Bugtracking

- Como priorizar
  - Bugs são "priorizados" pelo seu grau de severidade no sistema
    - ou seja, o quanto ele impacta a usabilidade do cliente
    - quanto mais "bloqueado" o cliente fica na aplicação, MAIOR é seu grau de severidade
  - níveis de severidade
    - os níveis podem mudar muito de ferramenta para ferramenta ou estrutura de time
    - níveis genéricos
      - CRITÍCO
        - usuário impedido de usar a aplicação ou um ponto critico dela
        - exemplo
          - logar em um aplicação
          - e-comerce não conseguir finalizar um pedido
      - ALTO
        - Não bloqueia o uso do cliente mais mesmo assim afeta a aplicação
        - exemplos
          - selecionar um item do menu
          - não exibir mensagem de erro
      - MÉDIO
        - Não tem bloqueio e nem alto impacto, normalmente são aquelas funcionalidade "segundarias"
          - ou que você pode executar a mesma ação de outra forma no sistema
        - exemplo
          - filtros
          - buscas
      - BAIXO/ESTÉTICO
        - exemplos
          - alinhamentos
          - cores
          - resoluções de imagem
        - tarefas que não geram impacto no sistema ou na funcionalidade
        - aquelas detalhes de layout que não vai matar ninguém
- Como gerenciar
  - lembram das categorias? Pois é você pode gerar um filtro e filtrar por times, causa raiz ou frente
  - Controle por versão ajuda você a saber quantos defeitos ainda estão abertos impactando um deploy
    - os sistemas sempre tem numero de versão
    - nome de pacote
- Como abrir
  - titulos coerentes
    - titulos devem ser enxuto e conter o resumo do defeito
    - eles devem ser claros a ponto de não precisar abrir o defeito para ler a descrição para saber o que está acontecendo
    - exemplos
      - Erro ao realizar login
      - Erro na mutation logarUsuário
  - descrição
    - a descrição do defeito deve conter o passo a passo para a sua reprodução
    - Use e abuse do Gherking aqui, e se a sua validação já está escrita em Gherking basta adiciona-la aqui
      - E vocês lembram do MAS do Gherking, agora nos vamos usar ele como contra ponto do resultado esperado
      - Então o login deve ser realizado
        - MAS observou-se uma erro
        - E o login não foi realizado
        - sacou?
  - evidência
    - Grave videos
    - Tire print
    - Desenhe
    - Ou até narre a sua execução
    - Vale tudo para ficar claro a todos
  - severidade
    - Informe a severidade do defeito
  - rastreabilidade
    - Vincule a tarefa que estava sendo testada que foi encontrado o defeito
    - Vincule a execução do seu teste ao defeito
    - Vincule o seu plano de teste ao defeito
- Material produzidor por
  - Priscila Caimi
  - para a plataforma Qualiters Club
  - todos os direitos estão reservados a Qualiters Club
- Objetivo
  - garantir a qualidade e confiabilidade
  - diminuir o número de defeitos
  - aumentar a satisfação do cliente
  - garantir rastreio de tarefas vs bugs
- O que é erro
  - um erro é uma ação humana incorreta
  - exemplo
    - não compreender correta a documentação da funcionaldiade
- O que é falha
  - uma função não realiza a ação desejada
  - exemplo
    - função de logar não loga na aplicação
- O que é defeito
  - Uma falha na lógica pode gerar um defeito na aplicação
  - exemplo
    - erro no código
      - função de receber senha não está validando se é uma senha valida dentro dos padrões estabelecidos na documentação
    - erro no processo
      - ao não realizar o protótipo da aplicação na fase correta, gerou um desenvolvimento incorreto da tela
- Como categorizar
  - Uma das formas de categorizar é informa o time responsável que será "dono" do defeito
    - ao fazer isso você ganha uma métrica de
      - X defeitos foram abertos ao time Y
    - crie uma lista com o nome das equipes e coloque um parâmetro no seu excell ou ferramenta de gerenciar defeitos
  - causa raiz
    - criar um processo de causa raiz ajuda a criar um processo para que aquele tipo de defeito não ocorra mais
      - exemplos
        - para isso ocorrer é preciso o time todo se envolver em uma discução para achar o ponto onde ocorreu o problema
        - protótipo
        - documentação
        - entendimento técnico
        - inviabilidade técnica
        - ambiente
        - recursos computacionais
    - também é conhecido como diagrama de Ishikawa
      - ou diagrama de espinha de peixe
  - "frente" responsável
    - aqui podemos informar se o defeito foi de backEnd, FrontEnd, Integração, etc ..

Bugtracking

1. Material produzidor por

1.1. Priscila Caimi

1.2. para a plataforma Qualiters Club

1.3. todos os direitos estão reservados a Qualiters Club

2. Objetivo

2.1. garantir a qualidade e confiabilidade

2.2. diminuir o número de defeitos

2.3. aumentar a satisfação do cliente

2.4. garantir rastreio de tarefas vs bugs

3. O que é falha

3.1. uma função não realiza a ação desejada

3.2. exemplo

3.2.1. função de logar não loga na aplicação

4. O que é defeito

4.1. Uma falha na lógica pode gerar um defeito na aplicação

4.2. exemplo

4.2.1. erro no código

4.2.1.1. função de receber senha não está validando se é uma senha valida dentro dos padrões estabelecidos na documentação

4.2.2. erro no processo

4.2.2.1. ao não realizar o protótipo da aplicação na fase correta, gerou um desenvolvimento incorreto da tela

5. O que é erro

5.1. um erro é uma ação humana incorreta

5.2. exemplo

5.2.1. não compreender correta a documentação da funcionaldiade

6. Como categorizar

6.1. Uma das formas de categorizar é informa o time responsável que será "dono" do defeito

6.1.1. ao fazer isso você ganha uma métrica de

6.1.1.1. X defeitos foram abertos ao time Y

6.1.2. crie uma lista com o nome das equipes e coloque um parâmetro no seu excell ou ferramenta de gerenciar defeitos

6.2. causa raiz

6.2.1. criar um processo de causa raiz ajuda a criar um processo para que aquele tipo de defeito não ocorra mais

6.2.1.1. para isso ocorrer é preciso o time todo se envolver em uma discução para achar o ponto onde ocorreu o problema

6.2.1.2. exemplos

6.2.1.2.1. protótipo

6.2.1.2.2. documentação

6.2.1.2.3. entendimento técnico

6.2.1.2.4. inviabilidade técnica

6.2.1.2.5. ambiente

6.2.1.2.6. recursos computacionais

6.2.2. também é conhecido como diagrama de Ishikawa

6.2.2.1. ou diagrama de espinha de peixe

6.3. "frente" responsável

6.3.1. aqui podemos informar se o defeito foi de backEnd, FrontEnd, Integração, etc ..

7. Como gerenciar

7.1. lembram das categorias? Pois é você pode gerar um filtro e filtrar por times, causa raiz ou frente

7.2. Controle por versão ajuda você a saber quantos defeitos ainda estão abertos impactando um deploy

7.2.1. os sistemas sempre tem numero de versão

7.2.2. nome de pacote

8. Como abrir

8.1. titulos coerentes

8.1.1. titulos devem ser enxuto e conter o resumo do defeito

8.1.2. eles devem ser claros a ponto de não precisar abrir o defeito para ler a descrição para saber o que está acontecendo

8.1.3. exemplos

8.1.3.1. Erro ao realizar login

8.1.3.2. Erro na mutation logarUsuário

8.2. descrição

8.2.1. a descrição do defeito deve conter o passo a passo para a sua reprodução

8.2.2. Use e abuse do Gherking aqui, e se a sua validação já está escrita em Gherking basta adiciona-la aqui

8.2.2.1. E vocês lembram do MAS do Gherking, agora nos vamos usar ele como contra ponto do resultado esperado

8.2.2.2. Então o login deve ser realizado

8.2.2.2.1. MAS observou-se uma erro

8.2.2.2.2. E o login não foi realizado

8.2.2.2.3. sacou?

8.3. evidência

8.3.1. Grave vídeos

8.3.2. Tire print

8.3.3. Desenhe

8.3.4. Ou até narre a sua execução

8.3.5. Vale tudo para ficar claro a todos

8.4. severidade

8.4.1. Informe a severidade do defeito

8.5. rastreabilidade

8.5.1. Vincule a tarefa que estava sendo testada que foi encontrado o defeito

8.5.2. Vincule a execução do seu teste ao defeito

8.5.3. Vincule o seu plano de teste ao defeito

9. Como priorizar

9.1. Bugs são "priorizados" pelo seu grau de severidade no sistema

9.1.1. ou seja, o quanto ele impacta a usabilidade do cliente

9.1.2. quanto mais "bloqueado" o cliente fica na aplicação, MAIOR é seu grau de severidade

9.2. níveis de severidade

9.2.1. os níveis podem mudar muito de ferramenta para ferramenta ou estrutura de time

9.2.2. níveis genéricos

9.2.2.1. CRITÍCO

9.2.2.1.1. usuário impedido de usar a aplicação ou um ponto critico dela

9.2.2.1.2. exemplo

9.2.2.1.2.1. logar em um aplicação

9.2.2.1.2.2. e-comerce não conseguir finalizar um pedido

9.2.2.2. ALTO

9.2.2.2.1. Não bloqueia o uso do cliente mais mesmo assim afeta a aplicação

9.2.2.2.2. exemplos

9.2.2.2.2.1. selecionar um item do menu

9.2.2.2.2.2. não exibir mensagem de erro

9.2.2.3. MÉDIO

9.2.2.3.1. Não tem bloqueio e nem alto impacto, normalmente são aquelas funcionalidade "segundarias"

9.2.2.3.1.1. ou que você pode executar a mesma ação de outra forma no sistema

9.2.2.3.2. exemplo

9.2.2.3.2.1. filtros

9.2.2.3.2.2. buscas

9.2.2.4. BAIXO/ESTÉTICO

9.2.2.4.1. exemplos

9.2.2.4.1.1. alinhamentos

9.2.2.4.1.2. cores

9.2.2.4.1.3. resoluções de imagem

9.2.2.4.2. tarefas que não geram impacto no sistema ou na funcionalidade

9.2.2.4.3. aquelas detalhes de layout que não vai matar ninguém