

Curso

Aplicações JAVA com SPRING BOOT



Prof. Msc. Antonio B. C. Sampaio Jr
engenheiro de software & professor

@abctreinamentos
@amazoncodebr

www.abctreinamentos.com.br
www.amazoncode.com.br



CONTEÚDO PROGRAMÁTICO



- UNIDADE 1 – INTRODUÇÃO
- UNIDADE 2 – FUNDAMENTOS DO SPRING BOOT
- UNIDADE 3 – PERSISTÊNCIA DE DADOS NO SPRING BOOT
- UNIDADE 4 – PROJETO WEB NO SPRING BOOT
- UNIDADE 5 – PROJETO REST API NO SPRING BOOT
- UNIDADE 6 – PROJETO REST API NO SPRING BOOT COM REACTJS

PROJETOS DO CURSO



- 1º Projeto Spring Boot – Impressão de Mensagens
- 2º Projeto Spring Boot – Impressão de Mensagens na WEB
- 3º Projeto Spring Boot – Aplicação Servidor Público
- 4º Projeto Spring Boot – Aplicação Servidor Público na WEB
- 5º Projeto Spring Boot – Aplicação Servidor Público no SGBD MYSQL
- 6º Projeto Spring Boot – Aplicação Servidor Público no MONGO DB

PROJETOS DO CURSO



- 7º Projeto Spring Boot – Aplicação Servidor Público WEB
- 8º Projeto Spring Boot – Aplicação Servidor Público REST API e MySQL
- 9º Projeto Spring Boot – Aplicação Servidor Público REST API com REACT
- 10º Projeto Spring Boot – Aplicação Servidor Público/Curso REST API – Monolítico
- 11º Projeto Spring Boot – Aplicação Servidor Público REST API - Microserviços

UNIDADE 4

PROJETO WEB NO SPRING BOOT

INTERNET E WEB

PADRÃO MVC

THYMELEAF

SPRING MVC

ARQUITETURA SPRING BOOT

PROJETO PRÁTICO

7º Projeto Spring Boot – Aplicação Servidor
Público WEB com Thymeleaf e MySQL

INTERNET E
WEB

INTERNET

- **O que é a INTERNET?**

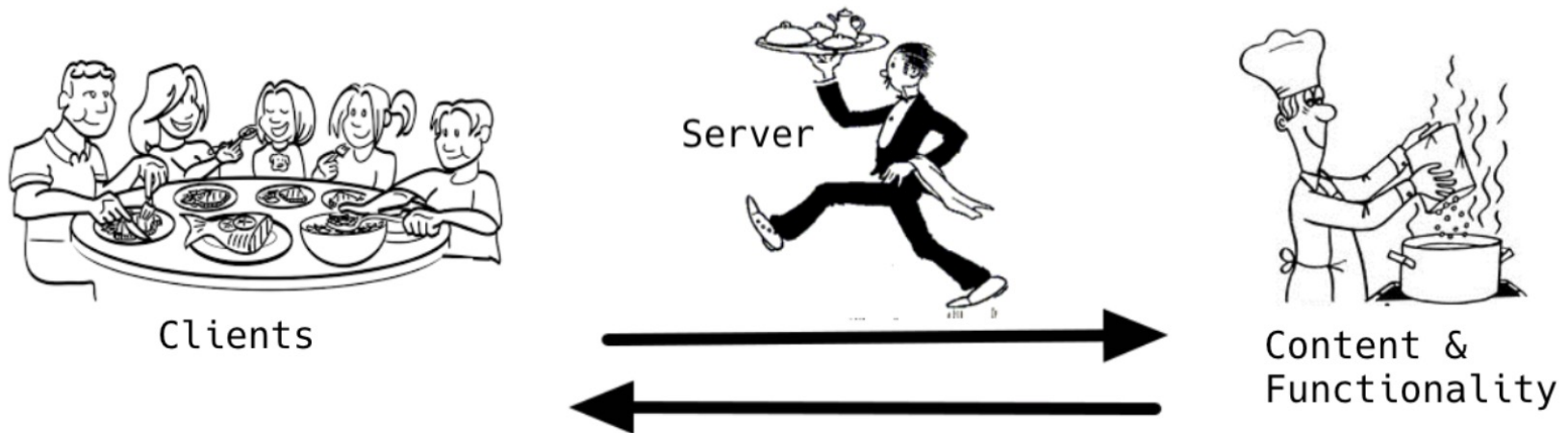
- A internet é uma rede mundial de computadores interconectados que permite a comunicação e o compartilhamento de informações entre usuários em todo o mundo. Ela é composta por milhões de dispositivos, servidores e roteadores, que trocam dados e informações usando um conjunto de protocolos de comunicação comuns.



INTERNET

- **Arquitetura Cliente / Servidor**

- Os bilhões de dispositivos ligados à Internet podem ser classificados como **clientes** ou **servidores**. Os **Clientes** solicitam conteúdos/serviços; os **Servidores** provêm esses conteúdos/serviços.



INTERNET

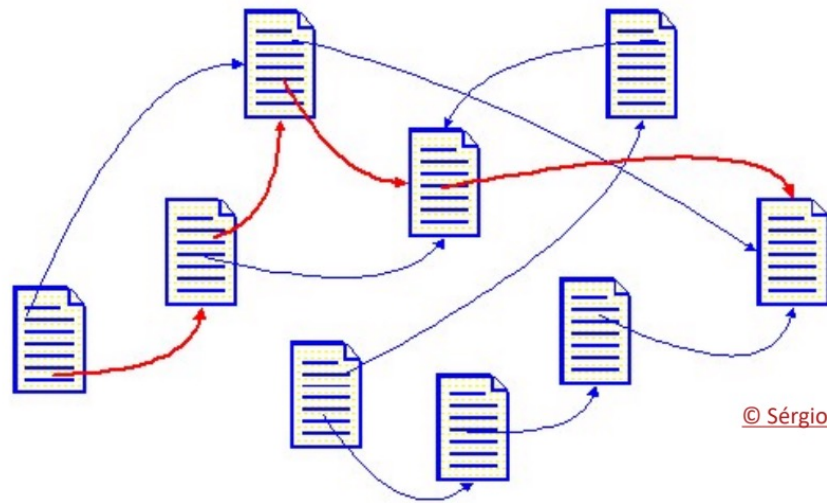
- Principais Serviços
 - Mídias Sociais
 - World Wide Web
 - Correio eletrônico
 - Acesso remoto
 - Colaboração
 - Compartilhamento de arquivos
 - *Streamming*
 - VOIP



WEB

- **Definição**

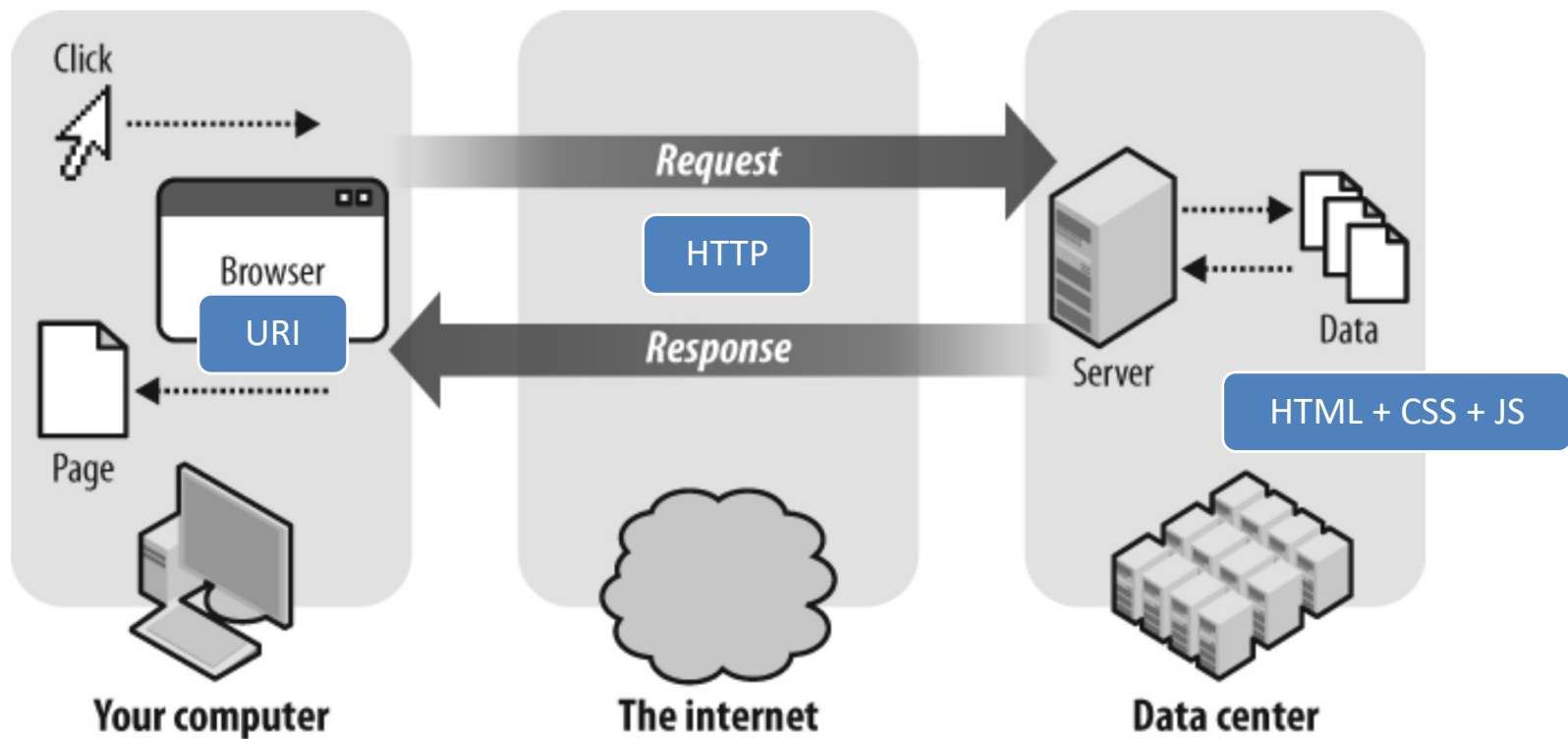
- A **World Wide Web (WWW ou WEB)** é um sistema de documentos em hipertexto que são interligados e acessados pela internet. Através da WWW, os usuários podem acessar uma ampla variedade de informações e recursos, como sites de notícias, redes sociais, blogs, fóruns e muito mais.



© Sérgio Sobral Nunes

WEB

- Arquitetura Cliente / Servidor



PRINCIPAIS TECNOLOGIAS DA WEB

- **URL**

- **URL (*Uniform Resource Locator*)** é uma sequência de caracteres que identifica um recurso na internet de maneira única e universal. Ela pode ser usada para identificar recursos como páginas da web, imagens, arquivos, serviços web, entre outros.

- **HTTP**

- **HTTP (*Hypertext Transfer Protocol*)** é um protocolo de comunicação utilizado para transferência de dados pela World Wide Web (www). Ele define a forma como as mensagens são formatadas e transmitidas entre os servidores e os clientes, permitindo o acesso a recursos como páginas web, imagens, vídeos, arquivos, entre outros, disponíveis na internet.

PRINCIPAIS TECNOLOGIAS DA WEB

- **HTML**

- **HTML** (*Hypertext Markup Language*) é a linguagem padrão usada para criar páginas da web na internet. É uma linguagem de marcação, o que significa que consiste em um conjunto de tags de marcação que são usadas para estruturar o conteúdo em uma página da web.

- **CSS**

- O **CSS** (*Cascading Style Sheets*) é uma linguagem de estilo utilizada para definir a aparência e o layout de uma página da web, incluindo cores, fontes, tamanhos de texto, posicionamento de elementos e outras características visuais.

PRINCIPAIS TECNOLOGIAS DA WEB

- **JAVASCRIPT**

- **JavaScript** é uma linguagem de programação de alto nível que é usada principalmente para criar interatividade em páginas da web e desenvolver aplicativos web.
- Também é utilizada em desenvolvimento de aplicativos do lado do servidor, através do **Node.js**, que permite aos desenvolvedores usar JavaScript em ambientes de servidor.

URL

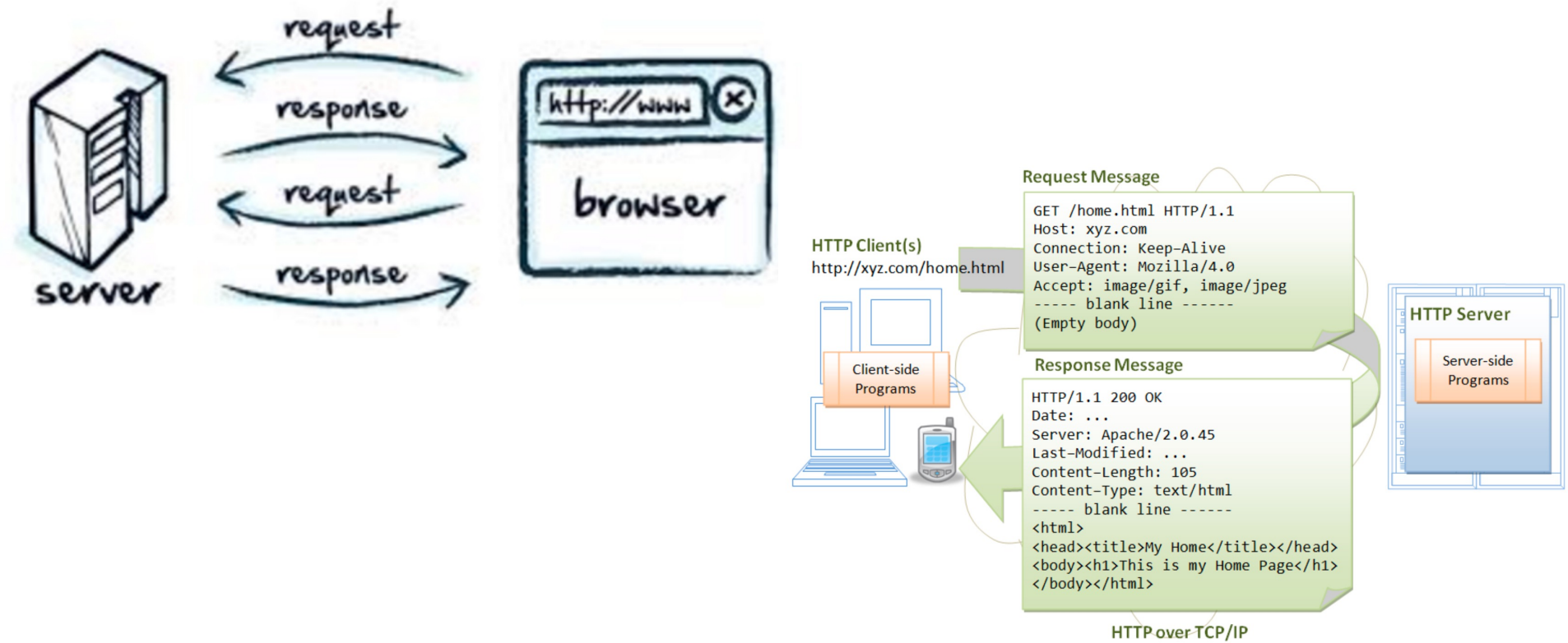
- **Sintaxe Padrão**

[illegible]

- **<protocolo>** é o esquema de comunicação usado para acessar o recurso, como o protocolo “http”.
- **<domínio>** é o endereço do servidor que hospeda o recurso: “www.example.org”.
- **<caminho>** é o caminho para o recurso no servidor: “/hello/world/foo.html”.

HTTP

- Modelo Requisição / Resposta



HTTP

- **Requisições HTTP**

- As requisições HTTP definem as ações que podem ser realizadas sobre um recurso identificado pela URI em uma aplicação Web. Existem **oito verbos** definidos pelo protocolo HTTP:
 - **GET**: solicita a representação de um recurso identificado pela URI;
 - **POST**: envia uma entidade para ser processada pelo recurso identificado pela URI;
 - **PUT**: substitui todas as atuais representações do recurso identificado pela URI pela entidade que acompanha a solicitação;
 - **DELETE**: remove o recurso identificado pela URI;
 - **HEAD**: solicita uma resposta que inclua apenas cabeçalhos.

HTTP

- **Requisições HTTP**

- **OPTIONS:** solicita informações sobre as opções de comunicação disponíveis para o recurso identificado pela URI.
- **TRACE:** solicita um loopback de diagnóstico da mensagem solicitada.
- **CONNECT:** inicia uma conexão em túnel para o servidor identificado pela URI.

HTTP

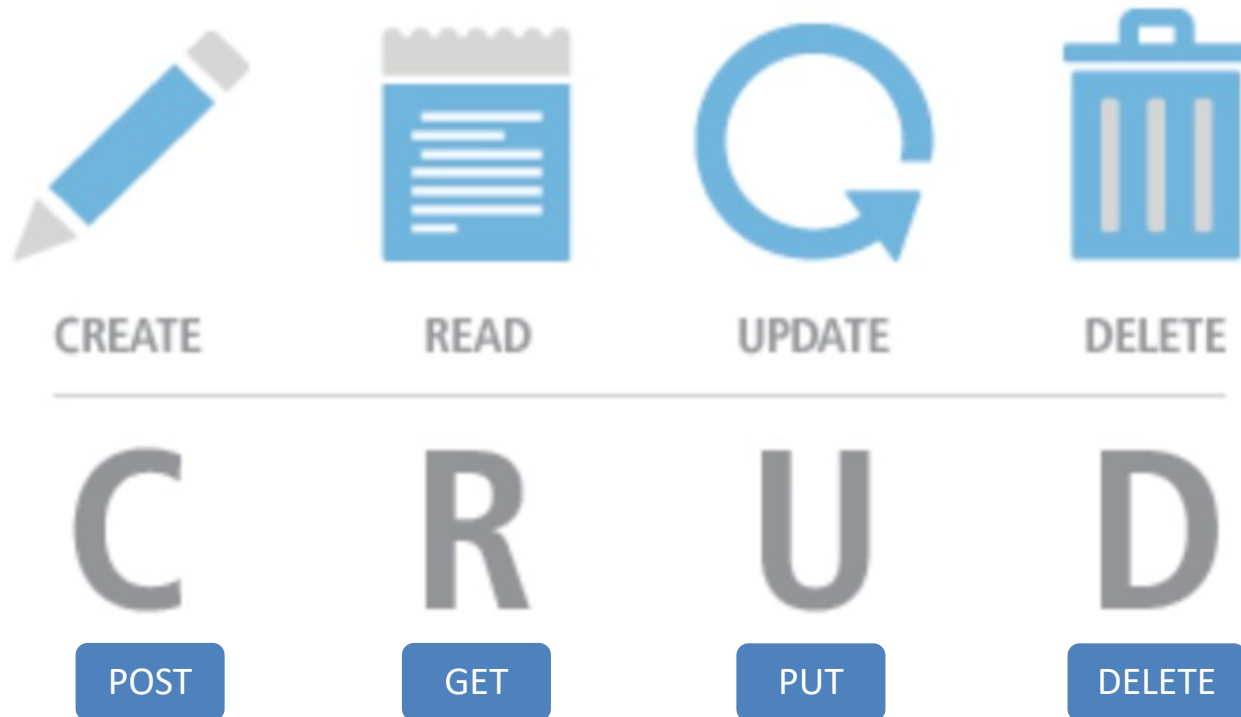
- Respostas HTTP



Code	Message
200	OK
301	Moved Permanently
302	Moved Temporarily
404	Not Found
500	Internal Server Error
503	Service Unavailable

HTTP

- Verbos HTTP e CRUD



HTTP

- Acesso à API via HTTP

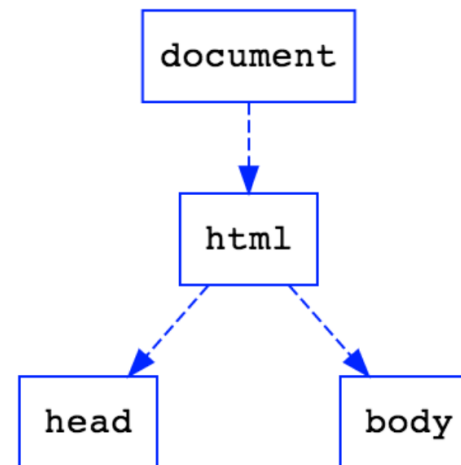
Endpoint	Método	Ação
/users	GET	Retorna a lista de usuários
/users	POST	Insere um novo usuário
/users/{id}	GET	Retorna o usuário com id = {id}
/users/{id}	PUT	Substitui os dados do usuário com id = {id}
/users/{id}	DELETE	Remove o usuário com id = {id}

HTML

- **Estrutura de um Documento HTML**

- Todo documento HTML é representado por uma estrutura de árvore conhecida como DOM (*Document Object Model*) que contém todos os seus elementos.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>My first document</title>
5      <meta charset="UTF-8" />
6    </head>
7    <body>
8      ...
9    </body>
10 </html>
```



HTML

- Principais Elementos

<i>Tags Base</i>		<i>Formatação de texto</i>	
<!DOCTYPE>	tipo de documento		Texto em negrito
<html>	Início e fim do documento	<i>	Define texto em itálico
<body>	Corpo do documento		Dá ênfase ao texto (itálico)
<h1> to <h6>	Cabeçalho do documento	<a>	Define uma âncora
<p>	Parágrafo		Dá ênfase ao texto (negrito)

	Insere uma quebra de linha no texto	<small>	Define um texto pequeno
<hr />	linha Horizontal	<sup>	Define um texto sobrescrito
<!-- ... -->	Define um comentário	<sub>	Define um texto subscrito

HTML

- **Formulários**

- O uso de formulários é o meio mais simples para a prestação de serviços interativos na WEB.
- O usuário precisa ter uma boa experiência, por esse motivo, torna-se imprescindível que o formulário seja **compreensível, não só visualmente, mas também na sua formatação interna do HTML.**

Nome(s):		RG:	Nº de Matrícula
<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>		<input type="text"/>	<input type="text"/>
<input type="text"/>		<input type="text"/>	<input type="text"/>
Telefone:	Celular:	E-mail:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	
Instituição:		Curso:	
<input type="text"/>		<input type="text"/>	

HTML

- Formulários – Principais Elementos

form: o contêiner do formulário;

```
<form>
...
</form>
```

Input: especifica os campos de um formulário como: **text**, **submit**, **button**, **radio**, **checkbox**, **file**, **image**, **reset**;

```
<input type="text" name="Nome" />
<input type="checkbox" name="checkbox" />
<input type="radio" name="radiobutton" />
```

textarea:
campo de texto multi-linha;

```
<textarea name="critSugest"></textarea>
```

select: utilizado para listas selecionáveis;

```
<select name="select">
  <option value="1">Unidade 1</option>
  ...
</select>
```

button: botão do formulário.

```
<input type="button" name="enviar" />
```

CSS

- **CSS**
 - Folhas de Estilos (*Cascading Style Sheets* ou CSS) é uma linguagem utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento.
 - Ao invés de colocar a formatação dentro do documento, o desenvolvedor cria um link (ligação) para uma página que contém os estilos, procedendo de forma idêntica para todas as páginas de um portal. Quando quiser alterar a aparência do portal basta portanto modificar apenas um arquivo.

CSS

- Sintaxe Básica

```
#geral {width:700px;}
```

seletor { **propriedade**: **valor** }

↑
Em qual tag(s)
HTML será
aplicada a
propriedade
(p. ex.: "body")

↑
A propriedade pode
ser por exemplo: cor
do fundo
("background-color")

↖ O valor da
propriedade cor do
fundo, pode ser por
exemplo: vermelha
("#FF0000")

CSS

- Exemplo de Uso

Código HTML:

```
<div id="geral">
  <h1>Layout com 3 Colunas</h1>
  <div id="esquerda">
    <p>Conteúdo da coluna da esquerda</p>
  </div>
  <div id="direita">
    <p>Conteúdo da coluna da direita</p>
  </div>
  <div id="conteudo">
    <p>Conteúdo</p>
  </div>
</div>
```

Código CSS:

```
#geral {width:700px;}
#esquerda, #direita {width:150px;}
#conteudo {width:370px;}
#esquerda {float:left;}
#direita {float:right;}
#conteudo {margin-left:160px;}
```

JAVASCRIPT

- **JAVASCRIPT**

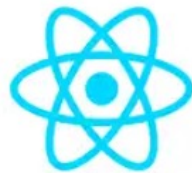
- Javascript é uma linguagem de programação popular e amplamente usada na criação de aplicativos da web, bem como em outras áreas, como aplicativos móveis, jogos e desenvolvimento de desktop.
- JavaScript é frequentemente usado em conjunto com outras tecnologias da web, como HTML e CSS, para criar aplicativos da web interativos e dinâmicos. Ele pode ser usado para validar formulários, criar animações, manipular o conteúdo da página e interagir com APIs de terceiros.

JAVASCRIPT

- **JAVASCRIPT**

- Além disso, JavaScript tem uma ampla variedade de bibliotecas e frameworks disponíveis, como **ReactJS**, **Vue JS** e **Angular**, que ajudam a tornar o desenvolvimento da web mais fácil e eficiente.

REACT JS



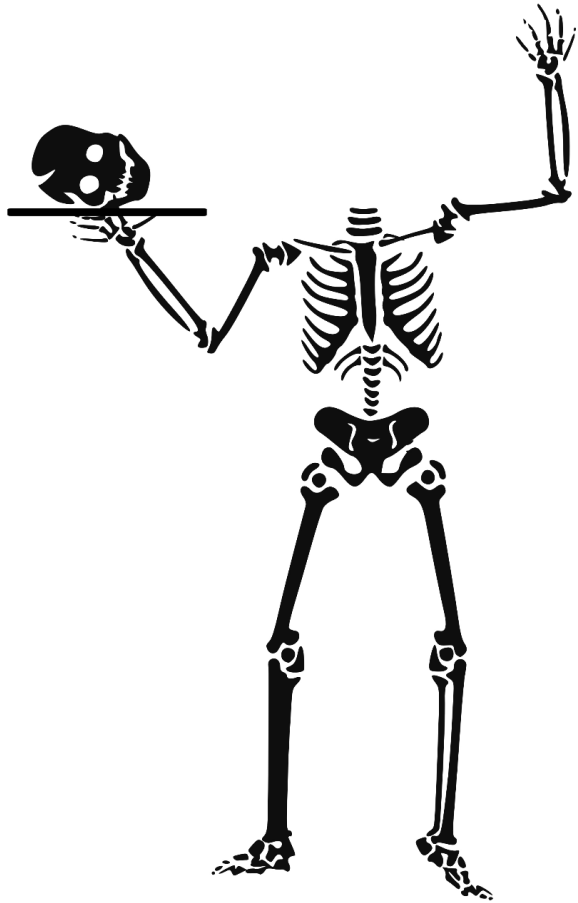
ANGULAR



VUE JS



ANATOMIA DE UMA PÁGINA WEB



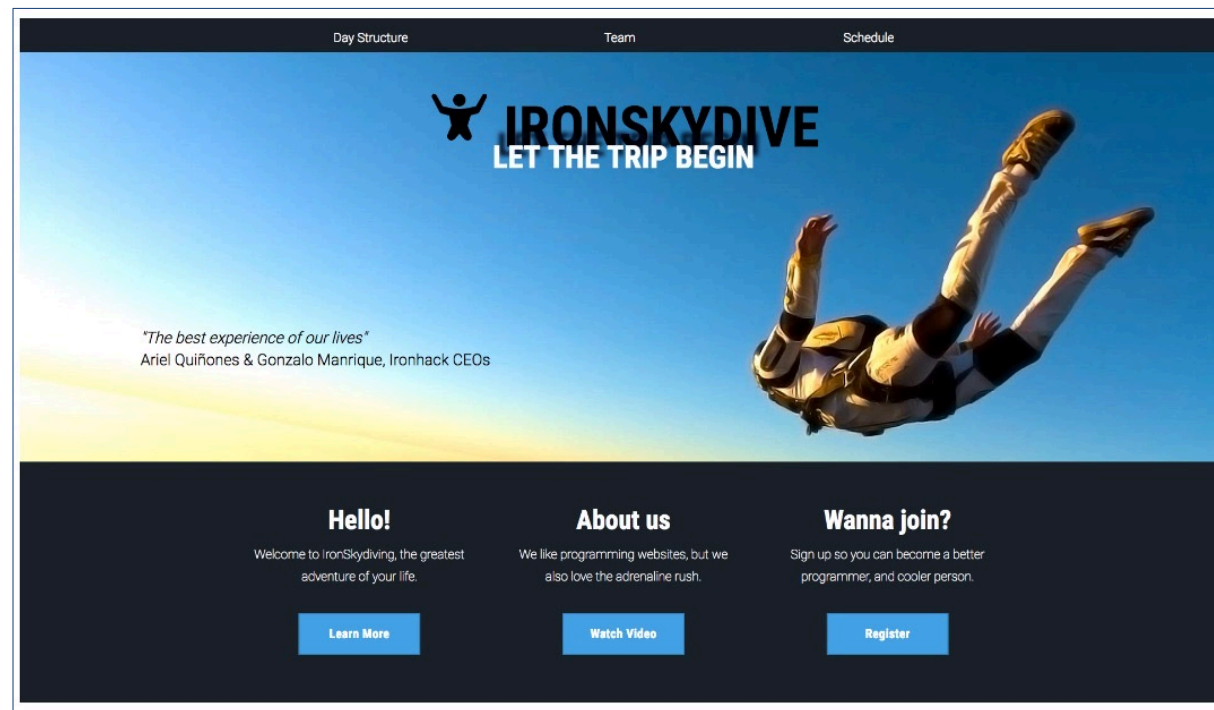
Os “ossos”: HTML

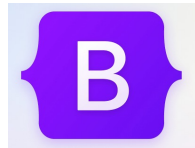
A “pele/cabelo/roupas”: CSS

Os “músculos”: JavaScript

FRONT-END

- **HTML, CSS e JavaScript**
 - Formam o grupo de tecnologias que definem o **front-end** de uma aplicação WEB.





BOOTSTRAP

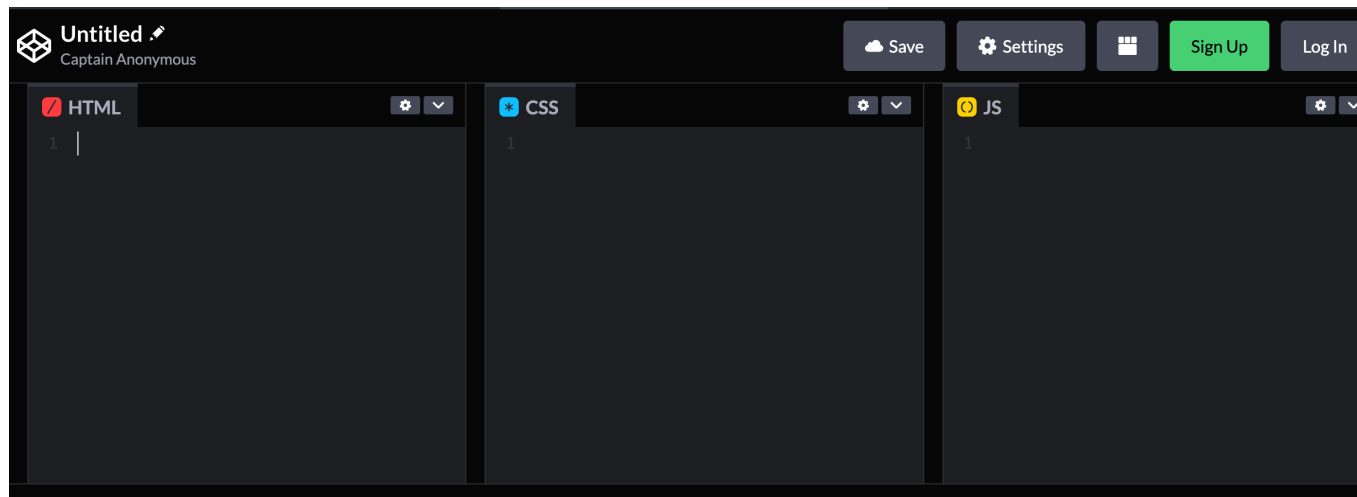
- **Definição**

- É um framework de **front-end** para desenvolvimento de sites e aplicativos da web responsivos e *mobile-first*. Ele inclui um conjunto de componentes HTML, CSS e JavaScript pré-construídos que permitem criar rapidamente páginas da web com uma aparência consistente e profissional em uma variedade de dispositivos e tamanhos de tela.
- Os componentes do Bootstrap incluem botões, formulários, tabelas, tipografia, navegação, paginadores e muito mais. Ele também fornece uma variedade de utilitários de classe, como espaçamento e margem, cores, posicionamento e exibição, para ajudar na criação de layouts personalizados.

CODEPEN

- **Definição**

- CodePen é uma plataforma online que permite aos desenvolvedores de front-end compartilhar e testar código HTML, CSS e JavaScript em tempo real. É uma ferramenta útil para prototipagem rápida e colaboração, bem como para aprender novas técnicas de codificação.

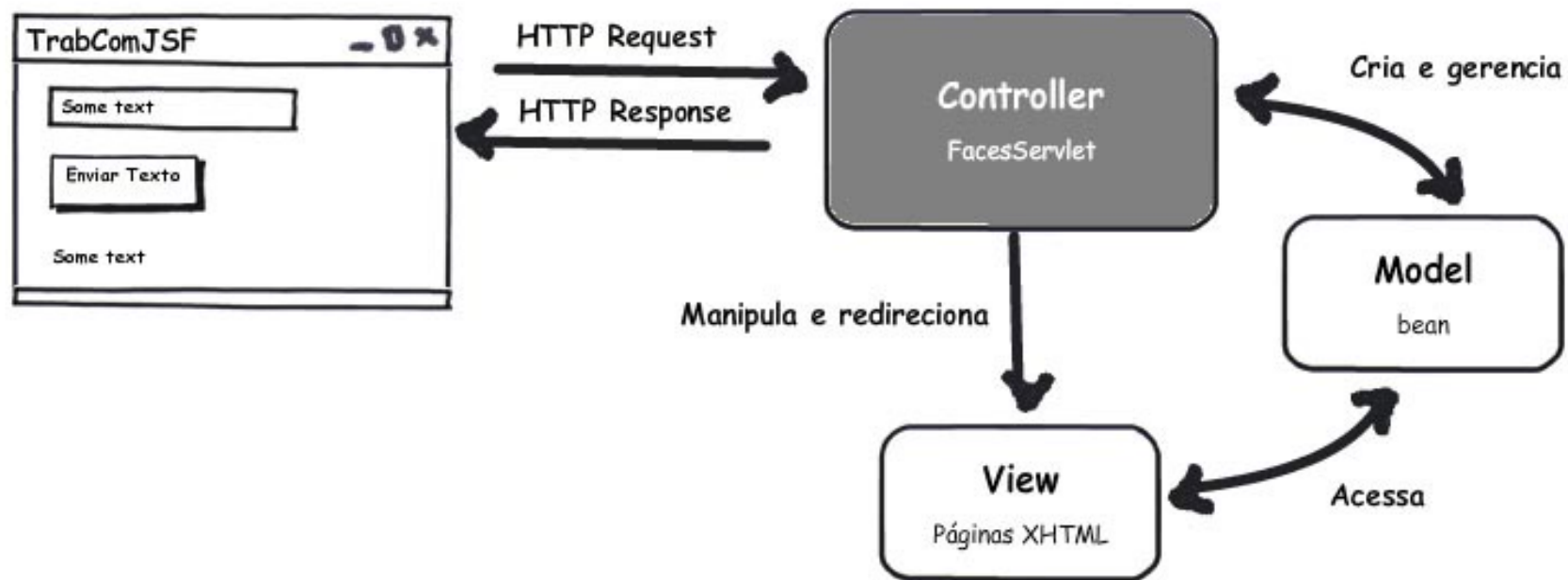


<https://codepen.io/pen/>

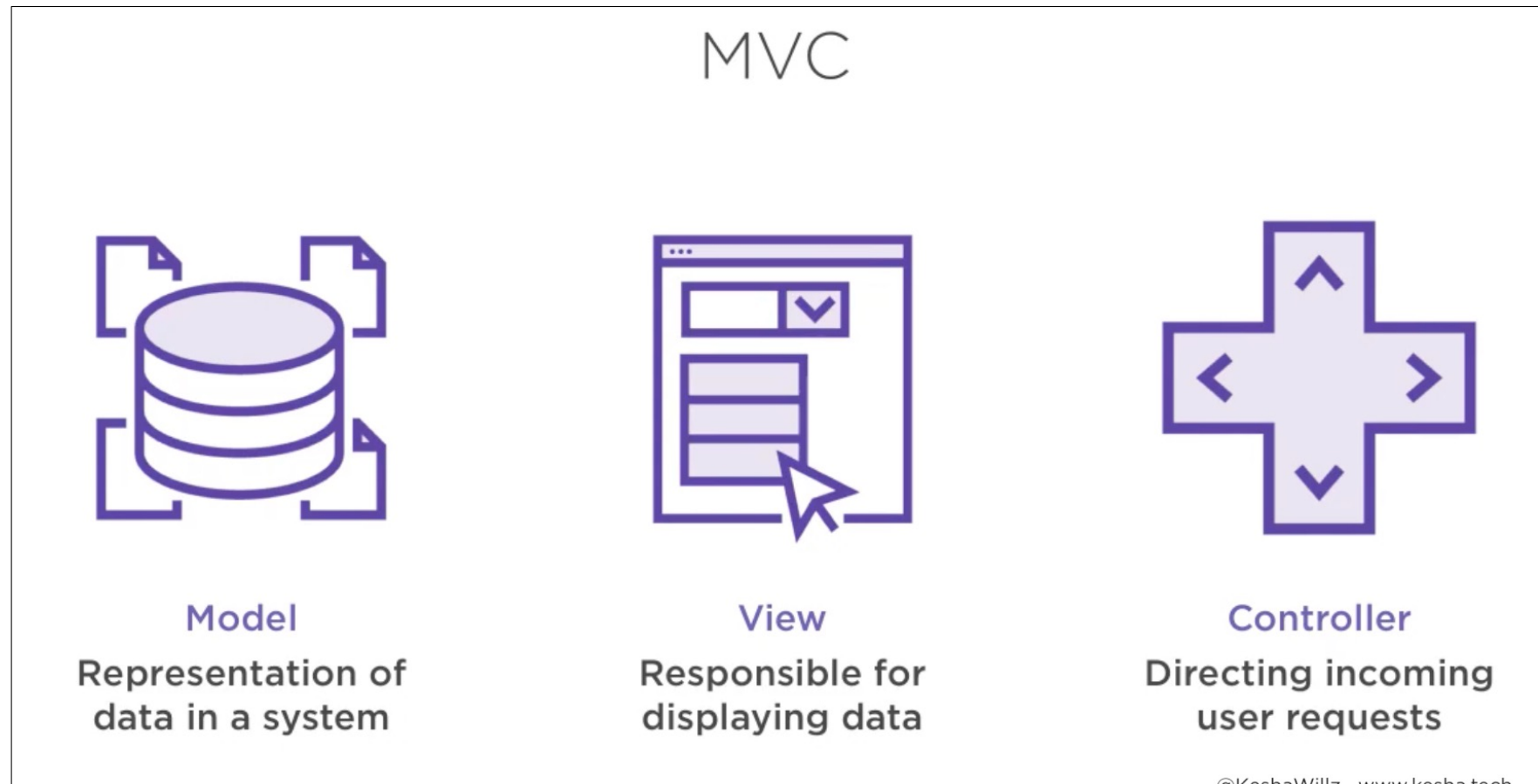
PADRÃO MVC

PADRÃO MVC

- O **MVC** é um padrão de projeto que separa a lógica da aplicação WEB em 03 camadas: **Model**, **View** e **Controller**.



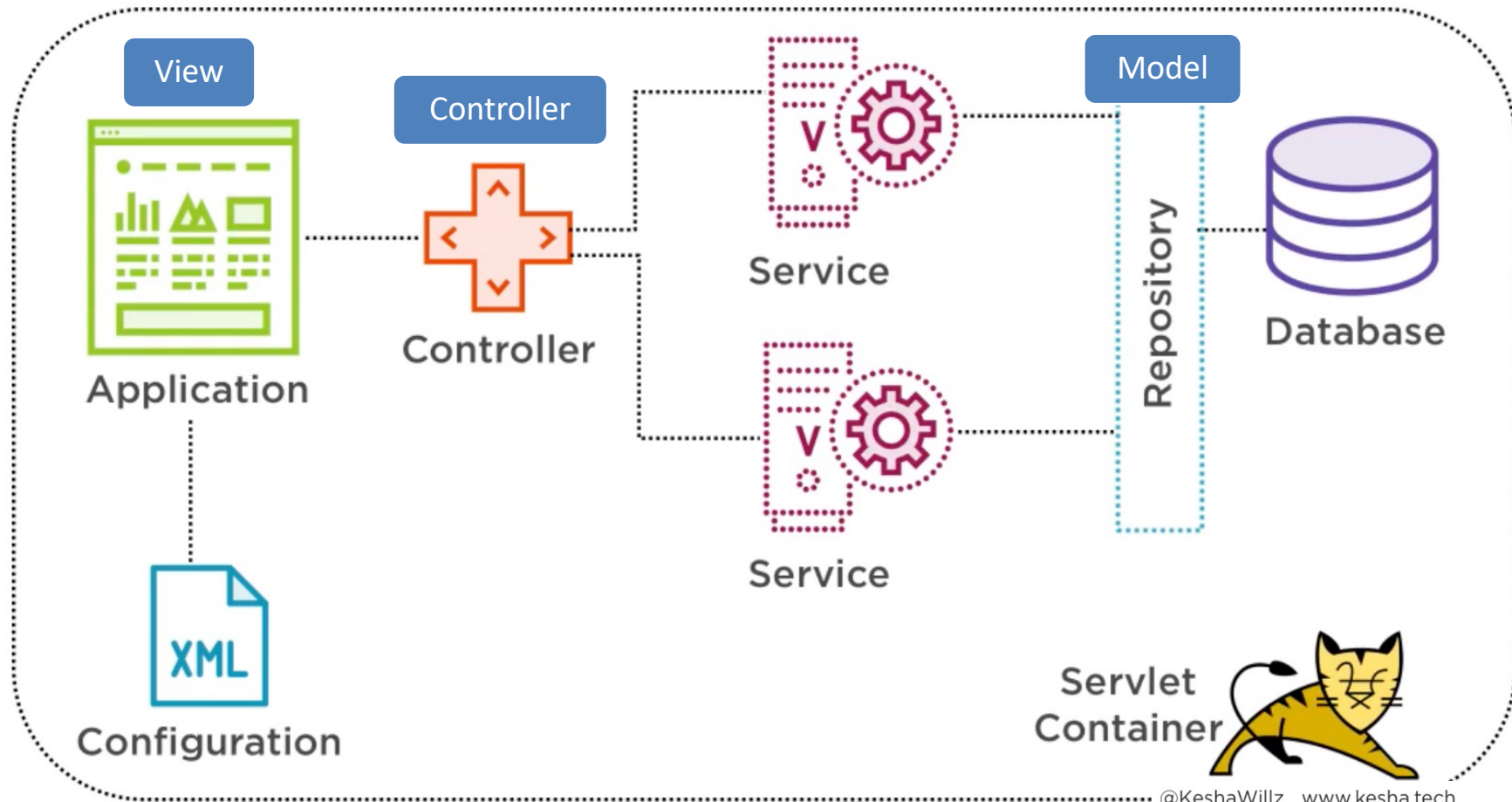
PADRÃO MVC



VANTAGENS DO MVC

- É fácil de manter, testar e atualizar.
- Aumenta a produtividade da equipe.
- A aplicação é naturalmente escalável.
- Mudanças no **Controller** não afetam os componentes do **View**.
- Mudanças no **View** não afetam as classes do **Controller**.

MVC NA ARQUITETURA SPRING BOOT



THYMELEAF



- **Definição**

- O Thymeleaf é uma template engine para projetos Java que facilita a criação de páginas HTML. Ele gera páginas HTML no lado servidor de forma dinâmica, permitindo a troca de informações entre o código Java e as páginas HTML.
- É muito utilizado em todo ecossistema Java, principalmente nas aplicações Spring Web MVC, possuindo uma dependência específica para o Spring Boot.

Dependencies

ADD DEPENDENCIES... ⌘ + B

Thymeleaf **TEMPLATE ENGINES**

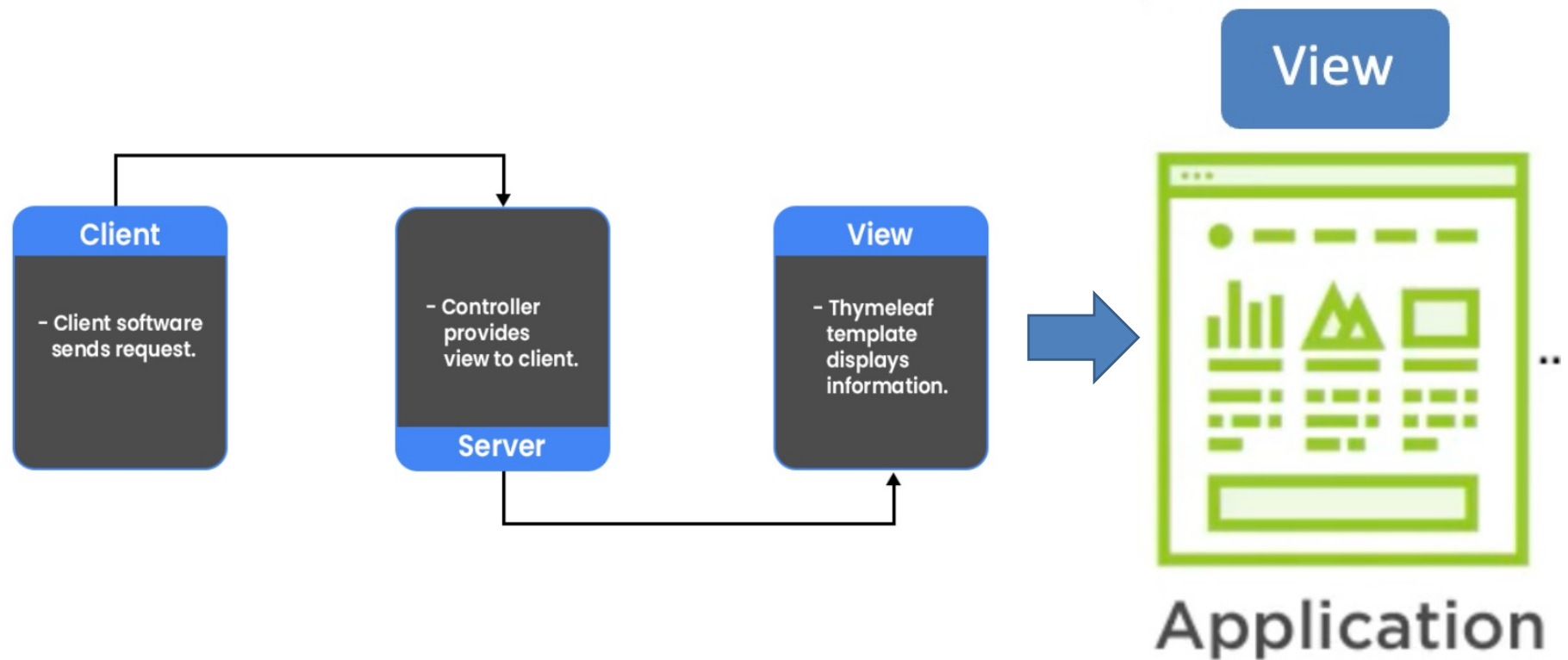
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.



USO DO
THYMELEAF

USO DO
BOOTSTRAP

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      crossorigin="anonymous">
</head>
<body>
...
  <tbody>
    <tr th:each="servidor : ${servidorespublicos}">
      <td></td>
      <td th:text="${servidor.nome}"></td>
      <td th:text="${servidor.cargo}"></td>
      <td>
        <div class="btn-group" role="group" aria-label="Ações">
          <a th:href="@{'/listarServidor/' + ${servidor.matricula}}" class="btn
            btn-primary">Detalhar</a>
        </div>
      </td>
    </tr>
```





- **Arquivo APPLICATION.PROPERTIES**

- O arquivo **application.properties** é um arquivo de configuração utilizado pelo Spring Boot para configurar diferentes aspectos da aplicação, como conexão com banco de dados, configurações de servidor web, etc.
- No Thymeleaf, as configurações para o mecanismo de template devem ser definidas no arquivo **application.properties**.

```
# ThymeLeaf  
spring.thymeleaf.template-loader-path: classpath:/templates  
spring.thymeleaf.suffix: .html  
spring.thymeleaf.cache: false
```

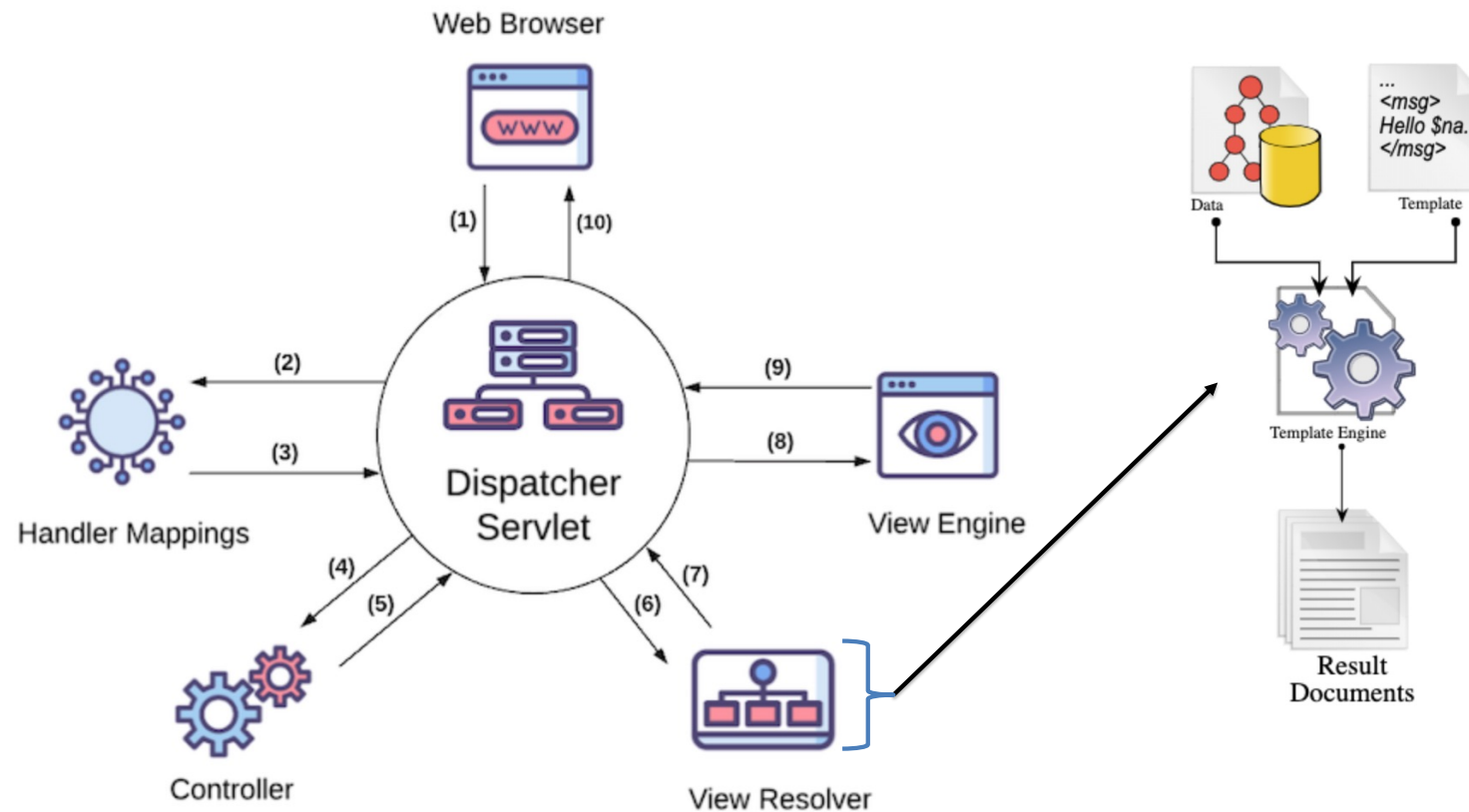
SPRING MVC

SPRING MVC

- **Definição**

- O **Spring MVC** é um módulo do Spring Framework que implementa o padrão MVC, combinando as vantagens de utilizar esse padrão de projeto com a robustez da família de tecnologias Spring.
- O Spring implementa o MVC utilizando o componente **Dispatcher Servlet** que irá gerenciar as solicitações HTTP recebidas pelo servidor da aplicação. Ele atua como um controlador frontal (*front controller*) e é responsável por encaminhar as solicitações para o controlador adequado e, em seguida, retornar a resposta apropriada ao navegador do usuário.

SPRING MVC



<https://openclassrooms.com/en/courses/5684146-create-web-applications-efficiently-with-the-spring-boot-mvc-framework/6124741-follow-spring-mvc-architecture-to-turn-static-html-into-a-thymeleaf-template>

SPRING MVC

- **Vantagens**

- O Spring MVC oferece recursos poderosos, como a injeção de dependência, a validação de dados, a autenticação e autorização, entre outros recursos avançados. Por isso é amplamente utilizado em projetos com o Spring Boot, devido à sua escalabilidade, modularidade e facilidade de integração com outras tecnologias.
- Além disso, possui uma comunidade ativa de desenvolvedores e uma documentação bastante abrangente.

SPRING MVC

- Para utilização do **Spring MVC** em um projeto Spring Boot, faz-se necessário a utilização do **spring-boot-starter-web**.

spring-boot-starter-web

```
<artifactId>  
    spring-boot-starter-web  
</artifactId>
```

◀ Spring MVC

◀ REST

◀ Tomcat

◀ Jackson

- Com **Spring MVC** configurado em um projeto Spring Boot é possível criar aplicações WEB escaláveis e flexíveis, usando uma abordagem baseada em componentes.



ANOTAÇÕES SPRING MVC

- Principais Anotações

- **@Controller**: é uma anotação utilizada na classe para criar um controlador (*controller*) que irá receber as requisições dos clientes (*requests*) e enviar as respostas (*responses*).

```
@Controller  
public class ServidorPublicoController {  
    //métodos para receber as requisições  
}
```



ANOTAÇÕES SPRING MVC

- Principais Anotações

- **@RequestMapping**: é uma anotação utilizada nos métodos que irão receber as requisições dos clientes.
- Pode ser usada para mapear solicitações HTTP de qualquer tipo (**GET, POST, PUT, DELETE**, etc.)

```
@Controller
public class ServidorPublicoController {

    @RequestMapping("/listarServidores")
    public String listarServidores(Model model)
    {
        model.addAttribute("servidorespublicos", servidorService.listAll());
        return "servidorespublicos";
    }
}
```



Menu - SISCAPACIT Servidor Público ▾

Listagem dos Servidores Públicos

Foto	Nome	Cargo	Órgão	Lotação	Email	Ações
	Caio Santos	Engenheiro	DNIT	São Paulo	caio@dnit.gov.br	Detalhar Alterar Excluir
	Antonio Teixeira	Auditor-Fiscal	RFB	Rio de Janeiro	antonio@rfb.gov.br	Detalhar Alterar Excluir

servidorespublicos.html



ANOTAÇÕES SPRING MVC

- Principais Anotações

- @GetMapping**: é uma anotação utilizada nos métodos que irão receber as requisições **HTTP GET**.


```
@Controller
public class ServidorPublicoController {

    @GetMapping("/listarServidores")
    public String listarServidores(Model model)
    {
        model.addAttribute("servidorespublicos", servidorService.listAll());
        return "servidorespublicos";
    }
}
```



Menu - SISCAPACIT Servidor Público ▾

Listagem dos Servidores Públicos

Foto	Nome	Cargo	Órgão	Lotação	Email	Ações
	Caio Santos	Engenheiro	DNIT	São Paulo	caio@dnit.gov.br	Detalhar Alterar Excluir
	Antonio Teixeira	Auditor-Fiscal	RFB	Rio de Janeiro	antonio@rfb.gov.br	Detalhar Alterar Excluir

servidorespublicos.html



ANOTAÇÕES SPRING MVC

- Principais Anotações

- **@PostMapping**: é uma anotação utilizada nos métodos que irão receber as requisições HTTP POST.

```
@Controller
public class ServidorPublicoController {

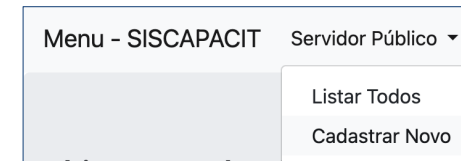
    @PostMapping("/cadastrarServidor")
    public String save(@ModelAttribute ServidorPublico novoServidor)
    {
        servidorService.save(novoServidor);
        return "redirect:/listarServidores";
    }
}
```



ANOTAÇÕES SPRING MVC

- **Interface Model**

- O **Model** é uma interface do **Spring MVC** que representa o modelo de dados em um aplicativo da web.
- O Model é um objeto que armazena dados no escopo de uma solicitação HTTP. É utilizado para compartilhar dados entre o controlador e a visualização, fornecendo uma maneira eficiente de passar dados dinâmicos para a visualização.



```
@Controller
public class ServidorPublicoController {

    @GetMapping("/formularioNovoServidor")
    public String formNovoServidor(Model model) {
        model.addAttribute("servidorPublico", new ServidorPublico());
        return "novoservidorpublico";
    }
}
```

localhost:8080/formularioNovoServidor



ANOTAÇÕES SPRING MVC

- Principais Anotações

- @ModelAttribute**: é uma anotação que vincula um objeto a um nome e o adiciona ao modelo para ser usado em uma visualização. [formularioNovoServidor.html](#)

@Controller

```
public class ServidorPublicoController {  
  
    @PostMapping("/cadastrarServidor")  
    public String save(@ModelAttribute ServidorPublico novoServidor)  
    {  
        servidorService.save(novoServidor);  
        return "redirect:/listarServidores";  
    }  
}
```

Menu - SISCAPACIT Servidor Público

Cadastro de Novo Servidor Público

Matricula: Nome: Foto:

Órgão: Vínculo: Cargo:

Lotação: Exercício: Email:

Telefone: Celular: CPF:

Naturalidade:



ANOTAÇÕES SPRING MVC

- Principais Anotações

- `@ModelAttribute` `ServidorPublico novoServidor`
- Quando um manipulador de solicitações é acionado pelo Spring MVC, o framework verifica se há algum parâmetro anotado com `@ModelAttribute` no método. Se houver, o Spring MVC instancia um objeto para esse tipo de modelo e o preenche com os valores do formulário HTML enviado na solicitação. Em seguida, o Spring MVC adiciona o modelo preenchido ao modelo geral para a visualização.



ANOTAÇÕES SPRING MVC

- Principais Anotações

- **@RequestParam**: é uma anotação utilizada nos métodos para mapear os parâmetros HTTP a argumentos dos métodos.

`http://localhost: 8080/listarServidor?matricula=900848893`

```
@Controller
public class ServidorPublicoController {

    @GetMapping("/listarServidor")
    public String listByMatricula(@RequestParam("matricula") long matricula, Model model)
    {
        model.addAttribute("servidorpublico",
                           servidorService.listByMatricula(matricula).get());
        return "servidorpublico";
    }
}
```



ANOTAÇÕES SPRING MVC

- Principais Anotações

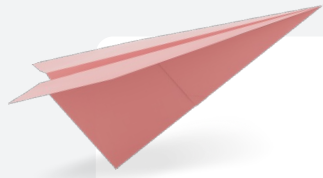
- **@PathVariable**: é outra anotação utilizada nos métodos para recuperar dados da URL.

`http://localhost: 8080/listarServidor/900848893`

```
@Controller
public class ServidorPublicoController {

    @GetMapping("/listarServidor/{matricula}")
    public String listByMatricula(@PathVariable("matricula") long matricula, Model model)
    {
        model.addAttribute("servidorpublico",
                           servidorService.listByMatricula(matricula).get());
        return "servidorpublico";
    }
}
```

PROJETO PRÁTICO

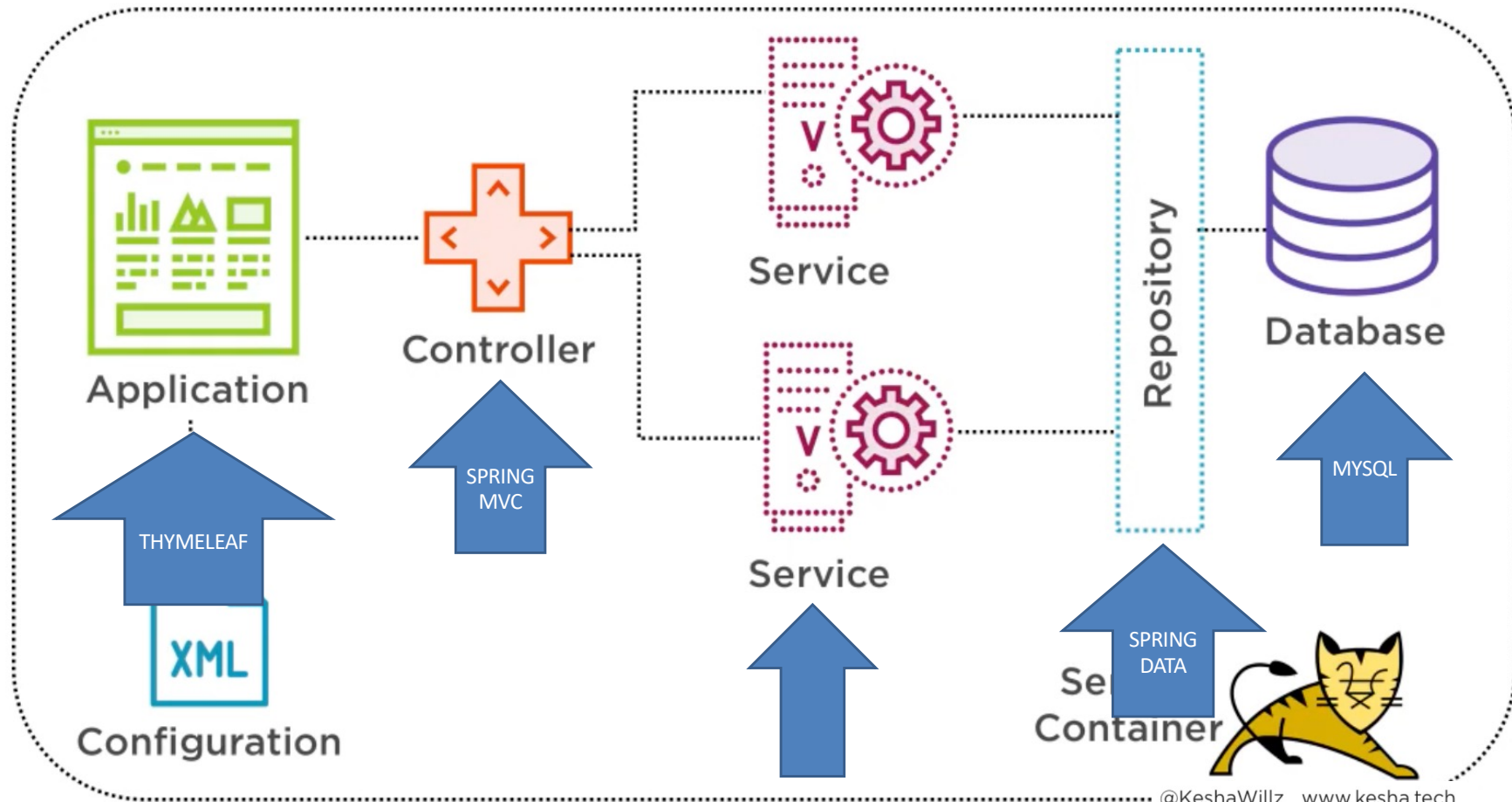


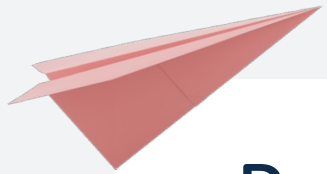
7º Projeto Spring Boot – Aplicação Servidor Público WEB com Thymeleaf e MySQL





ARQUITETURA SPRING BOOT





Passos:

- Criar o projeto com todas as suas dependências
- Criar os arquivos HTML com Thymeleaf
- Alterar o arquivo **application.properties**
- Criar a classe Controller





Passos:

- Criar o projeto com todas as suas dependências

Dependencies ADD DEPENDENCIES... ⌘ + B

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.

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

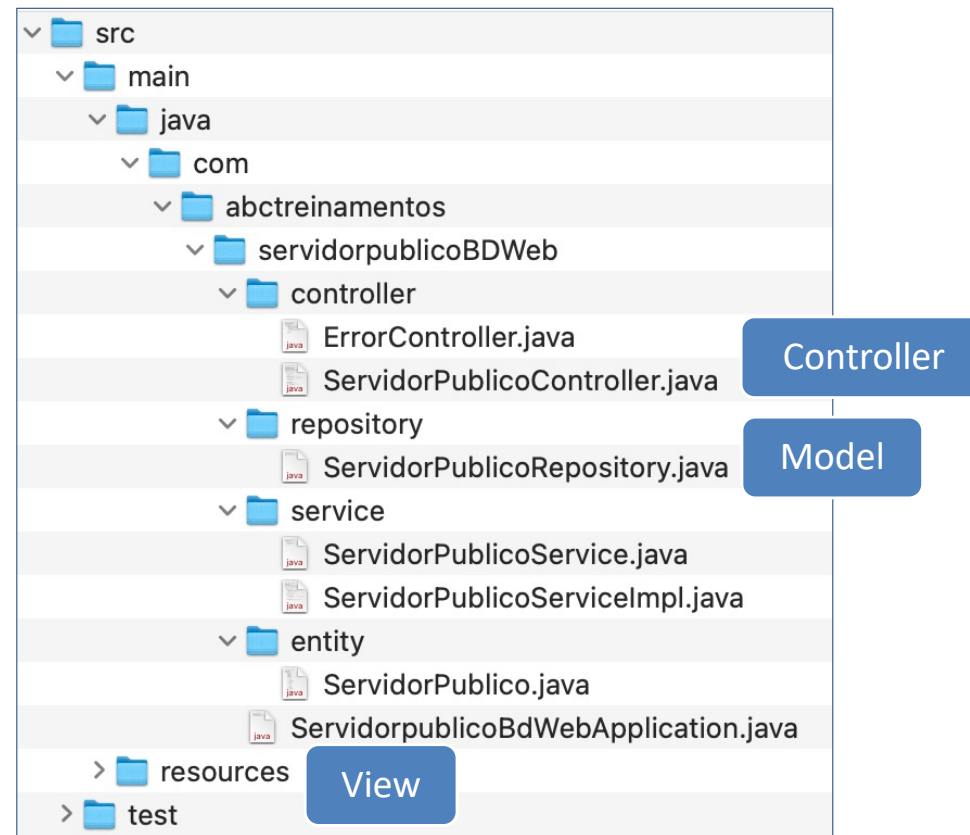
Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.





ARQUITETURA DO PROJETO





ARQUITETURA DO PROJETO

▼	src	
▼	main	
>	java	
▼	resources	
>	script	
▼	templates	
	editarservidorpublico.html	Update
	novoservidorpublico.html	Create
	servidorpublico.html	Read
	servidorespublicos.html	ReadAll
▼	erro	
	mensagem.html	
	500.html	
	404.html	
▼	fragments	
	navbar.html	
	index.html	
	application.properties	
▼	static	
▼	images	
	siscapacit.jpeg	
>	test	