Entrevista 04 - Entrevistado

Eu atualmente estou com um Senior Software Engineer. Basicamente, customização, então, trabalho com um software que já existe, que precisa adicionar novas features e tudo mais, ou realmente novas funcionalidades do zero para uma empresa que trabalha bastante com logística, basicamente, hoje em dia. Atualmente, na mesma empresa, estou com dois, que são duas iniciativas separadas, times separados, mas, nesse final de ano, a gente está tentando correr contra o tempo para ver se consegue lançar ainda no final do ano.

Sim, tem um que está com uma equipe até razoavelmente grande, acho que tem umas 18 pessoas. Dessas, acho que pelo menos uns 12 desenvolvedores juntos. E outra, a gente só está fazendo algumas melhorias, porque já foi lançado alguns meses atrás, a gente está acompanhando em produção e fazendo alguns bug fixes e melhorias para o que já está andando.

Boa. A empresa para quem eu trabalho, eles têm um processinho, que é o seguinte, a gente abre um pull request para poder colocar o código público para outros colegas verem, fazer um code review em cima daquele pull request. A gente usa o GitHub para isso.

No próprio GitHub, a gente já tem um processo de CI, que nesse processo a gente roda todos os testes automatizados e cria um ambiente de testes com todos os componentes, então, tem vários micro-serviços, tem um monolito e a gente levanta tudo junto num ambiente separado de testes. Esse PR estando positivo, todo mundo aprovou, passou todos os testes, um QA foi lá e no ambiente de testes conseguiu rodar um teste integrado, vários. A gente volta para o Kanban, que a gente usa o Jira, coloca essa atividade, esse card para revisão da equipe de produto, eles dando um ok, a gente faz o deploy.

Para fazer o deploy, basicamente, a gente tem uma filinha de espera para não ter várias pessoas, vários times diferentes fazendo deploy ao mesmo tempo para o mesmo componente. Então, a gente tem uma filinha no Slack e quando é sua vez, você entra na fila, quando chega na sua vez, você vai lá e faz o merge do PR. Fazendo isso, ele já gatilha o deploy contínuo e aí, se tudo der certo, os pods são rotacionados pelo Kubernetes mesmo e a nova versão entra em produção.

Diversas vezes, inclusive, eu fiz um freelance recentemente que foi a volta ao não processo. Basicamente, eu perguntei para o cliente como é que o pessoal fazia o deploy e ele disse, o deploy não é feito desde 2020. Então, faz aí.

Ok, tentei procurar se tinha algum tipo de estrutura, se tinha alguma coisa montada e não tinha. Consegui chegar no histórico de comunicação da empresa e vi que o deploy era feito basicamente no Heroku, fazendo um Heroku deploy. Então, literalmente, baixo

o repositório no meu projeto, faço a alteração que eu quero fazer, faço, no caso, fiz o commit direto na master e gatilhei o deploy do próprio Heroku.

Como assim? Ah, boa. Eu já trabalhei com, há um tempo atrás, com um GitFlow mais puro, usando feature branches, tendo a development e a master e as feature branches todas apontando para development. Eu já trabalhei com um GitFlow simplificado, que é basicamente, você elimina a development, é tudo na master, e o que muda, além disso, de você não ter a branch development, mas o próprio deploy, a diferença do deploy vai ser se vai para um ambiente de staging, um ambiente de homologação, ou se vai para um ambiente de produção, mas a branch é sempre a mesma, no caso do que está sendo feito o deploy, que é a master.

Hoje, no meu principal trabalho, que é esse que já tem toda a estrutura de CI e de CD, a gente não trabalha com a feature branch em si, a gente trabalha com poções menores de código, o ideal é que seja o menor tipo de alteração possível, e é sempre um merge para master, para main, mas a gente sempre tem, é obrigatório, essa etapa do PR, do code review em cima de um PR, o que é um pouco, não é exatamente, está longe de um GitFlow, mas também não é um trunk-based, porque a gente sempre tem essa, como eu posso dizer, a gente sempre tem uma branch separada, para poder fazer o PR, e a gente tem problemas constantes, no sentido de, como tem muitas equipes, muita gente trabalhando no mesmo projeto, no mesmo repositório, a gente está tendo que sempre atualizar com a main, todos os nossos PRs, então, é um negócio que toma um certo tempo, todos os dias, atualizar com a main, possivelmente resolver algum tipo de conflito, que apareça, porque esses PRs, eles levam um certo tempo, para sair do momento de code review, até realmente ser feito o deploy, dura dias, nesse caso, então, não é um trunk-based. Eu acho que ele está no meio do caminho, mas não é um trunk-based, especialmente por causa da demora, a branch main, ela vai muito mais rápida, do que essas branches de features, e aí você tem que estar o tempo todo, a ideia do trunk-based, até onde eu sei, ela é para ser commit rápidos, e você está sempre atualizando a main, e não é o caso, tem vezes que passa, uma semana, duas semanas, um commit que você fez, duas semanas atrás, para poder entrar na main hoje. Eu já participei, nessa mesma empresa, de um projeto, que eu participei, que ainda está rolando, mas enfim, basicamente, da metade do ano para novembro.

Como tinha menos pessoas trabalhando, então era eu, e mais um cara, constantemente, e tinha mais outros dois, que de vez em quando vinham, e faziam alguma alteração, então esses PRs, eles rodavam muito rápido, que aí se assemelhava no trunk-based, a gente só fazia o PR, só para fazer um checklist, e marcar, abrir um ambiente de teste rápido, então o PR durava um dia, assim, em geral. E aí, isso se assemelhava mais no trunk-based, e o principal benefício é, se alguém precisasse começar alguma coisa nova, um outro colega do trabalho, precisasse começar alguma coisa nova, era quase certeza que a main ia estar atualizada com o código mais recente. Em comparação, e para mim isso é maravilhoso, porque em comparação ao projeto que eu estou agora,

frequentemente eu tenho que ir lá no GitHub, ver todos os PRs que estão abertos, abrir todos eles, ver os arquivos que eles estão modificando, para ver se tem algum arquivo meu, algum arquivo que eu, assim, vai entrar, sei lá, essa semana, alguma modificação em algum arquivo que vai entrar nessa semana, que afeta o que eu estou fazendo agora.

Então, toda vez que eu começo uma atividade nova, eu não posso confiar na main, eu tenho que ir nos PRs para ver. É, porque, como existe um processo na empresa, dessa questão do Code Review, que, inclusive, uma das regras do Code Review é que pelo menos duas pessoas precisam aprovar, duas pessoas, idealmente, que, sei lá, se você estiver fazendo uma implementação em pair programming com outra pessoa, seria outra pessoa que não fez pair programming que tem que aprovar. Então, por existir esse requisito do processo da empresa em si, eu acho que isso dificulta muito usar o trunk-based, porque termina diminuindo a velocidade de tudo.

Não favorece o trunk-based. Outra coisa que acho que não favorece é como a gente tem um certo a gente tem microserviços, enfim, é uma arquitetura mista, mas existem microserviços. Quando você faz alguma alteração em um dos microserviços, a gente não tem forma fácil de validar, a não ser fazendo um teste integrado, tipo, ok, a gente escreve testes unitários, beleza, ali está validado.

Mas para quem tem uma área de QA, eles não tocam no código, eles não olham o código. Então, eles precisam ver alguma tela que apresente aquela alteração de comportamento. Para isso acontecer, a gente tem que levantar todos os microserviços e o monolito, isso leva tempo, então, leva uns 30 minutos para levantar esse ambiente de testes, e aí só assim eles conseguem validar aquela mudança específica.

Então, para toda mudança pontual, a gente faz um teste, a gente levanta os serviços inteiros para poder fazer um teste integrado, e aí isso dificulta também, tipo, não daria para fazer um trunk-based. É, acho que é mais ou menos isso. A gente tem testes automatizados unitários, e a prática que a gente faz é os testes, a gente faz mocks das chamadas externas.

Então, se eu estiver fazendo uma chamada para um microserviço separado, aquela camada ali, eu faço um mock para não fazer no momento de rodar os testes automatizados, não ter que ficar chamando o serviço externo. Essa é uma boa pergunta. Eu acho que em uma pequena escala dá para mudar, pelo menos na empresa atual que eu estou, que é o exemplo do projeto que eu tive anteriormente.

Como eu estava muito bem integrado com as outras pessoas que estavam fazendo aquele projeto, e a gente tinha feito um trabalho bom de planejamento, de pesquisa, antes para entender o que é que precisava ser feito, a gente estava muito bem alinhado, todo mundo que estava participando sabia o que precisava acontecer, quais eram as regras do negócio. Então, nesses pontos, nesses micro-teams, eu acho que dá para pular

algumas etapas do processo atual, e fazer o negócio rodar mais rápido, inclusive chegar a um ponto de fazer um trunk-based, que a gente fez, a gente só não fez 100% porque a gente não fez um microserviço separado, a gente fez um módulo novo, só que em um projeto já existente. Para essas alterações, a gente ainda precisava fazer o PR e tudo mais, mas era tipo, quando chegava no ponto de abrir o PR, a gente já sabia qual era o código novo, já sabia qual era a modificação, já viu que tinha rodado bem, porque a gente já estava rodando os testes automatizados na hora de fazer o PR Programming, então era só para marcar o checkbox.

Então, eu acho que sim, mas no projeto atual, que tem muita gente, de times que são diferentes, então gente que nunca tinha trabalhado junto, está todo mundo trabalhando no mesmo projeto agora, e frequentemente a gente encontra tarefas ou stories que elas modificam o mesmo pedaço do código, então a gente está tendo muito conflito de meios, porque talvez a abstração não tenha sido feita muito bem feita, não tenha sido bem abstraído, e aí frequentemente a gente está alterando o mesmo pedaço do código e alterando o código do colega, enfim, eu acho que o Git Flow não é de comum todo, talvez manter mais como está hoje, ou, pode ser que seja o caso de a gente fazer algumas feature branches para algumas coisas específicas. Tipo, partes da solução que é tudo uma coisa só e a gente só está adicionando funcionalidade àquela coisa, fazer uma feature branch para aquilo e poder fazer APRs para a feature branch e depois a gente bota tudo para master. Eles se consideram como startup ainda, mas eu acho que é muito mais para os investidores, a parte financeira, porque na prática é bem o processo em si ainda é bem o waterfall.

A pessoa do produto fala lá com os stakeholders, eles juntos lá, sem ninguém do desenvolvimento definem o que é para ser feito, aí entra o engineering manager que em reunião com a pessoa do produto, começa a tentar quebrar as atividades em tasks, stories, etc. E aí só desse ponto é que entra a equipe de desenvolvimento para pontuar as stories que também são uma mera formalidade e aí a equipe de desenvolvimento começa a ver ok, é isso que precisa ser feito, beleza, vamos lá, começa a fazer. Da equipe de desenvolvimento depois, code review, equipe de testes, que também não estava integrado com nenhuma parte do processo anteriormente, vai lá e pega o negócio e precisa falar com o pessoal do produto só naquele momento para ver esse comportamento aqui é esperado, então grava um vídeo e tal, manda para o pessoal do produto e aí é isso aí que a gente estava pensando waterfall completo.

Raramente a gente consegue e foi o que eu fiz no projeto anterior, entrar já no começo, então me coloca por dentro aí, eu vou assumir isso aqui na parte de desenvolvimento, beleza, mas me coloca por dentro de todas as reuniões, vai me colocando ali nas reuniões, eu vou começando a entender o negócio já do começo e aí quando a gente teve, começou o momento de desenvolvimento mesmo a gente já tinha só não tinha escrito o código, mas já tinha feito toda uma documentação por trás de a solução vai ser essa e aí fluiu muito melhor, mas só porque eu como desenvolvedor me enfiei lá no

processo do começo, né não era o padrão em relação à acho que bom um problema constante que eu não sei se entraria no no hall da entrevista é a questão de resolução de conflitos de enfim, quando tem duas pessoas trabalhando no mesmo pedaço de código, no mesmo módulo ou alguma coisa do tipo e a gente vai ter obviamente um conflito lá no vacinamento e tem muitas coisas diferentes, já lidei com várias pessoas que resolvem esses problemas de conflitos diferentes você faz um merge com a main, no caso eu vou fazer um merge da main para a branch ou você faz um rebase com a main e se você fizer um rebase, como é que você vai garantir que você não está adicionando nenhum código extra ou perdendo algum código existente né