Aula 30

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

TDD

TDD

TDD é uma sigla em inglês para Test Driven Development, que significa Desenvolvimento Orientado a Testes.

O TDD é uma metodologia de desenvolvimento, que consiste em guiar todo o fluxo de desenvolvimento em cima dos processos de testes.

Todos os processos de testes são feitos em forma de código, são de fato programados, podendo todos serem executados de forma automatizada.

TDD

A ideia é que, toda vez que for implementado uma nova funcionalidade na aplicação, os processos de testes serão criados simultaneamente, muitas vezes até mesmo antes da funcionalidade ser implementada.

Como a metodologia sugere inverter a ordem dos trabalhos, fazendo o teste primeiro e depois a funcionalidade, ela acaba se tornando um pouco impopular entre alguns desenvolvedores.

TDD

A princípio, pode parecer estranho escrever um teste sem ao menos ter a respectiva função pronta. Mas a ideia por trás disso é justamente facilitar a criação do código e evitar a inserção de erros e bugs no sistema.

Depois de se acostumar e pegar o jeito com o fluxo do trabalho, o processo de desenvolvimento pode fluir muito melhor, e traz muitos resultados positivos ao projeto.

TDD

- Ciclo do TDD

O TDD trabalha com pequenos ciclos de execução, divididos em três etapas:

- Red
- Green
- Refactor

TDD

- Ciclo do TDD (Red)

Na etapa red, é codificado um processo de teste que irá falhar.

Essa falha ocorre, não porque o processo é mal feito, mas porque a funcionalidade que o processo irá testar ainda não existe.

TDD

- Ciclo do TDD (Green)

Na etapa green, são realizados todos os ajustes necessários para fazer com que o processo de teste seja executado corretamente.

TDD

- Ciclo do TDD (Refactor)

Na etapa refactor, são realizadas refatorações no código, retirando duplicidades, renomeando variáveis, fazendo uso de padrões de codificação, e diversas outras ações que irão melhorar o código da funcionalidade.

TDD

- Ciclo do TDD

Todo o fluxo do TDD é feito em pequenos ciclos de repetições, podendo o mesmo método, passar por diversos ciclos, até que todas as suas funcionalidades sejam devidamente testadas.

TDD

- Vantagens

Cobertura de Código: Como existe um teste associado a cada funcionalidade, é possível ter a certeza de que todo o código foi executado e os erros são encontrados no início do desenvolvimento.

Segurança: A implementação de novas funcionalidades é um processo delicado, pois há o risco de que a alteração gere bugs no sistema. Com o TDD, esse risco é reduzido, dando mais segurança para quem está desenvolvendo o software.

TDD

- Vantagens

Feedback Rápido: Visto que as funcionalidades são testadas logo após a sua criação, é possível obter um feedback quase instantâneo do seu funcionamento.

Depuração Simplificada: O processo de depuração se torna mais intuitivo, uma vez que, quando um teste falha, é mais fácil identificar onde se encontra o problema.

TDD

- Vantagens

Documentação do Sistema: Os próprios testes podem servir como um tipo de documentação do software. Afinal, ao lê-los, é possível entender facilmente como o código funciona.

Maior Produtividade: Como o código é flexível e limpo, gasta-se menos tempo com a correção de bugs e implementação de novas funcionalidades, consequentemente, a produtividade da equipe é maior.

TDD

- Vantagens

Melhoria no Raciocínio: O TDD também é benéfico para as pessoas desenvolvedoras. Isso porque é necessário lapidar o processo de raciocínio a fim de elaborar códigos de qualidade que, ao mesmo tempo, atendam os requisitos, não insiram erros na aplicação, sejam simples e passem nos testes.

TDD

- TDD x Testes ao Final da Aplicação

A principal diferença entre TDD ou testar ao final está no tipo de feedback que obtivemos ao realizar uma nova funcionalidade.

No TDD, os testes estão escritos e são sempre executados. Por isso, desde o momento em que é criado o novo recurso, obtemos feedbacks.

TDD

- TDD x Testes ao Final da Aplicação

Já na abordagem de escrever o teste depois que o código foi escrito, temos um feedback mais demorado e muitas vezes até mais complexo, já que a lógica pode ter sido escrita de uma maneira complexa e a refatoração acaba sendo um retrabalho muito grande.

Por consequência da demora, criamos testes que avaliam a funcionalidade, mas que podem também comprometer a qualidade do código e o feedback obtido.

NUnit

NUnit

O NUnit é um framework de testes unitários para a plataforma .Net. Criado com base no JUnit, que é um framework de testes para o Java.

Ele pode ser usado para automatizar testes, e pode ser utilizado para seguir a metodologia TDD.

NUnit

- NuGet

O NUnit, assim como quase todos os frameworks .Net, é gerenciado pelo NuGet.

Para as nossas aulas, vamos precisar de três pacotes NuGet diferentes, para que possamos executar os nossos processos de testes sem problemas.

NUnit

- NuGet

NUnit: O framework principal, para executar os nossos processos de testes.

NUnit3TestAdapter: Um framework auxiliar, que vai disponibilizar algumas ferramentas no visual studio para executarmos os nossos processos de testes.

Microsoft.NET.Test.Sdk: Um framework auxiliar, que vai configurar a nossa aplicação para que seja possível executar os processos de testes.

NUnit

Por causa do pacote Microsoft.NET.Test.Sdk, que vai configurar a execução dos processos de testes, é altamente recomendado, que todas as classes de testes fiquem em um projeto separado, para não existir conflitos de execuções.