Planejamento de Testes: aplicativo para digitalização de notas fiscais paulistas e repasse de créditos gerados a instituições de caridade.

BELATO, Marcos Antonio
BISPO, Vinicius Henrique
TODESCO, Antero Sewaybricker

Centro Universitário Hermínio Ometto – FHO, Araras – SP, Brasil

1. Introdução

Este documento descreve o planejamento de testes para o projeto de TCC "Criação de um aplicativo para digitalização de notas fiscais paulistas e repasse de créditos gerados a instituições de caridade". O objetivo deste plano é definir a estratégia, os tipos de testes, os recursos necessários e os critérios para garantir a qualidade e a conformidade do aplicativo com os requisitos especificados.

2. Objetivos dos Testes

- Verificar se todas as funcionalidades do aplicativo estão operando conforme especificado.
- Garantir que a integração com a API da Nota Fiscal Paulista é funcional e segura.
- Assegurar que o aplicativo possui boa usabilidade e desempenho.
- Identificar e corrigir defeitos antes da entrega final do projeto.
- Validar se os requisitos não-funcionais são atendidos.

3. Escopo dos Testes

Funcionalidades a serem testadas:

- Cadastro e login de usuários.
- Escaneamento de notas fiscais via câmera do dispositivo.
- Extração de dados de notas fiscais (OCR).
- Validação dos dados da nota fiscal.
- Seleção de instituição de caridade.
- Transferência de créditos para instituições de caridade (simulação ou integração real, conforme escopo do TCC).
- Visualização de histórico de doações.
- Funcionalidades educativas sobre o programa Nota Fiscal Paulista e o impacto das doações.
- Listagem e busca de instituições de caridade.

Funcionalidades fora do escopo (Exemplo, ajustar conforme o projeto):

- Cadastro de novas instituições de caridade diretamente pelo usuário final (se for uma função administrativa).

- Testes de carga exaustivos simulando milhões de usuários (foco em testes funcionais e de usabilidade para o TCC).

4. Estratégia de Testes

A estratégia de testes será baseada em uma combinação de abordagens para cobrir diferentes aspectos do aplicativo.

4.1. Níveis de Teste

- **Testes de Unidade:** Foco em testar os menores componentes do software (módulos, funções, classes) de forma isolada. Serão realizados pelos desenvolvedores durante a codificação.
  - **Técnica:** Teste estrutural (caixa-branca), verificando a lógica interna do código, caminhos de execução, e manipulação de dados.
- **Testes de Integração:** Verificar a interação entre diferentes módulos e componentes do sistema, incluindo a integração com a API da Nota Fiscal Paulista e o sistema de OCR.
  - **Técnica:** Teste funcional (caixa-preta e caixa-cinza), focando nas interfaces entre os componentes.
- **Testes de Sistema:** Validar o sistema completo em relação aos requisitos funcionais e não-funcionais. Simular o ambiente de produção o mais próximo possível.
  - **Técnica:** Teste funcional (caixa-preta), baseado nos casos de uso e requisitos.
- **Testes de Aceitação:** Realizados com o objetivo de verificar se o sistema atende às necessidades dos usuários e aos critérios de aceitação definidos. Podem envolver usuários reais (ou colegas atuando como usuários) para feedback.
  - **Técnica:** Teste funcional (caixa-preta), focado na perspectiva do usuário.

4.2. Tipos de Teste

- **Testes Funcionais:**
  - **Objetivo:** Verificar se as funcionalidades do aplicativo se comportam conforme o esperado, de acordo com os requisitos definidos.
  - **Abordagem:** Baseada em casos de uso, histórias de usuário e especificações funcionais. Serão criados casos de teste para cada funcionalidade.
  - **Exemplos:** Testar o fluxo de escaneamento de nota, seleção de instituição, confirmação de doação, visualização de histórico.
- **Testes Estruturais (Caixa-Branca):**
  - **Objetivo:** Avaliar a estrutura interna do código, garantindo que todos os caminhos lógicos sejam cobertos.
  - **Abordagem:** Realizados principalmente durante os testes de unidade pelos desenvolvedores. Foco na cobertura de código (statements, branches, paths).
  - **Exemplos:** Verificar se todas as condições em uma função de validação de dados da nota fiscal são testadas.
- **Testes de Usabilidade:**
  - **Objetivo:** Avaliar a facilidade de uso, a intuitividade da interface e a satisfação do usuário.

    - **Abordagem:** Realização de testes com usuários (ou colegas), observando a interação com o aplicativo e coletando feedback através de questionários ou entrevistas.
    - **Critérios:** Facilidade de aprendizado, eficiência na execução de tarefas, clareza da interface, feedback do sistema.
- **Testes de Desempenho (Básico para TCC):**
    - **Objetivo:** Avaliar a responsividade do aplicativo em operações chave (ex: tempo para escanear e processar uma nota, tempo para carregar histórico).
    - **Abordagem:** Medição de tempos de resposta para operações críticas sob condições normais de uso.
- **Testes de Compatibilidade:**
    - **Objetivo:** Garantir que o aplicativo funcione corretamente em diferentes dispositivos Android (versões do SO, tamanhos de tela) definidos como alvo.
    - **Abordagem:** Execução dos principais casos de teste em uma variedade de emuladores e, se possível, dispositivos físicos.
- **Testes de Segurança (Básico para TCC):**
    - **Objetivo:** Identificar vulnerabilidades básicas, especialmente relacionadas à manipulação de dados sensíveis (dados da nota, dados do usuário) e à integração com a API da NFP.
    - **Abordagem:** Revisão de código para práticas seguras, teste de manipulação de entrada de dados (ex: SQL Injection básico, se aplicável), verificação de armazenamento seguro de credenciais.
- **Testes de Regressão:**
    - **Objetivo:** Garantir que as modificações ou correções de bugs não introduziram novos defeitos ou reativaram defeitos antigos em funcionalidades existentes.
    - **Abordagem:** Reexecução de um subconjunto de casos de teste previamente executados após cada alteração significativa no código ou correção de bug.

5. Plano de Testes para Requisitos Funcionais

ID RF	Requisito Funcional	Cenário de Teste Principal	Resultado Esperado	Prioridade
RF001	Escanear notas fiscais paulistas via câmera	1. Abrir a função de scanner. 2. Posicionar a câmera sobre uma nota fiscal válida. 3. Capturar a imagem da nota.	O aplicativo deve capturar a imagem da nota fiscal de forma clara e legível.	Alta
RF002	Extrair dados da nota fiscal (OCR)	1. Após escanear a nota, acionar a função	Os dados relevantes da nota fiscal	Alta

		de OCR. 2. Verificar os dados extraídos (CNPJ do estabelecimento, valor, data).	devem ser extraídos corretamente e exibidos para o usuário.	
RF003	Validar dados da nota fiscal	1. Tentar submeter uma nota com dados inválidos (ex: data futura, valor zerado). 2. Tentar submeter uma nota já processada.	O sistema deve identificar dados inválidos e informar o usuário. Não deve permitir o processamento de notas duplicadas.	Alta
RF004	Integrar com API da Nota Fiscal Paulista (NFP)	1. Submeter uma nota válida para processamento (simulado ou real). 2. Verificar o status da transferência dos créditos.	A comunicação com a API da NFP deve ser bem-sucedida e os créditos (simulados) devem ser processados.	Alta
RF005	Permitir ao usuário selecionar instituição de caridade	1. Acessar a lista de instituições. 2. Buscar uma instituição específica. 3. Selecionar a instituição desejada.	O usuário deve conseguir visualizar, buscar e selecionar uma instituição de caridade da lista.	Alta
RF006	Promover instituições de caridade dentro do app	1. Verificar se as informações das instituições (nome, descrição, causa) são exibidas corretamente.	As informações das instituições devem ser claras e acessíveis.	Média
RF007	Desenvolver	1. Acessar a	O conteúdo	Média

	funcionalidades educativas	seção educativa. 2. Verificar se o conteúdo sobre a NFP e o impacto das doações é apresentado.	educativo deve ser informativo e de fácil compreensão.	
RF008	Permitir acompanhamento do histórico de doações	1. Realizar uma doação (simulada). 2. Acessar o histórico de doações.	A doação realizada deve constar no histórico com os detalhes corretos (instituição, valor, data).	Alta
RF009	Cadastro e Login de Usuários	1. Tentar criar uma nova conta com dados válidos. 2. Tentar fazer login com credenciais corretas e incorretas.	O usuário deve conseguir se cadastrar e fazer login com sucesso. Falhas de login devem ser tratadas adequadamente.	Alta
RF010	Recuperação de Senha	1. Solicitar a recuperação de senha para um email cadastrado. 2. Seguir o fluxo de redefinição de senha.	O usuário deve conseguir redefinir sua senha de forma segura.	Média

6. Plano de Testes e Critérios de Aceitação para Requisitos Não-Funcionais

ID RNF	**Requisito Não-Funcional**	**Teste Proposto**	**Critério de Aceitação**	**Prioridade**
RNF001	Usabilidade	1. Teste de "pensar alto" com 3-5 usuários (colegas). 2.	- 80% dos usuários devem conseguir completar as tarefas chave	Alta

		Avaliar tempo para completar tarefas chave (escanear e doar). 3. Questionário de satisfação (SUS ou similar simplificado).	sem ajuda. - Tempo médio para escanear e doar uma nota < X segundos (definir X). - Interface considerada intuitiva pela maioria dos testadores.	
RNF002	Desempenho	1. Medir tempo de resposta para: abertura do scanner, processamento OCR, carregamento da lista de instituições, carregamento do histórico.	- Abertura do scanner < 2 segundos. - Processamento OCR < 5 segundos (para uma nota nítida). - Carregamento de listas < 3 segundos.	Alta
RNF003	Segurança	1. Testar entradas de dados para evitar XSS básico ou injeção (se houver campos de texto livre). 2. Verificar se a comunicação com a API NFP usa HTTPS. 3. Verificar armazenamento de senhas (se hashed).	- Ausência de vulnerabilidades básicas identificadas. - Comunicação com APIs externas deve ser criptografada. - Senhas de usuário armazenadas com hash seguro.	Alta
RNF004	Confiabilidade	1. Executar fluxos críticos múltiplas vezes (ex: escanear 10 notas diferentes). 2.	- Taxa de sucesso do OCR > 90% para notas legíveis. - Aplicativo não deve apresentar	Alta

		Testar processamento de notas com diferentes qualidades de imagem (dentro do razoável).	falhas (crashes) durante os fluxos principais em X horas de teste.	
RNF005	Compatibilidade	1. Executar os principais casos de teste em 2-3 versões diferentes do Android (ex: Android 10, 12, 14). 2. Testar em 2-3 resoluções de tela diferentes (emuladores).	- O aplicativo deve ser funcional e a interface deve se adaptar corretamente nas versões e resoluções de tela testadas, sem quebras de layout significativas.	Média
RNF006	Acessibilidade (Básica)	1. Verificar contraste de cores. 2. Verificar se os principais botões e campos são identificáveis por leitores de tela (usando ferramentas de desenvolvedor do Android).	- Contraste adequado entre texto e fundo. - Elementos interativos principais devem ter descrições para leitores de tela (contentDescription).	Baixa/Média (Conforme foco do TCC)
RNF007	Transparência	1. Verificar se o histórico de doações é claro e exibe todas as informações relevantes (instituição, data, valor/status do crédito).	- O usuário deve conseguir entender facilmente para onde e quando seus créditos foram direcionados através do histórico.	Alta

7. Recursos de Teste

- **Hardware:** Dispositivos móveis Android (físicos ou emuladores) com câmera.
- **Software:** Android Studio (para emuladores e debug), ferramentas de design para protótipos (Figma/Adobe XD, se usados para referência), possivelmente ferramentas para logging e monitoramento básico.
- **Humanos:** Os próprios desenvolvedores do TCC. Colegas ou outros estudantes para testes de usabilidade e aceitação.
- **Dados de Teste:** Amostras de notas fiscais paulistas (reais ou simuladas com dados válidos e inválidos). Lista de instituições de caridade fictícias ou reais (com permissão, se necessário).

8. Cronograma de Testes (Exemplo)

- **Testes de Unidade:** Contínuo, durante todo o ciclo de desenvolvimento de cada módulo.
- **Testes de Integração:** Após a finalização de módulos que interagem entre si (ex: Módulo Scanner + Módulo OCR; Módulo de Doação + API NFP).
- **Testes de Sistema:** Semanas X-Y (definir conforme o cronograma do TCC, geralmente nas fases finais de desenvolvimento).
- **Testes de Aceitação:** Semana Z (após os testes de sistema e correções).
- **Testes de Regressão:** Realizados continuamente após cada build/correção.

9. Critérios de Entrada e Saída

- **Critérios de Entrada para Testes de Sistema:**
  - Todos os módulos principais desenvolvidos e com testes de unidade aprovados.
  - Ambiente de teste configurado.
  - Casos de teste preparados e revisados.
- **Critérios de Saída (Conclusão dos Testes):**
  - Todos os casos de teste de prioridade Alta e Média executados.
  - Nenhum defeito crítico (bloqueador) em aberto.
  - Número de defeitos de prioridade Alta em aberto abaixo de um limite definido (ex: < 3).
  - Cobertura de requisitos funcionais atingida (ex: 100% dos requisitos cobertos por casos de teste).
  - Relatório final de testes aprovado.

10. Entregáveis dos Testes

- Plano de Testes (este documento).
- Casos de Teste (detalhados para cada funcionalidade e requisito não-funcional).
- Relatórios de Execução de Testes (com status de cada caso de teste).
- Relatório de Bugs/Defeitos (com descrição, passos para reproduzir, severidade, prioridade e status).
- Relatório Final de Testes (sumarizando os resultados, defeitos encontrados, cobertura e avaliação geral da qualidade).

11. Gestão de Defeitos

- **Registro:** Todos os defeitos encontrados serão registrados em uma planilha ou ferramenta

simples, contendo: ID, descrição, passos para reproduzir, severidade (Crítico, Alto, Médio, Baixo), prioridade, responsável, status (Aberto, Em Correção, Corrigido, Fechado, Reaberto).

- **Correção:** Os desenvolvedores analisarão e corrigirão os defeitos conforme a prioridade.
- **Verificação:** Após a correção, o testador verificará se o defeito foi resolvido e se não houve impacto em outras áreas (teste de regressão).