RELATÓRIO DE DESIGN DE CÓDIGO

1. INTRODUÇÃO

É evidente que para qualquer projeto são necessários dois pontos principais: Qual a situação atual e qual o objetivo que será buscado ao fim do projeto.

A partir disso, conseguimos quebrar o projeto em pequenas partes, dividindo as diferentes ações a serem realizadas de maneira que, naturalmente, é muito mais fácil, visível e claro definir em que passo o projeto está e qual será o próximo passo em direção ao fim.

Existem vários métodos de gerenciamento de projeto que visam atacar justamente esses pontos vitais. Fazendo um ciclo de checagem do estado atual e qual o próximo objetivo. Dois nomes importantes nesse sentido são a metodologia Scrum e o ciclo de PDCA. Que dizem, em suas bases, que o planejamento é o pronto principal para o bom desenvolvimento de tarefas de maneira rápida, assertiva e eficiente.

Portanto, ao iniciar um projeto, fica evidente que ter uma boa ideia de qual é, especificamente, o objetivo almejado, é o primeiro passo a ser executado. Assim, este documento visa demonstrar a arquitetura e design de código planejados durante o desenvolvimento do projeto apelidado de “Lancheria”.

O design e arquitetura de código são de suma importância para a confecção de aplicações funcionais, ao passo que, ao serem de fato pensados de maneira concisa, ajudam tanto na escrita do código em si, quanto na sua organização.

Conhecer o fluxo de uma aplicação em seus detalhes antes mesmo do início de sua fase de implementação é um passo muito importante para a modelagem dos componentes que a comporão. Esses termos, não por acaso, são utilizados durante o desenvolvimento, pois representam partes importantes do código.

Enfim, este documento conterá toda a arquitetura da aplicação, alguns conceitos serão brevemente explicados, afinal, o foco é demonstrar o desenvolvimento do projeto e não fornecer as bases da programação aplicada.

2. Back-End

Back-End pode ser traduzido literalmente como, processo interno. Na programação, se trata quase que exclusivamente de um servidor. Existem vários tipos possíveis de servidores, de arquivos, banco de dados, mídia, e-mail, entre outros. Então, basicamente ele trata da parte que não vemos, das requisições, tratamento de dados, comunicações etc.

O tipo de servidor escolhido para um projeto pode ser de fundamental importância para o desenvolvimento e comportamento da aplicação.

No caso desta aplicação em específico escolhi um servidor que fornece uma API (Aplication Programing Interface). Essa escolha se deu pois, como o projeto seria para uma startup do ramo alimentício que realiza vendas online, essa pareceu ser a melhor opção, pois a parte mais importante seriam os dados enviados, nesse tipo de servidor todas as rotas de acesso retornam exclusivamente dados, deixando toda a responsabilidade de renderização e cálculo para o front-end.

Sabendo disso pude planejar a estrutura que segue abaixo.

2.1. Da estrutura

Estruturar os dados é tão mais importante do que os programar, e esse foi o primeiro passo, definir as classes de dados a serem utilizadas. As estruturas das classes foram feitas como abaixo:

Conforme visto acima, cada produto tem vários materiais, cada material possui um ingrediente e uma quantidade daquele ingrediente e cada ingrediente por sua vez, possui um preço.

Essa estrutura é útil pois com referencias simples podemos definir com precisão a função de cada componente. Ao componente ingrediente não cabe saber a quantidade que irá na receita do produto, isso possibilita que a mesma referência de ingrediente seja usada para mais de um produto, otimizando a aplicação.

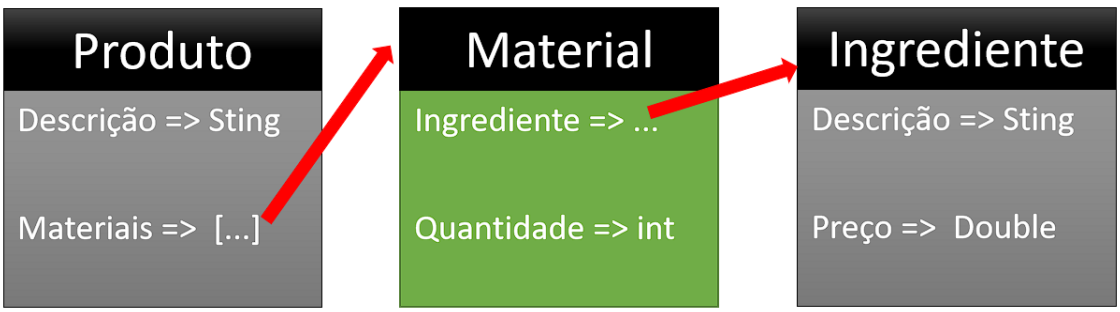

Figura 1 - Diagrama das classes

É importante notar que o preço total do produto não está presente em sua classe, isso se dá pela possível mudança nos valores individuais de cada ingrediente, gerando recalculo dos valores totais, isso ainda acontece, mas essa arquitetura impossibilita a existência de dois produtos, com os mesmos materiais gerando preços diferentes.

2.2. Das rotas e recursos

Após projetadas as classes, foi definido quais seriam as rotas a serem acessadas, quais recursos teriam etc.

Foram definidas duas rotas principais, as de produtos e as de ingredientes, ambas possuem também, um endpoints acessíveis por uma rota de administradores.

Sobre os recursos, ambas possuiriam teriam a capacidade de realizar um CRUD básico no banco de dados e retornar os valores obtidos.

As rotas ficaram de tal forma que a rota “/products”, responderia de maneira diferente dependendo do verbo HTTP utilizado requisição, por exemplo, se for feita uma requisição GET o retorno seria a lista completa de produtos, com uma requisição POST, o servidor aguardaria um novo produto para ser cadastrado e retornaria um código de estado positivo, caso ocorresse tudo corretamente.

2.3. Do banco de dados

O banco de dados foi implementado por conta da possível mudança de preços descrita no briefing do projeto. Com o banco seria mais fácil persistir os dados modificados pelo usuário.

3. Front-End

O Front-End é a cara da aplicação, sua apresentação, a janela de interação com os usuários. É de suma importância que se comunique bem com os usuários, que seja apresentável e funcional.

Foi utilizado React para programar esta parte. O React junta javaScript e HTML para chegar em uma solução que facilite a criação de aplicativos. O modelo principal que dita o funcionamento de aplicações React é a componentização, onde cada parte do código pode ser exportado como um componente separado.

3.1. Da estrutura

Como citado anteriormente, o Front-End foi programado pensando em componentes, pequenas partes que se entrelaçam, ligam e comunicam. A estrutura geral de todos é sempre a mesma, todos importam suas dependências, podem receber parâmetros para a execução e no final, retornam o componente desejado.

Cada pasta de componente possui dois arquivos que são iguais para todos, são eles: index.jsx e styles.css, esse design foi criado para facilitar a criação desses componentes.

Além disso, existem algumas pastas que possuem diversas utilidades para a aplicação, por exemplo, a função de cálculo de preços que está presente no Front-End.

Cada parte do programa tem suas responsabilidades bem definidas, facilitando o gerenciamento de cada modo de falha.

4. Conclusão e lessons learned

A aplicação está funcional, atende aos requisitos pedidos no briefing, o design do código utilizado é simples, por conta disso os comentários sobre ele acabam sendo curtos. Tanto no Back-End, quanto no Front-End as estruturas são reutilizáveis, passiveis de serem escaladas e expandidas, e vários dos conceitos SOLID de programação foram implementados nesse projeto.

Quanto ao aprendido, é evidente que o treino com uma aplicação real, melhora e muito as habilidades do programador. Mas em especial houve muito aprendizado nas questões de gerenciamento de tempo e tarefas, redes, novas tecnologia etc.