Super Categorias

Categories (general)	Categories	Codes
TBD	Utiliza a metodologia hubflow (Git Flow no Github)	P2 - [3] Utilizam a metodologia de hub flow.
		P5 - [11] Utiliza branches no modo hub flow. Apenas entra na branch master aquilo que está preparado para ir para produção.
	Integra o código na branch principal apenas no final da sprint	**Utiliza a metodologia hubflow (Git Flow no Github)**
		P1 - [3] Lança no fim de cada sprint (2 semanas) - [5] Entrega código na master e para os clientes a cada sprint"
		P3 - [1] Utiliza o SCRUM e tem um processo bem definido. Sprint de 2 semanas - [4] O código de cada sprint é colocado na branch principal apenas no final da mesma.
		P6 - [4] Cada feature é feito em uma branch separada. Apenas quando concluída ela é revisada pelo gerente e colocada na branch principal.
FT	Utiliza o conceito de épicos, mas o código só é integrado quando terminado	P2 - [25] Há o conceito de funcionalidades como épico, que se prolongam por múltiplas sprints, mas não há o conceito de FT: o código é mergeado apenas após a feature estar completa.
		P3 - [6] Tem a ideia de fazer uma funcionalidade entre duas sprints, mas nunca entregam pela metade, é sempre apenas quando ela estiver completa.
	A feature só vai para o ambiente de produção quando concluída	**Utiliza o conceito de épicos, mas o código só é integrado quando terminado**
		P1 - [20] Se a feature nao está pronta, ela nao entra em produção
		P6 - [12] Apenas quando tiver tudo pronto é que é lançado para homologação
		P7 - [25] Se a feature não tá pronta não entrega. Não utiliza condicionais para esconder código não pronto
		P9 - [27] As features só entram quando estão finalizadas
		P11 - [25] Quando uma funcionalidade não está pronta ela não é entregue, não utiliza Feature Toggle. Ou entrega quando está pronto ou particiona o código e adiciona pequenas partes até ter a funcionalidade de fato.
AWA	O time de desenvolvimento é o mesmo que faz deploy e mantém a aplicação rodando. Não há um time separado de operações	P1 - [8] É ele quem faz a entrega nos ambientes - [10] Há apenas ele na equipe, e ele é responsável pelo deploy e manutenção dos ambientes.
		P2 - [8] Existe uma ferramenta de CLI do projeto que automatiza a criação de algumas coisa e facilita o provisionamento de ambientes. - [9] Há a cada sprint o disparo de um release notes para toda a equipe e uma versão simplificada para os usuários do sistema interessados nas novas versões. - [13] Não existe o papel de Devops, há um revezamento entre todos da equipe para que o conhecimento seja difundido
		P5 - [14] O responsável por monitorar as entregas é o líder técnico da equipe. Todos conseguem deployar, sabem como fazer, mas geralmente quem faz é o lider - [15] Todos os desenvolvedores são devops.
		P6 - [17] O gerente que libera novas versões, mas todos do time sabem como fazer isto. - [18] Não há um time de operações. - [19] Depois de um problema, o passo a passo para a entrega de novas versões foi repassado para todos da equipe.
		P8 - [11] O time de desenvolvimento é o mesmo que faz deploy e recebe o feedback do cliente. - [12] O desenvolvedor é responsável por monitorar a entrega.
		P9 - [12] Há um ownership com relação a feature que está sendo entregue. A pessoa que faz é a mesma que entrega. Todos do time sabem fazer tudo: “Não tem divisão entre o dev e o ops”.
	Há um time de Devops específico, mas não há grandes silos entre este e a equipe de desenvolvimento	P3 - [7] Quando há uma nova versão no ar é enviado um email para todos os interessados. - [9] Há uma equipe dedicada para devops. Mas sempre temos uma conversa entre a equipe do projeto e o SRE para melhorar o produto final. - [11] O deploy é feito pelo projeto, não por SRE
		P7 - [11] Sobre a divisão de papéis, apesar de ter um time específico de SRE, os desenvolvedores também são devops. Qualquer pessoa do time pode entregar uma nova versão. - [14] Os desenvolvedores não entregam só algo feito para SRE entregar, eles também entendem como funciona a infraestrutura.

Super Categorias

HC	**Há apenas poucas verificações não muito complexas a respeito do sistema de produção**	P1 - [17] Não há HC rodando atualmente, mas já houve (está quebrado). Há apenas um check sobre o certificado https.
		P5 - [24] Não tem muitos testes automáticos de health check, apenas um caso para testar se o servidor do jogo está rodando.
DOC	**Não é uma prática definida, mas acontece implicitamente pela equipe**	P7 - [12] Não há uma definição para que a pessoa que fez a mudança fique um pouco mais depois de ela ser liberada, mas isso acontece implicitamente.
		P8 - [13] Como o entrevistado é responsável pela entrega, algumas vezes foi necessário que a gerente conversasse com ele fora do expediente para ajeitar algo de produção. - [21] Após uma nova feature entrar em produção, não é esperado que o desenvolvedor fique de plantão, mas já aconteceram alguns casos. O entrevistado procura estar disponível.
		P9 - [23] Em alguns casos há desenvolvedores que ficam analisando as métricas e o sistema depois de um deploy em produção, mas depende da mudança.
PIP	**Há um processo bem definido, mas este contém poucos ou nenhum passos automatizados**	P1 - [1] O ciclo consiste em: Funcionalidade nova, testes manuais em staging e deploy - [6] Consegue fazer a entrega de forma rápida (menos de 30 minutos), mas não tem um processo automatizado - [7] Ele acredita que poderia ter um processo mais automatizado - [12] Existe um processo estagiado mas não automatizado
		P4 - [1] Processo de HW é bem pouco maleável. Geralmente as coisas são decididas logo no começo, pois mudar no final é custoso. - [7] Dependendo do tamanho da mudança, é necessário ou não uma outra prova de conceito e prototipação. Outras mudanças podem não passar por todas as fases. O time decide sobre o tamanho da mudança. - [11] O que pode ser automatizado eles automatizam, mas, devido ao contexto, essa parte se torna bem mais difícil.
	Não tem um processo bem estruturado e não há passos automatizados.	P6 - [11] Não há um processo fixo. A ideia é fazer tudo que tem que ser feito para aquela nova entrega. - [13] Não há passos de deploy automatizados.
		P8 - [1] Não tem um processo bem estruturado. - [2] Após o commit e o review das features na sprint review, as mudanças são enviadas para produção. Não tem nada automatizado.
	Tem um processo estruturado e passos automatizados, mas este não é seguido para todo tipo de mudança.	P5 - [2] O processo de entrega é bem definido, mas alguns passos dependem da funcionalidade. Existem funcionalidades que são testadas manualmente para validação. Mas tem algumas outras que são pequenas o suficiente para não necessitar de testes manuais. - [3] A equipe define o tamanho da mudança e que pipeline ela vai seguir.
		P10 - [2] Quando o commit chega na master o deploy é feito automaticamente através do google kubernetes. - [3] Coisas pequenas ou grandes não utilizam o mesmo processo. Coisas pequenas seguem um caminho “mais curto” para chegar em produção. - [4] O processo de rodar os testes também é automatizado.

Super Categorias

CAN	**Utiliza um ambiente de homologação apenas para testes e validação de requisitos, mas este utiliza um banco de dados diferente do de produção**	P1 - [2] A funcionalidade é lançada para todos que forem afetados, já que o código é baseado na linha de produto - [4] Em um hotfix entra na mesma hora para todos - [13] Existe um ambiente de staging parecido com produção, mas este é utilizado apenas para testes da equipe interna e apresentações
		P2 - [5] Depois que é gerada a release, ela é entregue no ambiente de homologação para checks de regressão e de visualização inicial do cliente. Ao fim da sprint, essa release é levada para produção
		P6 - [8] Uma mudança passa primeiro para homologação e é testada apenas por uma parte dos usuários (que são os clientes). - [9] Homologação tem um banco de dados diferente de produção
	As features são sempre entregues para todos os usuários ao mesmo tempo	P3 - [5] Uma feature é sempre lançada para todos os usuários
		P8 - [4] As features são enviadas para todos os usuários de uma só vez.
		P11 - [6] Todos os usuários recebem as mudanças ao mesmo tempo
DAR	**Não conhecia a técnica de dark launches**	P2 - [26] Não conhecia a técnica de dark launches
		P6 - [28] Não conhecia a técnica de Dark Launches
		P8 - [27] Não conhecia e não utiliza a técnica de dark launches
	Nunca utilizou a técnica de dark launches	**Não conhecia a técnica de dark launches**
		P1 - [22] Conhece a técnica de dark launches, mas nunca fez uso dela.
		P10 - [24] Nunca utilizou dark launches.
		P3 - [30] Nunca utilizou a técnica de dark launches. - [31] A equipe de desenvolvimento não tem acesso aos dados de produção.
AB	**Não utiliza A/B pela baixa quantidade de usuários**	P2 - [24] Não utiliza A/B pois a infra hoje tem apenas um servidor único onde todos os usuários acessam e não há uma quantidade tão grande de usuários
		P5 - [22] Não utiliza testes A/B. - [27] A base de usuários é pequena para poder utilizar o processo de Testes A/B
		P6 - [27] Testes A/B não são utilizados pois a base de usuários é muito pequena.
		P11 - [24] Não utiliza testes a/b pois a base de usuários é pequena
	Testes A/B não se aplicavam ao contexto da aplicação	P7 - [4] Não utiliza partial rollouts. - [24] Testes A/B não são utilizados por achar que não se aplica ao contexto da aplicação
		P9 - [26] Não utiliza Testes A/B pois é difícil aplicar nos serviços que usam, visto que são sistemas de uso interno
Code Review	**Acredita que o code-review funciona para troca de conhecimento**	P3 - [17] A entrevistada não tem muita experiência com código por ser razoavelmente nova na área, e vê a revisão de código como bastante agregadora de conhecimento. - [18] O code review dissemina conhecimento de diferentes áreas do sistema para toda a equipe.
		P10 - [15] O code review aumenta o aprendizado dela como engenheira junior
		P11 - [19] Utiliza code review principalmente pela troca de conhecimento.
Testes automáticos	**Acredita que o aumento da cobertura de testes automáticos pode diminuir a quantidade de bugs em produção**	P1 - [15] Para melhorar QA: Deveria adicionar testes automatizados
		P5 - [10] Ele acha o processo de entrega bom, dá certo, mas acredita que seria bom ter testes mais automatizados, só que o tempo da sprint é curto.
		P8 - [23] Mais testes manuais e automáticos poderiam diminuir a quantidade de bugs
		P9 - [24] Se houvesse uma suíte mais estruturada de teste, o número de erros em “run time” seriam diminuídos, mas o processo de entrega seria bem mais devagar.
		P11 - [20] Sente que tem muita regressão, talvez devido a uma cobertura de testes pequena. Outro fator que aumenta a quantidade de bugs é a comunicação ruim entre os times (QA e dev).