

BOOTCAMP ANGULAR

Apresentado por: Nicoly Almeida



Sobre mim



Nicoly Almeida

Front-End Software Engineer

Picpay

Ambassador

Women Techmakers

Organização

Mulher Tech Sim Senhor



Overview

- *Padrão MVC*
- *Introdução ao Angular*
- *Configuração do ambiente*
- *Angular Material*
- *Roteamento*
- *Observables*
- *Comunicação com Back-End*



Introdução

Sobre o angular

O Angular, desenvolvido pela Google, é um framework que segue a arquitetura MVC e simplifica a criação de aplicações web interativas com componentes predefinidos.

TypeScript

Angular é composto por um conjunto de bibliotecas que permitem aos programadores criar aplicativos do lado do cliente, utilizando tanto JavaScript quanto TypeScript

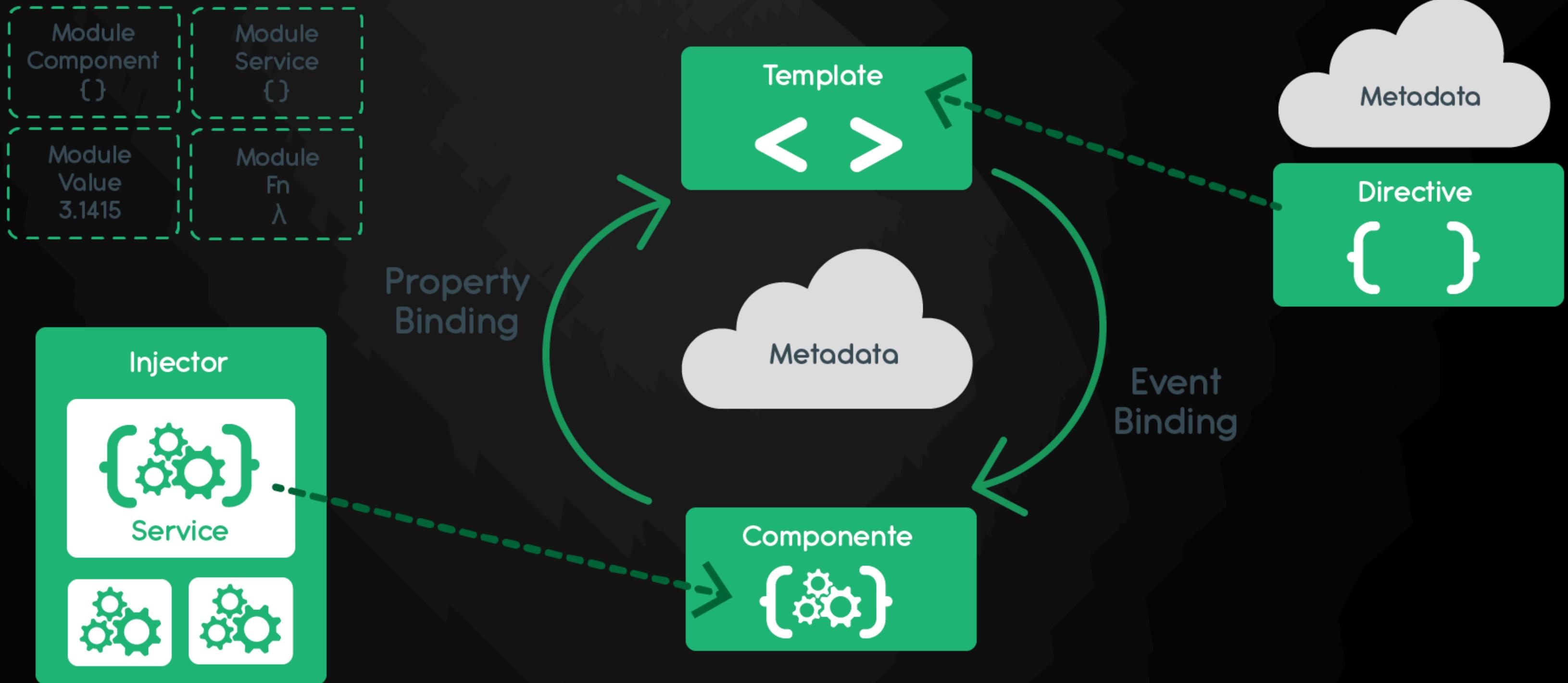
Passos para desenvolvimento de aplicações Angular

Criar templates HTML

Escrever classes (componentes) para gerenciar os templates

Adicionar lógica de negócio em serviços

Empacotar componentes e serviços em módulos



Módulos

Uma aplicação Angular é formada pelo menos por um módulo, o módulo raiz, tipicamente chamado de AppModule.

Poderíamos dizer que a área de relatórios seria um módulo, contendo diversos componentes, já o controle de contas seria outro módulo.

Não há uma regra específica para se dividir a aplicação em módulos, mas dividir o problema negocial em unidades coesas é o caminho mais seguro.



Componentes

Um componente controla um trecho da tela (chamada visão).

Por exemplo, podemos fazer um componente para listar todas as contas bancárias de um cliente, ou mesmo um componente que altere os dados de uma conta bancária.

```
export class ContaListComponent implements OnInit {  
    contas: Conta[];  
    contaSelecionada: Conta;  
  
    constructor(private service: ContaService) { }  
  
    ngOnInit() {  
        this.contas = this.service.getContas();  
    }  
  
    selecionarConta(conta: Conta) { this.contaSelecionada = conta; }  
}
```



Templates

Um template é um trecho de tela (ou uma tela inteira) escrita em "HTML". A parte da visão do componente, ou seja, o que será mostrada para o usuário é definida no template.

Perceba que há itens que não são HTML, mas sim de Angular. O *ngFor, o (click), as chaves {{}}. A propriedade contas vem da classe que trata essa tela, assim como o método selecionarConta().

```
<h2>Listagem de Contas</h2>

<p><i>Selecione uma conta</i></p>
<ul>
  <li *ngFor="let conta of contas" (click)="selecionarConta(conta)"
      {{conta.agencia}}-{{conta.numero}}
  </li>
</ul>
```

Metadados

Metadados é a forma de dizer ao Angular como processar uma classe. Uma classe em Typescript é apenas uma classe, até que você diga ao Angular que ela faz parte de um componente.

```
@Component({
  selector: 'conta-list',
  templateUrl: './conta-list.component.html',
  providers: [ ContaService ]
})
export class ContaListComponent implements OnInit {
  /* . . . */
}
```



Data binding

Interpolação de string: ao fazer {{conta.numero}}

Property binding: ao fazer uso do [conta]

Event binding: ao definir o (click)

Two-way binding [(ngModel)]

```
<li>{{conta.numero}}</li>
<conta-detalhe [conta]="contaSelecionada"></conta-detalhe>

<li (click)="contaSelecionada(conta)"></li>

<input [(ngModel)]="conta.numero">
```

Diretivas

Diretivas estruturais alteram o layout (o template) alterando, inserindo ou removendo elementos do DOM.

Diretivas de atributos mudam o layout (aparência) ou comportamento de elementos que já existem.

```
<li *ngFor="let conta of contas"></li>

<conta-detalhe *ngIf="contaSelecionada"></conta-detalhe>

<input [(ngModel)]="conta.numero">
```

Serviços

Serviço engloba qualquer valor, função ou feature (característica) que sua aplicação precisa. Não há algo específico no Angular que diga que algo é um serviço, mas sua aplicação definirá diversos serviços para ajudar na comunicação com o backend

```
export class ContaService {  
    private contas: Conta[] = [];  
  
    constructor(  
        private backend: BackendService,  
        private logger: Logger) {}  
  
    getContas() {  
        this.backend.getAll(Conta).then((contas: Conta[]) => {  
            this.logger.log(`Carregadas ${contas.length} contas.`);  
            this.contas.push(...contas);  
        });  
        return this.contas;  
    }  
}
```



THANK YOU