Pesquisador. Muito obrigado por participar dessa entrevista. Ela tem como objetivo compreender os processos de criação e manutenção de código de testes de unidade e entender como funciona a verificação da qualidade desse código de teste. Não há respostas certas ou erradas ou desejáveis ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que pensa. Reforçando um item do termo de consentimento, nossa entrevista está sendo gravada. O objetivo disso é para que eu possa obter todos os detalhes, mas ao mesmo tempo ser capaz de manter uma conversa atenta com você. Garanto que todos os seus comentários permanecerão confidenciais. O relatório final conterá os comentários dos participantes sem nenhuma referência aos indivíduos. OK?

E. Perfeito. Combinado.

P. Como você começou a trabalhar com teste de software?

E. Eu comecei a trabalhar com teste de software em 2012, quando eu tive a minha primeira experiência com desenvolvimento de forma profissional. Já trabalhava, já tinha experiências com projetos pessoais, mas minha primeira experiência foi com o desenvolvimento de um gravador telefônico, uma espécie de gravador telefônico de empresas de telemarketing, e o sistema não tinha nenhum tipo de cobertura de testes, então eu atuei num time onde o foco era aumentar essa cobertura de 0 para 90%, e a gente teve um desafio considerável nessa empreitada, que na época não tínhamos um framework de testes bem construído como temos hoje m dia para o .NET. Então foi um desafio bem marcante na minha carreira.

P. Quais foram suas fontes de aprendizado sobre código de teste?

E. Eu acompanho algumas pessoas, alguns canais no Youtube. O canal .NET, do Renato Groffe e os amigos dele. E eu li alguns livros, TDD, e muita coisa também eu vejo em relação a padrões de projetos de times que eu acabo participando então eu tento entender a forma que eles pensaram para resolver aquela solução, aqueles padrões que eles usavam e tento extrair o melhor dos mundos. Fazer algum tipo de compilado e montar o meu próprio padrão de estudos mesmo.

P. Em quais linguagens você cria testes?

E. C#, com .NET core. .NET framework. PHP, Ruby n Rails, NodeJS.

P. Qual plataforma do seu projeto atual?

E. .NET Core no back and e Angular 8 no Front.

P. Me conte como é seu processo de criação de teste.

E. A gente pratica o TDD, logo a gente tem a definição das regras de negócio, a gente parte direto para os testes e não direto para a implementação. Eu crio o código de teste como se eu tivesse implementando o código da implementação final, eu sigo padrões de separação de responsabilidades, eu sigo o solid no meu teste unitário também, padrões de nomenclatura, mas basicamente eu crio um projeto ou existe um projeto de teste aonde eu crio um arquivo de teste referente aquela funcionalidade e deixo nos limites em que essa funcionalidade precisa ser testada. Eu começo a desenvolver códigos da forma em que a gente consiga garantir que isso seja atendido. Obviamente no início eles vão falar, mas depois eles vão passar e à medida que eles vão sendo checados a gente vai garantindo que a aplicação atende a esses requisitos.

P. Existe algum fluxograma, template... algum documento que padronize esse processo de criação?

E. Não existe. A gente tem entre os desenvolvedores meio que uma convenção, mas isso não é documentado nem padronizado pela empresa. A gente que definiu, mas mesmo assim é um acordo nosso.

P. Quais ferramentas são utilizadas para criar teste?

E. Os testes unitários são criados no próprio Visual Studio com XUnit. A gente tem teste de integração com Postman Newman e tem os testes também com o Jest no Front end, e com uma ferramenta chamada Selenium, se não me engano, é que essa parte do Selenium eu não faço, mas sei que existem testes feitos com Selenium também para o Front end e a parte de tela.

P. Especificamente sobre teste unitário, como é o processo de verificação de qualidade desse código?

E. Normalmente os softwares para varreduras de código, tipo SonarQube, Code Climate, eles fazem essa varredura no código que vai ser entregue e não nos códigos de teste. A gente definiu que esse tipo de código vai ser varrido pelo Code Climate e pelo SonarQube. Então a mesma qualidade que a gente tem que entregar, que vai ser a funcionalidade em si, tem que ser aplicada também no código de teste. Elas seguem um padrão definido pela própria ferramenta. A gente consegue customizar isso também, aonde alguns critérios são levados em consideração para medir a qualidade do código, medir quando se tem alguma deficiência, algum débito técnico ou algo do tipo.

P. Passando para o processo de manutenção do código de teste, como ele ocorre?

E. Entre os desenvolvedores back end, a gente tem uma convenção do padrão de como vai ser feito, então para gente não é nada problemático, quando tem que fazer uma manutenção no código, a gente segue o TDD, então a primeira coisa que a gente vai fazer é ver se os testes estão corretos, se estão de acordo com a necessidade, senão é lá que a gente vai começar a fazer a alteração. Então para a gente, o teste é sempre o início do processo. É dali que se “starta” tudo. Eu não vejo nenhum tipo de problema, nenhum tipo de barreira em relação a manutenção de teste, mesmo tanto os que eu faço quanto os que outros desenvolvedores fazem e que eu tenho que atuar. Exemplo: Uma vez que a implementação for aceita, quer dizer, que os testes estão e acordo com a necessidade, eu só vou mexer nesses testes de novo se a funcionalidade sofrer alguma alteração de regra de negócio: “Agora não vai ser mais dessa forma”, então vou ter que adequar o teste para que atenda aquela necessidade, e aí a aplicação passe a fazer com que o teste seja aprovado.

P. O que você conhece sobre test smell?

E. Sinceramente eu não sei. Pelo nome assim. Possa ser que eu conheça de alguma outra forma, mas eu não conheço essa nomenclatura.

P. Eu vou te passar o conceito e dois exemplos pois as próximas perguntas estarão relacionadas a isso. “test smells são pontos fracos na estrutura do código de teste, tendo um impacto negativo na sustentabilidade da atividade de teste, pois os tornam complexos, difíceis de entender e modificar, prejudicando a repetibilidade, independência e estabilidade dos testes.” Vou te mandar dois exemplos e dizer porque são considerados smells. E caso você tenha alguma pergunta, é só perguntar. O primeiro exemplo de test smell que eu te mandei, ele é o chamado de Empty Test, ele é considerado um smell porque não tem nenhum script executável. Em relação ao outro, ele é chamado Conditional Test Logic. Ele é considerado um smell pois tem estruturas de condição e de repetição dentro dele. Então uma vez que esse teste falhe, torna-se mais difícil identificar o porquê que esse teste falhou, em qual momento dessa estrutura condicional ou de repetição que ele falou. Então o mais indicado seria separar essas condições dentro de diferentes testes para que uma vez que eles falhem seja mais fácil de identificar qual foi o teste e porque que ele falhou.

E. Perfeito, entendi.

P. Esses só alguns exemplos de smells, existem outros relacionados a asserts, a outras informações de estruturas, mas eles sempre estarão relacionados a tornar os testes mais complexos, a serem difíceis de entender a sua escrita ou de modificar e que acabem prejudicando a repetição, a sua independência e estabilidade.

P. Você possui alguma dúvida sobre esse tema?

E. Não. Realmente eu não conhecia por esse nome, mas eu fiz a relação com code smells, que você tem as mesmas situações dentro do código da aplicação sem ser o teste em si. Mas agora eu consegui captar a mensagem.

P. Os test smells descendem dos code smells, mas eles têm uma suite específica de test smells.

E. Perfeito.

P. Como test smells são tratados no processo de criação do código de teste de unidade?

E. Uma das saídas desses softwares que eu mencionei, que são o SonarQube e o Code Climate, são esses test smells. Eles conseguem encontrar alguns deles, não consigo te garantir que eles vão encontrar todos, mas com certeza esses dois exemplos eles pegariam, pois a gente não consegue publicar um projeto em que esteja com esses tipos de estrutura de testes, testes com conteúdo comentado por exemplo, teste vazio, teste que tenha complexidade maior que 1. O que é um teste com complexidade maior que 1? Você ter mais de uma camada de código, você ter um “if” dentro do código ele já fala "olha, não é assim que funciona isso. Você não pode ter um “if” dentro de um teste". Se tem um “if” são dois testes. Então a gente tem essa tratativa, a saída dele é direto do Code Climate, por exemplo.

P. Esse processo também ocorre para manutenção do código de teste?

E. Sim. Essa tratativa, essas varreduras ocorrem toda vez que você vai fazer, você pode fazer manualmente, você pode rodar uma varredura na hora que você quiser, e também roda automaticamente toda vez que você faz uma solicitação de pull request /code review. Então quando você solicita um code review, antes dele ser mandado para o code review, é feita essa verificação e se tiver esses itens a serem ajustados a serem pontuados, ele não possibilita que ele abra, solicite esse code review para um pull request e para ir para o pipeline.

P. Esse code review é feito pela própria ferramenta?

E. Essa varredura sim, é feita pela própria ferramenta, depois que é feita a varredura da ferramenta aí é feito por outros dois desenvolvedores.

P. Então antes dele realizar o merge ou ir para o pipeline, é feita a revisão por dois desenvolvedores além da ferramenta?

E. Isso. A ferramenta faz automática e quando a ferramenta não encontra nada, passa por dois desenvolvedores que tem que dar o aval deles que está tudo certo de acordo com o que a gente considera correto.

P. Isso ocorre só na manutenção ou na criação do código de teste também?

E. A qualquer momento. Todo o código que vai, tanto manutenção quanto criação, ele tem que passar por esse processo. Nada vai para “dev” nem para “homolog” nem para produção se não passar por esse fluxo.

P. Para você, como seria possível prevenir/evitar a inserção de test smell no código de teste no momento que ele é criado?

E. Eu acho que é muito a relação à cultura, porque quando você está num ambiente onde há um cuidado com os testes, onde você tem esse cuidado de não gerar code smells, você acaba que quando você faz um, você nota. Você olha para um e percebe que está errado, que não é dessa forma, então você consegue visualizar de forma muito fácil que o teste está grande demais, tem complexidade demais ou então, porque deixar um teste vazio, porque fazer algo assim? Eu só quero cobrir porcentagem se eu não estou cobrindo nada? Essa é uma maneira que eu vejo.

A segunda maneira é a relação de code review. Code review é muito importante, pois no code review, se passa alguma coisa despercebida quando a gente está desenvolvendo no dia a dia, com certeza outra pessoa com uma visão mais ampla, uma visão fora da caixa consegue pegar isso de uma maneira mais fácil.

P. Você tem alguma pergunta para mim ou quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar nossa entrevista?

E. Não. Realmente achei a entrevista muito boa, as perguntas foram bem dentro do assunto. É bem dentro do que a gente vive no dia a dia mesmo. Achei que é bem legal. Não tenho nada que poderia somar.