
Angular
No Caminho do Lado Claro
da Web

01 INTRODUÇÃO AO ANGULAR

Conhecendo a Força do Framework

Conhecendo e preparando

O que é Angular

O **Angular** é um framework de desenvolvimento de aplicações web criado pelo Google. Seu foco está na criação de aplicações dinâmicas, SPA (Single Page Applications), que rodam em uma única página web. Ele utiliza TypeScript, que adiciona tipagem estática e outros recursos avançados ao JavaScript, facilitando a manutenção e escalabilidade do código.

Instalando o Angular CLI

O Angular CLI (Command Line Interface) é uma ferramenta para gerar e gerenciar projetos Angular de forma rápida. Para instalá-lo, abra o terminal e execute:bashCopiar códigonpm install -g @angular/cli

02 ESTRUTURA E ORGANIZAÇÃO

Módulos, Componentes e Templates

Módulos, Componentes e Templates

Módulos no Angular

Módulos organizam o código em blocos lógicos, agrupando componentes, serviços e outros recursos. Todo projeto Angular começa com um módulo raiz, AppModule, definido em app.module.ts:

```ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

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

O declarations lista os componentes disponíveis no módulo, enquanto imports adiciona funcionalidades (como BrowserModule) e bootstrap define o componente raiz da aplicação.

Módulos, Componentes e Templates

Componentes: A Alma da Interface

Componentes representam as partes visuais da aplicação. Cada componente é composto de três partes principais: o arquivo TypeScript (.ts), o template HTML (.html) e o estilo CSS (.css ou .scss).Exemplo básico de componente (app.component.ts):

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  titulo = 'Bem-vindo ao Angular!';
}
```

Templates contêm o HTML que define a estrutura visual do componente. Eles permitem o uso de bindings para inserir variáveis

03 COMUNICAÇÃO E INTERATIVIDADE

Template Syntax e Bindings

Comunicação e Interatividade

Template Syntax e Bindings

O interpolation binding insere valores do componente diretamente no template usando {{ }}, como no exemplo anterior.

O property binding define atributos de elementos HTML com valores do componente.htmlCopiar código

O event binding permite que o Angular responda a eventos de usuário, como cliques e movimentos do mouse. Por exemplo, para exibir uma mensagem quando um botão é clicado:htmlCopiar código

Two-Way Data BindingO Angular também permite o two-way data binding, ou seja, a comunicação em duas vias entre o modelo de dados e o template, usando o diretório [(ngModel)]. Esse tipo de binding é muito útil para formulários.

```
<input [(ngModel)]="nome" placeholder="Digite seu nome">
<p>Olá, {{ nome }}!</p>
```

04 SERVIÇOS E DEPENDÊNCIAS

Injeção de Dependências e Roteamento

Serviços, Injeção de Dependências

Serviços

Serviços são classes que encapsulam funcionalidades e dados que podem ser compartilhados entre componentes. Eles são injetados nos componentes por meio da injeção de dependências, simplificando o acesso a funcionalidades externas.

```
import { Component } from '@angular/core';
import { ExemploService } from './exemplo.service';

@Component({
  selector: 'app-saudacao',
  template: `<p>{{ saudacao }}</p>`
})
export class SaudacaoComponent {
  saudacao: string;

  constructor(private exemploService: ExemploService) {
    this.saudacao = this.exemploService.getSaudacao();
  }
}
```

Serviços, Injeção de Dependências

Roteamento

O roteamento permite navegação entre diferentes visualizações de forma eficiente, mantendo a aplicação em uma única página.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: '', redirectTo: '/home', pathMatch: 'full' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

05 BOAS PRÁTICAS E FERRAMENTAS

Um Caminho para Maestria

Boas Práticas e Ferramentas

Maestria

Organização de Pastas: Separe os arquivos por funcionalidades.

Testes Automatizados: Implemente testes unitários para garantir a qualidade.

Código Limpo: Use interfaces, slint e prettier para manter o código limpo e padronizado.

AGRADECIMENTOS

OBRIGADA POR LER ATÉ AQUI!

Esse Ebook foi gerado por IA e diagramado por humano.

Conteúdo para fins de projeto de Bootcamp, não realizada validação humana no conteúdo e pode conter erros pela IA.