**Fundamento - Componentes e Estilização**

**Componente**

Em Angular, um componente é uma parte fundamental da estrutura de uma aplicação. Ele é uma classe TypeScript que contém propriedades e métodos que definem o comportamento e a estrutura de uma parte específica da interface do usuário. Além disso, um componente é associado a um template HTML que determina como o componente é renderizado na tela.

Um componente é responsável por encapsular a lógica de apresentação e interação do usuário em um único módulo reutilizável. Ele pode incluir lógica de negócios, manipulação de eventos, chamadas a serviços, entre outras funcionalidades relacionadas à sua área específica de atuação na aplicação.

Além disso, os componentes em Angular seguem o princípio de "união de responsabilidades", o que significa que cada componente deve ter uma única responsabilidade bem definida, tornando o código mais organizado, fácil de entender e de dar manutenção.

\*Obs.: componentes sempre irão dentro da propriedade declarations no module.ts.

**Módulo**

Em Angular, um módulo é uma unidade de organização e encapsulamento que agrupa componentes, diretivas, pipes e serviços relacionados de uma aplicação. Ele serve como um contêiner para organizar e configurar partes específicas da aplicação.

Um módulo é definido por uma classe TypeScript decorada com o decorator **@NgModule**. Este decorator fornece metadados que o Angular usa para entender como compilar o aplicativo e como executá-lo em tempo de execução.

Existem dois tipos principais de módulos em Angular:

1. **Módulo de Raiz (Root Module):** Este é o módulo principal da aplicação Angular. Geralmente é chamado de AppModule e é definido no arquivo **app.module.ts**. Ele importa outros módulos necessários e declara os componentes, diretivas, pipes e serviços que serão usados em toda a aplicação.
2. **Módulos de Funcionalidade (Feature Modules):** Estes são módulos adicionais que agrupam recursos relacionados de uma aplicação. Eles podem ser usados para organizar a aplicação em funcionalidades ou áreas específicas. Cada módulo de funcionalidade possui sua própria classe NgModule e pode importar/exportar componentes, diretivas, pipes e serviços específicos a essa funcionalidade.

Os módulos ajudam a modularizar e organizar o código, tornando a aplicação mais fácil de entender, manter e escalar. Além disso, os módulos fornecem benefícios de reutilização, permitindo que partes específicas da aplicação sejam facilmente compartilhadas entre diferentes partes do projeto ou até mesmo entre diferentes projetos.

\*Obs.: dentro do **@NgModule**, todo módulo ficara dentro do imports, já os componentes, dentro do declarations.

**Problema**

As vezes temos componentes que são praticamente iguais e muda apenas o texto ou background, não é uma boa prática criar um componente pra cada estilo. O melhor a se fazer é criar um único componente e passar o estilo/texto de forma dinâmica.

Por exemplo, os dois cards abaixo. Nesse caso, foram criados dois componentes e quando for necessário alteração, será preciso fazer nos dois componentes, se alterasse a fonte, o peso ou algo do tipo.

Texto

Descrição gerada automaticamente com confiança média

**Criação de Componentes Filhos**

Agora vamos remover esse botão “adquirir” e criar um componente pra ele. Desse modo, quando for necessário alterar algo do botão, faremos isso no seu componente e em todos os locais que ele for chamado, a alteração será refletida.

Uma parte do problema foi resolvida, agora temos um componente para o botão e temos o mesmo resultado visual. No entanto, ainda estamos repetindo código.

Texto

Descrição gerada automaticamente

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente