Entrevista 08 - Entrevistado

Hoje eu atuo como um engenheiro sênior em um dos três times da empresa. Minha responsabilidade atualmente é garantir evoluções no time de plataforma. Hoje eu acredito que em torno de seis projetos.

Você diz que o principal é um Principal Software Engineer? Não, não. Atuo igualmente em todos os projetos. É porque vai muito de momento.

A depender do foco da empresa, alguns dos projetos podem ter um enfoque maior. Mas se eu fosse dizer qual o maior foco que eu já tive no meu histórico na empresa, seria na parte de vendas e aquisição de clientes, trabalhos com parcerias. Somando a todo, somos em torno de uns 15 desenvolvedores.

Até um tempo atrás, uns seis meses, variava um pouco devido a necessidade de escalabilidade de última hora. Tinha um parceiro que exigia que a gente subisse algum código em um bitplot, e a gente tinha que improvisar. Mas hoje em dia o processo é homogêneo para todos os projetos dos quais a gente cuida.

Eu falei de seis projetos agora há pouco, mas um desses projetos é como se fosse um monohack de estruturas de domínios diversos de front. Então é um pouco mais amplo o nosso grupo de responsabilidade. Mas todos eles utilizam a mesma estrutura de deploy.

No momento, como é que a gente faz? Vou falar tanto do processo tecnológico quanto do processo com pessoas. Dado que vamos fazer uma melhoria em um determinado projeto da empresa, ele sobe para pull request, é avaliado, revisado. Dado que está tudo ok, ele é aprovado e mergeado de uma branch qualquer para a branch main.

Antigamente a gente usava o conceito de feature branch, hoje a gente está deixando de usar isso e já faz o merge direto na main. Certo? Uma vez que está mergeando na main e a gente identifica se existe uma janela de deploy interessante para deploy, o que seria uma janela de deploy interessante? Existem momentos um pouco críticos devido à natureza do nosso trabalho, em que dois a três dias do mês sempre vão ser muito agitados. E a gente prefere fazer deploys apenas do que é necessário nesses momentos.

Dado que não estamos nesses dias, nesse período, a gente identifica se faz sentido fazer um deploy naquele momento e se fizer, a gente faz. E como é que funciona? A gente gera uma tag a partir da main. Essa tag vai conter um identificador de qual projeto a gente está querendo deployar.

Só elucidando uma coisa, boa parte dos nossos projetos estão na estrutura de monorepo e depois temos outros projetos que estão em estruturas separadas. Mas todos eles usam a mesma ideia. A gente gera uma tag a partir da main do projeto que a gente está acessando.

Essa tag é jogada no GitHub Actions. O GitHub Actions, dado a tag que a gente gerou, ele identifica se a gente está querendo fazer um deploy para produção ou para homologação. Se for para produção, ele vai pegar aquela tag, vai começar a gerar os recursos na AWS, vai acionar o CloudFront e tudo mais, se for front-end.

Se for back-end, ele vai direto no Docker, gera uma imagem do Docker, vai no Elastic Beanstalk da AWS e solve o nosso ambiente lá. Mas basicamente o GitHub Actions gera os recursos que a gente precisa e solve isso para a produção. A gente vai ter uma tag que vai identificar qual é aquela versão que foi deployada.

No caso, encaixa um pouco com o que você comentou mais cedo no versionamento. Dado que eu queira fazer um novo deploy para a produção, o que eu vou fazer? Eu vou pegar aquela última tag, vou dar um cherry pick ou puxar a main e jogar nessa tag, eu gero a nova tag a partir dela e faço o deploy. É o mesmo processo.

Se for back-end, ele vai gerar um Docker, vai jogar no Elastic Beanstalk da AWS e vai gerar um novo ambiente. Se for front-end, ele vai gerar todos os recursos que são necessários, vai acionar o CloudFront, atualizar as rotas e tudo mais, e vai subir o nosso front-end. Mas é basicamente isso em relação aos nossos deploys.

Já cheguei a usar. Hoje, o que a gente usa mais próximo de um GitFlow? Cara, no geral, sempre foi um... A gente sempre fez deploy de tag. Sempre fez deploy de tag nos lugares em que eu trabalhei.

Sempre foi gerenciado assim o versionamento. Não teve um formato diferente. Já trabalhei em empresas que faziam deploy para a produção uma vez por mês, mas era sempre a partir de uma tagzinha para gerenciar o versionamento.

Em outros lugares, ocorriam vários deploys no mesmo dia, mas era todo um gerenciamento bem parecido com o que eu faço agora. Só mudou a ferramenta, basicamente. Certo? Cara, entrando um pouquinho nessa questão.

Como eu comentei, a gente tem um foco muito forte em usar Feature Branch na empresa. E isso diminuiu um pouco, certo? No momento em que a gente está, atualmente, para tentar fazer um deploy de uma branch e, se possível, já gerar um deploy caso faça sentido. Meio Trank e Based.

Mas, na essência, a gente ainda está utilizando aquela estrutura do GitFlow. Por que a gente acha interessante isso? Às vezes, você vai... A depender da organização da empresa e do momento em que ela se encontra, a gente não consegue ter uma visão clara da demanda completa que vai entrar. Isso também pesa um pouco quando o código é um código que tem problemas de escalabilidade.

Então, quando se inicializa uma demanda, às vezes é mais interessante você criar uma Feature Branch, ou criar uma branch apartada, ou trabalhar com apenas uma branch e

subir ela de uma vez com toda a demanda do que fazer pequenos PRs que podem vir a quebrar alguma coisa. Eu, particularmente, gosto de subir tudo de uma vez só na branch de produção. Então, acho que, se você está em um ambiente um pouco desorganizado e o código não tem a escalabilidade muito legal, eu acho bacana você trabalhar com um flow de Git mais organizado baseado em Feature Branch, ou alguma coisa parecida com isso.

Pelo menos é a visão que eu tenho. Não favorece o uso? Ah, cara, eu acho que a sua base de código está muito boa, está muito bacana. Você tem uma visão muito clara do que está subindo.

E as tarefas foram refinadas a um ponto de granularidade em que você tem a visão do micro de cada tarefa de código que vai subir. Você pode fazer pequenos merges na main diretamente, desde que você já saiba qual vai ser o impacto daquilo que está subindo. Eu acho que depende muito da base de código que está sendo trabalhada.

Uma base mais desorganizada se beneficia mais de um modelo semelhante a esse. Uma base muito bem organizada, você pode fazer pequenos merges na main incrementais e não deveria ter nenhum impacto. A gente está mudando, não é? A gente antigamente fazia apenas... A cada Store a gente já trabalhava com a Feature Branch, não é? Eu disse que a gente usa o Git Flow porque a gente ainda usa um pouco desse modelo.

A gente está deixando ele para o outro lado, mas... A gente tem uma base que está muito desorganizada. Nosso código tinha uma certa dificuldade no Process and Deploy. Então a gente trabalhava com Feature Branch para poder jogar novas histórias, novos épicos de uma vez e conseguir garantir que não iríamos ter problemas.

Como a nossa base está melhorando muito, nosso Process and Deploy está muito mais bem estruturado do que era, sei lá, 4, 5 meses atrás, a gente já está conseguindo migrar para um modelo em que a gente está fazendo merges incrementais na main. Mas tendo que se preocupar com uma estrutura muito mais organizada de breaches para fazer a integração contínua. Não está sendo necessário.

É, sim. Não é uma startup. Definitivamente não é uma startup.

Ela começou com um monohack, com a Devs e o sonho, basicamente. Mas... Mas... Há mais ou menos uns 6 meses ela já tinha provado o valor dela no mercado. A gente está perto do momento de virada em relação... a gente vai começar a dar lucro daqui a pouco.

Então... Já começou a ter um shift tecnológico. A gente agora está se preocupando com escalabilidade e evolução do código. Então, os últimos 6 meses foram basicamente só melhorias no nosso Process and Deploy, de estruturação e arquitetura de código.

Então... É uma startup. Começou como uma startup, bagunçada como todas as outras. E

agora a gente está tendo esse momento de virada para começar a se preocupar mais com a escalabilidade da operação.

Já tivemos, já tivemos, na época da... que o uso de feature breach era mais pesado. A gente teve bastante conflito naquela época. Hoje, com a mudança para um modelo mais trunk-based, a gente está tendo menos problemas.

Porque, em pequenos meses, dificilmente a gente vai ter grandes conflitos.