Desafios Enfrentados

Entendimento do Código Inicial: A tarefa mais árdua foi desvendar o raciocínio intricado e as condições complexas na função update_quality. Captar precisamente como as normas de negócio foram postas em prática no código confuso original demandou tempo e minúcia.

Falta de Testes Individuais: Apesar do teste "Golden Master" ser eficiente para certificar que a ação geral permanece igual, ele não examina as partes lógicas de maneira separada. Se um erro surgir, seria mais complicado achar a razão precisa sem testes individuais detalhados para cada classe de item.

Perigo de Retorno de Problemas: A cada linha alterada, pairava o receio de romper uma norma de negócio delicada ou uma situação limite não evidente. Isso reforçou a relevância de rodar os testes de "Golden Master" com extrema frequência, após cada mínima alteração.

Lições Retiradas

A Essencialidade dos Testes na Refatoração: A lição mais crucial é que refazer sem testes é imensamente arriscado. A proteção concedida pelo teste "Golden Master" foi o que fez o processo possível e seguro.

A Força dos Passos Curtos: Fazer alterações mínimas e testá-las de imediato reduz drasticamente a dificuldade e o perigo. Em vez de intentar reescrever tudo de uma só vez, a tática incremental é bem mais eficaz.

Código Claro é Código Sustentável: O resultado final não é só um código que funciona, mas um código que é simples de ler, compreender e, acima de tudo, ampliar. A ligeireza com que a função "Conjured" foi somada após a refatoração é a prova do valor do trabalho.

Regras de Design (SOLID): A refatoração nos aproximou de regras importantes de design de software:

Regra da Responsabilidade Singular (SRP): Cada função nova agora possui uma única incumbência (atualizar um tipo específico de item).

Regra Aberto/Fechado (OCP): O sistema agora está mais "aberto" para extensão (somar novos tipos de item) e "fechado" para mudança (não necessitamos mudar a lógica dos itens existentes para adicionar um novo).