

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

Fundamentos Angular

Módulo 5

Introdução ao Angular

Sessão 25

Objetivo geral

Capacitar os formandos na compreensão da arquitetura fundamental do Angular e do seu ecossistema, estabelecendo o modelo mental necessário para o desenvolvimento de aplicações modernas baseadas em componentes. A sessão introduz o Angular CLI, a estrutura de um projeto Angular, o papel dos componentes, decorators, e o processo de bootstrap da aplicação, permitindo ao formando entender como uma aplicação Angular é organizada e inicializada.

Objetivos específicos

Ao final da sessão, os formandos deverão ser capazes de:

- Compreender o posicionamento do **Angular** no ecossistema frontend, distinguindo-o de bibliotecas como React;
- Utilizar o Angular CLI para criar e gerir projetos Angular;
- Identificar a anatomia de um projeto Angular, incluindo **módulos** e **componentes**;
- Explicar a estrutura e função de um componente Angular, incluindo **metadata** e **templates**;
- Compreender o processo de **bootstrap**, desde o **main.ts** até à renderização no **index.html**;
- Reconhecer o papel dos **decorators**, como mecanismo fundamental da arquitetura Angular.

Ecossistema: Angular vs React

O que é o Angular?

Angular é um framework frontend completo para criar aplicações web modernas, mantida pela Google. Ela fornece tudo o que é necessário para construir aplicações profissionais, incluindo:

- Criação de interfaces e componentes;
- Sistema de navegação (Routing);
- Comunicação com APIs (HTTP Client);
- Gerenciamento de formulários (Forms);
- Injeção de dependências (Dependency Injection);
- Ferramentas de desenvolvimento (Angular CLI)

O que é o React?

React é uma **biblioteca** JS para a construção de interfaces de utilizador, mantida pela Meta (Facebook), ela é focada em:

- Criação de interfaces com componentes;
- Atualização eficiente da interface (renderização reativa).

Para funcionalidades como navegação, formulários e comunicação com APIs, normalmente utilizamos bibliotecas externas.

Framework vs Biblioteca

Framework (Angular):

- Define estrutura da aplicação;
- Define padrões de desenvolvimento;
- Define arquitetura;
- Controla o fluxo da aplicação.



Biblioteca (React):

- Oferece ferramentas específicas;
- Você define a arquitetura;
- Você escolhe as bibliotecas adicionais;
- Você controla o fluxo da aplicação.

Analogia prática

Angular decide:

- Estrutura de projeto padrão;
- Organização baseada em componentes;
- Sistema de routing integrado;
- Comunicação com APIs integrada;
- Arquitetura consistente.

React permite:

- Definir a estrutura de pastas;
- Escolher a biblioteca de routing;
- Escolher o gerenciamento de estado;
- Definir a arquitetura do projeto.

Angular é um framework baseado em TS

Angular utiliza TypeScript como linguagem principal de desenvolvimento, isso permite:

- Tipagem forte;
- Uso de classes e interfaces;
- Detecção de erros durante o desenvolvimento;
- Maior escalabilidade e manutenção.

Angular CLI

O que é o Angular CLI?

Angular CLI significa – **Angular Command Line Interface**, ela é a ferramenta oficial para:

- Criar novos projetos;
- Gerar Componentes, serviços e outras estruturas;
- Executar a aplicação localmente;
- Construir a aplicação para a produção.

Por que usar o Angular CLI?

Sem o Angular CLI, precisaríamos:

- Criar toda a estrutura do projeto manualmente;
- Configurar o sistema de build da aplicação;
- Configurar o compilador Typescript;
- Configurar o sistema de build da aplicação;
- Configurar ferramentas como o bundler (webpack);
- Executar e gerenciar o projeto manualmente.

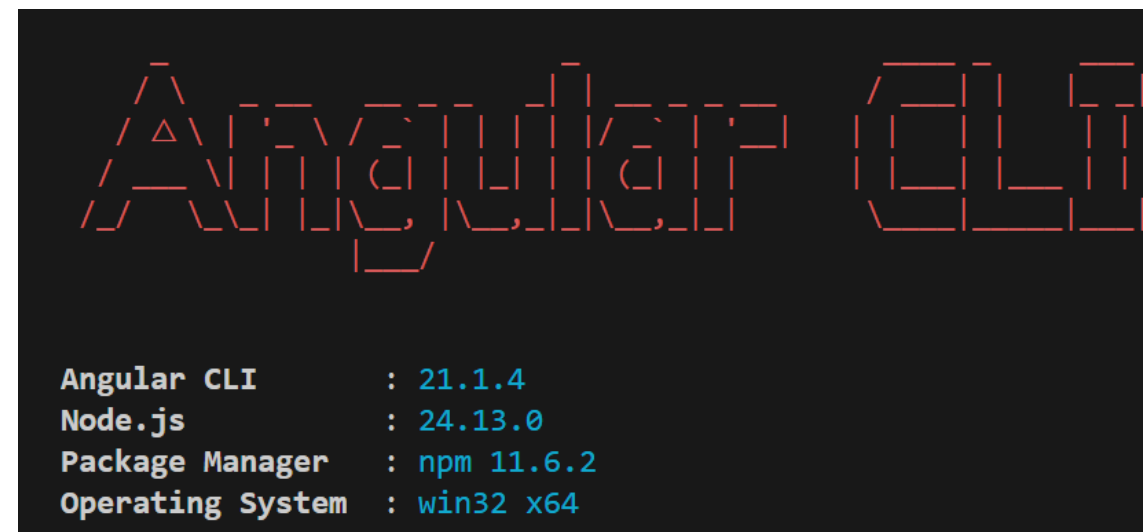
Instalar o Angular CLI

No terminal execute:

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

```
ng version
```

O comando ng do angular fica disponível e podemos assim, verificar a versão do Angular que foi instalada.



```
Angular CLI  
  
Angular CLI      : 21.1.4  
Node.js          : 24.13.0  
Package Manager  : npm 11.6.2  
Operating System : win32 x64
```

Criar uma nova aplicação

No terminal digite:

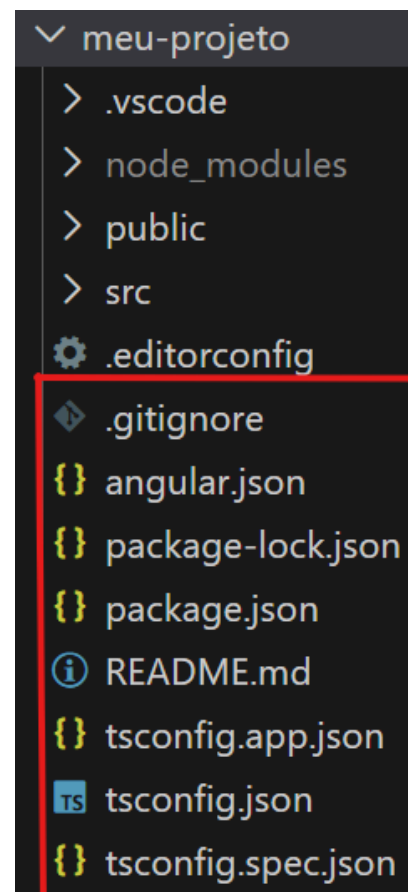
```
ng new meu-projeto
```

O Angular CLI irá fazer algumas perguntas:

- Deseja habilitar o routing? Digite y
- Escolha o formato de CSS -> Pressione enter para utilizar o padrão;

Estrutura criada automaticamente

O CLI cria a seguinte estrutura:



Executar a aplicação

Dentro da pasta meu-projeto
(nome default para a criação
do primeiro projeto) iremos
rodar o comando ng serve

```
ng serve
```

```
→ Local: http://localhost:4200/  
→ press h + enter to show help
```

O que acontece com o ng serve?

O Angular CLI:

- Compila o código Typescript para Javascript;
- Inicia um servidor local;
- Executa a aplicação no navegador;
- Observa alterações no código;
- Atualiza automaticamente a aplicação.

Gerar componente automaticamente

O CLI:

```
ng generate component header
```

```
CREATE src/app/header/header.spec.ts
CREATE src/app/header/header.ts
CREATE src/app/header/header.css
CREATE src/app/header/header.html
```

Ou o comando reduzido

```
ng g c header
```

O Angular CLI cria automaticamente todos os arquivos necessários.

Anatomia do Projeto Angular

Estrutura geral do projeto

Quando criamos um projeto com o Angular CLI, temos:

```
✓ MEU-PROJETO
> .angular
> .vscode
> node_modules
> public
> src
⚙ .editorconfig
📄 .gitignore
{} angular.json
{} package-lock.json
{} package.json
📖 README.md
{} tsconfig.app.json
📄 tsconfig.json
{} tsconfig.spec.json
```

Cada pasta e ficheiro possui uma função específica na aplicação.

A pasta mais importante: src/

A pasta src/ contém o código da aplicação. É desta pasta que desenvolvemos a aplicação.

```
✓ src
  > app
    <> index.html
    TS main.server.ts
    TS main.ts
    TS server.ts
    # styles.css
```

Index.html

O arquivo index.html é o primeiro arquivo carregado pelo navegador. Dentro dele existe a tag app-root, e é dentro dessa tag que o Angular renderiza toda a aplicação.

```
10   <body>
11   |   <app-root></app-root>
12   </body>
```

Main.ts

O main.ts será o ponto de entrada do Angular, será neste ficheiro que a aplicação será inicializada, e o responsável por fazer esse trabalho é o método `bootstrapApplication(AppComponent)`

```
bootstrapApplication(App, appConfig)  
  .catch((err) => console.error(err));
```

Pasta app/

Contém os componentes da aplicação. O componente principal da aplicação também está localizado aqui. É dentro dessa pasta que criamos novos componentes.

```
▼ app
  > header
  TS app.config.server.ts
  TS app.config.ts
  # app.css
  <> app.html
  TS app.routes.server.ts
  TS app.routes.ts
  TS app.spec.ts
  TS app.ts
```

App.ts

É o componente responsável por definir o comportamento e a configuração da aplicação. Atenção ao seletor, ele diz o nome da tag html que utilizamos, quando queremos chamá-lo em outro documento HTML, e ele está presente no nosso index.html.

```
@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrls: ['./app.css']
})
export class App {
  protected readonly title = signal('meu-projeto');
}
```

App.spec.ts

O arquivo app.spec.ts é utilizado para testes automatizados da aplicação. Esse arquivo é criado automaticamente pelo Angular CLI. Não é necessário modificá-lo no início do desenvolvimento.

```
1  import { TestBed } from '@angular/core/testing';
2  import { App } from './app';
3
4  describe('App', () => {
5    beforeEach(async () => {
6      await TestBed.configureTestingModule({
7        imports: [App],
8      }).compileComponents();
9    });
```

Componentes & Metadata

O que é um componente

É a unidade fundamental de construção de uma aplicação Angular, ele será responsável por controlar:

- Interface (HTML);
- Estilos (CSS);
- Comportamento (TS).

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-header',
5    imports: [],
6    templateUrl: './header.html',
7    styleUrls: ['./header.css'],
8  })
9  export class Header {
10
11  }
```

O que é o @Component?

É um **Decorator** que define as configurações de um componente Angular, ele informa ao Angular:

- Qual é o nome da tag do componente (selector);
- Onde está o template (HTML);
- Onde estão os estilos (CSS).

Selector

O selector define o nome da tag HTML do componente. É através dessa tag que utilizamos o componente no HTML.

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

@Component({
  selector: 'app-header',
  imports: [],
  templateUrl: './header.html',
  styleUrls: ['./header.css'],
})
export class Header {

}
```

TemplateUrl e styleUrls

Definem os ficheiros utilizados pelo componente:

- templateUrl → ficheiro HTML do componente
- styleUrls → ficheiro CSS do componente

```
@Component({  
  selector: 'app-header',  
  imports: [],  
  templateUrl: './header.html',  
  styleUrls: ['./header.css'],  
})  
export class Header {  
  
}
```

Classe do componente

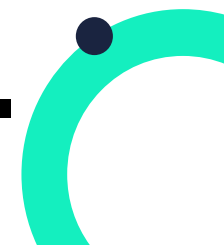
Iremos escrever a lógica do nosso componente nessa classe, ela estará disponível em outros componentes, ao ser importada.

```
@Component({
  selector: 'app-header',
  imports: [],
  templateUrl: './header.html',
  styleUrls: ['./header.css'],
})
export class Header {
  titulo: string = 'Meu primeiro componente Angular';
}
```

```
1 <h1>{{ titulo }}</h1>
```

```
1 <app-header></app-header>
```

Bootstrap da Aplicação



O que é o Bootstrap?

É o processo de inicialização da aplicação Angular, é o momento em que o Angular inicia o build, carrega o primeiro componente e renderiza a interface. Ele é configurado no `main.server.ts` e invocado no método `bootstrapApplication`, dentro do `main.ts`, é nesse momento que o Angular é inicializado, carrega o `AppComponent`, ligando-o ao seletor `<app-root>`.

Decorators

O que é um Decorator?

É uma função especial que adiciona metadado a uma classe, ele informa ao Angular como a classe deve ser usada. Sem o decorator o componente é apenas uma classe comum. No HTML o <head> possui os metadados que configuram o ficheiro HTML, como o título, e a localização dos ficheiros auxiliares ao seu funcionamento, como o css e javascript. No Angular, a configuração do componente fica encapsulado no decorator @Component.

```
@Component({  
  selector: 'app-root',  
  imports: [RouterOutlet, Header],  
  templateUrl: './app.html',  
  styleUrls: ['./app.css']  
})
```

Decorators importantes

- @Component: Configuração do componente;
- @Injectable: Injeta algum tipo de serviço em um componente;
- @Directive: Configura as diretivas do componente;
- @Pipe: Configura os Pipes do componente.

Resumindo, um decorator transforma uma classe comum em uma parte funcional do Angular.

Síntese

- **Ecosistema Angular e seu posicionamento:** Compreensão do Angular como um framework opinativo e completo de desenvolvimento;
- **Angular CLI como ferramenta central:** Utilização da interface de linha de comandos oficial para criação, execução e gestão de aplicações Angular;
- **Anatomia de um projeto Angular:** Identificação da estrutura fundamental da aplicação, incluindo ficheiros como index.html, main.ts e o componente raiz;
- **Arquitetura baseada em Componentes:** Introdução ao componente como unidade fundamental da aplicação Angular;
- **Processo de Bootstrap da Aplicação:** Compreensão do fluxo de inicialização da aplicação, desde o carregamento do index.html, passando pelo main.ts, até à renderização do componente raiz no DOM;
- **Decorators como mecanismo estrutural do Angular:** Reconhecimento dos decorators como elementos essenciais que fornecem metadata ao Angular;

Conclusão

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa