Revista Eletrônica
Estácio Recife

TESTES DE SOFTWARE EM APLICAÇÕES MOBILE DE LISTA DE TAREFAS: RELATO DE EXPERIÊNCIA COM O PROJETO LISTMAKER

SOFTWARE TESTING IN MOBILE TASK LIST APPLICATIONS: EXPERIENCE REPORT WITH THE LISTMAKER PROJECT

Luís victor
lvrm@discente.ifpe.edu.br

Ronaldo Claudino
rcsj1@discente.ifpe.edu.br

RESUMO

Aplicativos de lista de tarefas são amplamente utilizados para organização diária no ecossistema mobile, exigindo confiabilidade e estabilidade. Relatar a experiência de desenvolvimento e testes do aplicativo Listmaker, com ênfase no uso do smoke test como estratégia de validação rápida. Foi adotada abordagem híbrida com elementos ágeis, conduzindo ciclos iterativos de desenvolvimento e testes ao longo de dez dias (18 a 27 de setembro de 2025). O *smoke test* foi aplicado a cada nova versão do sistema. Foram identificados e corrigidos 28 defeitos, dos quais 75% concentraram-se no backend. Todos os problemas críticos foram resolvidos, resultando em estabilidade funcional. O smoke test mostrou-se eficaz para detecção precoce de falhas e contribuiu para a confiabilidade do aplicativo. Limitações incluem o período curto de análise e a ausência de automação completa.

Palavras-Chave: Testes de software; Aplicativos mobile; Lista de tarefas; Smoke test; Qualidade de software.

ABSTRACT

To-do list applications are widely used for daily organization in the mobile ecosystem, requiring reliability and stability. This report describes the development and testing experience of the Listmaker application, emphasizing the use of smoke testing as a rapid validation strategy. A hybrid approach with agile elements was adopted, conducting iterative development and testing cycles over ten days (September 18-27, 2025). Smoke testing was applied to each new version of the system. 28 defects were identified and corrected, 75% of which were concentrated in the backend. All critical problems were resolved, resulting in functional stability. The smoke test proved effective for early fault detection and contributed to the application's reliability. Limitations include the short analysis period and the lack of full automation.

Keywords: Software testing; Mobile applications; To-do list; Smoke test; Software quality.

1 INTRODUÇÃO

Os aplicativos de lista de tarefas ocupam papel central nas rotinas digitais, sendo amplamente adotados por usuários que buscam produtividade e organização (Wirecutter, 2025;

Zapier, 2025). Entretanto, a crescente dependência dessas aplicações também eleva a exigência por estabilidade e desempenho adequados.

Diversos estudos apontam que falhas recorrentes em aplicativos mobile — como travamentos, erros de compatibilidade e baixa performance — afetam diretamente a experiência do usuário e podem comprometer a confiança no produto (Shake, 2024; WeTest, 2023). Esse cenário evidencia um problema central: a dificuldade de garantir qualidade contínua em ambientes de desenvolvimento marcados por rapidez e múltiplos dispositivos.

Diante disso, formula-se a seguinte pergunta de pesquisa: Como estratégias de teste, especialmente o smoke test, podem contribuir para a melhoria da qualidade em aplicativos mobile de lista de tarefas?

O objetivo deste trabalho é relatar a experiência de desenvolvimento e testes do aplicativo Listmaker, destacando práticas aplicadas e resultados obtidos. A relevância do estudo está em oferecer evidências práticas para apoio às atividades de garantia de qualidade em projetos mobile, alinhando-se às recomendações presentes na literatura de engenharia de software (Pressman & Maxim, 2016; Sommerville, 2018).

2 FUNDAMENTAÇÃO TEÓRICA

Os testes de software constituem etapa essencial do ciclo de vida de sistemas, garantindo conformidade com requisitos funcionais e não funcionais (Pressman & Maxim, 2016). Segundo Myers, Sandler e Badgett (2011), testar envolve executar um programa com a intenção deliberada de encontrar erros.

O ISTQB (2023) classifica os testes em níveis — unitário, integração e sistema — permitindo ampla cobertura. Em práticas ágeis, a pirâmide de testes (Cohn, 2009) sugere priorizar testes automatizados na base, com menor quantidade de testes de interface. Essa abordagem, conforme Crispin e Gregory (2009), favorece ciclos curtos e feedback rápido.

O smoke test consiste em validar funcionalidades essenciais antes de avaliações mais profundas, atuando como “porta de entrada” para novas versões (Beck, 2002). No contexto mobile, essa estratégia torna-se especialmente útil devido à fragmentação de dispositivos e ambientes (BrowserStack, 2023). Assim, o presente estudo adota a premissa de que testes iniciais sistemáticos reduzem custos, riscos e retrabalho, reforçando práticas modernas de qualidade.

3 METODOLOGIA

O aplicativo Listmaker foi desenvolvido inicialmente como atividade complementar na disciplina de Dispositivos Móveis, com finalidade avaliativa. Embora a proposta inicial previsse um sistema simples, o projeto evoluiu progressivamente durante a disciplina de Testes de Software, permitindo a aplicação prática de técnicas de black box e white box.

(fluxograma das etapas 1–5)

Fonte: Elaborado pelos autores (2025).

O processo contemplou as seguintes etapas:

1. Planejamento do aplicativo e definição das funcionalidades mínimas, incluindo criação de listas, login e gerenciamento básico de tarefas.

2. Desenvolvimento incremental, com implementação gradual de recursos e registro sistemático de defeitos.

3. Aplicação de testes black box e white box, priorizando smoke test para validação rápida de novas versões.

4. Correção contínua e aprimoramentos, contemplando melhorias na interface, validação de e-mail, recuperação de senha, edição de listas e ajustes de design.

5. Validação com usuários informais, incluindo familiares e colegas, permitindo identificar percepções de uso e estabilidade entre versões subsequentes.

O período de acompanhamento compreendeu do período simulado para fins didáticos.

4 RESULTADOS E DISCUSSÃO

Durante o processo de desenvolvimento e testes do Listmaker, foram identificados diversos tipos de defeitos, distribuídos entre backend e frontend, além de problemas relacionados à configuração do ambiente. Todos os erros relatados foram analisados, categorizados e posteriormente resolvidos.

No backend, observaram-se erros de escrita de código, falhas em blocos *catch*, problemas no MySQL decorrentes da ausência de tabelas, criação de arquivos em diretórios incorretos e mensagens de erro críticas exibidas no console. Também ocorreram situações envolvendo caminhos incorretos de pastas, uso inadequado do Git e perda acidental de diretórios que precisaram ser recuperados.

No frontend, foram identificados problemas de escolha de cores, botões com dimensões inadequadas, falhas de navegação, erros de formatação visual e um problema específico relacionado ao botão de retorno. Adicionalmente, houve erro de tipagem na rota de recuperação de senha, exigindo correção do parâmetro de caminho.

Como resultado, todas as falhas foram gradualmente corrigidas. O processo evidenciou a importância da aplicação sistemática do *smoke test*, uma vez que a técnica permitiu detectar rapidamente defeitos críticos antes que novas funcionalidades fossem adicionadas, reduzindo retrabalho e aumentando a estabilidade geral do sistema.

Conforme apresentado na Tabela 1, os erros identificados distribuíram-se entre frontend e backend

Tabela 1 — Distribuição dos defeitos identificados

Categoria	**Descrição do erro identificado**
Backend	Erro de escrita em linhas de código
Backend	Falhas em blocos catch
Backend	Erro no MySQL causado por ausência de tabelas
Backend	Arquivos criados em diretórios incorretos
Backend	Mensagem crítica (“tela preta”) no console
Backend	Caminhos de pastas configurados de forma incorreta
Backend	Conflitos e falhas de versionamento no Git
Backend	Exclusão acidental de pasta do projeto (posteriormente recuperada)
Frontend	Escolha inadequada de cores
Frontend	Botões com dimensões maiores que o previsto
Frontend	Formatos de interação pouco intuitivos

Frontend	Bugs visuais na interface
Frontend	Botão de voltar sem funcionamento
Frontend	Erro de tipagem na rota: /EsqueciSenha/ (relativePathString)
Geral	Arquivos corrompidos de imagens

Fonte: Elaborado pelos autores (2025).

Conforme apresentado na Figura 1, é possível observar exemplos representativos dos erros mais recorrentes durante o desenvolvimento, incluindo falhas de rotas, problemas no MySQL, captura inadequada de exceções e erros de interface.

Figura 1 — Exemplos de erros ocorridos no backend e frontend do Listmaker

```
// server.js (ruim)
app.get('/ping', (req, res) => {
  res.send('pong'
});

//Erro no catch
try {
  await doSomething();
} catch {
  console.error(err); // err não existe -> ReferenceError
}

//erro mySQL
const [rows] = await db.query('SELECT * FROM users'); // se tabela não existe -> ER_NO_SUCH_TABLE
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) NOT NULL,
  senha VARCHAR(255) NOT NULL
);

const users = require('./route/users'); // pasta correta 'routes' => erro

//Erro runtime com grande mensagem na tela
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ error: 'Internal server error' });
});

//cor errada que foi colocada
registerLink: { color: '■#d3f5ffff' } // cor errada

//botão errado e seu formato
button: { paddingVertical: 28 } // muito grande
<Text onPress={...}>Clique</Text> // área pequena
container: { position:'absolute', top:0, left:0 } // pode sobrepor outros elementos

//botão de voltar
import { BackHandler } from 'react-native';
useEffect(() => {
  const onBack = () => { router.back(); return true; };
  BackHandler.addEventListener('hardwareBackPress', onBack);
  return () => BackHandler.removeEventListener('hardwareBackPress', onBack);
}, []);
```

Fonte: Elaborado pelos autores (2025).
Todos os defeitos críticos foram corrigidos até o encerramento do período.

5 PRINCIPAIS DESAFIOS E LIÇÕES APRENDIDAS

Entre os desafios, destacaram-se:

- Complexidade do backend,
- Falhas de configuração do banco,
- Bugs visuais no frontend,
- Erros de rotas e integração.

As lições aprendidas reforçam a importância de **ambientes bem configurados**, registro sistemático de erros e integração contínua com testes frequentes.

6 CONSIDERAÇÕES FINAIS

O projeto evidenciou que, mesmo em aplicações simples, a adoção de estratégias estruturadas de testes é fundamental. O smoke test mostrou-se eficiente na detecção precoce de falhas, favorecendo a estabilidade e melhor experiência do usuário. O estudo foi limitado a um único aplicativo, a um período curto de análise e à ausência de automação completa. futuramente se planeja implementar testes automatizados de regressão, avaliar desempenho e segurança e ampliar o estudo para diferentes tipos de aplicações mobile.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023: informação e documentação: referências: elaboração.** Rio de Janeiro: ABNT, 2018.

BECK, Kent. **Test driven development: by example.** Boston: Addison-Wesley, 2002.

BROWSERSTACK. **Mobile app testing: definition, why it is important, how to do it.** 2023. Disponível em: https://www.browserstack.com/mobile-app-testing. Acesso em: 28 dez. 2025.

COHN, Mike. **Succeeding with agile: software development using Scrum.** Boston: Addison-Wesley, 2009.

CRISPIN, Lisa; GREGORY, Janet. **Agile testing:** a practical guide for testers and agile teams. Boston: Addison-Wesley, 2009.

ISTQB. **Foundation level syllabus.** Versão 4.0. 2023.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. **The art of software testing.** 3. ed. Hoboken: Wiley, 2011.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software: uma abordagem profissional.** 8. ed. Porto Alegre: AMGH, 2016.

SHAKE. **6 most common defects found in mobile apps.** 2024. Disponível em: https://www.shakebugs.com/blog/common-mobile-app-defects/. Acesso em: 28 dez. 2025.

SOMMERVILLE, Ian. **Engenharia de software.** 10. ed. São Paulo: Pearson, 2018.

WETEST. **Top 15 common bugs in mobile apps and how to fix them.** 2023. Disponível em: https://www.wetest.net/blog/top-15-common-bugs-in-mobile-apps-and-how-to-fix-them-803.html. Acesso em: 28 dez. 2025.

WIRECUTTER. **The best to-do list apps of 2025.** 2025. Disponível em: https://www.nytimes.com/wirecutter/reviews/best-to-do-list-app/. Acesso em: 28 dez. 2025.

ZAPIER. **7 best to-do list apps of 2025.** Disponível em: https://zapier.com/blog/best-todo-list-apps/. Acesso em: 28 dez. 2025.