**PROJETO APP COMEDORIA DA TIA**

Testes e Qualidade de Software (DevOps)

**INTEGRANTES DO PROJETO e RA'S**

Cassio Gonçalves – 22023292
João Pedro – 24026658
Renan Damprelli – 24026623
Saulo Ribeiro - 24026911

São Paulo

2025

## Sumário

**1.** 4**2.** 4
**2.1.** 4
**2.2.** 7
**2.3.** 7
**2.4.** 8
**3.** 9
**3.1.** 9
**3.2.** 10
**3.3.** 10
**3.4.** 11
**4. REFERÊNCIAS BIBLIOGRÁFICAS** **5**

| Data das entregas | 22/set. | 10/nov. | 17/11 - 25/11 |
|---|---|---|---|
| **Disciplinas** | **Entrega 1 (3,0 Pontos)** | **Entrega 2 (4,0 Pontos)** | **Apresentação (3,0 Pontos)** |
| **Testes de Software** | **Instruções de Entrega:**<br>Qualidade de Software<br>**Descrição:**<br>O aluno deverá demonstrar o entendimento dos conceitos de qualidade de software, aplicando-os ao projeto. Isso inclui a criação de um diagrama do processo de qualidade de software, a identificação de atributos de qualidade.<br>Apresentar um relatório que explica como a norma de qualidade de software 25010 é utilizada no processo de desenvolvimento.<br>Entrega em documento pdv seguindo o template fornecido pelo professor. | **Instruções de Entrega:**<br>Testes Unitários, de Integração e de usuário.<br>**Descrição**:<br>Os alunos deverão desenvolver 4 testes unitários e 2 testes de integração e 1 teste de usuário. Esses testes devem garantir que o sistema esteja funcionando corretamente e validando os principais componentes e integrações e a geração de um relatório de teste de carga do servidor.<br>Entrega em documento pdv seguindo o template fornecido pelo professor. | **Critérios:** Criatividade, Impacto Social,Tempo de apresentação e Embasamento. |

# 1. INTRODUÇÃO

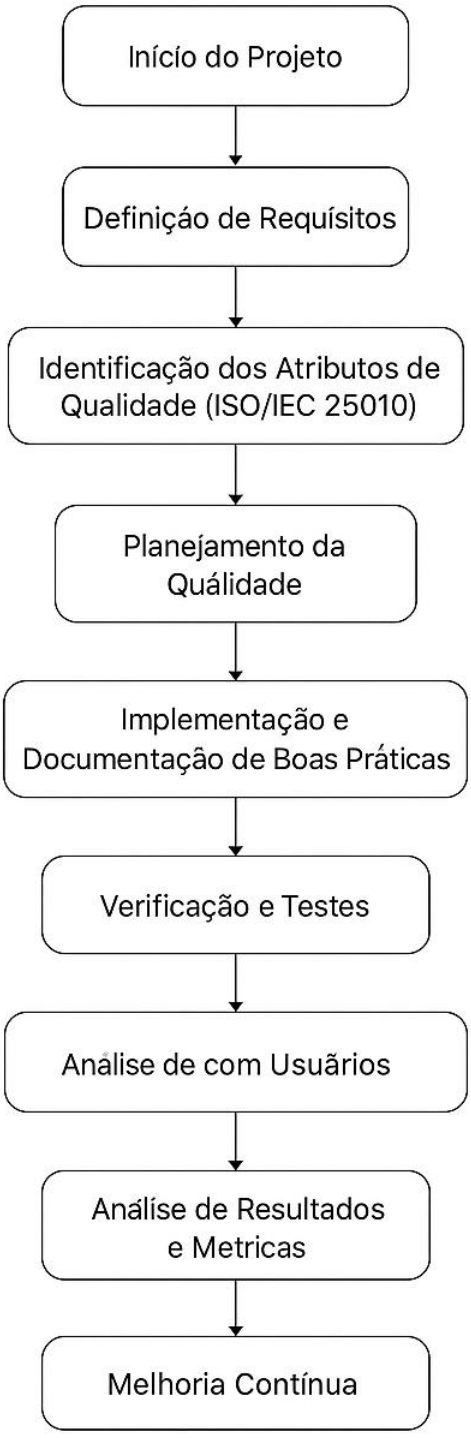
O presente relatório detalha a estratégia de gestão da qualidade para o desenvolvimento de um aplicativo para a Comedoria da Tia, que será projetado para a comunidade acadêmica da FECAP. A aplicação funcionará como uma plataforma de mobile commerce, permitindo que estudantes, professores e funcionários visualizem o cardápio da cantina, realizem pedidos, efetuem o pagamento de forma antecipada e retirem os produtos no balcão, um modelo conhecido como click and collect.

O principal problema que o app visa solucionar é a ineficiência operacional e a experiência negativa do cliente geradas por longas filas e tempos de espera, especialmente durante os horários de pico, como intervalos de aulas. Ao digitalizar e otimizar o processo de pedido e pagamento, o aplicativo busca não apenas melhorar a satisfação do usuário, mas também aumentar a eficiência e o volume de vendas da cantina. Os stakeholders do projeto incluem os usuários primários (estudantes com horários restritos), usuários secundários (professores e funcionários), a administração da cantina (responsável pela gestão de pedidos, estoque e finanças) e a administração da faculdade, que provê a infraestrutura e zela pela segurança dos dados da comunidade acadêmica.

# 2. Qualidade de Software

## 2.1. Modelo que qualidade de software (Diagrama/Design)

# Diagrama do Processo de Qualidade de Software (ISO/IEC 25010)




Fim do Processo

## 2.2. Processo (plano) de gerenciamento de qualidade de software (texto explicativo)

Para que a qualidade seja um resultado deliberado e consistente, é essencial a criação de um Plano de Gerenciamento da Qualidade de Software que servirá como o guia central para todas as atividades de qualidade do projeto, descrevendo os processos, padrões, ferramentas, métricas e responsabilidades que serão empregados para garantir que os níveis de qualidade definidos sejam alcançados.

**Definição de Padrões de Codificação:** A adesão a esses padrões melhora a legibilidade, a consistência e a colaboração, impactando diretamente a característica de **Manutenibilidade**.

**Revisões Técnicas Formais:** Será implementado um processo mandatório de revisão de código por pares para cada pull request. Esta é uma das práticas mais eficazes para a detecção precoce de defeitos lógicos, falhas de arquitetura e vulnerabilidades de segurança, além de promover o compartilhamento de conhecimento na equipe.

**Análise Estática de Código:** Ferramentas automatizadas que analisam o código-fonte em busca de bugs potenciais, vulnerabilidades de segurança e violações dos padrões de codificação definidos, fornecendo um feedback rápido ao desenvolvedor antes mesmo da fase de testes manuais.

**Auditorias de Processo:** Verificações periódicas serão realizadas para assegurar que a equipe de desenvolvimento está aderindo aos processos e padrões definidos no PGQS, como a correta documentação de requisitos e a execução das cerimônias ágeis planejadas.

## 2.3. Identificação de atributos de qualidade da norma 25010.

Embora todas as oito características de qualidade da ISO/IEC 25010 sejam relevantes, em um projeto com recursos e prazos finitos, é importante priorizar aquelas que exercem o maior impacto no sucesso do aplicativo. Esta priorização deve ser guiada pelos objetivos de negócio, pelas expectativas dos usuários e pelos principais riscos associados ao projeto.

**Usabilidade (Aprendizagem, Operabilidade, Proteção contra Erros do Usuário, Estética da Interface):** O público-alvo (estudantes) possui alta proficiência digital e baixa tolerância a interfaces confusas, esperando uma experiência fluida e intuitiva, similar a aplicativos comerciais de sucesso. O processo de pedido deve ser extremamente rápido para não anular o benefício de evitar a fila física.

**Eficiência de Desempenho (Comportamento Temporal, Capacidade):** O aplicativo enfrentará picos de carga extremos e de curta duração durante os intervalos de 20 minutos entre as aulas. A lentidão ou indisponibilidade nesses momentos críticos levará ao abandono imediato do serviço e à perda de confiança.

**Segurança (Confidencialidade, Integridade, Autenticidade):** O aplicativo processará transações financeiras (dados de cartão de crédito) e armazenará dados pessoais dos estudantes (nome,

matrícula, histórico de pedidos). Uma falha de segurança teria consequências graves, incluindo perdas financeiras, violação de leis de privacidade (LGPD) e danos à reputação da instituição.

**Confiabilidade (Disponibilidade, Tolerância a Falhas, Recuperabilidade):** O serviço deve estar operacional durante 100% do horário de funcionamento da cantina. Uma falha no sistema, mesmo que breve, significa perda direta de vendas e quebra da confiança do usuário, que pode não tentar usar o aplicativo novamente.

**Manutenibilidade (Modularidade, Analisabilidade, Testabilidade):** O ambiente de negócio de uma cantina é dinâmico: o cardápio muda sazonalmente, promoções são criadas, novos métodos de pagamento podem ser adicionados. O software precisa ser fácil e seguro de modificar e testar para se adaptar a essas mudanças de forma ágil e com baixo custo.

## 2.4. Relatório que explica como a norma de qualidade de software 25010 é utilizada no processo de desenvolvimento.

Detalhamento de como os atributos prioritários, definidos na seção anterior, serão incorporados e verificados desde a concepção até a manutenção do aplicativo, transformando a qualidade de um conceito abstrato em um resultado de engenharia tangível e contínuo.

**Tempo de Conclusão da Tarefa:** Um novo usuário deve ser capaz de selecionar um item, personalizar, adicionar ao carrinho e pagar em menos de 45 segundos (tempo estimado).

**Taxa de Erro:** Menos de 5% dos usuários devem cometer erros críticos durante os testes de usabilidade.

**Pontuação SUS (System Usability Scale):** Atingir uma pontuação média superior a 80 em questionários pós-uso.

**Tempo de Resposta da API:** 99% das requisições críticas devem ter um tempo de resposta inferior a 300ms (tempo estimado).

**Carga Suportada:** O sistema deve suportar 200 usuários concorrentes realizando pedidos com uma degradação de desempenho inferior a 15% em relação à linha de base.

**Conformidade:** Ausência de vulnerabilidades críticas e altas.

**Criptografia de Dados:** Todos os dados sensíveis devem ser criptografados em trânsito e em repouso.

**Disponibilidade):** O sistema deve manter uma disponibilidade superior a 99.9% durante o horário de funcionamento da cantina.

**MTTR (Mean Time to Recover):** O tempo médio para restaurar completamente o serviço após uma falha crítica deve ser inferior a 10 minutos (tempo estimado).

# 3. Teste de Software

## 3.1. Plano de Teste

O Plano de Teste é o documento que orienta todas as atividades de teste do projeto. Ele define o escopo, a abordagem, os recursos e o cronograma das atividades de teste.

**Objetivo:** Verificar se o aplicativo atende a todos os requisitos funcionais e não funcionais especificados, garantindo que o produto final seja robusto, seguro, eficiente e fácil de usar.

**Escopo dos Testes:**

**Funcionalidades em Escopo:**

- Cadastro e autenticação de usuário.
- Visualização do cardápio e detalhes dos produtos.
- Adição e remoção de itens no carrinho de compras.
- Personalização de pedidos.
- Aplicação de cupons de desconto.
- Processo de checkout e integração com gateway de pagamento.
- Histórico de pedidos.
- Notificações de status do pedido (e.g., "Pedido em preparo", "Pronto para retirada").

- **Funcionalidades Fora de Escopo (para a primeira versão):**
  - Programa de fidelidade.
  - Avaliação de produtos.

## 3.2. Apresentar 2 testes unitários.

**Exemplo 1: Teste da Função de Cálculo do Subtotal do Item**

- **Objetivo:** Verificar se a função `calcularSubtotalItem` retorna o valor correto ao multiplicar o preço do produto pela quantidade.
- **Componente Testado:** `funcao calcularSubtotalItem(produto, quantidade)`
- **Caso de Teste:**
  - **Entrada:** `produto = { nome: "X-Salada", preco: 15.50 }`, `quantidade = 2`
  - **Resultado Esperado:** `31.00`

**Exemplo 2: Teste da Função de Aplicação de Cupom de Desconto**

- **Objetivo:** Verificar se a função `aplicarCupom` calcula corretamente o novo total do carrinho após aplicar um cupom de desconto percentual.
- **Componente Testado:** `funcao aplicarCupom(totalCarrinho, cupom)`
- **Caso de Teste:**
  - **Entrada:** `totalCarrinho = 100.00`, `cupom = { codigo: "PROMO10", tipo: "percentual", valor: 10 }`
  - **Resultado Esperado:** `90.00`

## 3.3. Apresentar 2 testes de integração

**Exemplo 1: Integração entre Módulo de Pedidos e Módulo de Estoque**

- **Objetivo:** Garantir que um pedido não possa ser finalizado se um dos itens do carrinho não estiver mais disponível em estoque.
- **Módulos Envolvidos:** Serviço de Pedidos, Serviço de Estoque.
- **Cenário de Teste:**
  - O usuário adiciona o item "Suco de Laranja" (1 unidade em estoque) ao carrinho.
  - Outro processo (simulado pelo teste) remove a última unidade do "Suco de Laranja" do estoque.
  - O usuário tenta finalizar o pedido.
- **Resultado Esperado:** O sistema deve impedir a finalização do pedido e exibir uma mensagem ao usuário informando que o "Suco de Laranja" está esgotado. A API de Pedidos deve receber um erro do serviço de Estoque e tratar adequadamente.

**Exemplo 2: Integração entre Módulo de Checkout e Gateway de Pagamento**

- **Objetivo:** Verificar se, após uma transação de pagamento ser aprovada pelo gateway externo, o status do pedido no aplicativo é atualizado para "Pagamento Aprovado".
- **Módulos Envolvidos:** Serviço de Checkout, Gateway de Pagamento (simulado/mock).

- **Cenário de Teste:**
    - O usuário preenche os dados de pagamento e finaliza a compra.
    - O serviço de Checkout envia a requisição de pagamento para o Gateway de Pagamento.
    - O Gateway de Pagamento (simulado) retorna uma resposta de sucesso (transação aprovada).
- **Resultado Esperado:** O serviço de Checkout deve receber a resposta de sucesso, registrar a transação e atualizar o status do pedido no banco de dados para "Pagamento Aprovado".

## 3.4. Apresentar um teste de usuário (SISTEMA)

- **ID do Teste:** UAT-001
- **Título:** Realizar um pedido completo com sucesso utilizando cartão de crédito.
- **Resumo:** Este caso de teste simula o fluxo mais comum do aplicativo: um estudante faz login, escolhe seus produtos, paga e recebe a confirmação do pedido.

**Pré-condições:**

O usuário está cadastrado no sistema.
O aplicativo está instalado em um dispositivo de teste.
O produto "Salgado" (R$ 9,00) e "Refrigerante Lata" (R$ 6,00) estão disponíveis no cardápio.
O sistema de pagamento está configurado para aceitar um número de cartão de crédito de teste.

**Passos para Execução:**

Abra o aplicativo.
Faça login com o email e a senha correspondente.
Na tela principal, navegue até a seção "Lanches" e adicione 1 "Salgado" ao carrinho.
Navegue até a seção "Bebidas" e adicione 1 "Refrigerante Lata" ao carrinho.
Acesse o carrinho de compras.
Verifique se o total do pedido é R$ 15,00.
Clique em "Finalizar Pedido".
Na tela de pagamento, selecione a opção "Cartão de Crédito".
Insira os dados do cartão de crédito de teste.
Clique em "Pagar Agora".

**Resultado Esperado:**

O login deve ser bem-sucedido.
Os itens devem ser adicionados ao carrinho corretamente.
O total do pedido deve ser calculado corretamente.

Após clicar em "Pagar Agora", o sistema deve processar o pagamento e exibir uma tela de "Pedido Confirmado!".
A tela de confirmação deve exibir um número de pedido único.
O usuário deve poder visualizar este novo pedido em seu "Histórico de Pedidos" com o status "Em preparo".

## 4. REFERÊNCIAS BIBLIOGRÁFICAS

SOMMERVILLE, I. **Engenharia de Software.** 11ª Edição. São Paulo: Pearson Addison-Wesley, 2017.