Entrevista 06 - Entrevistado

Hoje eu sou engenheiro de software pleno no CESA. Hoje eu trabalho com um pouco de cada coisa, basicamente com bastante automação de IoT, um pouco de cloud, barra DevOps, barra Continuous Integration, Continuous Delivery. Se for para resumir, seria um pouquinho de cada coisa.

Um, atualmente um. Na prática, desenvolvedores atualmente são três. E QA são quatro, são cinco.

Hoje a gente está nesse final de ano, passando por uma mudança, porque a gente percebeu que ao longo de 2023, tivemos excessivos problemas. Então, atualmente estamos em um período de transição, mas como era feito até 2003, a lógica até o código chegar em produção, tinha o backlog, puxava a task do backlog, a pessoa era assinada para fazer aquela task, ela fazia a task, submetia o pull request no Bitbucket, que é a ferramenta que a gente usa. E aí existia todas as pessoas envolvidas naquele contexto, eram adicionadas como revisor, a revisão era feita, e aí existia a métrica de que pelo menos duas pessoas tinham que ter dado ok.

E dessas duas pessoas, uma tem que ser, obrigatoriamente, uma das pessoas que a gente considera que é responsável por aquele contexto. Existem grupos de responsáveis, pelo menos uma dessas pessoas, dos grupos de responsáveis, tem que dar uma approval, considerando que esses dois. Acredito que essas são as regras iniciais que a gente tem.

Baseado em dois approvals, o código está pronto para ser mergeado, e ele vai direto para a branch de desenvolvimento. No final da sprint da branch de desenvolvimento, ele passa para a produção. Um pouco diferente mais em relação à regra de aprovação de pull request, mas o fluxo sempre foi similar.

Ele parece com o que seria um GitFlow adaptado, porque a gente tem a branch de desenvolvimento, eu não lembro muito, porque eu não lembro todos esses nomes, tem uns três, quatro, meio que padrão. Eu acho que o que a gente usa é muito similar, se eu não me engano, ao GitFlow, que é o mais padrão, com a branch de desenvolvimento e a de produção. A gente não tem a branch de homologação, a gente é basicamente a branch de desenvolvimento por sprint e produção.

Então eu chutaria, barra lembrado, pelo que eu lembro, o GitFlow. Não, não. Talvez eu saiba se tu falar, mas eu não sei, de cabeça sim.

Então como eu comentei, como a gente está em processo de mudança, como a nossa equipe desenvolve framework para automação de dispositivos IoT, é um contexto um pouco diferente do contexto de engenharia de software em geral. O software has a service, então a gente está reconsiderando que talvez o GitFlow não fosse a melhor

opção. Porque esse GitFlow foi herdado pelo fluxo de treinamento de Git que a equipe de desenvolvimento Android utilizava.

E aí meio que a gente herdou todo o processo e agora está refazendo, porque a gente percebeu que o nosso contexto é um pouco diferente. Então eu diria que o GitFlow para o nosso contexto não é a melhor opção, e é por isso que a gente está transicionando para o ano que vem. Ah, então acho que uma das principais coisas é que a gente envolve hardware, a IoT envolve hardware.

Então a gente tem, quando se trabalha com hardware, existem algumas nuances que é preciso entender. Que quando você trabalha com a request do cliente, ele não é necessariamente uma funcionalidade como uma webpage ali, que você consegue ter um desenvolvimento. Normalmente envolve um hardware, e hardware eles tendem a ter problemas de falhar, ou que vai além do que você pode.

Então você tem que começar a fazer a feature inicial, que era para se comunicar com o dispositivo IoT, muitas vezes muda completamente de contexto, porque todos os hardware que estão associados àquele IoT, que faz todo aquele sistema mudar, não suporta aquele novo hardware. Então aí você precisa alterar as coisas. Então é uma coisa que até eu, particularmente aí, estou também descobrindo, que existe talvez um fluxo Git para alguns contextos repositórios, quando se trabalha com hardware, seja, eu não sei exatamente qual, mas com certeza, essa é uma das coisas que mais me deixa, que eu não curto muito a ideia do GitFlow, porque sempre que tem algum problema e envolve hardware, o GitFlow, eu sinto que poderia ter alguma coisa ali que favorecesse atualizar, talvez um GitTag, atualização do Git que é Tag, ver algum fluxo que eles usem mais coisas como GitTag, para versionar estabilidades de hardware, aquele hardware nesse trecho de código que estava rodando legal naquela versão de firmware, por exemplo, algo nesse sentido.

Eu acho que eu mudaria sim o modelo atual, mas não porque eu acho que o modelo atual é 100% culpa do modelo atual, mas devido à conjuntura do meu time, porque o time hoje tem mais QAs do que engenharia de software, e os QAs atuam no código, na automação, não na parte efetiva do framework, mas no desenvolvimento dos casos e testes da automação, já que o framework... O nível de conhecimento de engenharia de software da galera de QA tende a ser menor por experiência do que os engenharia de software, então a gente sempre tem um gargalo no nosso processo Git no nosso time, que é por isso que mais uma vez já estamos nesse processo de tentar mudar para o ano que vem. Mas eu acho que o que faria mais eu mudar seria mais pela imaturidade da equipe do que por um problema do framework e do GitFlow, porque ele é bem adaptável, então daria para talvez só reajustar usando mais algumas coisas pré-estabelecidas no mercado. Recorrente seria o que? O que você gostaria de ser recorrente? A gente teve um período que a gente teve muito problema, e aí a gente se atentou bastante a isso, principalmente porque a gente usa Git Rebase, a gente não usa

Git Merge, então a gente usa Rebase com amende.

Então o que a gente hoje em dia tem bem menos apenas quando acontece daquela... Tem feature que é meio que impossível assim, quando sobe não causar um impacto, mas tirando esses casos, recentemente, depois que existiu uma certa maturidade para entender que certos PRs precisam subir primeiro para evitar um impacto maior nos demais, quando teve essa organização usando Rebase e amende, os níveis de conflito não fizeram o que são o nível esperado, tem aquelas features que são inevitáveis, tem um Rebase e outro ali. Normalmente é um parto, porque a gente normalmente não faz isso, a gente não evita dar holdback, a lógica que é usada é tentar criar uma branch de manutenção, não necessariamente eu gosto dessa ideia, mas tudo bem, mas o processo atual é criar uma branch de manutenção, e nessa branch de manutenção, é focada, porque ela resolve os problemas que subiu de forma pontual, e já integra diretamente com a branch de desenvolvimento. São raras as vezes que acontece na produção, então normalmente quando acontece, isso acontece de forma até em desenvolvimento mesmo.

Acho que o tempo que a pessoa tem, o quanto tempo a pessoa trabalha com o versionamento código, o quanto tempo a pessoa tem experiência, acho que é uma pergunta boa, porque você consegue ter outros filtros, mas não é necessariamente o que a gente perceba, mas é só um problema.