Entrevista1- Entrevistado

Eu sou, na classificação, desenvolvedor sênior, especializado em back-end. Trabalho, por enquanto, só focado na cláusula IDA-AWS, para aplicações financeiras. Trabalho fixo para um único projeto, mas dividido em vários contextos.

Os projetos não têm começo e nem fim, são contínuos. É um projeto só, mas dentro dele deve ter 4 ou 5 contextos diferentes. São bem diferentes.

Nesse contexto principal, são 5 desenvolvedores. E existe um outro secundário, que são 3. Eu vou desconsiderar toda a parte de levantamento do problema. Seja para uma nova feature, seja para você corrigir um bug.

A primeira coisa que tem que ser feita, depois que a atividade cai no nosso fluxo de trabalho, é que o desenvolvedor escolhe ela, faz uma análise rápida, e pode ser para um grupo de um ou dois desenvolvedores, para ele comprovar qual seria a ideia para eliminar o problema. A partir daí, é código mesmo, local, sem nenhum problema. Se é necessário adicionar algum teste unitário, é adicionado.

Se precisa mudar algum teste que já existe, ele é mudado. E se for um caso novo de uso, ele vai ser criado um teste end-to-end, para ele, em uma ferramenta de teste separada nossa. E é feita a primeira subida para um ambiente de desenvolvimento, que é basicamente, para o nosso fluxo, você criar um PR para uma Bridge Developer.

Esse ambiente de develop, ele só serve, basicamente, só para testar como a aplicação está rodando, com relação à sua infraestrutura, porque a gente pode fazer mudanças de infra. É um teste bem preliminado, é um teste de regra de negócio. Quando esse cara é validado, a gente vê que a aplicação não caiu, nenhuma dependência que localmente funcionava, que não está funcionando dentro do nosso ambiente da cloud de desenvolvimento.

É um PR aberto para uma Bridge que a gente chama de release ou homologação. A partir disso, ele cai nesse âmbito de homologação, por causa da natureza do negócio que a gente trabalha. A gente trabalha com um grupo de TA, um grupo de TA separado do time.

Esses caras precisam validar se está tudo funcionando, eles entendem o contexto do que a gente trabalha. E só depois que eles validam, é que a gente pega esse PR que a gente fez para essa release, e a gente prepara para fazer um MR, literalmente abrir um MR para a produção. E aí tem toda a questão burocrática.

Não é rápido e não necessariamente... Ele é feito basicamente instantaneamente. Depois que a atividade é feita, ela pode ser estocada por um ou dois outros features para que a subida seja feita de uma vez só. E aí é basicamente o fluxo.

Essa é uma pergunta interessante. No começo do projeto, o nosso contexto, como era basicamente uma subsidiária, no caso trabalho com a parte financeira, mas é uma subsidiária, essa empresa trabalhou muito como se fosse uma startup. Então, a nossa pipeline de desenvolvimento era bem diferente.

Basicamente, não existia diferença desses ambientes de homologação e débito. E tudo era feito através, literalmente, de PRs. Se eu precisasse criar um ambiente novo, um PR seria criado, e aí uma instância desse PR viraria uma aplicação nova para ser testada.

Os testes eram feitos em poucas horas e, nesse mesmo tempo, a gente poderia fazer o MR para a produção. Não seria necessário uma estocagem de código nem nada associado. Com o tempo, essa forma de trabalho mais instantânea de você criar um PR e esse PR ser criado em uma aplicação para ser testada rapidamente e depois mediado em produção, ele acabou mudando por várias questões.

Uma delas é que a empresa teve que amadurecer e entrar em alguns contextos financeiros que exigem, necessariamente, subidas para a produção, eles passam por certos tipos de homologação, eles passam por análise de PR. Então, esse formato mais dinâmico que a gente tinha acabou sendo, retrocedendo para um modelo mais tradicional de ambientes separados. Porque essas equipes que fazem análises e fazem a checagem das subidas precisam que esses ambientes estejam rodando para ver o que está acontecendo.

O primeiro se baseava mais em uma trunk de base, porque, basicamente, cada MR se tornava literalmente uma aplicação para ser testada. Depois, tudo era margeado na branch principal. Essa concepção de ambientes separados não era tão clara lá e a própria ideia de que todas as nossas alterações eram sempre uma cópia da master.

Sempre que eu precisava fazer uma atividade, a cópia do código que eu precisava para fazer era em cima da master e somente nela. A partir do momento que a gente precisou fazer essa checagem de ambiente, a gente precisou criar ambientes separados. Então, a esteira ficou um pouco mais tradicional, com um stage, com um ambiente baseado em estágios.

O trunk de base tinha uma vantagem muito clara, que era, primeiro, a simplicidade da infra. Boa, então, posso falar disso. Também tem outra característica boa, que seria a agilidade dos processos.

Essa vantagem era mais clara no momento em que a gente tinha algum problema em produção, um incidente, ou dependendo da área que você fala, vai acabar mudando. Quando esse tipo de evento era alarmado, a equipe de desenvolvimento era acionada, a correção, a localização e correção do código era muito mais rápida. Até porque o acesso dos desenvolvedores para fazer realmente a subida, para tornar aquela feature real, porque ela só se torna real quando ela cai em produção.

Era muito mais rápida. Se você fosse um pouco mais conservador, menos de meio dia a correção já estava em produção. Então, essa velocidade era realmente a maior força e essa celeridade era o que fez mais diferença, principalmente porque o projeto estava no começo, não precisava de muita... As alterações eram muito mais furiosas, o tempo todo sendo... Era uma equipe que também não tinha experiência nessa área financeira, então sempre precisavam de ter features o tempo todo sendo acionados.

Ou também não ter experiência, bugs poderiam acontecer, então a correção precisava ser a mais celera possível. Essa era a maior vantagem. A partir do momento que a empresa precisou entrar mais nos ritos burocráticos, essa mentalidade de celeridade acabou sendo substituída pela ideia de que o rito é mais importante.

Você precisava alcançar todas as fases do processo para que o problema seja resolvido. E o problema vai ser resolvido, só que não precisa ser tão rápido. Primeiro, o trunk-bending não é muito útil em um ambiente que você precisa passar por uma cadeia de aprovações e avaliações.

Principalmente quando... Por exemplo, no nosso contexto, quando uma feature ou bug era corrigido e a gente criava um PR, a avaliação não era feita somente a nível da aplicação. Ela precisava ser colocada dentro de um ambiente inteiro de homologação e aí depois o fluxo teoricamente de ponta a ponta, porque a gente era só uma parte do processo, não o fluxo todo do processo, era chamado para ver se a operação funcionava de maneira correta. Era como se fosse um teste end-to-end real, porque teoricamente quando a gente fala end-to-end para teste de aplicações, é sempre ponta a ponta dentro do mesmo serviço e não do negócio inteiro.

Então, era necessário que houvesse literalmente uma cópia do ambiente inteiro rodando para que a gente pudesse literalmente fazer um teste sem ser mocado daquele fluxo de negócio estar rodando. Então, esse acabou sendo o maior detrator na época para que esse modelo Trunk Based fosse cair no fogo. E porque também como não tinha todo esse problema, não, que não é um problema, se você for parar a pensar, é só uma questão de rito burocrático, essa celeridade de você ter o código rapidamente e ele ser megeado acabou não fazendo mais tanto sentido, porque você tem que passar por processos.

Nesse caso, a maior facilidade para a gente foi essa parte do controle dos estados, porque como eu acabei de dizer, qualquer subida ela precisa passar por uma série de contextos burocráticos para serem subidas. Então, principalmente os ambientes de homologação, eles são os ambientes que a gente teoricamente, no meu time, tem menos controle, porque eles estarão tendo uma cópia toda do ambiente de produção e quem tem controle sobre eles são outras equipes especializadas para essa área. Então, a gente nem se preocupa muito com esse cara, diferente de quando a gente tinha o Trunk Based, só que quando precisava fazer o equivalente ao que seria um teste de

homologação, uma release de homologação, toda essa parte de infra ficava na nossa mão.

De analisar se o pote está funcionando direitinho, se está... A gente mocava as integrações com outras aplicações, então, menos preocupações tinha com relação a isso, mas... Como é que eu posso dizer? Eu acho que o controle é a melhor... O controle dos estados, eu acho que... Que eu acho que também não foi tanta diferença, foi a maior vantagem. Quais que não favorecem? Eu digo que toda a parte burocrática, principalmente com que eu trabalho hoje, ela foi moldada pensando que existem esses ambientes, sabe? Então, eu não diria que seria uma desvantagem, eu diria que todo o rito, toda a parte burocrática, ela foi feita pensada para um Branch Based, sabe? Não existe uma desvantagem. A maior desvantagem para mim, do ponto de vista de alguém que quer trazer mais correções, features, é que o meu time perdeu essa celeridade, esse acesso rápido a fazer subidas constantes para homologação, tem esse acesso, né? Agora é meio que existe um gatekeeper na frente, né? Que faz toda a checagem para dizer, não, agora o seu time, sua feature, ela realmente paga para que suba para homologação.

Essa é uma pergunta complicada. Se eu mudasse e voltasse para um modelo mais dinâmico, mais celere, eu ainda teria que passar por todo o componente não técnico, né? Que é o que eu estou resumindo como burocrático, né? Da área do desenvolvimento, da parte burocrática. Então, a mudança, do ponto de vista técnico, não teria muito ganho, sabe? Porque não é uma questão somente de você, vamos dizer assim, ah, meu time se sente mais confortável com isso.

Nem todas as áreas de desenvolvimento, elas têm essas, vamos dizer assim, esse privilégio, se eu posso chamar de um privilégio, né? Você poder escolher o seu modelo de trabalho, né? Dependendo da área, financeiro e saúde, existem, que são os exemplos mais fortes, né? Acho que tem a parte de aeronáutica também. Elas passam por baterias de avaliações, certificações e modelos que fazem com que esse modelo de tipo, sair rapidamente da sua máquina para a produção seja menos importante, né? É só a menor parte em todo o processo para fazer a subida. Acredito que hoje, no meu desenvolvimento, a parte que menos dura, vamos dizer assim, é o próprio desenvolvimento.

Todo o resto é maior do que ele, entendeu? Então, só dizer assim, ah, eu quero mudar porque meu time gosta, mas não ia ser muito útil, porque a gente teria que ainda lidar com todos os ritos de certificação e parte de que é apropriada, de subida, de avaliação, de passar por cabos, de grupos de aprovação. Então, não teria muito ganho. Eu acho que vale só segmentar por área, né? Porque, por exemplo, startups que vão fazer produtos para clientes, pessoa física, né? Eles têm muito menos, eles precisam de muito mais velocidade para fazer desenvolvimento do que, por exemplo, alguma área super bem estabelecida, como é que eu dou exemplo agora, de, sei lá, saúdes ou aeronáutica,

assim.

Então, eu acho que o setor em que a pessoa trabalha também afeta muito o modelo de trabalho dela. Ok, agradeço.