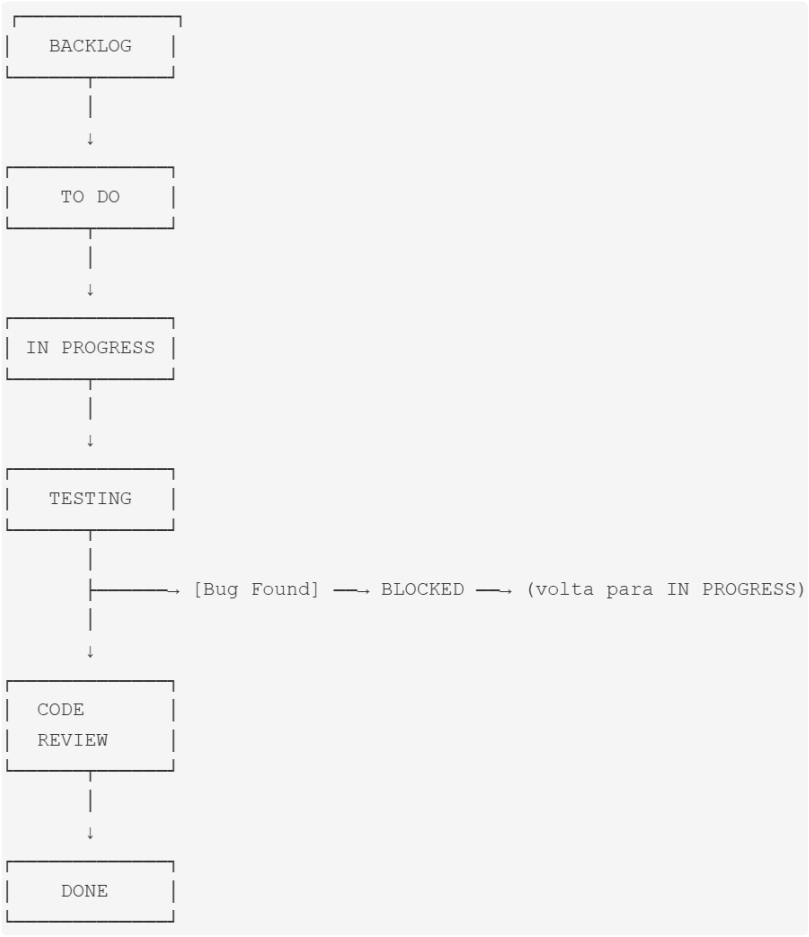
Fluxo de Trabalho e Ciclo de Vida do Bug

1. Fluxo de Trabalho (Workflow)

1.1 Visão Geral do Workflow



1.2 Descrição Detalhada dos Status

1.2.1 BACKLOG

Descrição: Issues aguardando priorização e planejamento.

Critérios de Entrada:

- User Story criada pelo Product Owner
- Requisito documentado
- Ideia ou sugestão de melhoria

Atividades:

- Refinamento de requisitos
- Estimativa de esforço
- Definição de critérios de aceitação
- Priorização pelo PO

Responsável: Product Owner

- Definição de critérios de aceitação
- Priorização pelo PO

Responsável: Product Owner

Critérios de Saída:

- User Story refinada e estimada
- Critérios de aceitação definidos
- Prioridade estabelecida

1.2.2 TO DO

Descrição: Tarefas priorizadas e prontas para desenvolvimento.

Critérios de Entrada:

- Sprint planning realizado
- Tarefa selecionada para a sprint
- Time committed com a entrega

Atividades:

- Aguardando desenvolvedor disponível
- Revisão técnica se necessário
- Preparação do ambiente

Responsável: Scrum Master / Tech Lead

Critérios de Saída:

- Desenvolvedor disponível
- Ambiente de desenvolvimento preparado
- Dependências identificadas

1.2.3 IN PROGRESS

Descrição: Desenvolvimento ativo da funcionalidade.

Critérios de Entrada:

- Desenvolvedor assignado
- Ambiente configurado
- Requisitos compreendidos

Atividades:

- Codificação
- Unit tests
- Commits no Git
- Atualização do status daily

Responsável: Developer

Critérios de Saída:

Responsável: Developer

Critérios de Saída:

- Código completo
- Unit tests passando
- Branch criada e atualizada
- Pull Request aberto

Tempo Máximo: Conforme estimativa da sprint

1.2.4 CODE REVIEW

Descrição: Revisão de código por pares.

Critérios de Entrada:

- Pull Request criado
- Testes unitários passando
- Build sem erros

Atividades:

- Revisão de código
- Análise de qualidade
- Verificação de padrões
- Sugestões de melhorias

Responsável: Tech Lead / Senior Developer

Critérios de Saída:

- Aprovação de pelo menos 2 revisores
- Comentários resolvidos
- CI/CD pipeline verde

Tempo Máximo: 1 dia útil

1.2.5 TESTING

Descrição: Testes funcionais e de qualidade.

Critérios de Entrada:

- Code review aprovado
- Build deployado em ambiente de testes
- Deploy notes disponíveis

Atividades:

- Execução de casos de teste
- Testes exploratórios
- Validação dos critérios de aceitação
- Testes de regressão (se necessário)
- Registro de bugs encontrados

- Validação dos critérios de aceitação
- Testes de regressão (se necessário)
- Registro de bugs encontrados

Responsável: QA Team

Critérios de Saída:

- Todos os casos de teste executados
- Critérios de aceitação validados
- Sem bugs bloqueadores ou críticos
- Evidências de teste anexadas

Tempo Máximo: 2 dias úteis

1.2.6 BLOCKED

Descrição: Tarefa impedida de prosseguir.

Critérios de Entrada:

- Bug crítico/bloqueador encontrado
- Dependência externa não resolvida
- Falta de informação
- Problema de ambiente

Atividades:

- Identificação do impedimento
- Comunicação ao time
- Ações para desbloqueio
- Follow-up diário

Responsável: Todos / Scrum Master facilita

Critérios de Saída:

- Impedimento resolvido
- Bug corrigido
- Informação obtida
- Dependência satisfeita

Ação: Retorna ao status anterior após desbloqueio

1.2.7 DONE

Descrição: Funcionalidade completa e aceita.

Critérios de Entrada:

- Todos os testes aprovados
- Product Owner aceitou a entrega
- Deployed em produção (ou pronto para)
- Documentação atualizada

- Deployed em produção (ou pronto para)
- Documentação atualizada

Atividades:

- Deploy em produção
- Comunicação aos stakeholders
- Atualização de documentação
- Monitoramento inicial

Responsável: DevOps / Product Owner

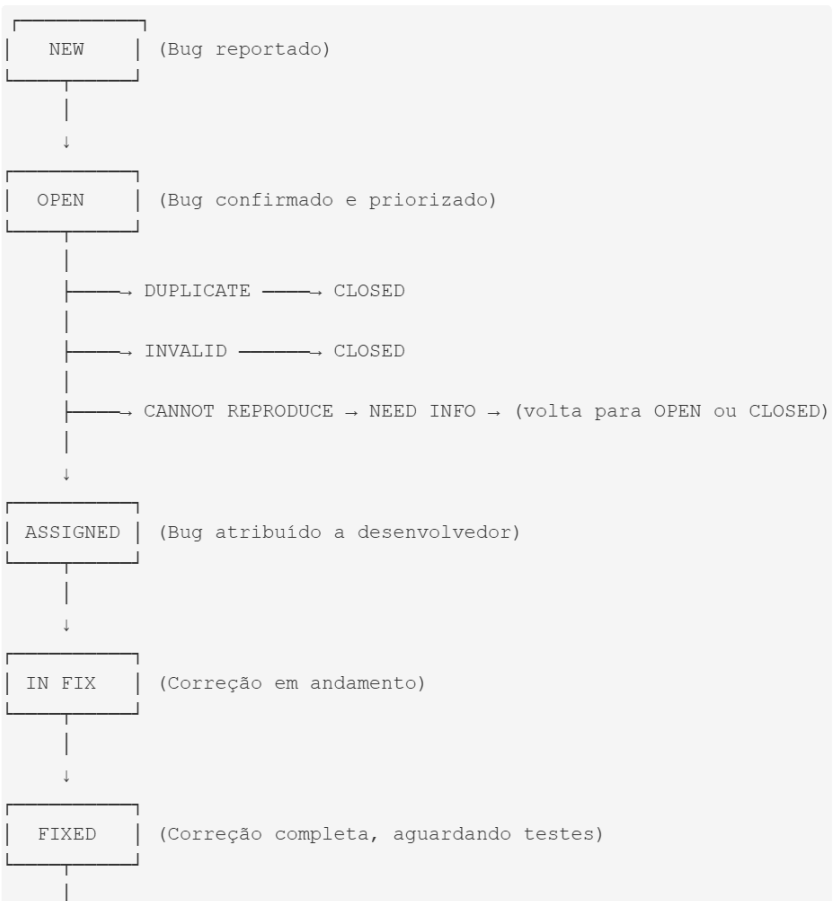
Definition of Done:

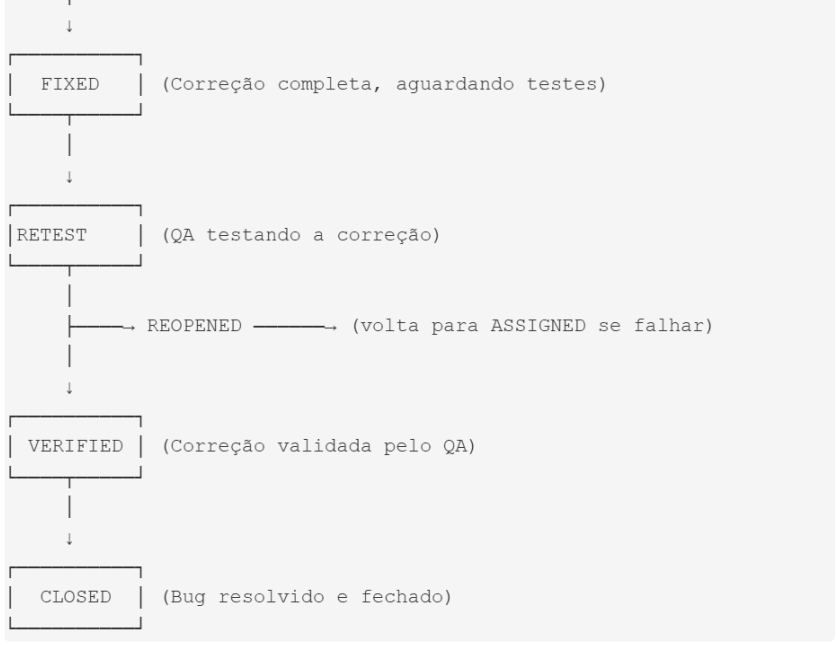
- ✅ Código desenvolvido e revisado
- ✅ Testes unitários criados e passando (cobertura > 80%)
- ✅ Testes de integração passando
- ✅ Testes funcionais aprovados pelo QA
- ✅ Product Owner aceitou a entrega
- ✅ Documentação técnica atualizada
- ✅ Deploy realizado em produção
- ✅ Sem bugs críticos ou bloqueadores
- ✅ Performance validada
- ✅ Segurança validada

---

2. Ciclo de Vida do Bug

2.1 Fluxo Completo do Bug





2.2 Descrição dos Status do Bug

2.2.1 NEW

Descrição: Bug recém-reportado, ainda não analisado.

Quem move: QA / Qualquer membro do time

Ações:

- Reportar bug no Jira
- Anexar evidências (screenshots, logs, vídeos)
- Descrever passos para reproduzir
- Identificar severidade e prioridade inicial

Próximos Status Possíveis: OPEN, DUPLICATE, INVALID

2.2.2 OPEN

Descrição: Bug confirmado como válido e aguardando correção.

Quem move: QA Lead / Tech Lead

Ações:

- Validar se o bug é reproduzível
- Confirmar severidade e prioridade
- Adicionar labels apropriadas
- Vincular a sprint se necessário

Informações Obrigatórias:

- Severidade (Blocker, Critical, Major, Minor, Trivial)
- Prioridade (Highest, High, Medium, Low, Lowest)
- Ambiente (Development, Staging, Production)
- Browser/Device (se aplicável)
- Versão do sistema

Próximos Status Possíveis: ASSIGNED, DUPLICATE, CANNOT REPRODUCE

- Browser/Device (se aplicável)
- Versão do sistema

Próximos Status Possíveis: ASSIGNED, DUPLICATE, CANNOT REPRODUCE

2.2.3 ASSIGNED

Descrição: Bug atribuído a um desenvolvedor para correção.

Quem move: Tech Lead / Scrum Master

Ações:

- Atribuir desenvolvedor
- Estimar esforço
- Adicionar a sprint se crítico
- Definir SLA de correção baseado na severidade

SLA por Severidade:

- Blocker: 4 horas
- Critical: 1 dia
- Major: 3 dias
- Minor: 1 semana
- Trivial: 2 semanas

Próximos Status Possíveis: IN FIX, CANNOT REPRODUCE

2.2.4 IN FIX

Descrição: Desenvolvedor trabalhando na correção.

Quem move: Developer

Ações:

- Analisar causa raiz
- Desenvolver correção
- Criar testes unitários
- Atualizar comentários no Jira
- Daily updates

Próximos Status Possíveis: FIXED

2.2.5 FIXED

Descrição: Correção completa, aguardando testes.

Quem move: Developer

Ações:

- Marcar como fixed no Jira
- Informar branch/commit da correção
- Atualizar comentários com detalhes
- Notificar QA para retesto

- Marcar como fixed no Jira
- Informar branch/commit da correção
- Atualizar comentários com detalhes
- Notificar QA para reteste
- Build deployado em ambiente de testes

Informações Obrigatórias:

- Commit hash
- Branch
- Ambiente disponível para teste
- Observações sobre a correção

Próximos Status Possíveis: RETEST

2.2.6 RETEST

Descrição: QA testando a correção do bug.

Quem move: Automático (quando developer marca como FIXED)

Ações:

- Executar cenários de teste originais
- Executar testes de regressão relacionados
- Validar se o bug foi corrigido
- Verificar se não há novos bugs (side effects)

Próximos Status Possíveis: VERIFIED, REOPENED

2.2.7 VERIFIED

Descrição: Correção validada pelo QA com sucesso.

Quem move: QA

Ações:

- Confirmar correção
- Anexar evidências do reteste
- Adicionar comentários finais
- Aprovação para produção

Próximos Status Possíveis: CLOSED

2.2.8 REOPENED

Descrição: Bug ainda existe após tentativa de correção.

Quem move: QA

Ações:

- Descrever por que o bug ainda existe
- Anexar novas evidências
- Atualizar severidade se mudou

- Descrever por que o bug ainda existe
- Anexar novas evidências
- Atualizar severidade se mudou
- Re-atribuir ao desenvolvedor

Próximos Status Possíveis: ASSIGNED, IN FIX

2.2.9 CLOSED

Descrição: Bug resolvido e validado em produção.

Quem move: QA / Product Owner

Motivos para Fechamento:

- **Fixed:** Bug corrigido e verificado
- **Duplicate:** Bug duplicado de outro
- **Invalid:** Não é um bug
- **Cannot Reproduce:** Impossível reproduzir
- **Won't Fix:** Decisão de não corrigir
- **Works as Designed:** Comportamento esperado

Ações:

- Adicionar resolution
- Comentário final
- Validação em produção (se fixed)
- Atualizar documentação se necessário

2.3 Status Especiais

DUPLICATE

Descrição: Bug já reportado anteriormente.

Ações:

- Linkar ao bug original
- Fechar como duplicate
- Consolidar informações no bug original

INVALID

Descrição: Não é um bug real.

Motivos:

- Erro do usuário
- Ambiente incorreto
- Dados de teste inválidos

CANNOT REPRODUCE

Descrição: Impossível reproduzir o bug.

CANNOT REPRODUCE

Descrição: Impossível reproduzir o bug.

Ações:

- Solicitar mais informações
- Tentar em diferentes ambientes
- Pode mover para NEED INFO

NEED INFO

Descrição: Faltam informações para analisar o bug.

Ações:

- Solicitar informações ao reporter
- Aguardar feedback
- Pode voltar para OPEN ou fechar como INVALID

---

3. Classificação de Bugs

3.1 Severidade

BLOCKER (Bloqueador)

- Sistema completamente indisponível
- Perda de dados
- Corrupção de dados
- Falha de segurança crítica
- Exemplo: Sistema não inicia, pagamentos não processam

CRITICAL (Crítico)

- Funcionalidade principal não funciona
- Sem workaround disponível
- Afeta muitos usuários
- Exemplo: Login não funciona, checkout falha

MAJOR (Maior)

- Funcionalidade importante não funciona
- Workaround disponível mas difícil
- Afeta alguns usuários
- Exemplo: Filtros de busca não funcionam

MINOR (Menor)

- Funcionalidade secundária não funciona
- Workaround fácil disponível
- Impacto limitado
- Exemplo: Tooltip incorreto, alinhamento errado

- Impacto limitado
- Exemplo: Tooltip incorreto, alinhamento errado

TRIVIAL (Trivial)

- Problemas cosméticos
- Não afeta funcionalidade
- Baixo impacto
- Exemplo: Erro de digitação, cor ligeiramente diferente

3.2 Prioridade

HIGHEST (Mais Alta)

- Precisa ser corrigido imediatamente
- Bloqueia release
- Afeta produção

HIGH (Alta)

- Deve ser corrigido na sprint atual
- Importante para o negócio

MEDIUM (Média)

- Pode ser corrigido nas próximas sprints
- Importante mas não urgente

LOW (Baixa)

- Pode ser corrigido quando houver tempo
- Nice to have

LOWEST (Mais Baixa)

- Backlog de melhorias
- Pode nunca ser corrigido

---

4. Template de Reporte de Bug

```
## Resumo
[Descrição concisa do bug]

## Ambiente
- **Versão:** 1.0.0
- **Ambiente:** Production / Staging / Development
- **Browser:** Chrome 119.0
- **Sistema Operacional:** Windows 11
- **Dispositivo:** Desktop / Mobile

## Severidade
[Blocker / Critical / Major / Minor / Trivial]

## Prioridade
[Highest / High / Medium / Low / Lowest]
```

```
## Severidade
[Blocker / Critical / Major / Minor / Trivial]

## Prioridade
[Highest / High / Medium / Low / Lowest]

## Pré-condições
- Usuário logado
- Carrinho com 2 itens
- [Outras condições necessárias]

## Passos para Reproduzir
1. Acessar a página X
2. Clicar no botão Y
3. Preencher campo Z com "valor"
4. Clicar em "Salvar"

## Resultado Atual
[O que está acontecendo - com evidências]

## Resultado Esperado
[O que deveria acontecer]

## Evidências
- Screenshot: [anexo]
- Vídeo: [link]
- Logs: [anexo]
- Network trace: [anexo]

## Informações Adicionais
- Frequência: Sempre / Às vezes / Raramente
- Impacto no usuário: [descrição]
- Workaround: [se houver]

## Relacionamentos
- Relacionado a: PROJ-123
- Bloqueia: PROJ-456
```

5. Métricas de Bugs

5.1 KPIs Importantes

Taxa de Detecção de Defeitos (DDR)

```
DDR = (Bugs encontrados no teste / Total de bugs) × 100
```

Taxa de Rejeição de Bugs

```
Taxa de Rejeição = (Bugs inválidos / Total de bugs reportados) × 100
```

Tempo Médio de Resolução

```
MTTR = Σ(Data Fechamento - Data Abertura) / Número de Bugs
```

Densidade de Defeitos

```
Densidade = Número de bugs / Tamanho do código (KLOC)
```

Taxa de Reopen

```
Taxa Reopen = (Bugs reabertos / Total de bugs fechados) × 100
```

Taxa de Reopen

```
Taxa Reopen = (Bugs reabertos / Total de bugs fechados) × 100
```

6. Boas Práticas

6.1 Para QA

✅ Reportar bugs assim que encontrados
✅ Sempre anexar evidências (screenshots, vídeos, logs)
✅ Descrever passos claros para reproduzir
✅ Verificar se não é duplicado antes de reportar
✅ Testar em múltiplos ambientes/browsers quando relevante
✅ Reteste completo após correção
✅ Validar side effects

6.2 Para Developers

✅ Atualizar status frequentemente
✅ Adicionar comentários técnicos úteis
✅ Incluir commit hash na correção
✅ Criar testes automatizados para evitar regressão
✅ Comunicar blockers imediatamente
✅ Pedir clarificações quando necessário

6.3 Para o Time

✅ Priorizar bugs críticos e bloqueadores
✅ Bug triage regular
✅ Comunicação clara e respeitosa
✅ Foco em prevenção, não apenas correção
✅ Análise de causa raiz para bugs recorrentes
✅ Retrospectivas sobre qualidade