Setor de Qualidade - BIUD

Estrutura, Processos e KPIs

Rogério Lopes - QA Lead

Fevereiro 2026

Contexto e Objetivo

- Por que estruturamos Quality Assurance?
  - **Problema**: Expectativas desalinhadas, sem setor de qualidade, gaps de qualidade e conflitos sobre processos
  - **Solução**: Definir escopo claro de atuação do QA em todas as fases do ciclo de desenvolvimento
  - **Objetivo Estratégico**: Transformar percepção de qualidade em dados acionáveis e reduzir escape de defeitos para Produção
  - **Impacto**: Alinhamento com desenvolvimento, PO e stakeholders desde o refinamento até o deploy

Estrutura do Setor de QA

- Princípios de Atuação
  - **Shift-Left**: Testar e questionar o mais cedo possível (principalmente no refinamento);
  - **Qualidade Compartilhada**: QA facilita e valida; o time constrói qualidade;
  - **Transparência:** Status, riscos, evidências e bugs visíveis para o time;
  - **Reprodutibilidade**: Todo bug com passos claros, evidência e rastreabilidade (CT/ciclo/release).

Responsabilidades por Fase

- Atuação do QA no Ciclo de Desenvolvimento
  - Refinamento: Validar DoR, questionar critérios de aceite, identificar riscos e dependências;
  - Planejamento: Levantar dúvidas, explicitar critérios testáveis e estimar esforço de testes;
  - Durante Desenvolvimento: Acompanhar progresso, criar/atualizar casos de teste, preparar massa de dados;
  - Fase de Testes: Executar testes funcionais, regressão, integração, responsividade e segurança;
  - Homologação e Deploy: Validar deploy, executar smoke tests, apoiar decisão de go/no-go;
  - Pós-Deploy: Apoiar validação de hotfixes, triagem de incidentes e coleta de feedback;

Processos Implementados - DoR/DoD

- Definition of Ready (DoR) - Padrão de Entrada
  - Formato Padrão: Como [persona] / Quero [funcionalidade] / Para que [valor de negócio]
  - Critérios de Aceite: Testáveis e claros (formato Gherkin quando complexo)
  - Dependências Técnicas: APIs, bibliotecas, cards relacionados identificados
  - Checklist de Ready: História no formato padrão, critérios testáveis, dependências mapeadas, prioridade definida, anexos disponíveis
  - Benefício: Evita retrabalho e garante que histórias cheguem ao desenvolvimento prontas para execução

Gestão de Bugs e Sub-bugs

- Classificação por Ambiente e Governança
  - **Sub-bug:** Falhas encontradas em Dev ou Homologação (durante ciclo de desenvolvimento)
  - **Bug**: Falhas encontradas em Produção (pós-deploy / escape)
  - **Rastreabilidade Obrigatória**: Todo bug associado a Ciclo/Release e, quando possível, ao Caso de Teste
  - **Urgência Máxima**: Bugs bloqueadores ou hotfixes são prioridade absoluta com comunicação imediata
  - **Gestão de SLA**: Monitoramento via Dashboard para evitar tickets parados em estados intermediários

Tipos de Bugs

- Taxonomia para Rastreabilidade e Direcionamento
  - **Funcional**: Desvio de regra de negócio (foco: PO e Devs)
  - **Interface/UI**: Problemas de layout, design, textos e responsividade (foco: Frontend e Designers)
  - **Integração/API**: Falhas na troca de dados entre sistemas e microsserviços (foco: Backend e Integradores)
  - **Dados**: Problemas de persistência e integridade no banco (foco: DBAs e Backend)
  - **Segurança**: Vulnerabilidades e acessos não autorizados - OWASP (foco: DevSecOps e QA)
  - **Performance**: Lentidão, travamentos ou consumo excessivo de recursos (foco: Backend e Infra)
  - **Ambiente/Configuração**: Erros de variáveis, permissões ou pipeline de deploy (foco: DevOps/Infra)

Severidade vs Prioridade

- **Severidade (QA - Impacto Técnico)**
  - S1 - Bloqueador: Sistema inacessível ou funcionalidade vital interrompida, sem workaround
  - S2 - Crítico: Falha grave em regra de negócio principal ou perda de dados
  - S3 - Major: Falha importante mas com workaround manual possível
  - S4 - Minor: Erros de UI, ortografia ou mensagens confusas
- **Prioridade (PO - Urgência de Negócio)**
  - P0 - Urgente: Correção imediata (Hotfix no mesmo dia)
  - P1 - Alta: Correção na Sprint atual (antes do cut-off)
  - P2 - Média/Baixa: Correção planejada (Backlog para próximas Sprints)
  - Regra de Ouro: Nenhuma release aprovada para Produção com bugs S1/S2 ou P0/P1 abertos

KPIs de Qualidade

- 5 Indicadores Iniciais para Baseline (Sprint 30+)
  - **Escape Rate:** Quantidade de Bugs em Produção por sprint/release (mede eficácia e risco operacional)
  - **Densidade de Sub-bugs**: Quantidade por sprint e módulo (evidencia áreas instáveis)
  - **Lead Time de Correção**: Tempo entre abertura e fechamento de Sub-bugs (mostra velocidade de remoção de impedimentos)
  - **Taxa de Reabertura**: Percentual de Sub-bugs reabertos após correção (mede qualidade da correção)
  - **Distribuição por Tipo**: Percentual por tipo de bug (direciona investimento técnico e revela padrões)

Dashboard de Acompanhamento – em construção

- Como Medimos Qualidade
  - Acompanhamento Semanal: Leitura rápida do dashboard, identificação de tendências e gargalos;
  - Por Sprint (Retrospectiva): Resumo com Top 3 módulos mais instáveis, tipos predominantes de bugs e ações de melhoria contínua;
  - Resultado Esperado: Qualidade deixa de ser subjetiva e vira dado;
  - Módulos instáveis ficam evidentes e priorizáveis;
  - Redução progressiva de escape para Produção e menos retrabalho;

Fluxo Operacional de QA

- Jornada de uma História - Do Refinamento ao Deploy
  - 1. Refinamento: Validar DoR + critérios testáveis + riscos e dependências
  - 2. Planejamento: Alinhar escopo de testes e necessidades (dados/ambiente)
  - 3. Durante Desenvolvimento: Acompanhar, revisar testabilidade e preparar CTs/massa
  - 4. Deploy em Homologação: Confirmar versão/correção correta para ciclo/release
  - 5. Execução de Testes: Funcional + regressão + integrações conforme risco
  - 6. Registro de Bugs: Com severidade, prioridade, evidência e rastreabilidade
  - 7. Re-teste: Validar correções, evitar reabertura por falta de evidência
  - 8. Aprovação: Liberar ou bloquear conforme critérios e riscos

Resultados Esperados

- Impacto no Negócio e na Operação
  - **Redução de Escape para Produção**: Menos bugs chegando ao usuário final;
  - **Módulos Instáveis Evidentes:** Dados direcionam refatoração e hardening técnico;
  - **Menos Retrabalho**: Queda na taxa de reabertura e melhoria no lead time de correção;
  - **Decisões Baseadas em Dados**: Qualidade deixa de ser opinião e vira métrica acionável;
  - **Alinhamento de Expectativas**: Escopo claro de QA em todas as fases do desenvolvimento;
  - **Melhoria Contínua:** Retrospectivas com foco em ações de qualidade e prevenção.

Próximos Passos

- Evolução Contínua do Setor de QA
  - **Automação de Testes**: Priorizar testes repetitivos e regressões frequentes;
  - **Manutenção de Suíte Automatizada**: Evoluir em conjunto com time técnico;
  - **Definição de Metas e SLAs**: Após 2-3 sprints com baseline confiável;
  - **Revisão de Padrões**: Atualizar DoR/DoD, templates de bug e checklists conforme evolução do produto;
  - **Participação em Retrospectivas**: Foco em ações de qualidade e prevenção de regressões;
  - **Documentação Técnica**: Manter atualizada para facilitar onboarding e transferência de conhecimento.

Conclusão

- Qualidade como Dado, Não Opinião
  - **Estrutura Sólida**: QA atuando desde o refinamento até o pós-deploy
  - **Processos Claros**: DoR/DoD, gestão de bugs, severidade e prioridade bem definidos
  - **Métricas Acionáveis**: 5 KPIs iniciais para baseline e tomada de decisão
  - **Transparência Total:** Dashboard semanal e retrospectivas por sprint
  - **Foco no Negócio**: Reduzir escape para Produção e melhorar experiência do usuário

Qualidade é responsabilidade compartilhada. QA facilita, valida e mede. O time constrói.