Qualidade de Software na ZapSign

Ao decorrer deste documento minha intenção é demonstrar como ter uma área e pessoas comprometidas com a qualidade das entregas pode impactar positivamente internamente na empresa, quanto externamente na forma que a sociedade e em especial os clientes da ZapSign enxerga os produtos oferecidos.

Mas antes de tudo gostaria de discorrer sobre o que é qualidade? Se formos buscar em algum dicionário o significado deste nome, está relacionado com: propriedade, atributo ou condição das coisas ou das pessoas capaz de distingui-las das outras e de lhes determinar a natureza.

Com base nisso trazendo para o contexto da ZapSign poderia dizer que podemos medir a qualidade tanto no quesito de como os colaboradores das ZapSign estão realizando seu trabalho, quanto como a ZapSign está realizando suas entregas aos seus clientes e no mundo.

Então como aplicamos o conceito de qualidade no contexto de uma empresa que cria sistemas e softwares e possui produtos digitais? Tendo isto criou-se a área de qualidade de software.

Na engenharia de software esta área está representada nos conceitos de atributos de qualidade e na disciplina de Verificação e Validação. Onde a

- verificação é uma atividade na qual envolve a análise de um sistema para certificar se este atende aos requisitos funcionais e não funcionais.
- validação é a certificação de que o sistema atende as necessidades e expectativas do cliente.

Onde tais conceitos precisam andar juntos e dependentes um do outro para conseguir medir e mensurar como está a qualidade.

Agora que sabemos mais sobre qualidade vou estar realizando a análise técnica proposta no desafio:

Baseado na documentação do processo de desenvolvimento de uma feature na ZapSign a seguir, disserte sobre onde e como o time de Q&A pode impactar positivamente na estabilidade da aplicação propondo mudanças no fluxo se necessário:

Análise: Como pessoa de qualidade, minha orientação é que antes de se preocupar com a estabilidade da aplicação, é importante a empresa trabalhar a cultura da empresa para que seus colaboradores apliquem o conceito de qualidade ao realizar suas atividades, que tenha a qualidade como requisito e característica da empresa. E com isso se distinguir das outras empresas pela a qualidade que realiza seus processos e entrega seus produtos. Tendo processos de verificação e validação em toda a empresa para garantir isso. E aplicar as 3 áreas da qualidade que são gerir, garantir e controlar. Tendo métricas para basear a evolução da qualidade e transformar de dentro para a fora a performance e estabilidade de seus produtos de software (aplicações).

Análise: Antes de começar o refinamento de uma tarefa que é onde começa o processo abaixo, gostaria de ressaltar a importância de duas etapas que antecedem o refinamento. Que é o levantamento dos requisitos (necessidades, problemas, dores dos stakeholders) e planejamento do desenvolvimento (escopo, custos, riscos, pessoas, etc) de qualquer aplicação e/ou funcionalidade. Estas etapas precisam estar muito bem fundamentadas em conceitos e atributos de qualidades para garantir ter a compreensão suficiente do real problema ou dor a ser resolvida por um software. Evitando gastar tempo e dinheiro com um software ineficiente.

Processo de desenvolvimento de Feature:

1. Definição de critérios de aceitação, layout e escrita do Use Story (card no Jira) pelo time de produto em conjunto com o Tech Lead

Análise: Eu gosto de levantar a bandeira de time e que com mais 'cabeças' pensando sobre um problema com suas diferentes visões, experiências e papéis agrega para ter uma solução mais completa em probabilidades e cenários, além de antecipar problemas técnicos, defeitos e riscos. Entendo que às vezes é difícil reunir um número maior de pessoas, mas gastar mais tempo no planejamento e refinamento além de diminuir os custos do projeto, ajuda na maior qualidade. Pois todo o time estaria mais alinhado em relação ao o que precisa ser feito e como deve ser feito.

Além do que as definições feitas nessa etapa vão direcionar todo o fluxo restante, então é melhor descentralizar para mais pessoas do time. Entendo que o Tech Lead possui grande experiência para tomada de decisões, mas porque não trazer algumas pessoas chaves do restante do time de

desenvolvimento para mais perto?

2. Refinamento da estratégia proposta por produto (buscar o melhor custo benefício de implementação considerando débitos técnicos em conjunto com o PM)

Análise: Nesta etapa é importante ter uma boa documentação dos débitos técnicos e limitações já existentes, diagramação da arquitetura atual e uma rastreabilidade de funcionalidades e requisitos funcionais e não funcionais do sistema. Pois isso vai agilizar a tomada de decisão arquitetural, ajudar medir os custos de implementação, antecipar limitações e adaptabilidades a serem feitas para atender os requisitos e critérios de aceitação e entender se é possível reutilizar alguma parte do projeto ou a necessidade de dependências de outros projetos.

3. Definição de esforço pela respectiva squad com poker planning seguindo a definição de pontos por esforço e marcar o valor definido no card do Jira

Análise: É muito comum ver dinâmicas de estimativa de esforço como esta, principalmente em processos e times ágeis, mas é raro processos que gerencia e controla se de fato a estimativa está correspondendo ao executado. Sem atividades de controle e garantia da qualidade nessa etapa faz com que tenhamos prazos não cumpridos ou um funcionalidades sendo entregues sem qualidade pois teve que ser entregue para cumprir um acordo baseado numa realidade de esforço e estimativa diferente da realidade e capacidade.

4. Definição do responsável pelo card

Análise: Esta etapa é importante e tem que ser levada em consideração pois temos uma grande oportunidade do Tech Lead aqui de evoluir a senioridade do time, onde junto com o PDI de cada integrante do time, entender as ambições e expectativas de futuro, e balancear a atribuição e distribuição das tarefas de modo em fazer com que pessoas com mais senioridade trabalhe junto com os iniciantes do time para agilizar o aprendizado e paralelamente evoluir as skills do sênior em liderar e do iniciante de atingir sua plenitude. Pois um time mais experiente e com mais senioridade coopera para uma entrega com mais qualidade e atingimento dos requisitos.

5. Definição de prazo pelo responsável (se a tarefa não for estimável, definir o tempo do spike)

Análise: Precisa levar em conta o esforço (baixo, médio, alto) para executar a tarefa e a experiência das pessoas que vão trabalhar na tarefa nas etapas de implementação, testes e entrega.

6. Abrir uma branch nova no github com o ID do card no Jira (ex: ZDEV-666)

7. Escrita dos testes cobrindo os critérios de aceitação

Análise: É importante ter no processo de engenharia do software abordagens já comprovadas que contribui para a qualidade do software, e nesta etapa é importante ressaltar o TDD (Desenvolvimento orientado a testes). Ter um processo que implemente as suas boas práticas e que monitore se realmente está sendo aplicado corretamente, ajudando a cobrir não só os critérios de aceitação mas também os requisitos funcionais e não funcionais.

Obs: Critérios de aceitação e requisitos são conceitos diferentes, sabia? Os requisitos descrevem funções e comportamentos que o cliente solicita, já os critérios de aceitação referem-se a um conjunto de requisitos pré definidos que devem ser atendidos para que uma história de usuário seja concluída.

8. Implementação dos comportamentos

Análise: Importante cobrir todos os cenários de testes e aplicar padrões e boas práticas de desenvolvimento de código para ter uma arquitetura limpa e eficaz, que permita a qualidade, escalabilidade, reutilização e desacoplamento quando possível.

E implementar o conceito de feature flags para caso acontecer alguma instabilidade ou erro posteriormente em ambiente produtivo, ter como rapidamente atuar na correção sem impactar no uso de outras funcionalidades do sistema. Após homologado e sem riscos excluí-las.

9. Garantir que todos os critérios estão passando nos testes. É obrigatório o teste manual, além dos automatizados, caso o card envolva uma das seguintes áreas da aplicação:

a. Fluxo de pagamento
b. Fluxo de criação de documento (api e web)
c. Posicionamento de rubrica
d. Relatório de assinaturas
e. Certificado digital
f. Reconhecimento facial
g. Envio de email/SMS/Whatsapp

Análise: Importante verificar e validar se a solução está atendendo os requisitos dos stakeholders e se

cumpri o que realmente foi designado para fazer com qualidade aceitável.

10. Criar um pull request da branch para homolog para code review e abrir uma thread no canal de Pull Requests do Discord com o ID do card no Jira como título

Análise: Para viabilizar a documentação, rastreabilidade e entendimento dos outros integrantes do time terá que atuar no review, testes e deploy em produção, é importante criar pull requests com boas descrição e deixando claro o que foi modificado e criado. E realizar uma bateria de testes de software respeitando as estratégias e técnicas pré estabelecidas no plano de testes.

11. Marcar todos os devs na thread e o respectivo tech lead para code review. É obrigatório no mínimo o approve do Tech Lead e mais um desenvolvedor para seguir.

12. Realizar o merge do pull request para homologação

Análise: Após realizar o merge realizar testes regressivos para garantir que as funcionalidades já existentes continuem funcionando e não foram impactadas com a nova versão. E realizar testes de aceitação para garantir que está atendendo os critérios de aceite e requisitos do cliente.

13. Abrir uma thread no canal de QA do Discord e marcar o respectivo PM e algum stakeholder (ex: alguém do time de CX/Sales/Marketing)

14. Criar um Pull Request para a master

Análise: Após merge na master acompanhar todo o processo de deploy. Monitorar os primeiros usos da funcionalidade. Se possível realizar um teste de fumaça. Criar tarefas para atuar nos débitos técnicos e erros com baixa prioridade. Documentar as evoluções do software. Dividir as lições aprendidas com o time de desenvolvimento.

15. Tech lead/Devops/PMs soltar o release

Análise: Criar uma boa documentação da release rastreando o que foi removido, modificado e criado na nova versão. Gerenciar as lições aprendidas e pontos negativos e positivos durante o desenvolvimento. Treinar e capacitar outras áreas da empresa e cliente no uso da nova funcionalidade.

Essas são algumas observações e sugestões que acho relevantes no desenvolvimento de uma funcionalidade e no ciclo de vida do software dentro de uma empresa, tendo como base o processo que me passaram.

E claro que uma característica importante de uma engenheira de software é a adaptabilidade em trabalhar com diferentes ambientes, tecnologias e problemas a serem resolvidos. E o mais importante em conjunto criar um processo que faça sentido para todos os envolvidos no processo. Pois somente assim com o comprometimento de todos, tendo como característica a qualidade, vai entregar um produto de software estável e com alta performance.