**Projeto Interdisciplinar: Desenvolvimento e Publicação de Aplicativo Mobile**

**Requisitos da disciplina Teste Qualidade e DevOps**

São Paulo

2024

**INTEGRANTES DO PROJETO e RA’S**

* Kauê Dantas – 23025287
* Giovanne Braga – 23025648
* Isaac Santos – 23025417
* Icaro Luiz - 23025413

Sumário

[1. INTRODUÇÃO 3](#_Toc177380672)

[2. Teste de Software 3](#_Toc177380673)

[2.1. Plano de Teste 3](#_Toc177380674)

[2.2. 4 testes unitários. 3](#_Toc177380675)

[2.3. 2 testes de integração 3](#_Toc177380676)

[2.4. Teste de sistema. 3](#_Toc177380677)

[3. Qualidade de Software 3](#_Toc177380678)

[3.1. 4 atributos de qualidade de software e como foi aplicado no projeto integrador 3](#_Toc177380679)

[3.2. Modelo de qualidade de software 4](#_Toc177380680)

[3.3. Plano de gerenciamento de qualidade de software 4](#_Toc177380681)

[4. REFERÊNCIAS BIBLIOGRÁFICAS 4](#_Toc177380682)

# INTRODUÇÃO

Tendo em vista a necessidade crescente de educar financeiramente pessoas de baixa renda, o nosso projeto visa desenvolver um aplicativo móvel que facilite o aprendizado e a prática de conceitos financeiros básicos. O foco é seria a erradicação da pobreza por meio de estratégias de controle financeiro, metas de economia e uma orientação personalizada. O aplicativo é voltado para o público que enfrenta dificuldades em gerenciar suas finanças, oferecendo ferramentas de fácil acesso e compreensão.

# Teste de Software

# Plano de Teste

# O plano de teste estabelece a metodologia e os critérios utilizados para garantir que o aplicativo funcione conforme esperado e atenda aos requisitos especificados. A abordagem adotada inclui testes unitários, de componentes e de sistema, com foco em garantir que o aplicativo cumpra suas funções principais e ofereça uma experiência fluida e segura ao usuário.

# 4 testes unitários.

**Teste Unitário: Adicionar Despesa com Dados Válidos**

**Objetivo:** Verificar se uma despesa válida é adicionada corretamente à lista de transações.

**Cenário:** O usuário preenche os campos de descrição, valor e data corretamente e clica no botão "Salvar" para adicionar uma despesa.

**Entrada:**

Descrição: "Almoço"

Valor: "25.00"

Data: "01/10/2024"

**Saída Esperada:** A transação é adicionada à lista de transações com o tipo "despesa", e o valor é corretamente interpretado como 25.00.

**Classe Testada:** Despesa

**Método Testado:** adicionarDespesa()

**Teste Unitário: Adicionar Receita com Valor Inválido**

**Objetivo:** Verificar se o sistema trata corretamente uma entrada inválida de valor ao adicionar uma receita.

**Cenário:** O usuário tenta adicionar uma receita com um valor não numérico (inválido).

**Entrada:**

Descrição: "Salário"

Valor: "abc"

Data: "01/10/2024"

**Saída Esperada:** Nenhuma transação deve ser adicionada, e uma mensagem de erro deve ser registrada.

**Classe Testada:** Receita

**Método Testado:** adicionarReceita()

**Teste Unitário: Cálculo de Balanço**

**Objetivo:** Verificar se o balanço é calculado corretamente ao adicionar transações.

**Cenário:** O usuário adiciona uma despesa e uma receita, e o balanço deve ser atualizado corretamente.

**Entrada:**

Receita de R$1000.00

Despesa de R$200.00

**Saída Esperada:** O balanço final deve ser R$800.00.

**Classe Testada:** Session

**Método Testado:** calcularBalanco()

**Teste Unitário: Abrir e Fechar Drawer Layout**

**Objetivo:** Verificar se o menu do drawer (navegação lateral) abre e fecha corretamente quando o ícone é clicado.

**Cenário:** O usuário clica no ícone de menu lateral para abrir e fechar o drawer.

**Entrada:** Ação do clique no ícone.

**Saída Esperada:** O drawer abre quando está fechado, e fecha quando está aberto.

**Classe Testada:** Session

**Método Testado:** onClick(View v) (Relacionado ao ícone de navegação)

# 2 testes de integração

Nossos dois testes de integração serão realizados em nosso backend, porém o sistema de backend ainda está em desenvolvimento. Mas já explicando, esses dois testes serão de Teste de API com Postman e um teste de CRUD no banco de dados.

**2.4. Teste de sistema**

* **Teste de Sistema: Fluxo Completo de Gerenciamento Financeiro**
  + **Objetivo**: Testar o fluxo completo de uso do aplicativo, desde o login até a adição de receitas e despesas, incluindo o cálculo do balanço final.
  + **Cenário**: O usuário acessa o aplicativo, faz login, adiciona uma receita e uma despesa, e o saldo final deve ser exibido corretamente.
  + **Entrada**: Receita de R$1000.00 e Despesa de R$200.00.
  + **Saída Esperada**: O balanço final exibido na tela deve ser de R$800.00, e o fluxo do usuário deve ser concluído sem erros.
  + **Classe Testada**: Session, Despesa, Receita
  + **Métodos Testados**: login(), adicionarReceita(), adicionarDespesa(), calcularBalanco()

# Qualidade de Software

# 4 atributos de qualidade de software e como foi aplicado no projeto integrador

No desenvolvimento do nosso aplicativo, os seguintes atributos de qualidade de software foram aplicados:

1. **Usabilidade**: A interface do aplicativo foi projetada para ser simples e intuitiva e divertida, facilitando a navegação, mesmo para usuários com pouca experiência em tecnologia.
2. **Confiabilidade**: Foram realizados testes muito rigorosos para garantir que o aplicativo funciona corretamente em situações normais e anormais (como entradas inválidas).
3. **Eficiência**: O aplicativo foi otimizado para ter um bom desempenho em dispositivos móveis, consumindo poucos recursos e garantindo tempos de resposta rápidos.
4. **Segurança**: Ainda será implementado.

# Modelo de qualidade de software

O modelo utilizado foi baseado no ISO/IEC 25010, o qual destaca atributos essenciais de qualidade de software como funcionalidade, confiabilidade, usabilidade, eficiência e segurança. No projeto, houve um enfoque especial em usabilidade e confiabilidade, visando garantir uma experiência agradável e muito divertida ao usuário e uma confiança para que ele possa aprender muito bem como administrar seu dinheiro com nossas dicas.

# Plano de gerenciamento de qualidade de software

O gerenciamento da qualidade do projeto foi baseado no ciclo de melhoria contínua **PDCA (Plan-Do-Check-Act)**:

* **Planejar**: Definimos metas de qualidade focadas em usabilidade, confiabilidade e eficiência.
* **Executar**: Implementamos testes de unidade, de integração e de sistema para garantir que o código atenda aos critérios estabelecidos.
* **Verificar**: Realizamos revisões de código, testes automatizados e manuais para garantir que os requisitos de qualidade foram cumpridos.
* **Agir**: Aplicamos melhorias contínuas com base nos resultados dos testes e feedback dos usuários.

# 4. REFERÊNCIAS BIBLIOGRÁFICAS

<https://blog.onedaytesting.com.br/iso-iec-25010/>

<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

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