Entrevista2 - Entrevistado

Atualmente sou desenvolvedor líder, gerenciando 26 sistemas de sustentação. Um resumo, o cliente tem um conjunto de ferramentas que são feitas através de outros fornecedores, então a gente está criando um centralizador, nada mais que o hub, que conecta essas ferramentas gerando uma versão atual. O cliente teve recentemente um problema de invasão, teve um ransomware na rede, teve dados corrompidos, vendíveis.

Então a estratégia agora são sistemas diferentes, a gente tem 26 de sustentação e mais outros 10 em desenvolvimento, e cada um deles com um propósito único, que são conectados a esse hub. Então o usuário ali dentro consegue fazer tudo automatizado, não usando mais planilha, não precisando nenhuma ferramenta externa ou dependendo de outras ações. Tudo automatizado, conectando tudo.

Isso, manter todas as aplicações funcionando e conectando com esse hub. O hub é desenvolvido por uma outra equipe, só para dar um contexto um pouco melhor. O sistema é dividido em três partes basicamente, a gente tem uma empresa que é proprietária do banco de dados e cuida do banco, toda a parte de DBA e banco é deles.

Tem a outra empresa que é responsável por esse hub, e a minha empresa no caso é responsável simplesmente pela manutenção e criação desses softwares automatizados, retirando aquele processo manual, entregando. Aí tem uma variação muito grande, a gente tem software ali, por exemplo, de meios de pagamento, como um pagar e tudo mais. Então tem uma pouca da variação, cada software é único, cada um tem um único propósito, mas todos eles funcionam em conjunto, por assim dizer.

Então, a minha equipe é dividida em duas partes, a gente tem um time de desenvolvimento puro que conta com três programadores e três desenvolvedores, e eu tenho uma outra equipe de evolução, onde pequenas evoluções que não são um hotfix, são apresentadas e essa equipe cuida de lá e tem em média seis desenvolvedores. São desenvolvedores comigo e tenho um time de desenvolvimento com dez. Isso vou ter uma certa variação, depende de qual time eu estou usando, por exemplo.

Vamos lá, a gente trabalha com a sustentação, as 26 aplicações lá que estão funcionando, é apresentado um bug, por exemplo, então a gente usa ali o trunk-based, por quê? Porque a gente tem uma única, a gente puxa dele lá a ramificação, trabalhamos ali na hotfix, uma vez que foi desenvolvido, é homologado pela própria branch de hotfix, é criado uma branch de release e é publicado, mas voltando diretamente para sempre utilizando a main, então a gente sempre está puxando ali na sustentação. Já no time de evolução, eles já contam com a branch de desenvolvimento, eles já utilizam o GitFlow, então temos ali desenvolvimento, temos homologação, temos pré-produção e temos a branch main lá, que é a branch de release em si. Então temos esses dois processos.

Isso, vamos lá, se a gente tem um sistema em desenvolvimento hoje, ele vai usar o GitFlow, se a gente tem um sistema que está em sustentação, ele vai usar o trunk-based, se é uma evolução complexa, vai ser feito com o GitFlow, se é uma evolução pequena, curta, vai ser feito pelo trunk, depende muito de qual time vai pegar essa evolução, não apenas hotfix, a sustentação cuida também, o time de evolução cuida de evoluções maiores, que demandam mais do que duas ou três sprints. Vai ser feito uma única sprint, a própria sustentação pode fazer, usando o trunk-based mesmo. É porque assim, no caso da sustentação, a complexidade é menor, então é muito melhor utilizar a trunk-based, baseado que a gente tem uma única branch, é simples, o processo torna-se muito suave, é menos propenso a erros, e é muito mais fácil estar em sincronia, até porque o time é menor, um time pequeno fica fácil de gerenciar dessa forma.

Também tem o ponto da integração contínua, a gente pode ter vários bugs na mesma aplicação, cada um pode trabalhar e todos eles estão atualizados, porque estão trabalhando com uma única base, uma única trunk. Então isso torna prático. Agora, quando estamos em evolução, a gente optou por ir para o GitFlow, porque fica mais fácil a gente quebrar, a gente pode ter algum sprint que fica em estoque, a gente pode ter algum sprint que precisa subir, então a gente quebrar ela melhor, a variação melhor, já que evolução tem uma complexidade maior no seu desenvolvimento como um todo, na própria gestão da sprint, o GitFlow encaixa melhor.

Geralmente a própria publicação, a gente não tem um processo muito estruturado, por mais que a gente possa estruturar toda uma organização, a gente tem um fluxo interno do processo de homologação, todas as etapas antes da publicação em si, ainda é um pouco, eu considero sujo, um pouco feio, toda a publicação, toda a nova release. Eu acho que é a parte que é pior. Ela é simples, ela é fácil de usar, mas na hora de você publicar, ela é meio que uma publicação contínua ali, então é meio difícil acompanhar.

Se a gente não tiver o cuidado de estar sempre atualizando as sprints ali, a gente acaba tendo alguns gaps, ou se alguém tentar, por exemplo, um cenário que fica mais simples de entender é assim, temos duas aplicações e os dois times estão trabalhando, tanto evolução quanto sustentação. Se a sustentação colocar na main, atrapalhou o pessoal da evolução, porque eles vão estar pegando um conteúdo que talvez não foi publicado ainda, então acaba gerando esse problema, por isso que eu acredito que a publicação é o principal ponto negativo da Trunk. TrunkBaser.

É, basicamente, a baixa complexidade dela, o ritmo dela, assim, a sensação que todos temos é que o ritmo é um ritmo de integração contínua. Chega uma coisa, chega outra coisa nova, entra uma coisa e outra, a gente consegue facilmente desenvolver e publicar. A gente consegue estar ali publicando sempre, a gente consegue estar entregando continuamente, então essa sensação é o que mais favorece ela, o ritmo dela.

Se o time não for um time muito disciplinado, você vai ter, de fato, assim, muitos problemas na hora de testar e na hora de integrar. Entra naquele lance do poluir a main, a master, a trunk lá, porque você está colocando conteúdo que não deveria estar lá naquele momento, então o time tem que ser atento ao momento de subir as coisas. Entra naquele ponto da release, né, sempre que você vai publicar a release fica um pouco confuso, porque alguém pode colocar uma coisa lá que não era para estar.

Então você, às vezes, tem que fazer uma serialização da branch, remover algum conteúdo, e como a gente tem um único tronco, se poluir esse único tronco, influencia todos os desenvolvedores envolvidos. Então a disciplina ali é importante e essa parte da release é o gargalo dela. Todo mundo trabalhando bem feito é um ponto positivo, mas se tem alguma pessoa nova a gente acaba atrasando ou tendo algum impacto ali na velocidade.

Sim, é uma empresa estruturada, já num ponto de... já é uma empresa antiga, não é mais uma startup, já tem um bom tempo de mercado e está expunindo agora para mercados internacionais, né, então... na categoria que ela tem uma certa estabilidade dentro do Brasil e está alcançando parcerias externas. Nesse momento não tem nenhuma pergunta, né? Não tem nada a acrescentar.