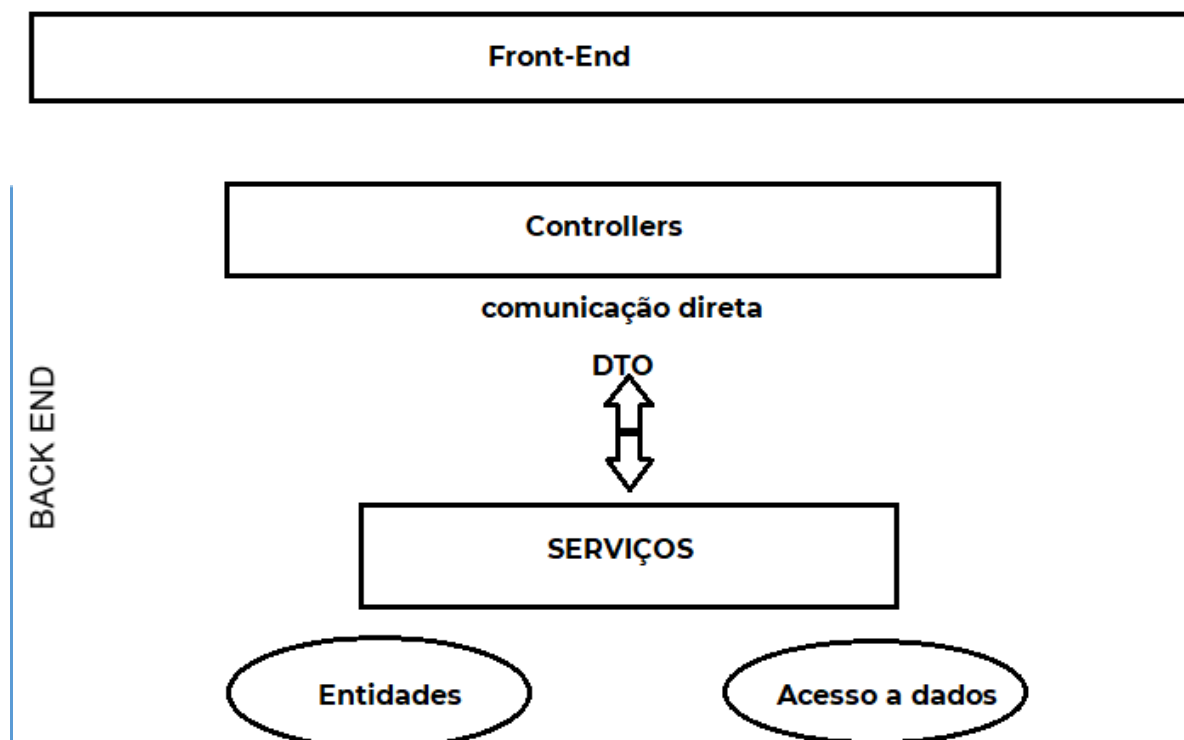
camadas. Se tudo o que você está fazendo é escrever um aplicativo CRUD simples, o padrão em camadas pode ser um exagero para você.

Ideal para Aplicativos de linha de negócios padrão que fazem mais do que apenas operações CRUD

### Exemplo prático do padrão de camadas em um projeto

Em um projeto será criado um endpoint para listar os dados do banco de dados MySQL.

O projeto terá a seguinte estrutura:



Na IDE ficará da seguinte forma:

Nome	Data de modificação	Tipo
config	12/09/2021 17:27	Pasta de arquivos
controllers	12/09/2021 17:27	Pasta de arquivos
dto	12/09/2021 17:27	Pasta de arquivos
entities	12/09/2021 17:27	Pasta de arquivos
repositories	12/09/2021 17:27	Pasta de arquivos
services	12/09/2021 17:27	Pasta de arquivos

O padrão de projeto DTO é muito versátil e importante, tanto para receber dados, quanto para enviá-los, pois a partir dele poderemos manipular os dados com mais segurança e não será necessário ter acesso as camadas mais inferiores, como as entidades.

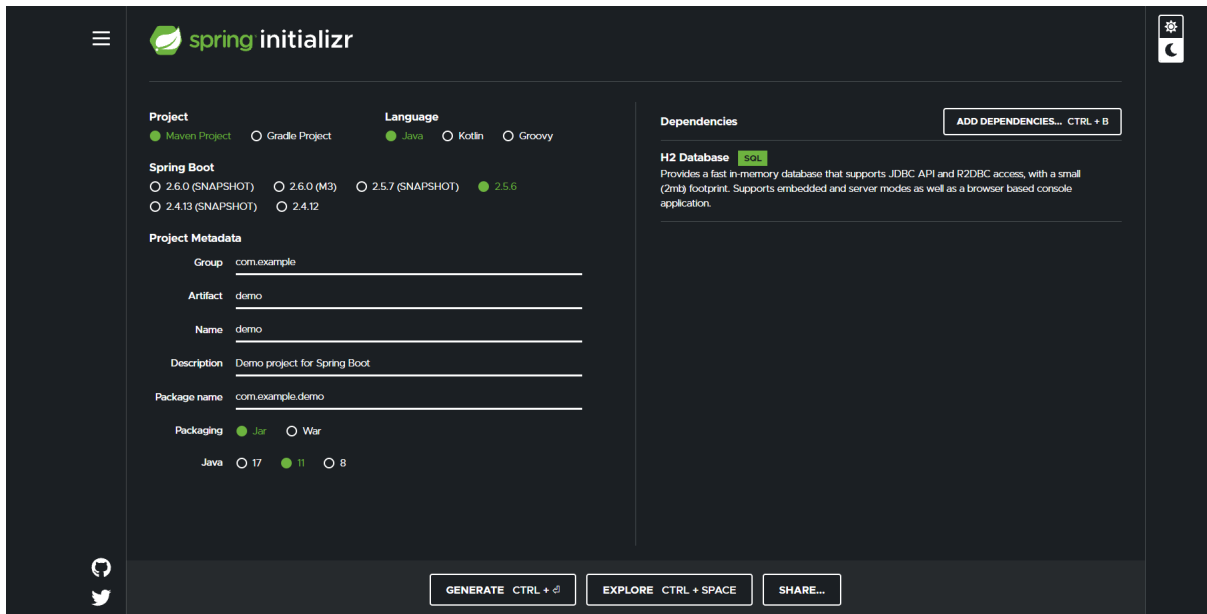
### O que são classes DTO

Data Transfer Object (DTO) é um padrão de projetos muito usado em Java para a comunicação de dados entre diferentes componentes de um sistema, diferentes instâncias ou processos.

O Data Transfer Object Design Pattern é um dos padrões de arquitetura de aplicativo corporativo que exige o uso de objetos que agregam e encapsulam dados para transferência. Um Objeto de Transferência de Dados é, essencialmente, como uma estrutura de dados. Ele não deve conter nenhuma lógica de negócios, mas deve conter mecanismos de serialização e desserialização.

Vamos para o exemplo:

Crie um projeto Spring, levando em consideração a estrutura de pastas nos parágrafos acima.



The image shows the Spring Initializr web interface. It is a dark-themed form for generating a Spring project. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.5.6' selected. The 'Project Metadata' section contains fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), and Package name (com.example.demo). The 'Packaging' section has 'Jar' selected. The 'Dependencies' section has 'H2 Database' selected. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE'.

Depois crie uma entidade Usuário:

```
@Entity
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String username;
    private String nome;
    private String email;

    // Getters and Setters
}
```

Crie um repositório:

```
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {}
```

Crie uma classe serviço que terá a função de regra de negócio e se comunicará com o DTO. Para isso mapearemos os resultados e retornaremos os DTOs:

```
@Service
public class UsuarioService {

    @Autowired
    private UsuarioRepository userRepository;

    @Transactional(readOnly = true)
    public Page<UsuarioDTO> findAll(Pageable pageable) {
        userRepository.findAll();
        Page<Usuario> result = repository.findAll(pageable);
        return result.map(x -> new UsuarioDTO(x));
    }
}
```

Vamos fazer um objeto de transferência de dados DTO para transferir apenas as informações necessárias.



```
public class UsuarioDTO {
    private Long id;
    private String username;
    private String nome;
    private String email;

    // Getters and Setters
}
```

Finalmente, vamos fazer um endpoint para permitir que alguém recupere os dados dos usuários:

```
@RestController
@RequestMapping(value = "/usuarios")
public class UsuarioController {
    @Autowired
    private UsuarioService service;

    @GetMapping
    public ResponseEntity<Page<UsuarioDTO>> findAll(Pageable pageable) {
        Page<UsuarioDTO> list = service.findAll(pageable);
        return ResponseEntity.ok(list);
    }
}
```

É de extrema importância que o nosso banco de dados contenha algumas informações fictícias para fins de teste:

```
INSERT INTO tb_usuarios (username,nome,email) VALUES
('maria','Maria Lima','maria@gmail.com');
```

Seguido todos os passos só testar no postman a api no seguinte endereço:

**<http://localhost:8080/usuarios>**



## Referências

---

SPRING Boot Reference Documentation. 1. [S. l.], 10 out. 2020. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Acesso em: 15 out. 2021

STACK OVERFLOW. Stack Overflow Developer Survey, 2020. Disponível em <https://insights.stackoverflow.com/survey/2020>. Acesso em 15 out. 2021.

HECKLER, Mark. Spring Boot: Up and Running: Building Cloud Native Java and Kotlin Applications. [S. l.: s. n.], 02/03/2021. ISBN 1492076988.