Entrevista 19 - Entrevistado

Eu atualmente estou atuando como analista de sistemas Custão de cargo, analista de sistemas, função, tech lead Seria como sênior? No momento estou atuando em um projeto, no caso um cliente Que tem cerca de 5 frentes, que é uma sustentação Assim como também dá uma evolução, criação de novos projetos Mas atua basicamente em 5 contextos Pegando o nosso contexto, que a gente dá suporte e avanço na aplicação São cerca de 5 desenvolvedores, 5 devs e um tech lead E existem também outras equipes que fazem parte de outros projetos Que envolvem integrações com o nosso, demais, mas são externas Sim, contando com o tech lead, sim Bem, desde o nascimento da solicitação Ou acaba surgindo através do negócio Uma demanda nova de alteração e implementação Ou até mesmo uma correção e melhoria de processos que a gente percebe no código E com isso acaba que o tech lead fica responsável Por poder fazer a definição da atividade, definir a regra de negócio junto ao PO E fazer a quebra das tarefas Depois disso, é feita toda uma divisão das tarefas que vão ser realizadas no código E a partir disso o desenvolvedor pega uma atividade Ele pega a atividade que está no nosso quadro de controle Atualmente a gente está usando o Kanban para isso E a partir dessa atividade ele faz toda uma... Ele faz uma criação de uma branch A gente trabalha usando o Git no Ezri Ele faz uma criação da branch atualmente A gente está criando as branches a partir da master mesmo Porque o nosso projeto, apesar de a gente ter um suporte em 5 contextos diferentes Existem outras equipes que fazem também modificações nesses projetos Então assim, tem muita coisa ali que está na release, que está em QA, que está em dev Que está sendo testado, está validando Mas ainda não foi autorizado, não foi validado para subir em produção Então sempre que a gente vai criar uma branch nova no nosso fluxo aqui A gente acaba criando a partir da master O desenvolvedor faz todo o desenvolvimento necessário nessa branch E depois disso a gente segue mais ou menos o GitFlow aqui Que a gente tem recomendado no nosso projeto A gente passa pelo desenvolvedor no caso, né? Faz o pull request O tech lead lá, ele faz a validação Sobe lá em QA O QA executa o teste E caso esteja ok, a gente abre um pull request direto para a produção Faz feature também, só que assim Como a estrutura que está montada também Existem várias equipes que mexem no projeto Tem muita alteração ali que a gente está testando em QA Tem alteração de features, testando em QA em conjuntos Mas não pode subir em produção, porque ainda não validou Ou por causa que tem mais modificações a serem feitas Então, para não gerar um impedimento na nossa feature A gente acaba subindo a nossa feature direto em produção Após passar em QA Mas subindo direto da feature para a master Desenvolvimento não, por causa que é um ambiente que a empresa não tem disponível Mas basicamente tem QA em produção Sim, sim, porque assim, como falou Como esse projeto não é de controle apenas interno aqui entre o time Existem outros times envolvidos no projeto Para poder fazer um melhor gerenciamento disso Para não ter

impedimentos nas nossas releases A gente acabou adotando essa prática Também acabei utilizando em outras empresas Como falou, um versionamento ali, uma prática mais voltada a próprio GitFlow Onde a gente seguia fazendo as subidas através da release para a master E isso foi mais um projeto onde toda a equipe era dona do código Então realmente, nossa equipe era responsável por toda a manutenção do código Assim como a aprovação das pull requests, a criação das releases E a liberação em produção Bem, o principal ponto da questão de a gente fazer esse versionamento É justamente para poder, não vou dizer assim Que o GitLang não vai te dar uma resposta Ele é uma ferramenta que igual o Scrum, qualquer metodologia, ferramenta Ele não está ali para solucionar o seu problema Depende de como você usa Então assim, para a gente aqui, hoje em dia ajuda bastante Porque a gente segue também algumas boas práticas aqui Que existem para isso, que é dar commit significativo A gente trabalha também, como falou, com avaliação de pull requests Então a gente consegue ter um gerenciamento bom, um controle bom, sabe? Das versões que a gente sobe E como eu falei, como a gente faz alguns commits significativos A gente até facilita, vamos dizer assim, a própria revisão de código Por causa que quando a gente está lendo uma revisão A gente não vai só simplesmente ler o código A gente quando lê o título da pull request Assim como o próprio nome da branch A gente tem alguns, vamos dizer assim, um glossário Que já abre a mente assim que você ler aquela pull request Então você já sabe do que se trata aquele contexto Você já sabe do que se trata aquela pull request E facilita o entendimento da revisão do código Bem, eu não vejo assim nenhum fator que não favorece, como eu falei Isso ali para a gente é apenas uma ferramenta E sim vai depender do uso dela Então um problema que pode ocorrer nesse caso É por possuir um projeto que é distribuído entre mais de um time É justamente o cuidado na gestão das branches Porque às vezes uma outra equipe acaba subindo uma release Que está levando código que não deveria ir para a produção Então o maior problema que a gente pode possuir nesse cenário É justamente o gerenciamento dessa ferramenta Eu não vejo assim complexidade Mas um próprio alinhamento mais funcional entre os times Porque nesse caso ali, como todos tinham acesso a ferramenta Ocorreu um cenário já onde a gente acabou tendo um código subindo em produção Que não deveria subir Mas justamente por ter o controle de versão, ter a visão disso Foi fácil resolver Onde a gente conseguiu facilmente identificar qual foi a versão Que causou o problema em produção E fazer o rollback lá de maneira rápida Do GitFlow, é porque a gente não tem alguns ambientes Mas eu tentaria manter, por enquanto Enquanto esse projeto for dividido com várias equipes Eu acabaria mantendo esse modelo Mas caso esse projeto fosse mantido por apenas uma equipe Onde essa equipe tivesse todo o controle do código Eu acabaria puxando mais para o próprio GitFlow Para que a gente consiga fazer uma melhor separação das nossas branches É uma empresa já estabelecida Recorrente não, porque apesar das várias equipes Tomarem conta do mesmo contexto, da mesma API, do mesmo projeto São em contextos diferentes dentro daquela aplicação Então como a nossa estrutura de arquitetura é um pouco mais semelhante Aqui em Slice, onde é dividido por contextos Cada contexto fica a parte Então na questão de conflito, mesmo

de desenvolvimento Pode ocorrer um ou outro assim Mas sempre entre a própria equipe Onde a gente na revisão consegue identificar e resolver facilmente