# PLATAFORMA DE MOBILIDADE CORPORATIVA

SISTEMAS PARA INTERNET

# Introdução ao Angular

Aula 08

# Angular

Angular é uma plataforma de desenvolvimento para construção de aplicativos móveis e desktop.

Angular permite expandir a sintaxe do HTML para expressar os componentes da sua aplicação de forma clara e sucinta.

A ligação angular e a Injeção de Dependência eliminam grande parte do código que você teria de escrever.

# Angular CLI

As boas ferramentas tornam o desenvolvimento de aplicativos mais rápido e fácil de manter do que se você fizesse tudo à mão.

A "Angular CLI" é uma ferramenta de interface de linha de comando que pode criar um projeto, adicionar arquivos e executar uma variedade de tarefas de desenvolvimento contínuo, como teste, agrupamento e implantação.

Para saber mais: https://cli.angular.io/

FACULDADE META

# Instalando Angular CLI

Com o npm instalado execute o comando

```
sudo npm install -g @angular/cli
```

FACULDADE META

# Criando um novo projeto

Ao instalar o CLI do angular o comando ng ficará disponível no seu terminal.

para criar um novo projeto a partir do CLI, navegue até a pasta onde você irá criar seu projeto e execute os comandos:

```
ng new my-app
```

Isso irá criar um novo esqueleto de projeto.

Para saber mais: https://github.com/angular/angular-cli#usage

# Servidor

Vá a pasta do projeto criado e execute o comando.

```
cd my-app
ng serve --open
```

O comando ng serve abre um servidor e observa seus arquivos e reconstrói o aplicativo à medida que você faz alterações nesses arquivos.

Usando a opção --open (ou apenas -o) abrirá automaticamente seu navegador em http://localhost:4200/

FACULDADE META

# Estrutura de arquivos

Seu aplicativo vive na pasta src. Todos os componentes angulares, modelos, estilos, imagens e qualquer outra coisa que o seu aplicativo necessite aqui.

Todos os arquivos fora desta pasta devem apoiar a criação de seu aplicativo.

Para saber mais: https://angular.io/guide/quickstart#the-src-folder

# Criando um novo componente

Os componentes são o bloco de construção mais básico de uma IU em uma aplicação Angular. Uma aplicação angular é uma árvore de componentes angulares. Os componentes angulares são um subconjunto de diretrizes. Ao contrário das diretivas, os componentes sempre possuem um modelo e apenas um componente pode ser instanciado por um elemento em um modelo. Um componente deve pertencer a um NgModule para que ele possa ser usado por outro componente ou aplicativo.

Para saber mais: https://angular.io/api/core/Component
https://www.webcomponents.org/introduction

# Criando um componente

Crie um pasta chamada novo-componente em src/app, e crie um novo arquivo chamado novo-componente.component.ts.

```
import { Component } from '@angular/core';
@Component({
 selector: 'novo-componente',
 template:`
    <div>
        <h1>Olá sou o novo componente!!</h1>
    </div> `
})
export class NovoComponente {

}
```

# Usando o componente

No arquivo app.component.html  em src/app, adicione sua o novo componente:

```
<novo-componente></novo-componente>
```

# Usando o componente

Para finalizar adicione as linhas abaixo no arquivo app.module.ts em src/app:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { NovoComponente } from './novo-componente/novo-componente.component'

@NgModule({
  declarations: [
    AppComponent,
    NovoComponente
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Exercício

Crie um novo componente que mostre seu nome na tela:

FACULDADE META

# Criando um componente via CLI

Para criar um novo componente utilizando o Angular CLI basta usar o comando.

```
ng g component nome-do-componente
```

# Templates

Angular possui um sistema de modelo muito expressivo, que leva o HTML como base e o estende com elementos personalizados:

Double chaves {{ }} permite que você inclua propriedades do componente;

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `<h1>Hello {{ nome }}!</h1>
  `
})
export class HelloWorldComponent {
    nome = 'Vitor Hugo';
}
```

Para saber mais: https://angular.io/guide/template-syntax

# Template

Você pode utilizar o {{ }} com valor de uma propriedade:

```
@Component({
  selector: 'hello-world',
  template: `<h1>Hello {{ nomeCompleto() }}!</h1>
    <img src="{{avatar}}" >
  `
})
export class HelloWorldComponent {
    nome = 'Vitor Hugo';
    sobreNome = 'Silva';
    avatar = 'assets/image/avatar.png';
    nomeCompleto() { return this.nome+ ' ' + this.sobreNome }
}
```

# Property Binding

Outra opção é usar a propriedade binding [attribute] = property

```
@Component({
  selector: 'hello-world',
  template: `<h1>Hello {{ nomeCompleto() }}!</h1>
    <img [src]="avatar" />
  `
})
export class HelloWorldComponent {
    nome = 'Vitor Hugo';
    sobreNome = 'Silva';
    avatar = 'assets/image/avatar.png';
    nomeCompleto() { return this.nome+ ' ' + this.sobreNome }
}
```

# Outras propriedades:

```
1  <!-- Esta é uma sintaxe HTML válida. -->
2  <input [value]="person.emailAddress">
3  <!-- Funciona na sintaxe de atributo. -->
4  <button [attr.aria-label]="help">help</button>
5  <!-- Permite ligar condicionalmente uma classe  -->
6  <div [class.special]="isSpecial">Special</div>
7  <!-- Ou propriedades de style -->
8  <button [style.color]="isSpecial ? 'red' : 'green'">
9  <!-- E trabalha com componentes personalizados! -->
10 <birthday-card [date]="person.birthday">
```

# DIRECTIVA

# Condicional (*ngIf)

```
@Component({
  selector: 'hello-world',
  template: `<h1>Hello {{ nomeCompleto() }}!</h1>
    <img [src]="avatar" *ngIf="onDisplay()" />
  `
})
export class HelloWorldComponent {
    nome = 'Vitor Hugo';
    sobreNome = 'Silva';
    avatar = 'assets/image/avatar.png';
    onDisplay(){  return false } // Try changing to true!
    nomeCompleto() { return this.nome+ ' ' + this.sobreNome }
}
```

# Repetindo elementos com *ngFor

```
import { Component } from '@angular/core';

@Component({
  selector: 'lista-aluno',
  template: `
    <h1>Lista de Alunos</h1>
    <ul>
        <li *ngFor="let aluno of alunos">{{ aluno }} </li>
    </ul>
  `
})
export class ListasAlunos {
    alunos = ['Aluno 1', 'Aluno 2', 'Aluno 3', 'Aluno4'];
}
```

# ngModel

# Adicionando bootstrap 4 no seu projeto.

Você pode adicionar outros recursos a suas aplicações angular, como o bootstrap 4, para adicionar ao seu projeto você pode utilizar os seguintes comandos.

```
npm install --save jquery
npm install --save popper.js
npm install --save bootstrap@4.0.0-beta
```

Para saber mais: http://getbootstrap.com/docs/4.0/getting-started/download/

# Adicionando bootstrap 4 no seu projeto.

No arquivo .angular-cli.json modifique as configuaçõ̃es de carregamento de styles e scripts:

```
21      "styles": [
22        "../node_modules/bootstrap/dist/css/bootstrap.min.css",
23        "styles.css"
24      ],
25      "scripts": [
26        "../node_modules/jquery/dist/jquery.min.js",
27        "../node_modules/popper.js/dist/popper.js",
28        "../node_modules/bootstrap/dist/js/bootstrap.min.js"
29      ],
```

# Referências e Bibliografias.


- https://github.com/angular/angular-cli
- https://angular.io/
- https://docs.npmjs.com/files/package.json
- https://www.webcomponents.org/introduction
- http://www.dotnetcurry.com/angularjs/1366/angular-4-app-typescript-bootstrap
- https://www.youtube.com/playlist?list=PLGxZ4Rq3BOBoSRcKWEdQACbUCAWLczg2G