TEST DRIVEN DEVELOPMENT

O que é o método TDD?

TDD, ou Desenvolvimento Orientado por Testes (Test Driven Development), é uma abordagem de desenvolvimento de software que coloca uma ênfase significativa na escrita de testes automatizados antes de escrever o próprio código do software. Criada por Kent Beck e um dos pilares do XP (Extreme Programming), essa técnica prioriza os testes de códigos baseada em pequenos ciclos de repetições, em que antes é criado um teste para cada funcionalidade do sistema. Assim, desenvolvemos o nosso software baseado em testes que são escritos antes do nosso código de produção.

Ciclo TDD

O processo de desenvolvimento de software conhecido como TDD (Desenvolvimento Orientado por Testes) segue um ciclo bem definido, composto por três etapas essenciais, conhecido como Red-Green-Refactor.

Red (Vermelho): Nesta etapa, o primeiro passo é escrever um teste automatizado que define o comportamento desejado do código a ser implementado. Como esse código ainda não foi criado, o teste irá falhar inicialmente.

Green (Verde): Com o teste falhando, passamos para a etapa de implementação do código mínimo necessário para fazer o teste passar. O objetivo aqui é escrever apenas o suficiente para que o teste não falhe mais, mantendo o foco no requisito específico em questão.

Refactor (Refatoração): Uma vez que o teste está passando, é hora de refatorar o código. Isso envolve reorganizar o código, melhorar sua estrutura, nomear variáveis de forma mais significativa e aplicar boas práticas de programação. O objetivo é garantir que o código continue limpo e legível, enquanto mantém seu comportamento correto.

Este ciclo Red-Green-Refactor pode ser repetido várias vezes, conforme necessário, em cada nova implementação ou alteração de funcionalidade. É importante executar todos os testes existentes após cada alteração no código, para garantir que as novas implementações não quebrem funcionalidades existentes.
Essa abordagem traz diversos benefícios, incluindo feedback rápido sobre as implementações, maior segurança nos processos de refatoração e adição de novas funcionalidades, e código mais limpo e robusto. Ao seguir este ciclo de desenvolvimento, os desenvolvedores podem garantir não apenas a funcionalidade do código, mas também sua qualidade e manutenibilidade a longo prazo.

Benefícios do TDD

1. Qualidade e Confiabilidade:
- Garantia de qualidade desde o início do desenvolvimento.
- Prevenção da introdução de bugs no código.
- Identificação precoce de erros e falhas.

2. Refatoração Segura:
- Permite alterações no código com confiança.
- Detecta regressões durante o processo de refatoração.
- Facilita melhorias contínuas no design do software.

3. Documentação Viva:
- Descreve o comportamento esperado do sistema.
- Serve como uma referência clara para entender e manter o código.

4. Eliminação do Medo:
- Proporciona confiança no desenvolvimento de aplicações.
- Suíte sólida de testes automatizados aumenta a segurança no código.

5. Feedback Rápido:
- Identifica imediatamente problemas na implementação.
- Agiliza a correção de problemas e falhas.

6. Melhoria no Design:
- Incentiva a modularidade e o desacoplamento do código.
- Promove um design mais eficaz e sustentável.

7. Facilita a Colaboração:
- Atua como uma forma clara de comunicação entre membros da equipe.
- Ajuda a entender o que está sendo implementado em diferentes partes do sistema.

8. Facilita a Manutenção:
- Permite alterações no código com mais confiança.
- Detecta se algo foi quebrado durante a modificação do código.

9. Redução de Custos a Longo Prazo:
- Economize tempo e recursos identificando bugs mais cedo.
- Torna a manutenção mais eficiente, reduzindo custos operacionais.

10. Maior Confiança no Código:
- Fornece uma suíte sólida de testes automatizados.
- Aumenta a confiança de que o código está funcionando conforme o esperado.

11. Foco nos Requisitos:
- Encoraja a pensar sobre os requisitos e comportamentos esperados antes da codificação.
- Ajuda a manter o foco na entrega de valor real aos usuários.

12. Hábito de Qualidade:
- Promove um hábito de escrever código de alta qualidade desde o início do desenvolvimento.
- Melhora as habilidades dos desenvolvedores ao longo do tempo.

Esses benefícios destacam a importância do TDD no desenvolvimento de software e como ele pode contribuir significativamente para a qualidade, eficiência e colaboração no processo de desenvolvimento.