frameworks e agular

Principais Frameworks Web (Angular, Vue e React)

Angular

angular é um framework web desenvolvido pelo google, sua funcionalidade é full stack (front-end e back-end) para SPAs (single page applications). Sua principal linguagem é o typescript.

características

1. estrutura MVC (Model-View-Controller) que divide a aplicação em três componentes principais, model para apresentar os dados e a lógica de negócios, view para a interface do usuário e o controller que integra o model e a view respondendo aos eventos da interface e atualizando o model.
2. injeção de dependência que promove a reutilização de código, uma vez que as bibliotecas e dependências podem ser compartilhadas várias vezes no sistema para aumentar a flexibilidade e a modularidade.
3. Two-way data binding, este mecanismo de programação é particular para frameworks em front-end, ele permite a automação entre model (data) e view, fazendo com que eles estejam interligados criando mais dinâmica e responsividade para a interface
4. Sistema de roteamento, formulários, testes, HTTP e muito mais embutido

React

uma biblioteca JavaScript para construção de interfaces de usuário, desenvolvida pelo Facebook (Meta), é muito utilizada em SPAs e sua curva de aprendizado é mais acessível comparado com o Angular

características

1. componentização: toda a interface é beaseada em componentes reutilizáveis, que encapsulam lógica e visual, melhorando a organização e reutilização do código assim como a escalabilidade das suas aplicações
2. one-way data binding: a comunicação entre os dados e a interface é em apenas uma direção, ou seja, o componente pai para o componente filho, essa abordagem torna o controle da interface mais previsível
3. Virtual DOM: React usa uma representação virtual da árvore DOM. ao invés de manipular diretamente o DOM, o react calcula a diferença entre o estado anterior e o novo e atualiza apenas os elementos necessários, aumentando o desempenho da aplicação.
4. JSX **(**JavaScript **+** XML**)**: Permite escrever HTML dentro do JavaScript, o que facilita a criação e visualização dos componentes de forma mais intuitiva e expressiva.

Vue.js

um framework criado por Evan You (ex funcionario do google), esse framework foi pensado para ser flexível, podendo ser usado tanto como uma biblioteca simples para melhorar a interface quanro como um framework completo para aplicações complexas

características

- Fácil de aprender e usar: Vue possui uma curva de aprendizado suave, tornando-se uma ótima opção para iniciantes. Sua estrutura é intuitiva e baseada em HTML, CSS e JavaScript puros.
- Two-Way Data Binding: Assim como Angular, o Vue também oferece ligação bidirecional entre dados e interface, facilitando o desenvolvimento reativo de aplicações.
- Componentes reutilizáveis: A estrutura do Vue também é baseada em componentes, o que permite maior organização e modularidade no projeto.

- Reatividade: O Vue implementa um sistema reativo eficiente que atualiza automaticamente a interface do usuário quando os dados mudam

Componentes do Angular

O que são componentes?

Componentes são **blocos de construção** da interface no Angular. Cada componente possui:

- Um **template** (HTML)
- Uma **classe TypeScript** com lógica
- Um **estilo CSS/SCSS**
- Um **decorator** `@Component` que conecta tudo

Exemplo básico de um componente:

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

@Component({
  selector: 'app-exemplo',
  template: `<h1>Olá, Angular!</h1>`,
  styles: [`h1 { color: blue; }`]
})
export class ExemploComponent {}
```

Tipos de Data Binding:

- **Interpolation**: `{{ variavel }}`
- **Property binding**: `[property]="variavel"`
- **Event binding**: `(click)="metodo()"`
- **Two-way binding**: `[(ngModel)]="variavel"`

4. HTTP do Angular

Para que serve?

É usado para **fazer requisições HTTP** (GET, POST, PUT, DELETE etc.) a APIs REST.

Como funciona?

Angular usa o **HttpClient** do módulo `@angular/common/http`.

Exemplo de uso:

1. Importar o módulo no app:

```
typescript
CopiarEditar
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  imports: [ HttpClientModule ]
})
export class AppModule {}
```

1. Fazer requisição:

```
typescript
CopiarEditar
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable()
export class MeuServico {
  constructor(private http: HttpClient) {}

  pegarUsuarios() {
    return this.http.get('https://api.exemplo.com/usuarios');
  }
```

```
}
```

1. Consumir no componente:

```
typescript
CopiarEditar
this.meuServico.pegarUsuarios().subscribe(dados ⇒ {
  console.log(dados);
});
```

---

5. Introdução aos Testes com Karma e Jasmine no Angular

O que são?

- **Jasmine**: Framework de testes com sintaxe descritiva (`describe`, `it`, `expect`).
- **Karma**: Ferramenta que executa os testes no navegador automaticamente.

Como funcionam juntos?

- O Angular CLI já vem configurado com Karma e Jasmine.
- Quando você roda `ng test`, o Karma executa os testes escritos com Jasmine.

Exemplo básico de teste:

```
typescript
CopiarEditar
describe('MeuComponente', () ⇒ {
  it('deve retornar verdadeiro', () ⇒ {
    expect(true).toBe(true);
  });
});
```

Testando um componente Angular:

```
typescript
CopiarEditar
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { MeuComponente } from './meu-componente.component';

describe('MeuComponente', () => {
  let component: MeuComponente;
  let fixture: ComponentFixture<MeuComponente>;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [ MeuComponente ]
    });
    fixture = TestBed.createComponent(MeuComponente);
    component = fixture.componentInstance;
  });

  it('deve criar o componente', () => {
    expect(component).toBeTruthy();
  });
});
```