Plano de Fluxo de Trabalho de Desenvolvimento e Ciclo de Vida do Bug

Sistema Bancário – Projeto QA

Autor: **Lucas A. Santos** | Versão: 1.0 | Data: 20/12/2025

1. Introdução

Este documento descreve o fluxo de trabalho de desenvolvimento e o ciclo de vida do bug, adotados no projeto de QA do sistema bancário (fictício) inspirado no Banco Bradesco.

Em sistemas financeiros, a definição clara de processos é essencial para garantir qualidade, confiabilidade e segurança das funcionalidades entregues ao usuário final.

A padronização desses dois processos permite melhor comunicação entre os membros do time, rastreabilidade das atividades e maior controle sobre a qualidade do produto ao longo de todo o ciclo de desenvolvimento.

2. Objetivos

O objetivo deste documento é apresentar:

- O fluxo de trabalho de desenvolvimento utilizado pelo time.
- As etapas do ciclo de vida de um bug.
- Os papéis envolvidos em cada fase do processo.
- A integração entre desenvolvimento e QA para garantir entregas com qualidade.

3. Overview do Projeto

O projeto simula um aplicativo bancário digital fictício, inspirado no Banco Bradesco, com foco nas seguintes funcionalidades:

- Autenticação de usuários (login e logout);

- Consulta de saldo e extrato;
- Transferência entre contas;
- Pagamento de contas;
- Gerenciamento de dados do perfil do usuário.

Este sistema será utilizado como base para a criação de User Stories, casos de teste, mind-map e definição de processos de QA.

> Observação: Este projeto não possui qualquer vínculo com o Banco Bradesco real e é destinado exclusivamente para fins de estudo e portfólio.

4. Tecnologias Utilizadas

As seguintes ferramentas são consideradas no contexto do projeto:

- **JIRA:** gerenciamento de User Stories, tarefas e bugs, permitindo rastreabilidade e controle do fluxo de trabalho.
- **Confluence:** centralização da documentação do projeto, requisitos, processos e decisões.
- **Git/GitHub:** versionamento e organização dos artefatos de QA.

Essas ferramentas simulam um ambiente real de trabalho adotado por equipes ágeis, em especial no método SCRUM, utilizado como inspiração neste trabalho.

5. Responsabilidades dos membros do Time

No projeto, são considerados os seguintes papéis:

- **Product Owner (PO):** responsável por definir e priorizar os requisitos do sistema.
- **Scrum Master (SM):** facilita o processo ágil e remove impedimentos.
- **Desenvolvedor (DEV):** implementa as funcionalidades do sistema.
- **QA/Testador:** planeja e executa os testes, além de registrar e acompanhar bugs.
- **Time de Desenvolvimento:** conjunto de profissionais responsáveis pela entrega da sprint.

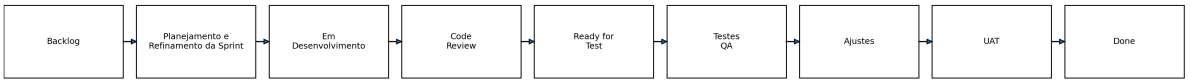
Cada papel contribui para que o processo seja colaborativo e focado na agilidade e qualidade.

6. Fluxo de Trabalho de Desenvolvimento

6.1 Visão Geral do Workflow

O fluxo de trabalho de desenvolvimento define as etapas que uma User Story percorre desde sua criação no backlog até sua entrega como funcionalidade pronta.

Fluxo resumido:



Esse fluxo garante organização, transparência e integração entre desenvolvimento e QA.

6.2 Descrição das Etapas do Workflow

1. Backlog

As User Stories são criadas e priorizadas no JIRA pelo Product Owner, com apoio do time. Detalhes e regras de negócio podem ser documentados no Confluence.

Status no JIRA: `To Do`.

---

2. Planejamento (+ Refinamento) da Sprint

Etapa unificada que combina planejamento e refinamento das histórias. As atividades ocorrem na seguinte ordem:

1. Histórias são selecionadas para a sprint.
2. O time revisa requisitos, critérios de aceite e riscos, enquanto o QA já antecipa cenários de teste.
3. São realizados ajustes nas histórias, quando necessário.
4. Definição de responsáveis pelas atividades.

Ao final, as histórias estão prontas para desenvolvimento.

Status no JIRA: `Selected for Development`.

---

3. Em Desenvolvimento

Os desenvolvedores implementam as funcionalidades conforme os requisitos definidos.

Status: `In Progress`.

---

4. Code Review

Outro desenvolvedor revisa o código implementado para garantir padrões de qualidade e boas práticas. Ajustes podem ser solicitados.

Status: `Code Review`.

---

5. Ready for Test

A funcionalidade é disponibilizada em ambiente de testes, ficando pronta para validação pelo QA.

Status: `Ready for Test`.

---

6. Testes de QA

O QA executa os casos de teste planejados, registra evidências e reporta defeitos no JIRA quando encontrados.

Status: `In Test`.

---

7. Ajustes

Os desenvolvedores corrigem os defeitos identificados pelo QA e retornam a funcionalidade para nova validação.

Status: `Bug Fix`.

---

8. Homologação (UAT)

O Product Owner valida se a funcionalidade atende às expectativas e critérios de aceite.

Status: `In UAT`.

---

9. Done

A funcionalidade é considerada pronta e apta para produção.

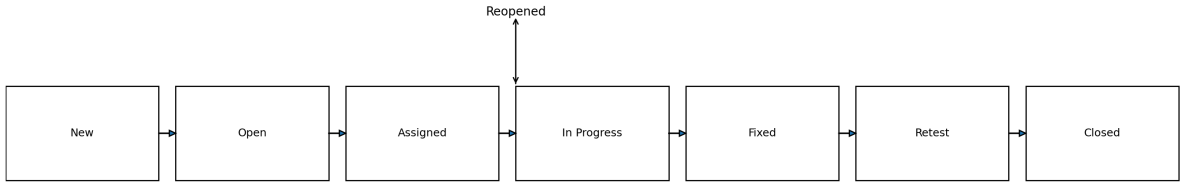
Status: `Done`.

7. Ciclo de Vida do Bug

7.1 Visão Geral

O ciclo de vida do bug descreve todas as etapas que um defeito percorre desde sua identificação até seu encerramento, garantindo controle e rastreabilidade.

Fluxo resumido:



Status possíveis: `Bug`, `Not a Bug`, `Duplicate`, `Won't Fix`, `Deferred`.

7.2 Descrição das Etapas do Ciclo do Bug

1. New

O bug é identificado pelo QA durante a execução dos testes e registrado no JIRA com descrição, passos para reprodução, resultado (esperado x atual) e evidências.

---

2. Open

O bug é analisado na triagem pelo time e validado como defeito legítimo.

---

3. Assigned

O bug é atribuído a um desenvolvedor responsável pela correção.

---

4. In Progress

O desenvolvedor trabalha na análise e correção do defeito.

---

5. Fixed / Resolved

A correção é finalizada e disponibilizada para reteste pelo QA.

---

6. Retest

O QA valida se o bug foi corrigido corretamente.

---

7. Reopened

Caso o problema persista, o bug é reaberto e retorna para correção (etapa 4).

---

8. Closed

Após validação positiva, o bug é encerrado.

8. Integração entre Workflow e Ciclo do Bug

Durante a etapa de **Testes de QA** do workflow de desenvolvimento, os defeitos encontrados são registrados no JIRA e passam a seguir o ciclo de vida do bug.

Uma User Story somente pode avançar para a etapa de **Homologação (UAT)** quando não houver bugs críticos ou bloqueadores abertos, garantindo que a funcionalidade esteja estável para validação de negócio.

9. Benefícios do Processo

A adoção deste fluxo de trabalho e ciclo de vida do bug proporciona:

- Maior organização e previsibilidade das entregas;
- Comunicação clara entre os membros do time;
- Rastreabilidade das atividades no JIRA;
- Centralização do conhecimento no Confluence;
- Redução de retrabalho e aumento da qualidade do produto.

10. Considerações Finais

Este documento apresenta um modelo de processo alinhado às boas práticas ágeis e de QA, adequado para projetos de sistemas críticos como aplicações bancárias. Sua aplicação contribui para entregas mais seguras, organizadas e orientadas à qualidade.

O conteúdo aqui descrito faz parte de um projeto fictício desenvolvido para fins educacionais e de demonstração de competências profissionais em Quality Assurance.