Tarefa Casa e Café - parte 3

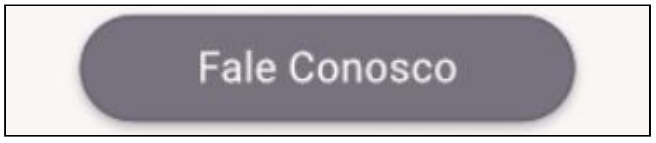
Matheus Gomes da Silva Horta

Nesta terceira tarefa, devemos explicar como criaríamos componentes para uma das telas da empresa CasaeCafe.com, mostrando os motivos da distribuição dos componentes, da criação de cada componente e as funcionalidades que cada um apresentaria.

Nas seções subsequentes, eu apresentarei cada componente que eu criaria explicando o que me levou a pensar de tal forma e como esse componente se comportaria em uma página. Para tomar a decisão de transformar em componente ou não, eu foquei principalmente em quatro conceitos fundamentais: Encapsulamento, Extensibilidade, Reusabilidade e Agregação.

Componentes

1. Fale Conosco



Um componente composto pelo botão na imagem, que, ao ser clicado, levaria para uma página ou aplicação, onde se pudesse fazer contato com a empresa. Poderia ser utilizado como uma tag `<fale-conosco></fale-conosco>`, ou algo do gênero.

Por que um componente?

O botão apresenta um estilo único na página e, na ausência de encapsulamento, deveria ter classes e seletores específicos aplicados a ele de modo a evitar conflitos de estilo. Englobar toda essa estilização e estrutura em uma única tag tornaria o código mais conciso e legível. Além disso, por se tratar de um botão com propósito de linkar para algum meio de contato da empresa, poderia ser utilizado também em outras telas do site, ou dentro de outras tags e componentes.

2. Cupom de desconto

Um componente constituído por uma label e um input, que checa a validade do cupom (chamadas a APIs, por exemplo) e aplica o desconto ao preço total. Poderia ser chamado no html como `<coupon-checker></coupon-checker>`. Esse componente, no caso, teria apenas uma funcionalidade específica. De checar por validade do que foi digitado.

Por que um componente ?

Nota-se que o botão possui estilos diferentes dos outros botões da página, além de um “addon” representado pelo botão “>” na direita. Em uma implementação de html, css e js comum, seria necessário atribuir algumas classes (no caso de se utilizar um framework css) para se alcançar esse estilo, além de que seria necessário atribuir seletores específicos para esse botão de modo a impedir conflitos, assim como um evento para ativar a ação de validar o cupom. Tudo isso seria encapsulado em apenas uma tag, que poderia ser reutilizada e agregada a outros elementos da página.

3. Display de preço

Esse componente não teria nenhuma funcionalidade específica além de mostrar o preço total alterado com o desconto. Poderia ser simplesmente colocado na página por meio de uma tag `<price-display amount=””></price-display>`.

Por que um componente ?

O foco aqui seria de encapsular o estilo e estrutura necessários para se criar esse display. Poderia ser utilizado para qualquer display de valores que o programador quisesse que tivesse esse estilo específico de preço. Talvez fosse útil apenas na página de pagamento porém é uma maneira de, visualmente e semanticamente, deixar o código mais limpo.

4. Pagamento com cartão

Esse componente englobaria o formulário para o pagamento com cartão de crédito, que incluiria o botão que faria a requisição para criar um novo pagamento no servidor. Seria semelhante a uma tag <form> que aceitaria os inputs como uma lista, algo parecido com:


```
<payment-form attributes=””>
  <all kinds of tags>
</payment-form>
```

Onde o botão que faria submit no formulário já estaria embutido no componente.

Por que um componente?

Formulários robustos são sempre algo que requer muito html e css além de poderem possuir muitos elementos que são utilizados também em outras partes da aplicação, o que pode tornar os arquivos bastante carregados visualmente tanto com marcações quanto com seletores. Fora isso, ainda tem a grande quantidade de javascript necessário para fazer as requisições ajax para a API do servidor.

Transformando o formulário em um componente, eu ocultaria toda a parte estrutural, de estilização e de javascript, reduzindo bastante a carga visual. Poderia se modificar o novo formulário para também aceitar tags além de inputs, se adequando a necessidade do programador quanto aos dados que ele necessita do usuário.

4. Pagamento com boleto

Analogamente ao componente do formulário, teria-se um para a opção de submeter com o boleto, que tomaria toda a parte de digitação de cpf, texto explicativo, vencimento e botão de submissão. Algo semelhante a uma tag

```
<payment-boleto>
    <any input>
</payment-boleto>
```

.

Por que um componente?

Teria-se o encapsulamento de toda a lógica de geração de boleto e estilos usados na seção, o que deixaria ainda mais visualmente e logicamente organizado, principalmente durante a lógica de seleção de tipo de pagamento feita pelos botões de select na parte de “método de pagamento”.

5. Observações

De fato, criar muito componentes em uma página pode dificultar a comunicação entre os mesmos e, além disso, causaria a duplicação de css, caso muitos estilos fossem comuns em mais de um componente. Porém, deve-se levar sempre em consideração a aplicação que se está desenvolvendo, tal como as vantagens em relação ao desempenho e portabilidade que o uso de web components traria.

Sobre o que não virou componente

Seletor de forma de pagamento

O principal motivo que me levou a desconsiderar o seletor do “método de pagamento” como componente é o fato dessa seção alterar outros elementos da página. Eu imagino que, ao se selecionar cada opção, o html da página mude dinamicamente para a opção formulário ou o pagamento com boleto. Isso é um indicador de que não seria recomendado transformar esse seletor em um componente, pois ele dependeria de outros elementos da página, impedindo a reusabilidade e indo contra o encapsulamento.