RELATÓRIO INDIVIDUAL
INF01120 - TÉCNICAS DE CONSTRUÇÃO DE PROGRAMAS

Rafael Baldasso Audibert - 00287695

1. Maintainer

Acredito que a parte do maintainer tenha sido a etapa mais fácil e simples para mim. Por já ter tido experiência trabalhando com linguagens orientadas a objeto (Ruby e Python) ou com conceitos similares à OO (JavaScript e Rust), os conceitos apresentados eram facilmente correlacionados com técnicas que eu já havia utilizado antes, bastando apenas se habituar a uma linguagem fortemente tipada (o que eu devo admitir que eu gosto bastante, já que me dá uma segurança muito maior de o que eu estou fazendo realmente vai funcionar).

Me atrapalhou um pouco o fato de que o projeto bank era muito grande, com muitas classes e funcionalidades, e levou um certo tempo para conseguir entender onde cada coisa se encontrava ou se relacionavam entre si. Do meu ponto de vista, os diagramas de sequências não auxiliam muito para isso, já que quando temos laços ou ifs, o diagrama se torna extremamente confuso, sendo que muitas vezes preferi usar features da IDE para ir para a declaração do método que eu estava chamando para tentar descobrir o que ele fazia. Falarei mais da minha experiência com os diagramas de sequência ao falar sobre a etapa 2 e 3.

Essa parte do trabalho, no meu grupo, foi realizada somente pelo colega Fábio Gomes e por mim, sendo que os outros dois integrantes do grupo não fizeram nada, mesmo após solicitarmos alguma ajuda. Esse foi um dos maiores problemas do trabalho, embora nessa primeira etapa, como era mais simples, não ocasionou grande prejuízo, já que gostei do trabalho apresentado por nós.

Acho que essa parte do trabalho foi importante pra mim, por me mostrar como é herdar uma base de código de tamanho considerável, e eu precisar me inteirar dela por completo, já que nas minhas experiências profissionais, eu sempre construí o código do zero. Foi muito interessante ver maneiras inteligentes de utilizar Polimorfismo e Herança para atingir coisas grandes, e, principalmente, ver quão útil é ter uma camada de serviços, fazendo com que somente um código base, pudéssemos criar 2 interfaces independentes entre si.

2. Designer

Essa etapa foi a mais complicada (vide nota que o grupo tirou) entre as três. Boa parte da culpa por termos tirado essa nota foi por acharmos que

fazia sentido estruturarmos o código da maneira da qual fizemos, mantendo regras de negócio fora das classes de domínio, o que, ao fazermos a Etapa 3, realmente notamos que não fazia sentido, e manter essa classe de serviço no meio, tornando a interface independente da camada de domínio com seu banco de dados, torna tudo imensamente mais fácil (e mais rápido para construir). Como comentei em aula, de minha parte, isso foi provavelmente um vício de ter entrado no mundo da orientação de objetos diretamente na prática na minha vida profissional, em uma equipe que também não possuía experiência prévia em desenvolvimento OO.

Como ressalva aos outros dois integrantes do grupo, acredito que nessa fase todo mundo colaborou de maneira igual, porém, em direção errada. Acredito que o projeto realizado para a terceira etapa é muito melhor do que o que foi entregue nesta segunda etapa.

3. Developer

Acho que foi a parte mais divertida do trabalho, embora tenhamos tido vários problemas no início, já que o jeito de pensar para realizarmos *"test-driven development"* é totalmente diferente do padrão.

Para começar, refizemos o diagrama de classes e de sequência criados na etapa 2, reformulando o para ficar de maneira similar àquele disponibilizado juntamente com o enunciado do trabalho, com pequenas alterações que julgamos necessárias.

A realização dos testes foi a maior dificuldade para mim, já que eu nunca havia feito testes ANTES de eu ter um código funcionando, se baseando apenas nas regras de negócio. Acho que com o tempo, essa técnica foi se aprimorando, fazendo com que, realmente, conseguíssemos evitar alguns bugs que teriam sido deixados no código, justamente por pensarmos de maneira diferente, criando classes de equivalência, idealizando casos limites e extremos, etc. Acredito que eu gostaria de trabalhar novamente com TDD, pois dá um segurança muito maior de que o código realmente está fazendo o que ele deveria fazer de acordo com as regras de negócio.

Novamente, essa etapa trouxe problemas por causa do grupo formado. Não gostaria de reclamar tanto deles, mas infelizmente eu não senti que todos os integrantes proveram o mesmo esforço para a realização do trabalho. Do meu ponto de vista, eu e o integrante Fábio fizemos 80% dessa etapa, sendo que havíamos dividido o trabalho de maneira consideravelmente igual. Ao partirmos para a realização do relatório e criação dos slides, mesmo após solicitado, ninguém comentou nada sobre se prontificar para ajudar, e eles foram realizados inteiramente por mim e, novamente, pelo Fábio.

Acredito que eu gostei do resultado final apresentado, embora eu não tenha achado tão elegante a maneira na qual imprimimos algumas coisas na tela, já que existem impressões na tela acontecendo dentro de classes do domínio, o que ao meu ver, talvez não seja o ideal. Foi a primeira vez também que trabalhei da maneira correta com error handling, e acho que foi uma boa experiência, já que foi uma experiência fortíssima.

Ao final, vejo que foi realmente um bom projeto para concluir a matéria, já que pude colocar em prática muito das coisas práticas da orientação a objetos que aprendemos, quanto ver de verdade sobre as matérias teóricas que vimos (principalmente na visualização dos problemas que o alto acoplamento pode trazer, já que temos dificuldades com isso).