Pesquisador. Gravação iniciada. Mais uma vez muito obrigado por ter aceito participar da entrevista. Ela tem como objetivo compreender os processos de criação, manutenção e verificação da qualidade do código de teste de unidade. Não há respostas certas ou erradas ou desejáveis ​​ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa.

Entrevistado. Uhum

P. Reforçando um item do termo de consentimento, nossa conversa 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 e que no relatório final conterá os comentários dos participantes sem nenhuma referência aos indivíduos.

E. Certo

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

E. antes de eu entrar na empresa O, eu participei de um evento da comunidade de Android, a Android Dev Conference, e uma das coisas que o pessoal falava muito era sobre testar, sobre arquitetura e daí eu comecei a estudar e comecei a escrever testes, isso tem 5, 6 anos. Antes disso eu já tinha escrito teste na universidade, já tinha pego algum software open source de alguma disciplina, que a gente tinha que escrever testes unitários, testes instrumentados, testes de aceitação, testes end to end e também fazer um CI (continuos integration). Eu já conhecia, só não entendia o devido valor que um teste podia trazer. Conhecia na universidade mesmo, mas o devido valor foi quando eu comecei a trabalhar profissionalmente, que tem volta de 5 anos já, onde alguns bugs eram resolvidos, eles voltavam e aí o teste com a escrita dos testes eu tinha uma segurança de que tipo ele não vai ter uma regressão do que aconteceu, voltar com o problema dado que com o tempo a gente esquece o que a gente implementou.

P. Quais foram suas fontes de aprendizado sobre código de teste, além da graduação?

E. Na graduação eu não lembro do livro que eu li, era um básico de engenharia de software, eram literaturas tradicionais de engenharia de software para graduação.. Hoje em dia, quando eu fui profissionalmente aplicar, eu lia muito esses tutoriais de Medium, muito sample de alguma arquitetura ou de um repositório no github, algo nesse caminho, e aí com o tempo eu consegui fazer uma associação de um pouco da teoria que eu tinha aprendido em engenharia de software, tipo teste de caixa preta, tenho que testar o caminho para entender se um fluxo funcionou do jeito que estava pensado, um teste onde eu simplesmente checo o resultado e isso está ok para mim. Então foi nesse mix, primeiro na literatura que eu aprendi na universidade e em combo com tutoriais da internet mais específicos pela plataforma que eu aderi ao trabalho, que enquanto desenvolvedor eu desenvolvo para Android especificamente, então eram como "ah como eu vou fazer uma asserção no Android usando JUnit ou Roboeletric da vida, ou Expresso da vida" era mais nesse sentido, mas a base foi da universidade mesmo. Os tutoriais eram mais voltados a ferramenta que a teoria em si.

P. Em quais linguagens você cria testes?

E. Hoje Kotlin, antes Java. Não me recordo de ter feito testes em Phyton e não me recordo de ter feito testes em PHP que foram linguagens que eu já trabalhei.

P. Qual linguagem do seu projeto atual?

E. Kotlin

P. Como é seu processo de criação de teste de unidade?

E. Hoje eu não faço o teste antes da escrita da funcionalidade, no meu dia a dia, não consigo, já tentei. Então como hoje eu tenho uma arquitetura que está estável a 2 anos e meio, eu meio que monto meu bloco de código e escrevo testes que tenham certa semelhança com testes que eu já escrevia antes, dado que o código é um pouco mais do mesmo do meu dia a dia. Eu escrevo primeiro os testes do fluxo feliz, depois os casos de teste em que eu vou ter exceptions e por último, casos bizarros que são mapeados dentro das minhas aplicações para cenários de erro direto e sem mutação para o lado do usuário. Primeiro escrevo o código e depois o teste.

P. Você cria o teste depois baseado no código do software ou baseado nos requisitos?

E. Eu faço o teste a partir do fluxo de dados que eu recebo, veja se uma requisição para um end point X, Y, Z da vida se eu conseguir fazer o mapeamento correto, se eu consegui “desserializar” a informação, se aquilo ali gerou um estado esperado de acordo com um mínimo possível de texto que eu vou ter relacionado a testes. Não tem tantas definições, mas no final a gente tem estados que são levantados e que meio que viram nosso diagrama de requisitos da vida.

P. Não é necessariamente no código, mas é focado num diagrama de fluxo de dados ou no código do que no requisito em si.

E. Exatamente, por exemplo, eu trabalho com aplicações bancárias, então o usuário quer fazer portabilidade, então eu sei que dado uma informação X, eu tenho que emitir um estado Y, então dado a informação X se eu emitir um estado Y então o código se comportou como esperado. Não sei se o requisito chega dessa forma para mim ou se talvez estou tão acostumado que o requisito já chegue e eu nem percebo.

P. Existe algum fluxograma, template documentado que padronize esse processo de criação do código de teste?

E. A gente no dia zero do projeto a gente cria uma DSL em que a gente padronizou como oficial dentro do código. Você tem prerrogativas, você tem o teste e você tem o resultado. A gente sempre escreve desse jeito. Primeiro você monta o cenário de teste, faz alguns mocks se você acha necessário fazer alguns mocks, algumas definições para deixar claro o teste que você está escrevendo, por exemplo, antes de você simular uma chamada, você pode dizer "essa chamada precisa que o usuário não tenha cadastro. Então você força esse cenário. Você define primeiro o escopo do teste que você quer escrever, depois você faz de fato a chamada para o que você quer realizar e por último você coleta os resultados. É meio que esse o jeitão que acabou ficando a DSL. A gente tem três steps, o dado, o então que é quando executa o teste e por último a checagem do resultado. Eu costumo fazer “given invalid user, then perform some requests and collect the results”.

P. Seria essa estrutura no caso.

E. Given, then, assert.

P. O que seria DLS?

E. Linguagem de domínio específico (Domain Specific Language). Eu pego uma API de teste, nesse caso específico, e eu criei ao invés de utilizar o JUnit da vida eu tentei deixar mais humanizado o fluxo de teste. Deixar mais próximo da engenharia de software em si. Então esse Given, é o bloco onde eu por baixo dos panos faço alguns mocks, Then é quando de fato eu faço a requisição, faço algum cálculo, alguma coisa da vida, e o Assert é a parte por fim quando eu checo os resultados igual a qualquer teste qualquer API de teste realiza, só que de uma forma que fique bem clara com o que a gente vê no dia a dia de engenharia de software. E que você tenta tornar a programação um pouco mais digitando em português ou em inglês, por exemplo.

P. Isso fica disponível para todos os membros da equipe e fica padronizado durante aquele projeto?

E. Isso, exatamente.

P. Isso muda de acordo com o projeto, esse acordo de equipe ou acaba, por mais que não seja um processo organizacional, acaba se tornando dentro da cultura? Vocês sempre acabam seguindo esse modelo?

E. A gente culturalmente segue esse modelo, porque a gente meio que impôs no princípio do start do projeto. A gente fez essa definição e quem entra depois tem a documentação de tipo "olha, fazemos testes dessa forma, utilizamos isso daqui". Mas se alguém quiser escrever teste de outra forma, tudo bem. Sem problema. A gente não impõe, não vou impedir. O cara chegou e escreveu 100 testes da forma que ele achou bacana. Não vejo problema em adicionar. Afinal quanto mais testes melhor e é o que importa no final das contas. Eu vou te mandar o link da lib para você verificar a forma que a gente faz as coisas. Você pode dar uma checada de como seria. Isso foi criado aqui dentro da empresa, pelo antigo tech leader que veio antes de mim e aí tem como fazer. Se você ver o overview, fica claro de certa forma o que está sendo feito. O que a gente busca com essa DSL? Que alguém na posição de QA ou alguém na posição de business consiga pegar um teste desse e entender o que a gente está validando. É essa a ideia. É tanto que se você ver aí onde tem a sessão dentro do overview onde tem "instead of this" que é algo como acontece no dia a dia e a forma acima disso é tipo a forma como a gente conseguiu deixar mais claro suficiente para um QA consiga ler um teste como se fosse simplesmente um texto em inglês dentro do que é possível também.

P. Quais ferramentas são utilizadas para criar/ executar teste?

E. eu vou falar especificamente como “dev” android, a gente usa para teste unitário o JUnit, Mockito, algumas APIs do Kotlin sobre o Mockito tipo, Mockkey, para testes que envolvem tela, mas são testes unitários a gente usa o Roboletric, em certa medida dá para fazer teste unitário utilizando o Expresso, mas aí já são teste de tela onde eu valido. Eu posso fazer teste unitário de uma tela onde eu valido cliques, ao clicar se trocou uma cor, um componente assim.

P. Como é o processo de verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, existe alguém que revisa esse teste depois de criado, compatibilidade com o requisito, assim por diante?

E. Hoje dentro dos times, cada um tem um processo, tem uns mais rígidos ao depender do produto que está sendo feito, tem times menos rígidos ao depender da velocidade que precisa ganhar para entregar uma funcionalidade porque não é simplesmente você testar você tem que agregar valor ao produto primeiro e justificar a sobrevivência daquilo que você está fazendo. Então dentro das equipes hoje eu tenho times que estão engatinhando no fluxo de teste, começando a escrever agora em prol de uma velocidade maior, temos equipes que ja escrevem testes a muito mais tempo e é obrigatório e tem time que ainda não escreve teste e não tem problema, dado que a gente tem controles de qualidade antes de chegar no usuário que envolve um trabalho feito por um QA. O código é revisado, todo o código que a gente produz é revisado, independente de ser teste ou não e aí os desenvolvedores da equipe tem a oportunidade de estar checando via pull request do github. Além disso, uma vez que a equipe possua testes, e essa é a regra de uma forma geral, você joga num continuos integration da vida onde ele sempre vai rodar os testes antes de você gerar um deploy daquilo que foi produzido. De forma automática você sempre vai passar por um teste, você não precisa que um humano verifique aquilo. Se um teste quebrar, simplesmente não há deploy porque algo deu errado. É assim que funciona o fluxo. e antes de executar o CI, você tem o review de código natural de fluxo de trabalho diário.

P. quantos desenvolvedores revisam um código submetido no pull request?

E. Pelo menos 2.

P. Passando para o processo de manutenção do código de teste, como ele é realizado dentro da equipe?

E. Hoje acontece muito da gente ter uma funcionalidade que está estável e a gente precisa atualiza-la. Adicionar alguma coisa nova ou algo do tipo, então é muito comum que o teste quebre. Dado os novos cenários. Então a galera tem que atualizar os testes para os novos cenários ou escrever novos testes uma vez que a funcionalidade está sendo atualizada.

P. O que você conhece sobre test smell?

E. É o teste, é o que a gente chama de flaky tests, se eu não me engano. Onde o teste pode passar ou não. Não se tem uma garantia de nada daquilo que está acontecendo. Já tive problema com esse tipo de teste em que ele passava na minha máquina e quando passava pelo CI ele não passava de jeito nenhum, nunca entendi. Então reescrevia o código até que ele passasse em todos os locais. Não sabia direito se o problema estava no meu código, no teste, mas de todo modo eu reescrevi tentando garantir que passasse. Test smell não me dá uma garantia de nada.

P. Como você conheceu este assunto?

E. Não lembro. Possivelmente lendo algum tutorial enquanto eu estudava testes ou então em algum momento de raiva em que eu não conseguia entender porque um teste não estava passando e eu ia ler sobre. Muito provavelmente pode ser alguma coisa assim.

P. Você possui alguma dúvida sobre esse assunto?

E. Se eu respondi corretamente sobre o assunto, nesse caso não. Se for mesmo esse teste flaky não teria dúvida não.

P. Tem a ver com flaky tests, existe uma relação entre eles. Eu vou te passar o conceito sobre o assunto da literatura, caso você tenha alguma dúvida você me fala, a gente conversa porque as perguntas a seguir têm a ver com esse tema específico, sobre test smell, tudo bem?

E. Tudo bem.

P. “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.” Então tem test smell relacionado a asserts, relacionado a construtor, a própria complexidade do código em si. Se você me permitir eu posso compartilhar 2 exemplos de test smells, explico porque eles são considerados smells e aí depois a gente continua. Pode ser?

E. Pode ser.

P. Esse é o chamado Conditional Test Logic smell. Ele é considerado um smell porque ele tem uma estrutura de repetição, condicional dentro dele. Então além de dificultar a compreensão dependendo da experiência do programador, uma vez que esse teste falhe é mais difícil identificar em que momento esse teste falhou devido a sua estrutura. O outro exemplo de smell é o chamado Empty test smell. Ele é considerado um smell porque não tem nenhum script executável. Então ele pode passar e não vai estar sendo feito nada e não ter a informação de porque ele está comentado ao invés de estar sendo executado ou porque ele não foi excluído e está apenas comentado. Isso interfere também na sustentabilidade da atividade de teste. Mas com eu comentei. Tem a ver com asserts, construtores, independência, ou seja, testes que estão relacionado com o mesmo recurso e dependendo da ordem que eles forem executados ele vai falhar ou passar por conta da dependência de recurso que eles estão compartilhando. Isso é considerado um smell porque impacta na sustentabilidade da atividade de teste.

E. Certo.

P. Tem alguma pergunta, alguma dúvida sobre?

E. Não.

P. Vou para a próxima pergunta então. Como test smells são tratados no processo de criação do código de teste?

E. Eu acho que a gente não tem esse problema hoje, dessa dificuldade, dado que a gente tem um padrão de escrita. Por a gente ter educado o pessoal de como a gente escreve, do que precisa fazer de asserção no final, acaba sendo um pouco mais do mesmo diariamente. Um teste se torna inútil, ou que só me atrapalha para manter, só me dá dor de cabeça e não prova um devido valor quando o código é muito alterado ou simplificado. E aí eu mantenho um teste complexo que não está validando um cenário real. Então ontem eu fiz um teste que não precisei escrever teste, eu precisei deletar umas coisas que eram testadas, e o pessoal cobrou. "precisa escrever testes" e eu disse "não, já estava escrito. Só estou simplificando o que está escrito" . Então seria nessa vertente.

P. Você comentou sobre a pergunta que eu ia te fazer que era sobre o tratamento de smells na manutenção do código de teste. É basicamente o que você comentou ou tem mais algum tipo de cuidado que você tem em relação a escrita de um teste quando vocês estão realizando a manutenção dele?

E. Não. Acho que não tem nada extra. Dada a base que a gente já criou.

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

E. Isso seria, eu estou até lendo aqui os possíveis tipos de test smells, seria muito eu ter que explicar mais sobre testes para o pessoal. Tem diferentes níveis de desenvolvedores e a quantidade de testes vem crescendo, mas seria mais no sentido de educar os desenvolvedores que tenho na equipe, talvez isso poderia surtir algum tipo de efeito.

P. Estou chegando ao final da entrevista. Você tem alguma pergunta para mim, quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar a entrevista?

E. A entrevista é muito boa. Está bem bacana. Eu fico muito feliz de ver que principalmente pesquisas indo pelo caminho de engenharia de software de pesquisam que façam sentido, nossa área. Eu não fiz mestrado porque não fiz uma coisa legal para estudar e eu vi que você achou uma coisa legal para estudar. Eu fico muito feliz pelo que você está fazendo, de coração.

P. Obrigado.

E. Eu só acho que mandaria, para ficar mais prático para você eu só mandaria o termo, o formulário antes. Nada de muito útil, a dica.

P. Ajuda realmente, mas eu tive caso de pessoas que eu fiz a marcação e na hora da entrevista não apareceram, cheguei a remarcar algumas vezes e não surtiu muito efeito, então eu acabaria tendo uma informação no formulário, mas não teria a entrevista, então acaba sendo uma informação perdida nesse processo. Então por isso que estou fazendo logo, momentos antes de começar para garantir que o dado que eu vou ter que tratar depois é o dado que eu já tenho também de entrevista, mas muito obrigado pelas informações, ela me ajudou bastante nessa pesquisa.

E. Tem algumas coisas que podem ser úteis para você. Eu não sei se exatamente para a sua pesquisa ou se para você enquanto profissional, mas por exemplo... eu gostava muito de escrever testes, eu hoje escrevo testes unitários e já tive alguns debates sobre a não obrigatoriedade sobre escrita de testes. Hoje eu gosto de escrever testes que de fato façam a diferença na minha vida. Por exemplo se eu tenho uma função de soma, eu jamais vou escrever um teste de uma função de soma, por razões óbvias do tipo que eu estou dizendo que tenho sei lá 2500 testes e 500 deles está verificando esse tipo de coisa inútil. Hoje para mim o teste unitário, principalmente tem um valor absurdo em coisas que são complexas, agora no meu dia a dia, sei lá, estou montando uma lista, checar se os elementos da lista de algum modo não foram listados como aconteceu é coisa que nunca vou fazer pois não vá me trazer valor. Agora testar um estado por exemplo, da portabilidade, isso traz muito valor. O que vou falar pode ser um pouco polêmico, mas o que eu sinto também é que a gente sai muito focado na escrita de testes e as vezes essa qualidade não é importante dado um momento do produto. É bizarro falar isso? É bizarro, mas as pessoas podem testar, garantir que está tudo ok no seu software, ele funcionar perfeitamente e você não ter um único teste unitário, mas você consegue fazer com que seu produto sobreviva e consegue nas trincheiras do dia a dia do empreendedorismo, de ganhar mercado, de trabalhar e sobreviver, você consegue fazer com que seu produto gere um valor. Uma vez que eu gerei um valor no mercado a gente consegue trazer para o nosso dia a dia, questões mais legais de engenharia de software como por exemplo: agora é obrigatório escrever pelo menos 10 testes em um código que você está tentando enviar para a produção. Um avançar do produto com a robustez que o produto vai adquirindo a escrita de teste vai ganhando cada vez mais um valor importante e eu já tive diversas fases dentro dos projetos que trabalhei, do tipo começar escrevendo um monte de testes e no final das contas ter que deletar o projeto, o projeto morrer. Assim como eu já não escrevi testes e o projeto virar um sucesso. Hoje, o teste não é a garantia do sucesso, mas ele é a garantia da qualidade quando você tem um produto que sobreviveu. Você começa o trabalho com um desenvolvedor, depois você coloca dois desenvolvedores, depois você tem 20 pessoas desenvolvendo e você precisa melhorar as práticas do seu dia a dia, então junto com a escalabilidade, do meu ponto de vista, você entra com os testes. Eu acho que vale a pena, se você tiver um espaço para colocar isso que, não é que a indústria não queira escrever testes, mas é que a gente também precisa sobreviver diariamente. Todo mundo gosta muito de testes, testes e arquitetura é o que todo mundo mais gosta de conversar, e conversa, escreve tutorial, mas ali no dia a dia, nem sempre é a realidade. Eu já pensei muito sobre, hoje as equipes estão crescendo e elas mesmo estão evoluindo os processos, eu enquanto responsável por algumas equipes eu olho que eu não preciso interferir na questão de qualidade porque a própria equipe percebe que "cara, somos 8 agora, precisamos melhorar isso aqui" daqui a pouco no total são 30 comigo, tipo... os times estão evoluindo. Acho que vale a pena considerar e óbvio que com essa evolução, tests smells vão aparecer.

P. É um relato bacana. Não entra muito no foco da minha pesquisa, pois ela entra no foco da qualidade do código de teste de unidade, mas de alguma forma a própria importância da atividade de teste ou necessidade da atividade de teste acaba sendo um fator que pode interferir nisso sim. Então ela não entra especificamente na minha pesquisa, mas é o um ponto interessante que pode gerar o tema de uma outra pesquisa de um outro membro do grupo ou minha mesmo num doutorado, quem sabe. Esse relato é bem bacana, bem interessante. Muito obrigado.