Plano de Implementação da Cultura de Qualidade

---

1. Pilares da Cultura de Qualidade

1.1. Comunicação Transparente e Colaborativa

Objetivo: Garantir que toda a equipe (desenvolvedores, QAs, analistas, PO e Tech Lead) esteja alinhada e ciente dos objetivos de qualidade, compartilhando informações de forma clara, acessível e proativa.

1.2. Desenvolvimento Orientado à Qualidade

Objetivo: Incorporar a qualidade em todas as etapas do ciclo de desenvolvimento, desde a concepção da funcionalidade até a entrega final, com responsabilidade compartilhada entre todos os membros da equipe.

1.3. Testes Abrangentes e Contínuos

Objetivo: Estabelecer um processo de testes eficiente, que permita a detecção e correção de defeitos o mais cedo possível no ciclo de desenvolvimento, assegurando entregas com qualidade contínua.

---

2. Melhores Práticas por Pilar

2.1. Comunicação Transparente e Colaborativa

Nos projetos **ApiceChat** e **MyID**, a *daily* é o único rito adotado com regularidade. A seguir, práticas recomendadas:

- **Refinamento de Requisitos:**
  Detalhamento das tarefas, clareza nas funcionalidades e critérios de aceite. Promove entendimento compartilhado e alinhamento.

- **Daily (otimizada):**
  Atualização rápida de status, impedimentos e necessidades imediatas.

Evitar discussões técnicas longas.

- **Feedback One-on-One:**
  Reuniões individuais para troca de feedbacks construtivos, alinhamento de expectativas e desenvolvimento interpessoal.

- **Pair Testing:**
  Sessões colaborativas entre QA e devs durante o desenvolvimento.
  Promove aprendizado, eficiência e descoberta de cenários.

- **Documentação Clara e Acessível (Ex.: ApiceChat):**
  Documento compartilhado entre QA e analistas contendo:

    - Requisitos funcionais e regras de negócio
    - Critérios de aceite objetivos
    - Relato claro de incidentes
    - Tarefas bem descritas
    - Prototipações navegáveis, atualizadas em colaboração com UX

2.2. Desenvolvimento Orientado à Qualidade

Qualidade é responsabilidade de todos. Deve estar presente desde a ideação até a manutenção.

Práticas Recomendadas:

- **TDD (Test Driven Development):**
  Escrita de testes antes da implementação.

- **Code Review Contínuo:**
  Revisões constantes para:

    - Detectar falhas precocemente
    - Compartilhar conhecimento
    - Melhorar manutenibilidade

    - Promover boas práticas

2.3. Testes Abrangentes e Contínuos e Atuação do Time de QA

Responsabilidades e Práticas do QA:

- Estratégia de Testes
- Automação de Testes
- Gestão de Bugs
- Atuação como facilitador da qualidade

---

3. Templates e Ferramentas de Apoio

3.1. Checklist de Qualidade por Tarefa/Funcionalidade

Checklist - História de Usuário: [ID] - [Título]

Refinamento e Planejamento (PO, Dev, QA, TL):

- Critérios de aceitação definidos?
- Cenários de sucesso/falha (inclusive bordas)?
- Requisitos não funcionais discutidos?
- Dependências e impactos identificados?
- Estimativa de esforço de teste planejada?

Desenvolvimento (Dev):

- Testes unitários escritos e passando?
- Cobertura de testes adequada?

- Testes de integração feitos?
- Código revisado?
- Boas práticas seguidas?
- Débito técnico documentado?
- Impactos em outras áreas analisados?

Testes (QA):

- Plano de teste elaborado?
- Testes funcionais executados?
- Regressão validada?
- Testes exploratórios realizados?
- Usabilidade, performance, segurança testados (se aplicável)?
- Compatibilidade verificada?
- Dados de teste adequados?
- Bugs reportados e retestados?

Entrega (PO, Dev, QA, TL):

- Testes passaram no CI/CD?
- Homologação validada?
- Documentação atualizada?
- Histórico de bugs analisado?

Observações:
[Espaço livre]

3.2. Documento de Planejamento de Testes

Criado para cada nova funcionalidade ou release, contendo:

- Estratégia e escopo de testes
- Mapeamento de cenários
- Identificação de riscos

3.3. Documento Mensal de Status Report - Qualidade

Produzido mensalmente pelo QA, contendo:

- Atividades realizadas
- Métricas (bugs, MTTR, cobertura, etc.)
- Impedimentos e ações
- Recomendações para o time

3.4. Templates de Comunicação de Qualidade

- **Bug Report:**
  Título, passos, resultado esperado, ambiente, severidade, etc.
- **Sugestão de Melhoria:**
  Problema, sugestão, impacto.
- **Comentário de Aprovação/Reprovação:**
  Justificativa clara e objetiva.

---

4. Plano de Implementação e Monitoramento

Etapas de Implementação

1. **Apresentação e Alinhamento Inicial:**
   Apresentar plano, pilares e responsabilidades a toda a equipe.

2. **Reestruturação das Dailies:**
   Inserir status de testes, bugs e aprendizados.
   Fortalecer a voz do QA.

3. **Refinamento Contínuo com Foco em Qualidade:**
   Aplicar a técnica dos Três Amigos (QA, Dev, PO).
   Refinamentos sob demanda.

4. **Uso do Checklist de Qualidade:**
   Validação obrigatória entre devs e QA.

5. **Expansão da Automação Existente:**
   Garantir execução dos testes automatizados no CI/CD.

6. **Fomento a Testes Unitários e Code Reviews:**
   QA como mentor; tornar revisões obrigatórias.

7. **Formalização de Processos:**
   Aplicar templates e realizar análises de causa raiz.

8. **Status Report Mensal:**
   Produzido e compartilhado pelo QA com stakeholders.

Métricas de Qualidade e Monitoramento

- **Número de Bugs Reportados:**
  Por severidade e origem.

- **Tempo Médio de Correção (MTTR):**
  Por severidade.

- **Cobertura de Testes Automatizados:**
  Unitários e UI. Responsabilidade de devs e QA.

- **Falhas em Produção:**
  Quantidade, impacto, recorrência.

- **Feedback do Cliente:**
  Coletado via suporte, pesquisas, comentários.

- **Satisfação da Equipe:**
  Via formulários ou conversas abertas.

---

5. Revisão e Melhoria Contínua

- **Reuniões Periódicas de Qualidade:**
  Mensais ou bimestrais, com análise do status report e definição de próximos passos.

- **Documentação de Lições Aprendidas:**
  Compartilhadas em reuniões e repositórios.

- **Ajustes Baseados em Feedback:**
  Reavaliação contínua de processos e ferramentas.

- **Avaliação de Maturidade:**
  Identificar áreas para evolução (ex.: testes de performance, documentação).

---

6. Estratégia de Testes

Pirâmide de Testes:

Priorizar testes unitários e de integração (desenvolvedores), seguidos por API e UI (automatizados ou manuais pelo QA). O QA pode apoiar com orientação técnica.

Testes de Regressão:

Executados via pipeline automatizada com Cypress, garantindo estabilidade após novas entregas.

Testes Exploratórios:

Identificar falhas não cobertas por scripts. Alta relevância em contextos com pouca documentação.

Testes de Performance e Segurança:

Inclusão básica e progressiva desses testes, com apoio do QA na pesquisa e introdução de ferramentas leves.

Automação de Testes

- **Priorização:**
  Testes repetitivos e críticos automatizados primeiro.

- **Frameworks:**
  Cypress (UI). Avaliar ferramentas de API.

- **Manutenção:**
  Garantir confiabilidade contínua dos testes.
  CI deve alertar falhas automaticamente.