UNIVERSIDADE SÃO JUDAS TADEU

APLICAÇÃO DA TÉCNICA DE TESTES ÁGEIS TDD

GIULIA GABRIELA DE LIMA SANTOS | 823122979
KAUE BRITO VIEIRA | 824211851
KAUÊ DIB DE SOUZA DIAS 823149871
MURILO BONUCCELLI DE OLIVEIRA | 823148988
VICTOR IGNACIO | 823125249
VINÍCIUS SANTANA TEIXERA | 82319112

O Desenvolvimento Orientado por Testes (TDD – *Test-Driven* *Development*) é uma prática ágil que propõe a criação de testes automatizados antes mesmo da implementação do código. O objetivo principal é garantir que o software atenda exatamente aos comportamentos esperados, validando continuamente cada funcionalidade implementada. Essa técnica se baseia no ciclo Red – Green – Refactor, composto por três etapas:

- Red (Vermelho): escrever um teste que inicialmente falha, já que o código ainda não foi implementado;
- Green (Verde): desenvolver o mínimo de código necessário para que o teste passe;
- Refactor (Refatorar): melhorar o código sem alterar sua funcionalidade, mantendo os testes bem-sucedidos.

No caso do algoritmo de busca binária, a aplicação do TDD permite um desenvolvimento iterativo, validando cada comportamento do algoritmo por meio de testes antes da codificação da lógica propriamente dita.

7.1 Planejamento dos Testes (Red)

Foram definidos, antecipadamente, os comportamentos esperados do algoritmo com base nos casos de teste descritos na seção 4 deste trabalho. Os seguintes cenários foram utilizados como base para a escrita dos testes:

- O valor buscado está presente no vetor e é encontrado na primeira tentativa;
- O valor buscado é menor do que o elemento do meio (iK < meio);
- O valor buscado é maior do que o elemento do meio (iK > meio);
- O valor buscado não existe no vetor;
- O vetor contém apenas um elemento;
- O vetor está vazio.

Para cada um desses comportamentos, foram elaboradas assertivas que representam o que o código deve retornar ao ser executado em tais condições**.**

7.2 Implementação guiada pelos testes (Green)

Após a criação dos testes, o algoritmo foi desenvolvido de forma incremental. A cada nova funcionalidade prevista por um teste, implementou-se apenas o suficiente para que este fosse aprovado. Por exemplo, ao testar a busca de um valor presente no meio do vetor, o código foi ajustado para atender especificamente a esse cenário antes de evoluir para os demais.

Essa abordagem garantiu que a implementação estivesse sempre em conformidade com as expectativas funcionais, reduzindo a possibilidade de falhas lógicas**.**

7.3 Refatoração com segurança (Refactor)

Com todos os testes passando, o código foi revisado para melhorias estruturais, como legibilidade, simplificação de expressões e clareza na definição de variáveis. A presença dos testes automatizados assegurou que nenhuma alteração impactasse negativamente a funcionalidade geral da busca binária.

7.4 Vantagens observadas

A adoção do TDD trouxe os seguintes benefícios ao processo de desenvolvimento:

- Confiabilidade: o código foi validado em todos os seus comportamentos esperados desde o início;
- Manutenibilidade: alterações no algoritmo puderam ser feitas com segurança, pois os testes garantiam a integridade da lógica;
- Documentação funcional: os testes serviram como documentação viva do que se espera do algoritmo;
- Cobertura completa: todos os fluxos lógicos descritos na metodologia de teste foram cobertos naturalmente ao longo do ciclo Red-Green-Refactor**.**