1. **Quais as vantagens do TDD?**

Redução de erros: O TDD identifica erros no código antes que eles aconteçam, permitindo que o desenvolvedor evite-os.

Código mais limpo: O TDD promove um código mais limpo e de fácil manutenção

Facilidade de refatoração: O TDD oferece segurança no processo de refatoração, pois os testes são rodados novamente a cada refatoração.

1. **O que você acha sobre o Programador desenvolver o teste antes de programar o software?**

Escrever testes primeiro ajuda a garantir que o código atenda aos requisitos especificados e funcione corretamente desde o início.

1. **O desenvolvedor do software realizar o teste unitário influencia na função de um Analista de Qualidade de Teste (QA)?**

Com os desenvolvedores realizando testes unitários, muitos erros são identificados e corrigidos antes de chegar ao QA, permitindo que o QA se concentre em testes mais complexos e de integração.

Testes unitários bem escritos podem servir como documentação adicional, ajudando os QAs a entenderem melhor o comportamento esperado do software.

Algo a se perceber sobre escrever testes, especialmente se você estiver fazendo TDD (onde você escreve o teste primeiro, com entradas e resultados esperados, e então implementa o código com base no teste), é que isso vai economizar tempo, esforço e até código mais tarde.

Por quê? Muitos motivos, mas alguns dos principais são:

Porque você terá código provavelmente "correto", já que testes adequados o provarão executando-o em todos os seus vários casos de teste, certificando-se de que o resultado é o que você espera.

Você terá que voltar e refatorar as coisas com menos frequência, porque escrever uma bateria de testes que exigem o resultado desejado o forçará a escrever código que o leva a esse resultado na primeira vez, em vez de implementar cegamente e depois descobrir que precisará reformular completamente a classe para obter tudo o que você queria em primeiro lugar. Isso não impede o escopo de rastejar de requisitos que mudam, mas essa é uma realidade frustrante do desenvolvimento de software, não importa como você o faça.

Quando você encontrar um bug que não levou em consideração em seus testes, você pode adicionar um novo teste ou modificar testes existentes para levar isso em consideração, garantindo que isso nunca aconteça novamente.

Você terá MUITO menos regressões (bugs que você corrigiu, mas que voltam em uma versão futura), porque, se o bug for reintroduzido (por meio de uma reversão ou vários outros meios), os testes existentes falharão e você não lançará uma versão com uma regressão. Regressões parecem muito ruins em um ambiente profissional, especialmente se um teste tivesse impedido que isso acontecesse uma segunda vez. E isso se tornou mais importante nos últimos 10 anos ou mais. Onde você pode ter conseguido se safar na década de 2000, se safar na década de 2020 é difícil, especialmente porque as ferramentas estão tão bem amadurecidas e fáceis de usar agora.

Testar é algo que não é realmente divertido, mas precisa ser considerado indissociável do processo de desenvolvimento. Se você escrever algo não trivial e ele não tiver um teste, você falhou. Infelizmente, porque não é divertido, porque é trabalho extra e porque normalmente não é ensinado em currículos formais de ciência da computação, tende a ser uma habilidade e um hábito que o desenvolvedor adquire à medida que amadurece profissionalmente. Seu parente está fazendo um favor MUITO sólido ao tentar colocá-lo nesse caminho AGORA.

O valor dos testes para um projeto pequeno e simples pode não ser aparente ou até mesmo significativo, mas também é normalmente muito mais fácil em um projeto pequeno. Quando você começar a trabalhar ou construir aplicativos complexos, realmente apreciará tê-lo - especialmente se várias pessoas estiverem trabalhando nesse projeto. É muito mais fácil juntar um monte de peças quando você sabe com certeza que essas peças individuais funcionam corretamente ou como esperado, isoladamente. Então vêm os testes de integração, que são as mesmas ideias, mas certificando-se de que as partes do código que precisam funcionar juntas, de fato, funcionam corretamente, quando usadas juntas.