COTI INFORMÁTICA
ESCOLA DE NERDS

TDD – Test Driven Development

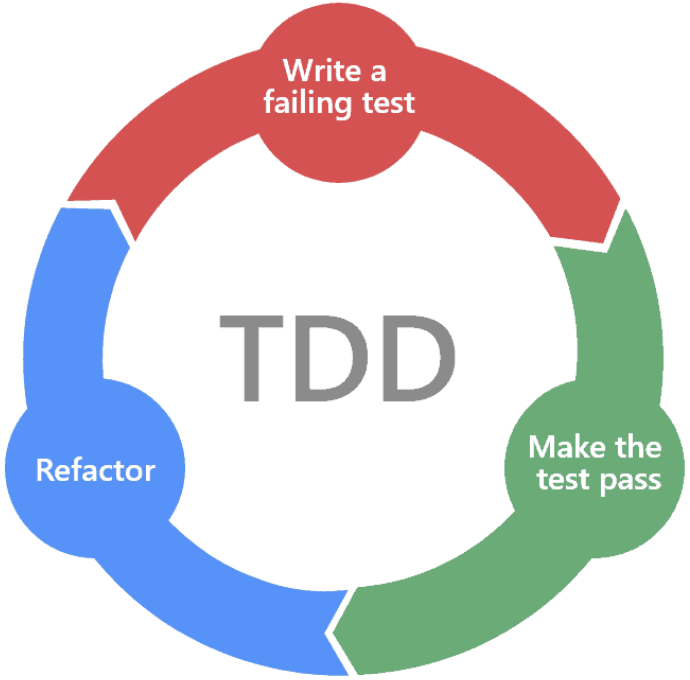
Java WebDeveloper - FullStack

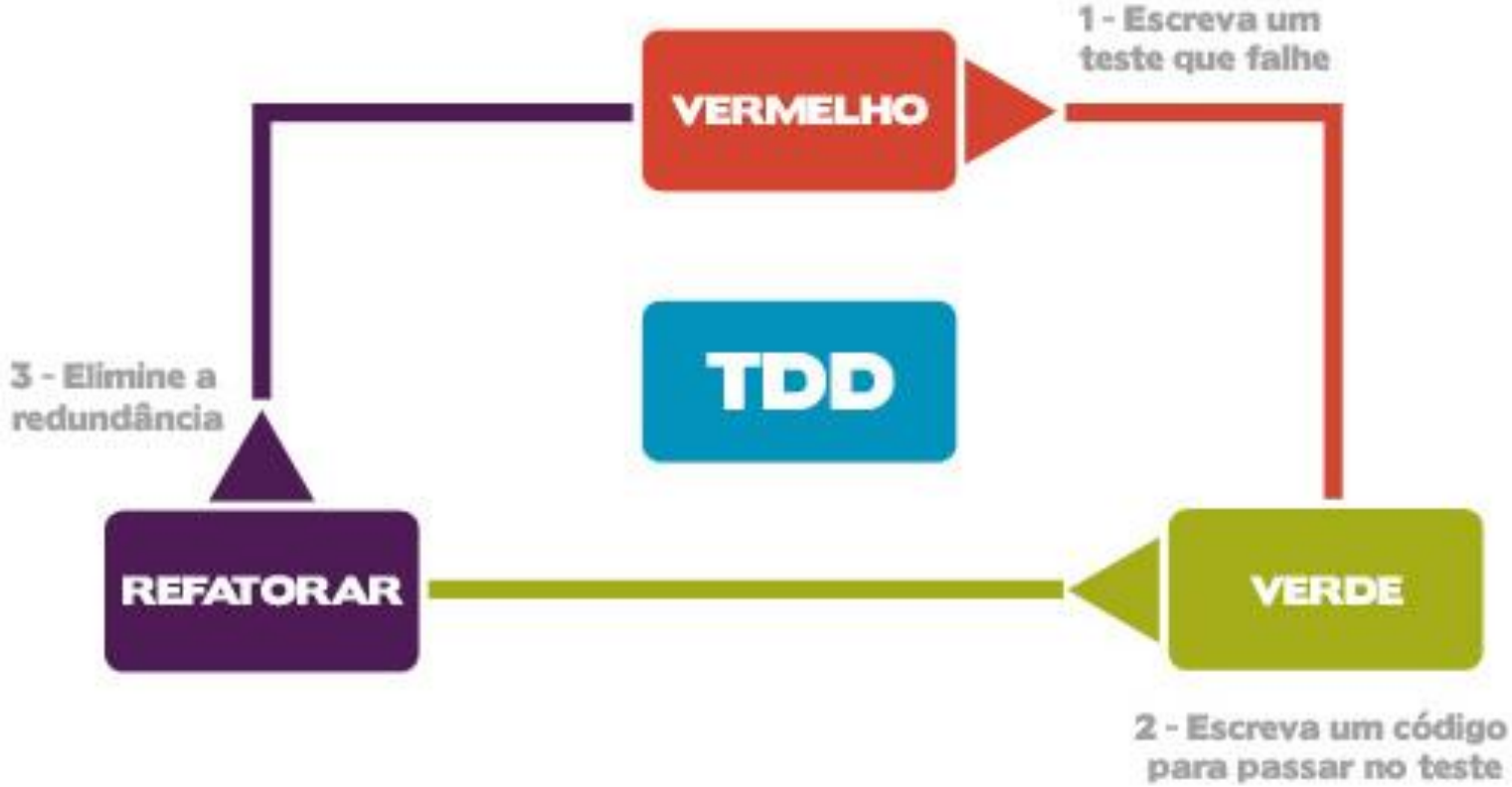
Professor Sergio Mendes

COTI INFORMÁTICA
ESCOLA DE NERDS

Test Driven Development

(Desenvolvimento orientado a testes)



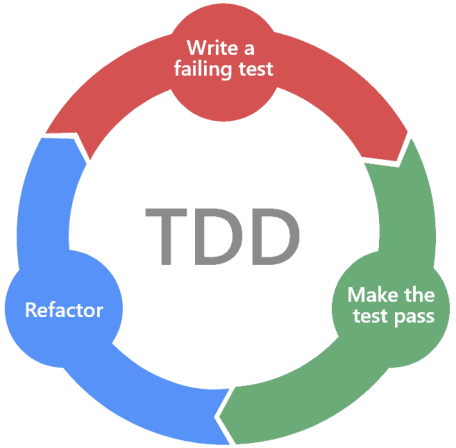


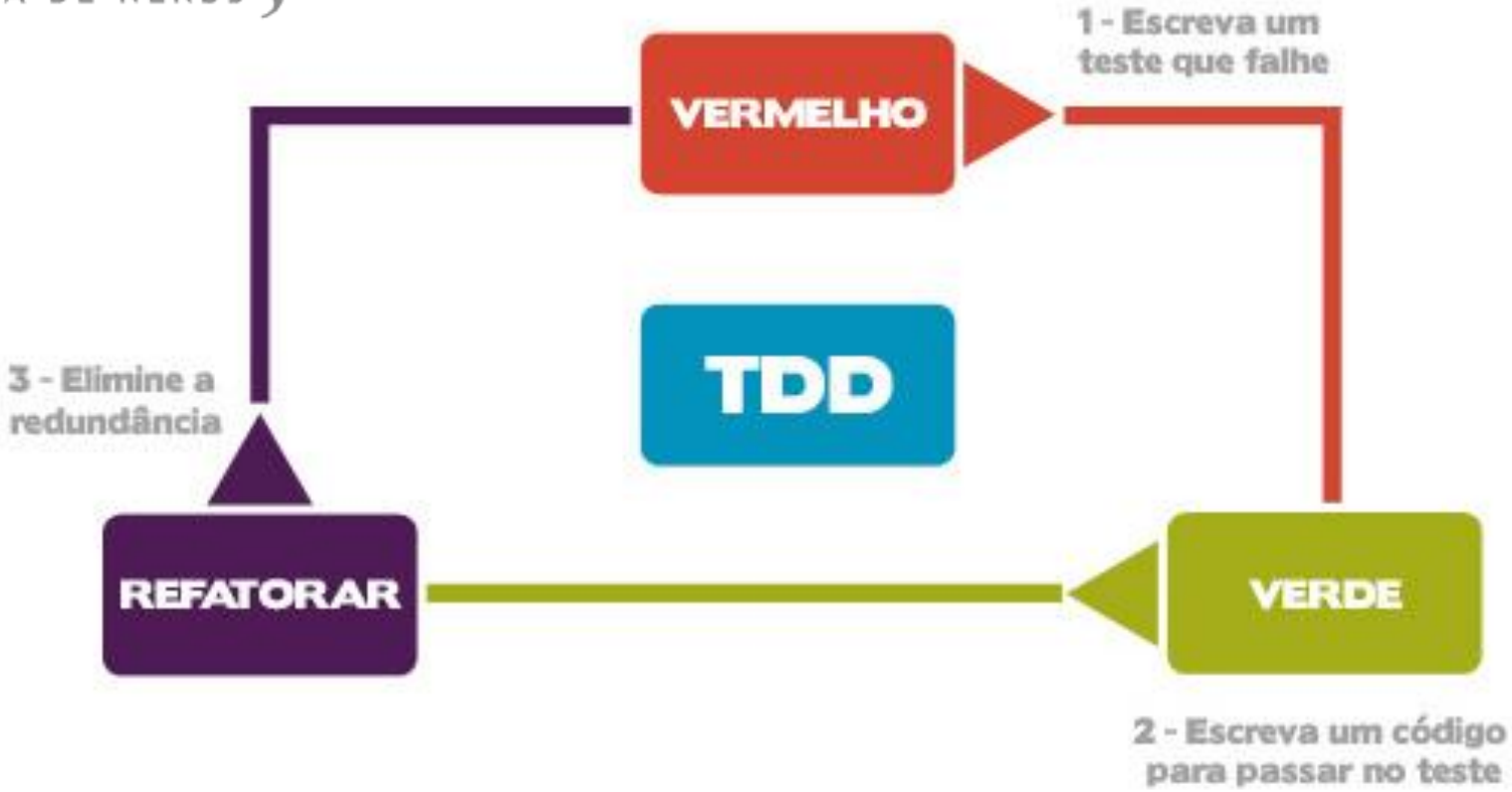
TDD é uma sigla para **Test Driven Development**, ou *Desenvolvimento Orientado a Testes*. A ideia do TDD é que você trabalhe em ciclos. Estes ciclos ocorrem na seguinte ordem:

1) Primeiro, escreva um teste unitário que inicialmente irá falhar, tendo em vista que o código ainda não foi implementado;

2) Crie o código que satisfaça esse teste, ou seja: implemente a funcionalidade em questão. Essa primeira implementação deverá satisfazer imediatamente o teste que foi escrito no ciclo anterior;

3) Quando o código estiver implementado e o teste satisfeito, refatore o código para melhorar pontos como legibilidade. Logo após, execute o teste novamente. A nova versão do código também deverá passar sem que seja necessário modificar o teste escrito inicialmente.





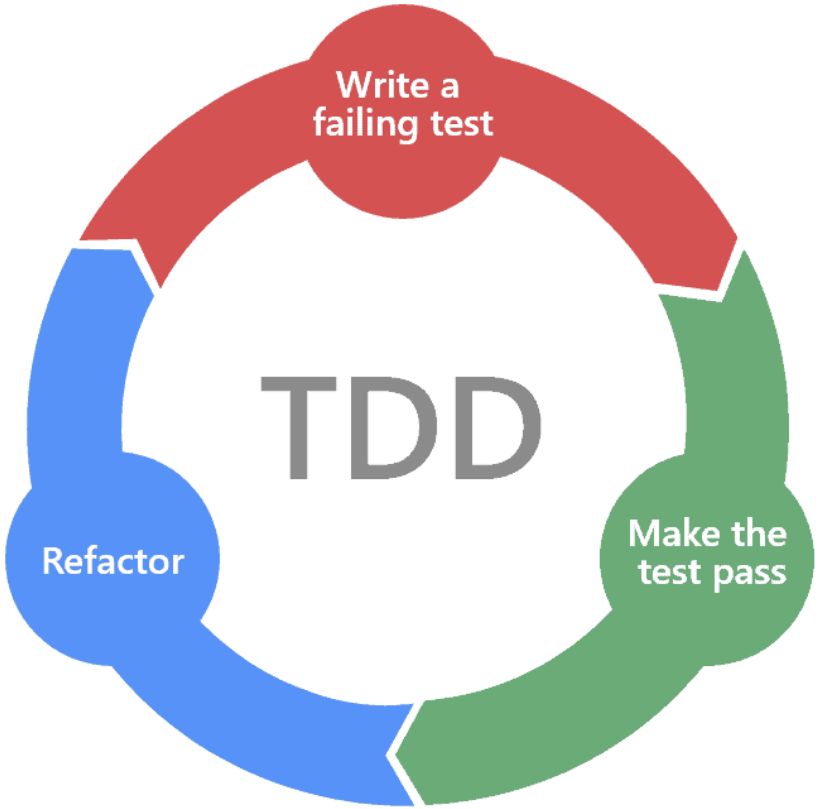
Red: escreva um pequeno teste automatizado que, ao ser executado, irá falhar;

Green: implemente um código que seja suficiente para ser aprovado no;

Refactor: refatore o código, deixando-o mais funcional e mais limpo.

1. Adicione um teste

Em **Test Driven Development**, cada nova funcionalidade inicia com a criação de um teste. Este teste precisa inevitavelmente falhar porque ele é escrito antes da funcionalidade a ser implementada (se ele falha, então a funcionalidade ou melhoria 'proposta' é óbvia). Para escrever um teste, o desenvolvedor precisa claramente entender as especificações e requisitos da funcionalidade.



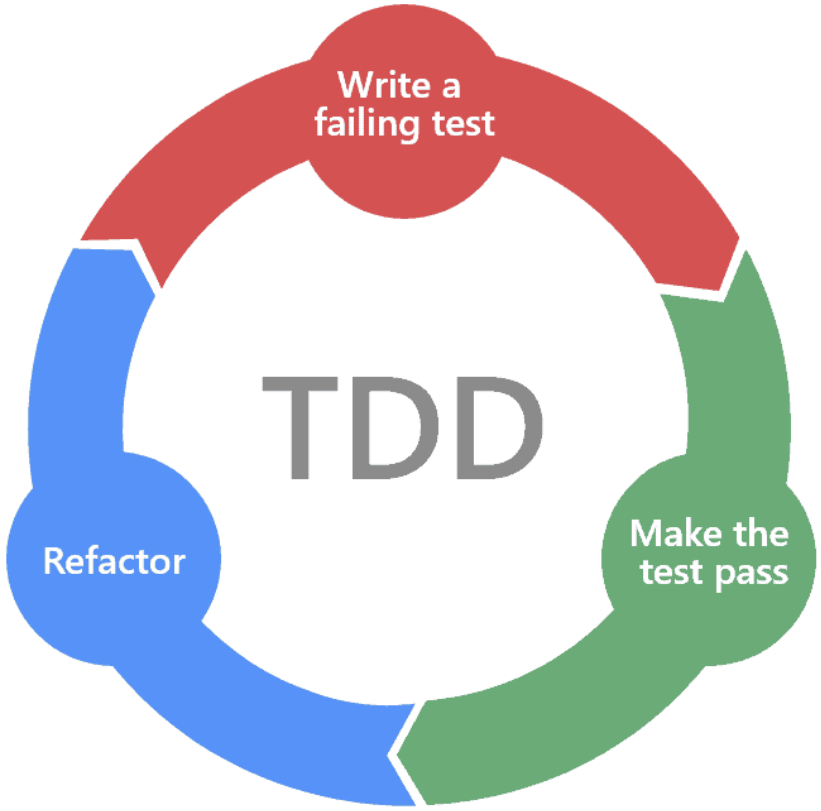
O desenvolvedor pode fazer isso através de casos de uso ou user stories que cubram os requisitos e exceções condicionais. Esta é a diferenciação entre desenvolvimento dirigido a testes entre escrever testes de unidade 'depois' do código desenvolvido. Ele torna o desenvolvedor focado nos requisitos 'antes' do código, que é uma sutil mas importante diferença.

2. Execute todos os testes e veja se algum deles falha

Esse passo valida se todos os testes estão funcionando corretamente e se o novo teste não traz nenhum equívoco, sem requerer nenhum código novo.

Pode-se considerar que este passo então testa o próprio teste: ele regula a possibilidade de novo teste passar.

O novo teste deve então falhar pela razão esperada: a funcionalidade não foi desenvolvida. Isto aumenta a confiança (por outro lado não exatamente a garante) que se está testando a coisa certa, e que o teste somente irá passar nos casos intencionados.

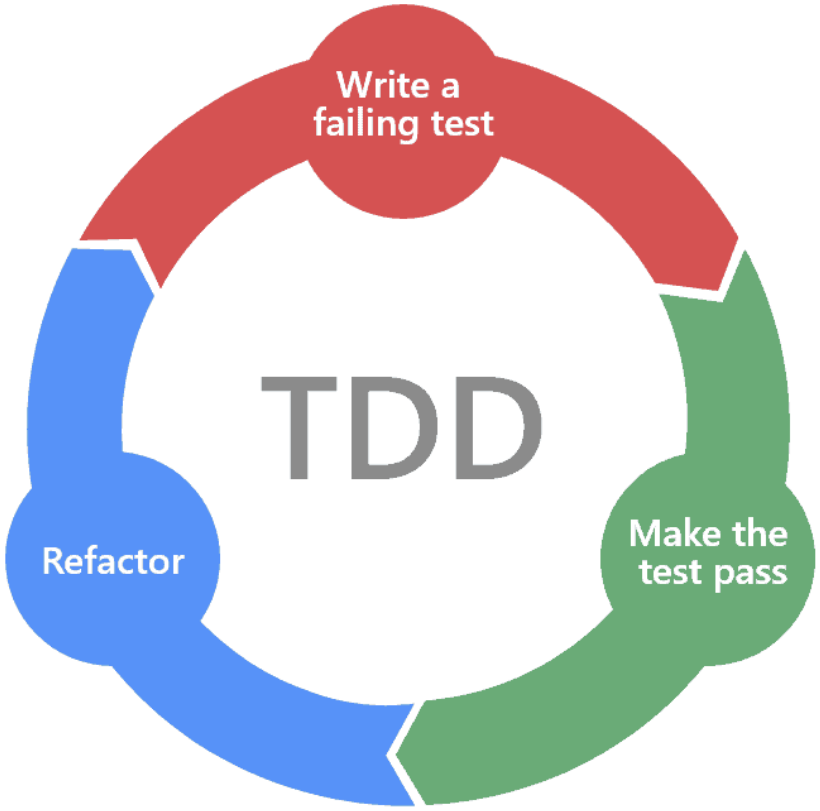


3. Escrever código

O próximo passo é escrever código que irá ocasionar ao teste passar.

O novo código escrito até esse ponto poderá não ser perfeito e pode, por exemplo, passar no teste de uma forma não elegante. Isso é aceitável porque posteriormente ele será melhorado.

O importante é que o código escrito deve ser construído *somente* para passar no teste; nenhuma funcionalidade (muito menos não testada) deve ser predita ou permitida em qualquer ponto.

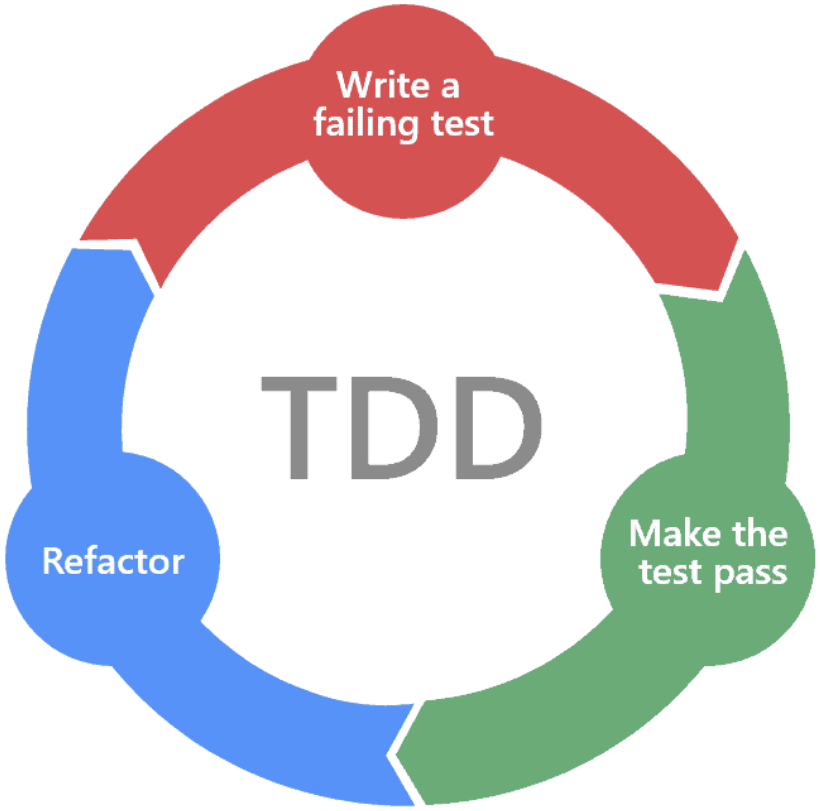


4. Execute os testes automatizados e veja-os executarem com sucesso

Se todos os testes passam agora, o programador pode ficar confiante de que o código possui todos os requisitos testados. Este é um bom ponto que inicia o passo final do ciclo TDD.

5. Refatorar código

Nesse ponto o código pode ser limpo como necessário. Ao re-executar os testes, o desenvolvedor pode confiar que a refatoração não é um processo danoso a qualquer funcionalidade existente. Um conceito relevante nesse momento é o de remoção de duplicação de código, considerado um importante aspecto ao design de um software.

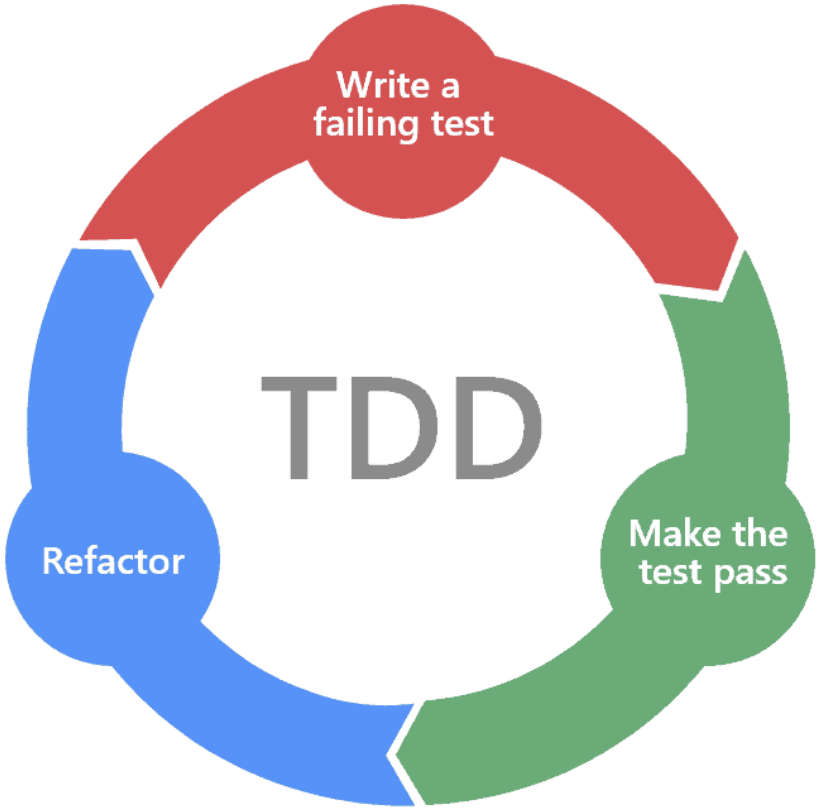


6. Repita tudo

Iniciando com outro teste, o ciclo é então repetido, empurrando a funcionalidade a frente.

O tamanho dos passos deve ser pequeno - tão quanto de 1 a 10 edições de texto entre cada execução de testes.

Se novo código não satisfaz rapidamente um novo teste, ou outros testes falham inesperadamente, o programador deve desfazer ou reverter as alterações ao invés do uso de excessiva depuração.



{ COTI INFORMÁTICA
ESCOLA DE NERDS }