

SPRING



mentoriaT3ch

tecnologia, tendência & transformação.

Sobre o autor

Giovanni de Carvalho

Minha carreira iniciou em 2016 quando comecei a cursar um curso de manutenção de computadores pelo Instituto do Banco do Brasil, ali já tinha decidido que ia seguir na área da tecnologia. Entre idas e vindas em 2017 comecei a trabalhar em uma empresa de Telecom no setor administrativo e consequentemente iniciei uma graduação de Administração de Empresas na Universidade Nove de Julho. Após 2 anos de curso, estava atuando em uma multinacional e sendo destaque em sala de aula com palestras e projetos que foram patrocinados por grandes empresas como B3 e Coca-Cola.

Bom depois de tudo isso você deve estar pensando, que mais um jovem pode querer? Bom eu ainda quero muito mais e um destes enormes desejos era retornar a área de tecnologia.

Conquistei uma bolsa de estudos no centro de inovação do Facebook, para estudar programação Web, não pensei duas vezes! Pedi demissão do meu emprego para agarrar um dos meus sonhos. De lá para cá fui aluno do Senai, realizei diversos cursos e estou no último ano de uma graduação de Análise e Desenvolvimento de Sistemas. Trabalhei em um das Startups mais reconhecidas do país e hoje trabalho no maior Banco da América Latina.

Tenho experiência em engenharia de software com integração e modelagem de sistemas, transformação digital e modernização de plataformas legadas atuando com tecnologias como: Html5, Css3, Javascript, Saas, Bootstrap, Materialize, Angular, React, Spring Boot, Django, NodeJs, MySQL, Db2, Oracle, Mainframe, Mvc, Rest, Docker. Além de experiência em: administração de equipes, administração de processos gerando KPI's relatórios e análise de dados para melhorias contínuas.



“Feito é melhor do que perfeito” e é por isso que uma ideia pior do que a sua está fazendo sucesso



- Mark Zuckerberg

visão geral, introdução

ecosistema spring

como funciona a injeção de dependências

instalação das ferramentas

estrutura de um projeto

desafio start spring

S T - A R T - I

MÓDULO 1

visão geral, introdução

SPRING

Bóra entender um pouco sobre Spring

Atualmente o java tem sido uma das grandes opções para entrar no mercado de desenvolvimento web. Sua stack é utilizada em diversos conceitos e diversas empresas de principalmente médio e grande porte. O Spring boot com todo o seu ecossistema é o principal framework de java, ou seja, uma ferramenta que otimiza o desenvolvimento de seus projetos, abstraindo a infraestrutura de seu projeto e disponibilizando métodos já prontos para utilização como acesso ao banco de dados, retorno e tratamento de dados.

A linguagem java

O java é a linguagem mais utilizada no mercado, tendo responsabilidade no desenvolvimento de projetos famosos como: Uber, Ifood, Banco Itáu, Banco Bradesco, Pic Pay e etc;



START

MÓDULO 2

ecossistema spring

Ecossistema Spring

O Spring consiste em uma plataforma completa de recursos para construção de aplicativos Java, que veio para simplificar o desenvolvimento em Java EE com diversos módulos que auxiliam na construção de sistemas reduzindo muito o tempo de desenvolvimento.

Essa plataforma conta com recursos avançados que abrangem várias áreas de uma aplicação com projetos/módulos prontos para uso.

Projetos Spring

Spring Framework;
Spring Boot;
Spring Web;
Spring Security;
Spring Data;
Spring Batch;
Spring Cloud;
outros.

Para visualizar e conhecer todos os projetos disponíveis na plataforma Spring, basta acessar o site Spring by Pivotal e conferir a lista completa através do link :

Projetos Spring

<https://spring.io/projects/spring-boot>

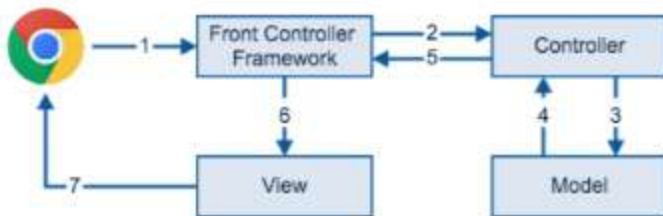
START

MÓDULO 3

como funciona a injeção de
dependências

SPRING

O que é a (injeção de dependência)?



Injeção de dependências (ou Dependency Injection - DI) é um tipo de inversão de controle (ou Inversion of Control - IoC) que dá nome ao processo de prover instâncias de classes que um objeto precisa para funcionar. A grande vantagem desse conceito é que nós conseguimos programar voltados para interfaces e, com isso, manter o baixo acoplamento entre as classes de um mesmo projeto.

Diferença entre instancia e injeção de dependência

```
public class Cliente {
```

```
// Nesse exemplo estou supondo que  
// "RepositoryCliente" é uma  
// interface. Mas poderia ser uma classe  
// abstrata ou  
// mesmo uma classe concreta.
```

```
private RepositoryCliente repositorio =  
new RepositoryClienteImpl();  
...  
}
```

Você vai fazer isso:

```
public class Cliente {  
    @Autowired  
    private RepositoryCliente repositorio;  
    ...  
}
```

S T - A R T

MÓDULO 4

instalação das ferramentas

SPRING

Principais ferramentas

Como já vimos nesta jornada sobre as IDE's (ambientes de desenvolvimento) para desenvolver códigos em java temos diversas opções como o Netbeans, Eclipse, IntelliJ IDEA ou Spring tool suite.

Para nosso projeto vamos utilizar:

Xampp (compilador do Mysql e do toomcat)
HeidiSql (SGBD do Mysql)
Spring tool suite
Thymeleaf

XAMPP

<https://www.apachefriends.org/index.html>

HEIDISQL

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

SPRING TOOL SUITE

<https://spring.io/tools>

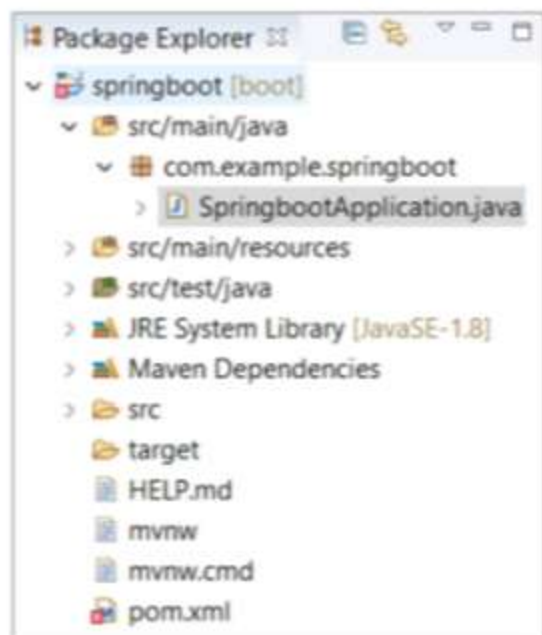
START

MÓDULO 5

estrutura de um projeto

SPRING

Estrutura de projeto

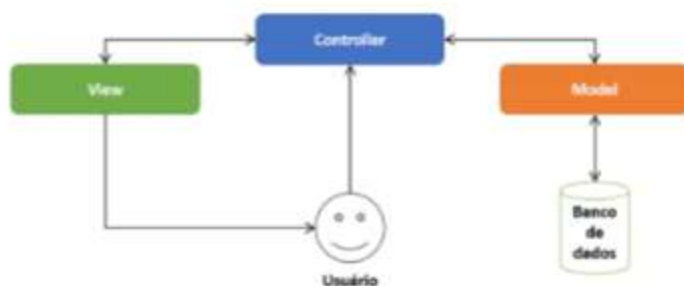


controller: A Controladora (controller), como o nome já sugere, é responsável por controlar todo o fluxo de informação que passa pelo site/sistema. Define quais informações devem ser geradas, quais regras devem ser acionadas e para onde as informações devem ir, é na controladora que essas operações devem ser executadas

model: Camada de modelo ou da lógica da aplicação (Model) é o coração da execução, responsável por tudo que a aplicação vai fazer a partir dos comandos da camada de controle em um ou mais elementos de dados, respondendo a perguntas sobre o sua condição e a instruções para mudá-las.

view: A view é responsável pela interface que será apresentada, mostrando as informações do model para o usuário.

repository: Um Repository (Repositório) é um objeto que isola os objetos ou entidades do domínio do código que acessa o banco de dados.



START

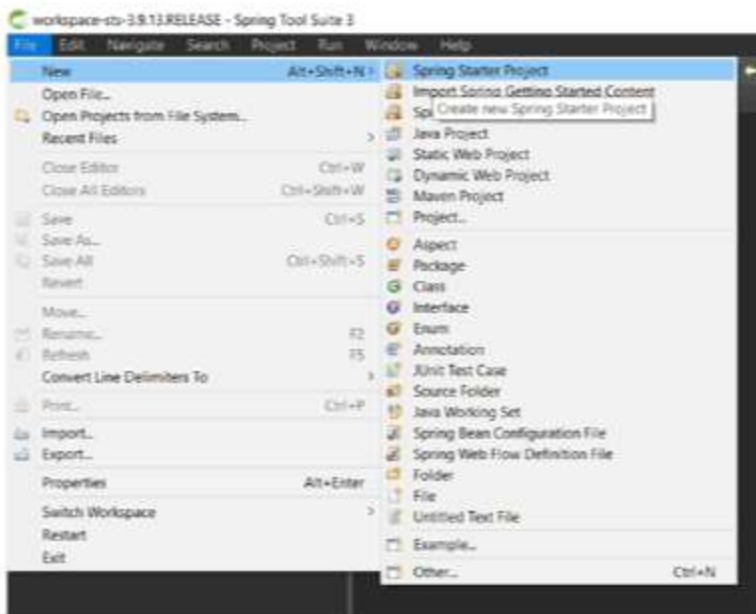
MÓDULO 6

desafio start spring

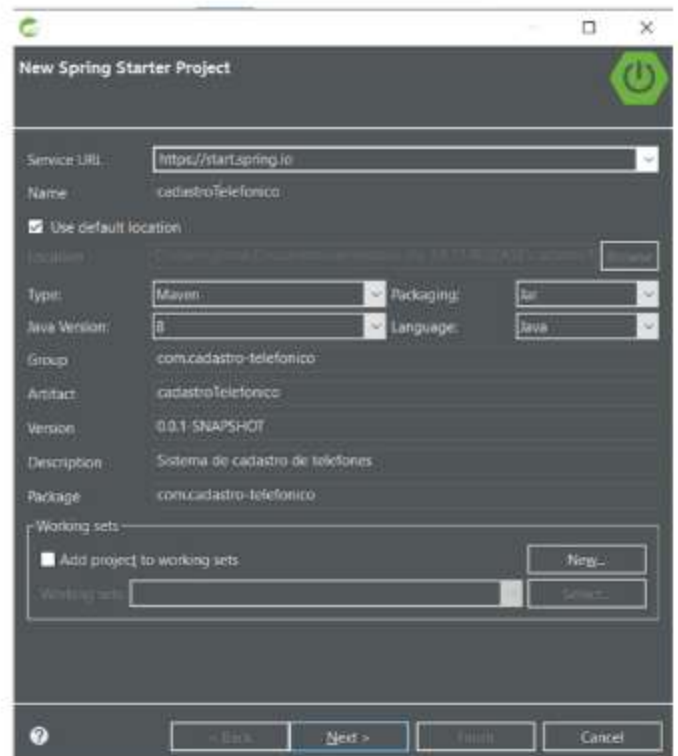
SPRING

Desafio #startSpring

Vamos começar pela abertura da Ide.
Criaremos um novo projeto.



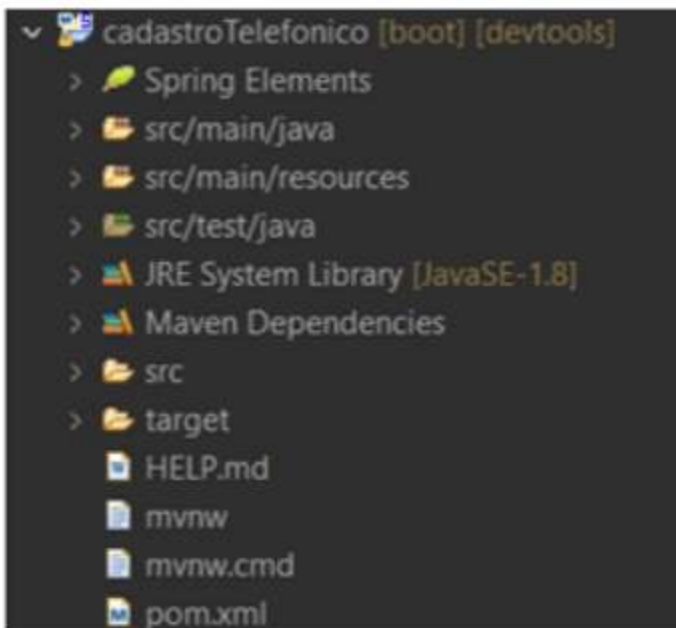
Vamos preencher ele da seguinte forma.



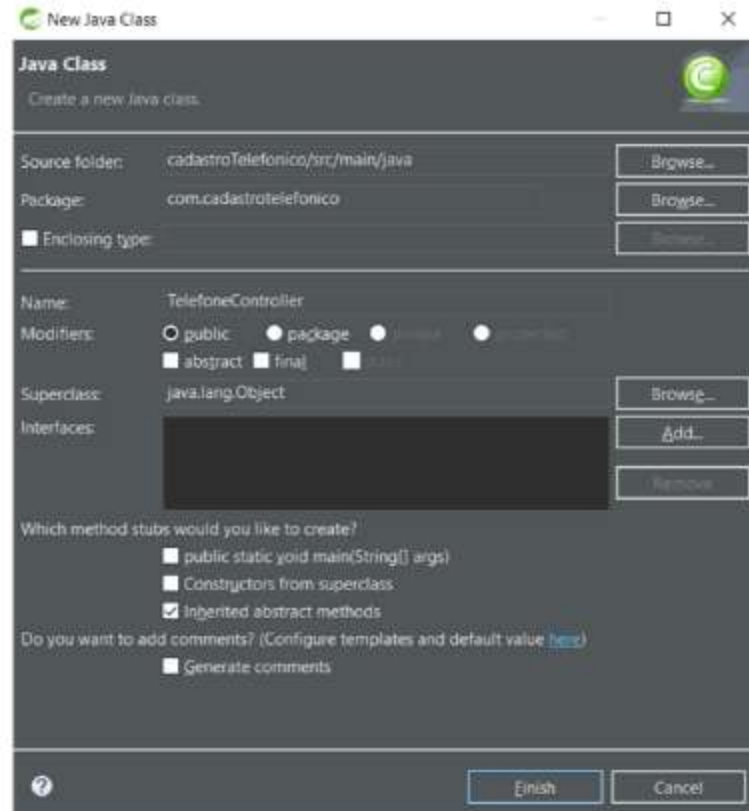
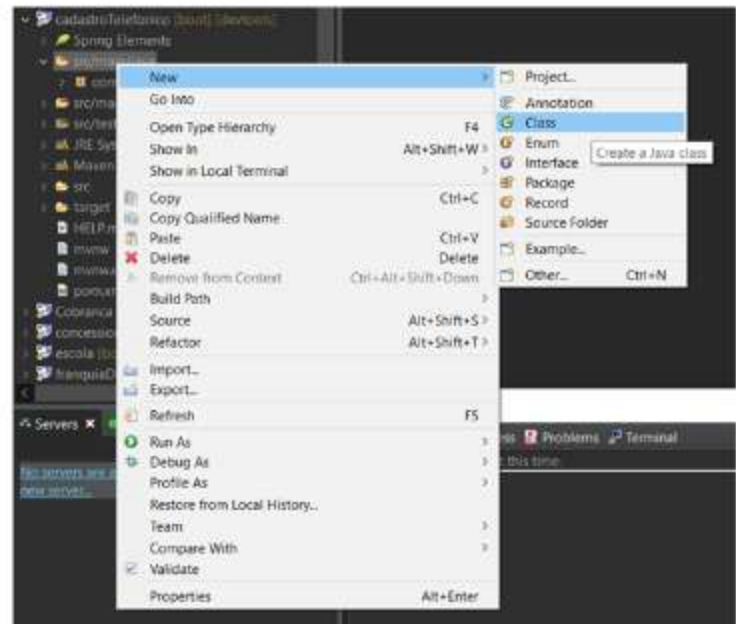
SPRING

Desafio #startSpring

Repare na estrutura do projeto.



Criaremos uma Classe chamada
TelefoneController.java



SPRING

Desafio #startSpring

Esta é a nossa class TelefoneController.java

```
TelefoneController.java
1 package com.cadastratelefonico;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5
6 @Controller
7 public class TelefoneController {
8
9     @GetMapping("/")
10    public String index() {
11        return "index";
12    }
13 }
14 }
```

```
package com.cadastratelefonico;
```

```
import
```

```
org.springframework.stereotype.Controller;
```

```
import
```

```
org.springframework.web.bind.annotation.Get  
Mapping;
```

```
@Controller
```

```
public class TelefoneController {
```

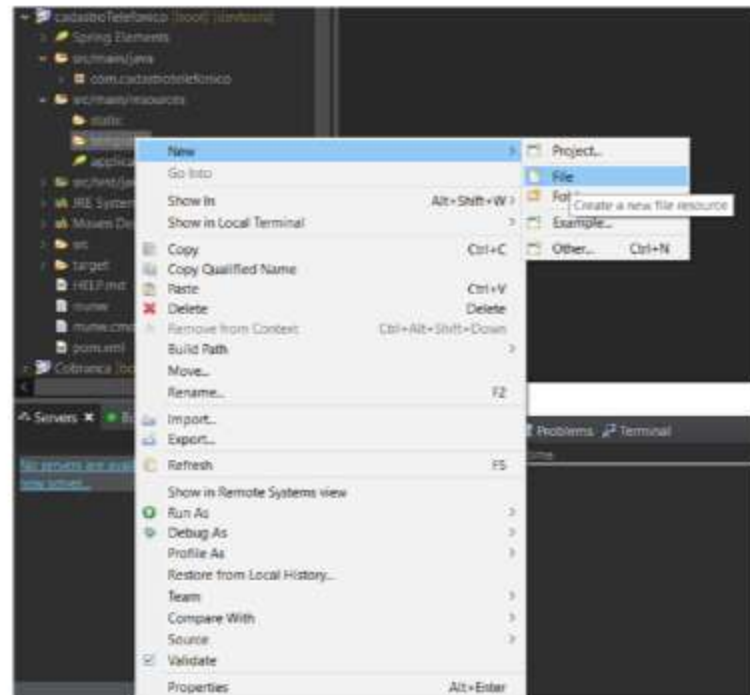
```
    @GetMapping("/")
```

```
    public String index() {  
        return "index";
```

```
    }
```

```
}
```

Agora criaremos o arquivo index.html



Este é o nosso arquivo index.html

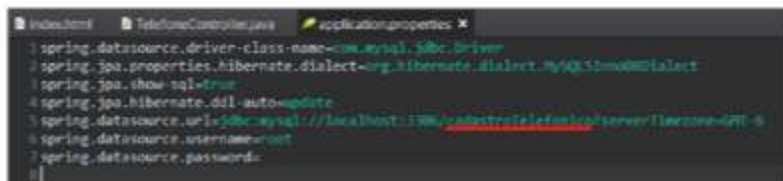
```
*index.html
1 <h1> Teste spring </h1>
```

```
<h1> Teste spring </h1>
```

SPRING

Desafio #startSpring

Vamos criar nossa conexão com o banco de dados MySQL no arquivo application.properties



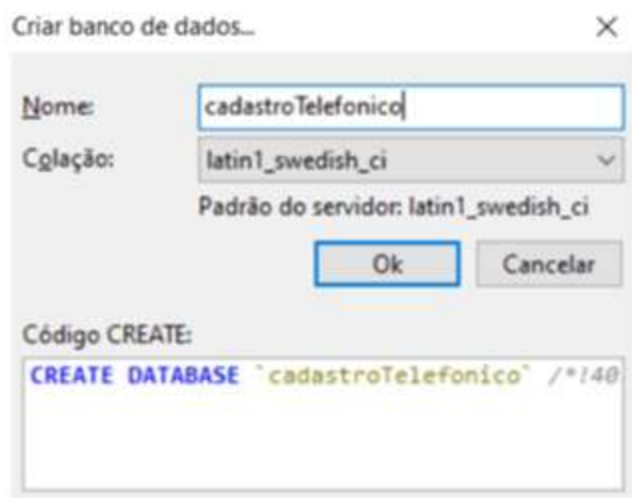
```
1 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
2 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
3 spring.jpa.show-sql=true
4 spring.jpa.hibernate.ddl-auto=update
5 spring.datasource.url=jdbc:mysql://localhost:306/cadastroTelefonico?serverTimezone=GMT-6
6 spring.datasource.username=root
7 spring.datasource.password=
```

O grifado em vermelho é o nome do banco:

[cadastroTelefonico](#)

```
spring.datasource.driver-class-
name=com.mysql.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hi
bernate.dialect.MySQL5InnoDBDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3
06/cadastroTelefonico?
serverTimezone=GMT-6
spring.datasource.username=root
spring.datasource.password=
```

Vamos ao nosso banco de dados e assim criamos um banco de dados com o mesmo nome: cadastroTelefonico.

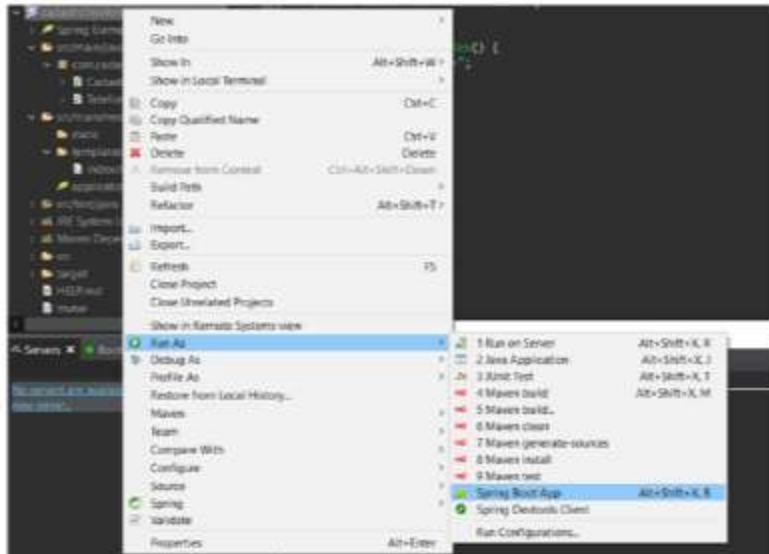


Esse padrão é para que ao momento em que realizarmos o build do projeto ele crie e registre todas as informações em nosso banco de dados.

SPRING

Desafio #startSpring

Agora é só executarmos o RUN, ou seja "rodar o projeto".

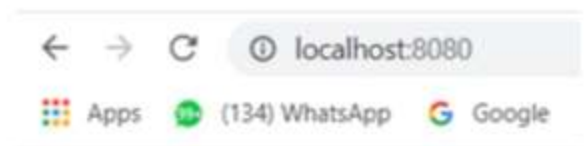


O RUN do projeto é a construção e execução dele por completa.



Esse padrão é da execução correta de nosso projeto, onde ele irá ficar disponível na porta 8080/

A partir disso, vamos a uma aba do google
[localhost://8080](http://localhost:8080)



Teste spring

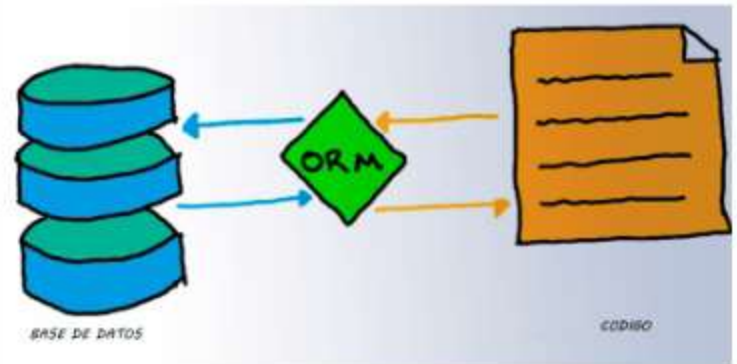
SPRING

Desafio #startSpring

UML do projeto

telefone		
pk int	auto increment	id
String		nome
String		sobrenome
String		numeroTelefone

O mapeamento objeto-relacional em ciência da computação é uma técnica de programação para converter dados entre sistemas de tipos incompatíveis usando linguagens de programação orientadas a objetos. Isso cria, com efeito, um "banco de dados de objetos virtuais" que pode ser usado a partir da linguagem de programação.

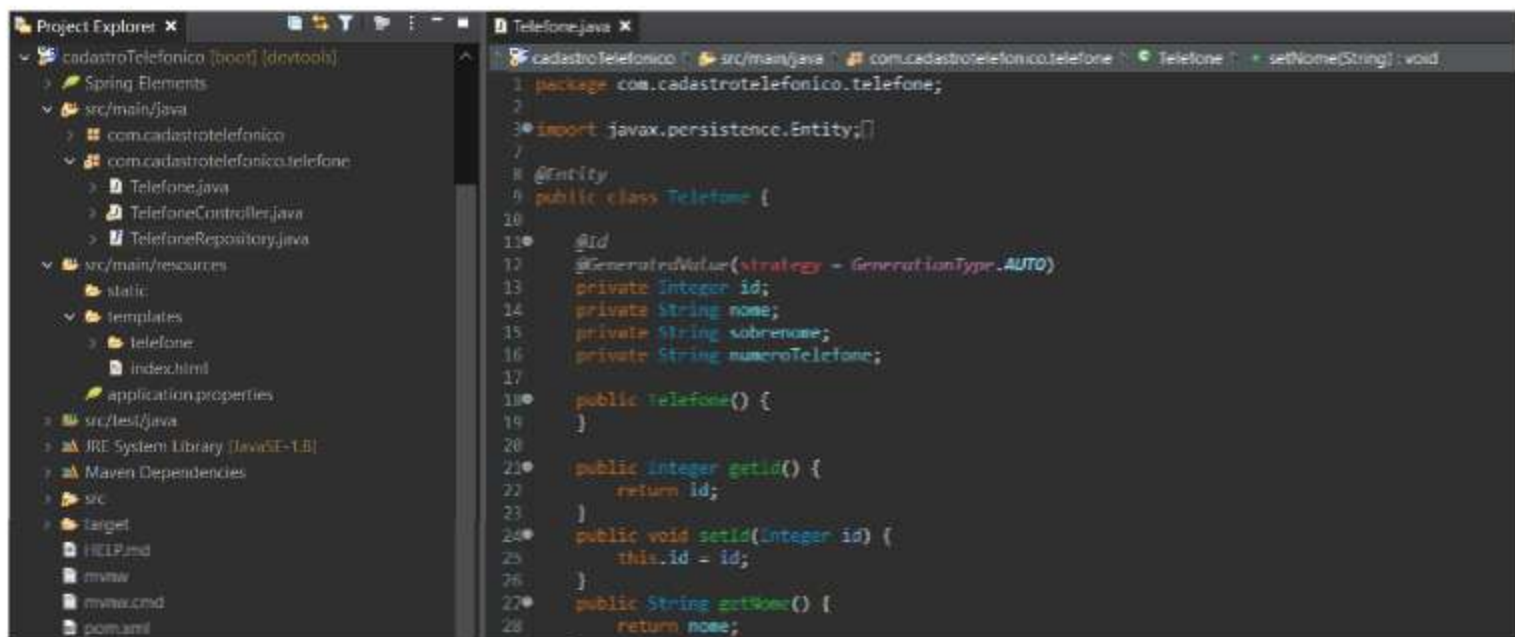


A orm serve para que nossos objetos sejam transferidos do banco para o código e vice e versa, neste exemplo usaremos o JPA e o Hibernate.

SPRING

Desafio #startSpring

Camada de entidade - mapeamento de dados



Para construir um cadastro vamos construir uma nova classe com o nome do objeto que irá executar as operações de cadastro, listagem e etc...

Todas nossas classes serão encapsuladas em um package com o nome :

[com.cadastrotelefonico.telefone](#)

A classe com a @anotation acima dela, significa que ela terá uma responsabilidade ao ser executada:

[@Controller](#) - camada de roteamento entre páginas e operações .

[@Entity](#) - camada de mapeamento de dados para o banco de dados.

[@Repository](#) - camada de intermediação entre o mapeamento de dados e o banco de dados.

SPRING

Desafio #startSpring

A entidade ou model é a classe com a anotation `@Entity` é baseada em nosso banco de dados, por isso é sempre muito importante que ao iniciamos o desenvolvimento de um sistema precisamos sempre pensar na modelagem dos dados.

Esse processo é chamado de engenharia de software, a UML é a estrutura base de um sistema que tenha backend ou seja acesso a banco de dados e a hospedagem em um servidor.

Podemos utilizar "`constraints`" e tipos de dados para nossa camada, como por exemplo:

```
@Id  
@GeneratedValue(strategy =  
GenerationType.AUTO)
```

Assim irá garantir que nosso Id seja registrado como primary key e como auto increment.

Camada Model

@Entity

```
public class Telefone {
```

```
    @Id
```

```
    @GeneratedValue(strategy =  
GenerationType.AUTO)
```

```
    private Integer id;
```

```
    private String nome;
```

```
    private String sobrenome;
```

```
    private String numeroTelefone;
```

```
    public Telefone() {  
    }
```

```
    public Integer getId() {  
        return id;  
    }
```

```
    public void setId(Integer id) {  
        this.id = id;  
    }
```

```
    public String getNome() {  
        return nome;  
    }
```

```
    public void setNome(String nome) {  
        this.nome = nome;  
    }
```

```
    public String getSobrenome() {  
        return sobrenome;  
    }
```

```
    public void setSobrenome(String sobrenome) {  
        this.sobrenome = sobrenome;  
    }
```

```
    public String getNumeroTelefone() {  
        return numeroTelefone;  
    }
```

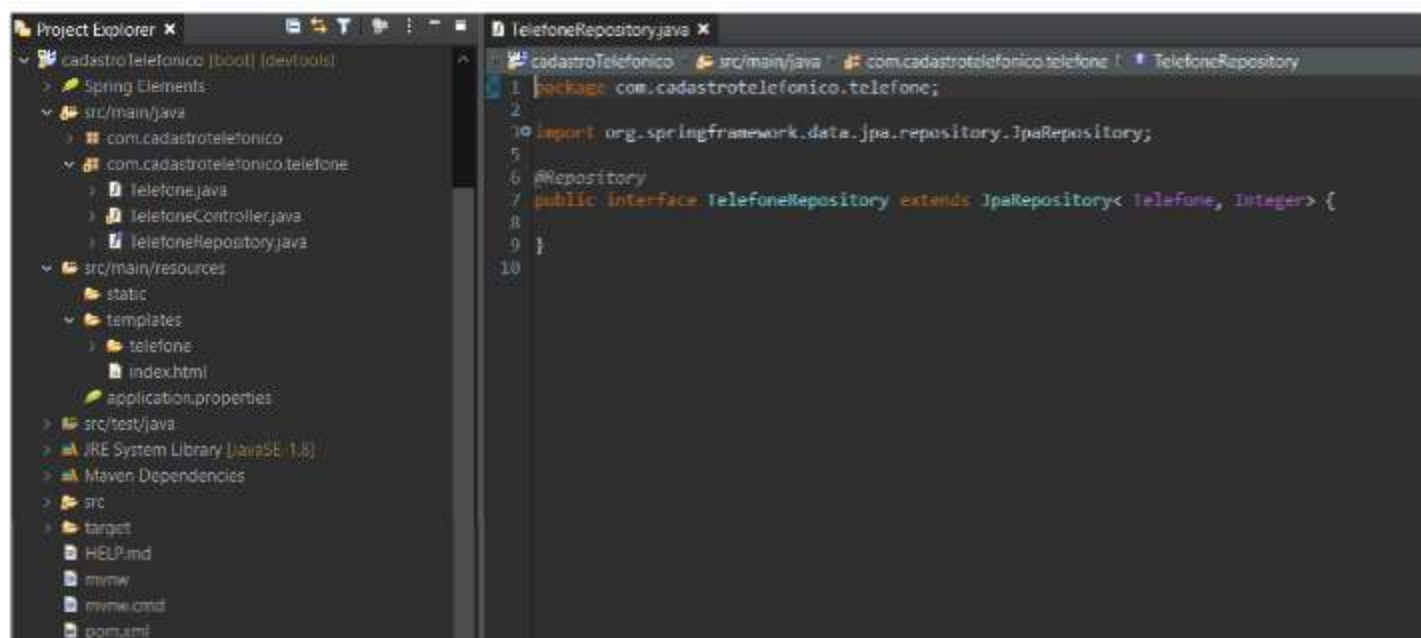
```
    public void setNumeroTelefone(String  
numeroTelefone) {  
        this.numeroTelefone = numeroTelefone;  
    }  
}
```

SPRING

Desafio

#startSpring

Camada de repository - extensão entre os dados e o baco de dados



Após a construção do nosso relacionamento entre a uml e nossa camada model, vamos criar uma interface que irá estender "[extends](#)" a nossa conexão com o banco de dados.

Quando formos realizar a extensão iremos realizar o [extends JpaRepository](#) para ter acesso ao mapeamento.

A interface com a @anotation acima dela, significa que ela terá uma responsabilidade ao ser executada:

[JpaRepository](#) - é um método que terá diversas funções e operações para usarmos no sistema.

SPRING

Desafio #startSpring

A interface repository irá pegar nosso mapeamento e interligar com o nosso banco de dados já criado.

fazendo que a conexão seja simples e tenhamos acesso aos métodos de operação CRUD de maneira simples.

Por exemplo, podemos salvar um objeto com o [.save](#)

Interface Repository

@Repository

```
public interface TelefoneRepository extends  
JpaRepository< Telefone, Integer> {
```

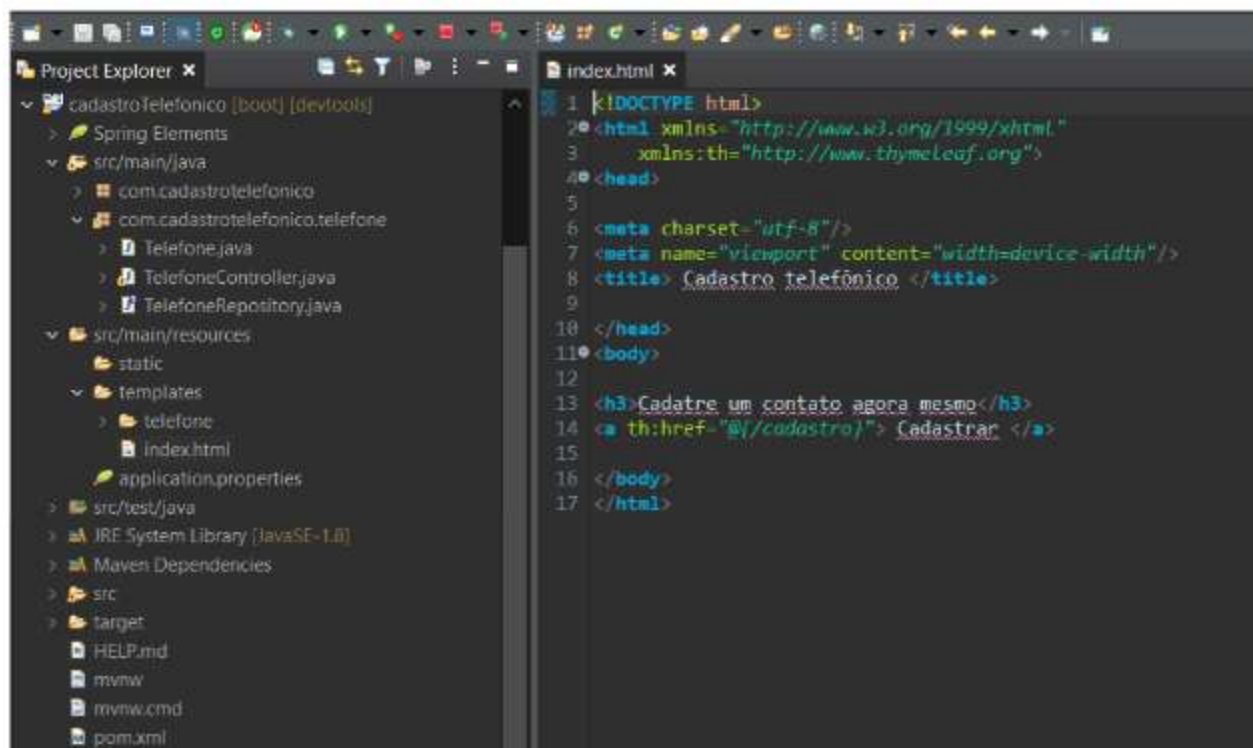
```
}
```



SPRING

Desafio #startSpring

Camada de view - comunicação do usuário com a aplicação



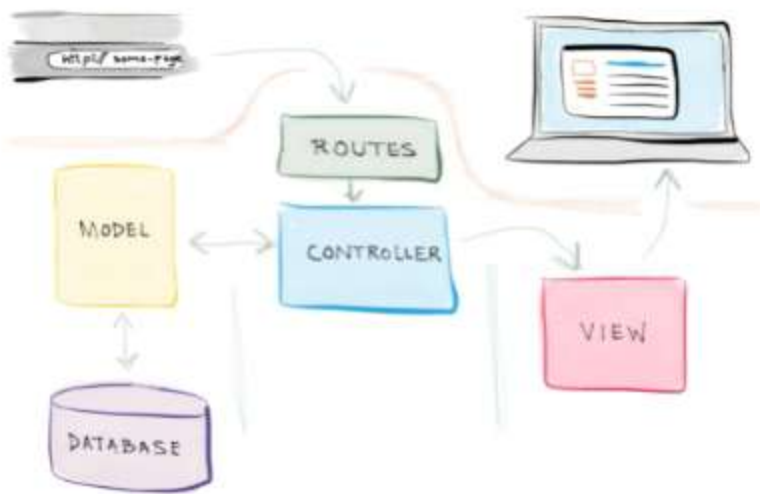
A [index.html](#) é a página raiz de nosso site/sistema. Em sistemas com arquiteturas temos a view que é a camada de visualização e comunicação do usuário com a aplicação.

Neste exemplo nossa camada view é desenvolvida com arquivos html, porém ele tem um [template engine \(THYMELEAF\)](#) ou seja um processador de objetos em java para html através do retorno de dados em XML.

SPRING

Desafio #startSpring

No exemplo da página index, vamos usar ela como porta de apresentação de nosso sistema com um link para uma página de cadastro, com operação create.



Camada view

```
<!DOCTYPE html>
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:th="http://www.thymeleaf.org">
<head>

  <meta charset="utf-8"/>
  <meta name="viewport"
    content="width=device-width"/>
  <title> Cadastro telefônico </title>

</head>
<body>

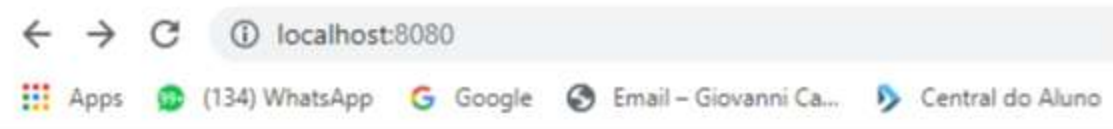
  <h3>Cadatre um contato agora mesmo</h3>
  <a th:href="@{/cadastro}"> Cadastrar </a>

</body>
</html>
```

SPRING

Desafio #startSpring

Arquivo index.html



Cadastre um contato agora mesmo

[Cadastrar](#)

A [index.html](#) é criada dentro de templates, porém ela não fica dentro de um package ou uma pasta, por ser a raiz do projeto.

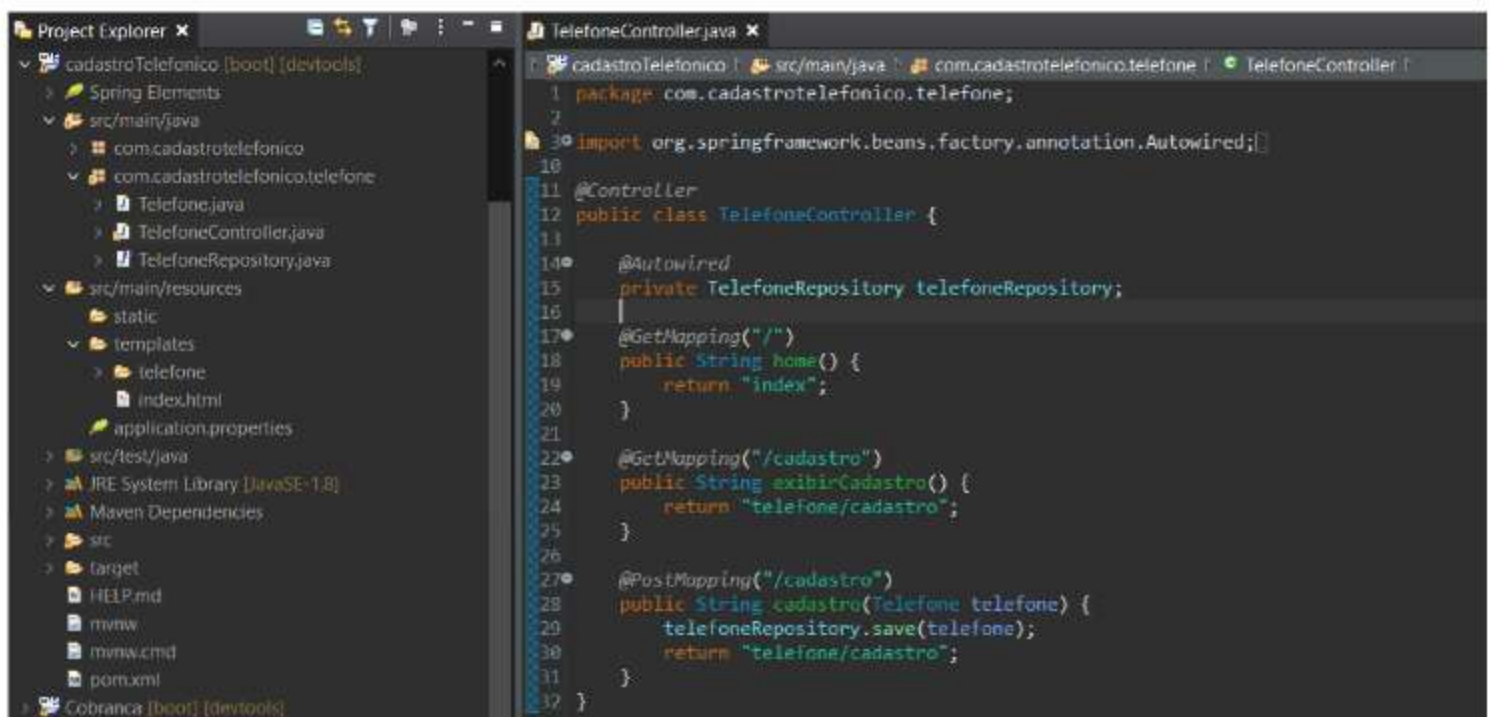
Usando o `<a th:href="@{/cadastro}">Cadastrar` com o processador do thymeleaf vamos rotear ele para uma rota, essa que será criada na classe `TelefoneController.java`

SPRING

Desafio

#startSpring

Camada de controller - roteamento entre a view e os dados do projeto



The screenshot shows an IDE with two panels. The left panel is the Project Explorer, showing the project structure for 'cadastroTelefonico'. The right panel shows the code of 'TelefoneController.java'.

```
1 package com.cadastrotelefonico.telefone;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Controller
6 public class TelefoneController {
7
8     @Autowired
9     private TelefoneRepository telefoneRepository;
10
11     @GetMapping("/")
12     public String home() {
13         return "index";
14     }
15
16     @GetMapping("/cadastro")
17     public String exibirCadastro() {
18         return "telefone/cadastro";
19     }
20
21     @PostMapping("/cadastro")
22     public String cadastro(Telefone telefone) {
23         telefoneRepository.save(telefone);
24         return "telefone/cadastro";
25     }
26 }
```

A controller é a responsável por rotear o projeto através das URI, que são os serviços fornecidos dentro de uma URL completa, ou seja:

URL: www.mentoriotech.com/cadastro

URI: /cadastro

Outro detalhe é a [injeção de dependência](#) onde nós injetamos a conexão que fizemos no repository para a controller, possibilitando o acesso a controller dos métodos que foram realizados o [extends JpaRepository](#), ou seja, vamos implementar as operações CRUD.

SPRING

Desafio #startSpring

Bom, nós já realizamos o mapeamento, conectamos o mapeamento ao banco de dados, criamos a primeira view agora o próximo passo é a apenas rotear isso e implementar o que cada página deve fazer com os dados.

Na index, nós vamos apenas fazer um método que retorna a página.

Na página cadastro, vamos criar 2 métodos, um para mostrar a página e um para que seja realizado enfim o cadastro das informações no sistema.

Camada controller

@Controller

```
public class TelefoneController {
```

```
    @Autowired
```

```
    private TelefoneRepository
```

```
    telefoneRepository;
```

```
    @GetMapping("/")
```

```
    public String home() {
```

```
        return "index";
```

```
    }
```

```
    @GetMapping("/cadastro")
```

```
    public String exibirCadastro() {
```

```
        return "telefone/cadastro";
```

```
    }
```

```
    @PostMapping("/cadastro")
```

```
    public String cadastro(Telefone telefone)
```

```
{
```

```
        telefoneRepository.save(telefone);
```

```
        return "telefone/cadastro";
```

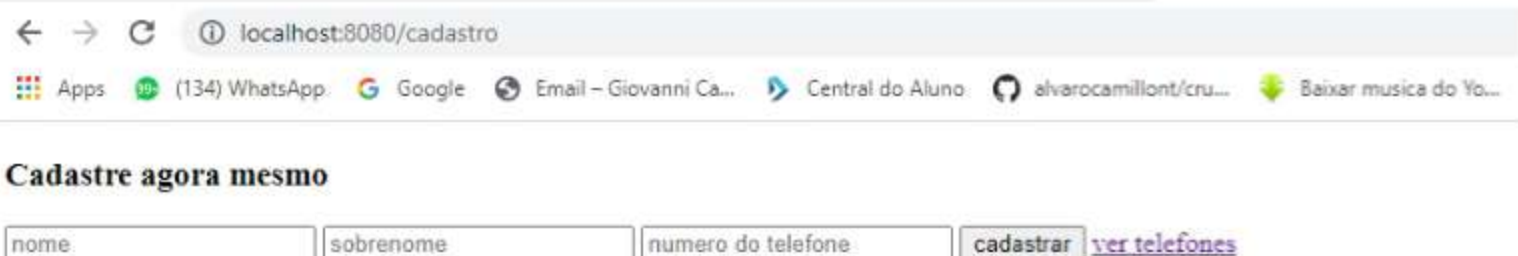
```
    }
```

```
}
```

SPRING

Desafio #startSpring

Arquivo cadastro.html



← → ↻ ⓘ localhost:8080/cadastro

Apps (134) WhatsApp Google Email – Giovanni Ca... Central do Aluno elverocamillont/cru... Baixar musica do Yo...

Cadastro agora mesmo

nome sobrenome numero do telefone [ver telefones](#)

A [cadastro.html](#) é criada dentro de templates > telefone.

Utilizamos um formulário para que seja enviado as informações para nosso banco de dados.

Usando o `th:action="@{/cadastro}" th:object="${telefone}" method="post"` com o processador do thymeleaf vamos usar:

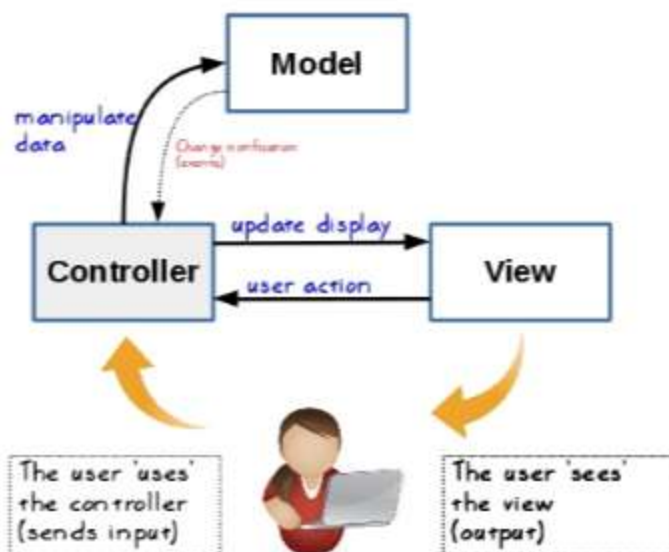
`th:action` para informar qual vai ser a ação daquele formulário.

`th:object` para informar o objeto que será cadastrado.

`method: post` indicando o método HTTP que o formulário vai utilizar.

SPRING

Desafio #startSpring



Camada view

```
<!DOCTYPE html>
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org">
<head>

<meta charset="utf-8"/>
<meta name="viewport"
content="width=device-width"/>
<title> Cadastro telefônico </title>

</head>
<body>

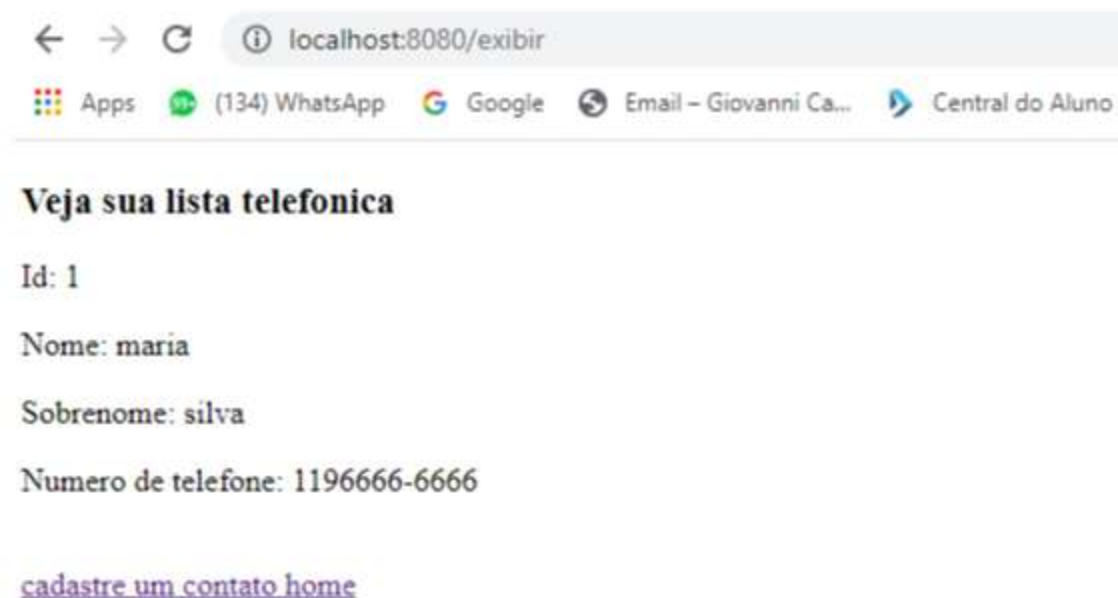
<h3> Cadastre agora mesmo </h3>
<form th:action="@{/cadastro}"
th:object="${telefone}" method="post">
    <input type="text" name="nome"
placeholder="nome" />
    <input type="text"
name="sobrenome"
placeholder="sobrenome" />
    <input type="text"
name="numeroTelefone"
placeholder="numero do telefone" />

    <button type="submit" > cadastrar
</button>
    <a th:href="@{/exibir}"> ver telefones
</a>
</form>
</body>
</html>
```

SPRING

Desafio #startSpring

Arquivo telefone.html



A [telefone.html](#) é criada dentro de templates > telefone.

Utilizamos um for each para realizar a iteração dos dados, ou seja, sempre que ele for no banco, ele irá realizar uma varredura em todos os registros e nos devolver essas informações.

Usando o `th:each="telefones: ${telefone}"` com o processador do thymeleaf.

SPRING

Desafio #startSpring

Camada controller

```
@GetMapping("/exibir")
public ModelAndView listar() {
    ModelAndView model = new
ModelAndView("telefone/telefone");
    Iterable<Telefone> telefone =
telefoneRepository.findAll();
    model.addObject("telefone",
telefone);
    return model;
}
```

Método para devolver a página exibir com todos os dados.

Camada view

```
<!DOCTYPE html>
<html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8"/>
    <meta name="viewport"
content="width=device-width"/>
    <title> Cadastre um telefone</title>
</head>
<body>
    <h3> Veja sua lista telefonica </h3>

    <div th:each="telefones: ${telefone}">
        <div class="row" >
            <p> Id: <span
th:text="${telefones.id}"></span></p>
            <p> Nome: <span
th:text="${telefones.nome}"></span></p>
            <p> Sobrenome: <span
th:text="${telefones.sobrenome}"></span>
</p>
            <p> Numero de telefone: <span
th:text="${telefones.numeroTelefone}">
</span></p>
            <br/>
        </div>
    </div>

    <a th:href="@{/cadastro}"> cadastre
um contato </a>
    <a th:href="@{/}"> home </a>

</body>
</html>
```



Obrigado

Queremos agradecer todo seu empenho e disposição em participar deste projeto que é a mentoria tech uma comunidade construída de alunos para alunos com grandes ambições e objetivos dentre eles temos:

- Fazer com que os alunos alcancem altos níveis em suas formações.
- Mentoria de alunos experientes para apoiar e moldar os alunos mais novos.
- Agregar valor na busca de seu primeiro emprego.
- Produzir e compartilhar conteúdo de qualidade feito por alunos com foco em alunos.
- Disseminar e promover acessibilidade e igualdade em nossos conteúdos ajudando assim toda cota de alunos.
- Realizar encontros com outras comunidades e eventos que gerem impactos positivos como networking, experiências, preposições técnicas entre outros.



Busque nossas redes sociais



+55 (11) 9 61985346



contatomentoriatech@gmail.com



@mentoriatech



www.mentoriatech.com.br

Bootcamp



ACESSE AGORA

Siga nossas mídias sociais
[@mentoriatech](#)