# Plano de Testes

# Tap&Send

Grupo24:

Regina da Paz

Bruna Fernandes

Ali Kaan Ersoy

# Índice

1. Introdução ..... 3

2. Objetivos dos testes ..... 3

3. Estratégias e Abordagem de Testes ..... 3

3.1. Estratégias: ..... 3

3.2. Abordagens ..... 4

4. Condições para testes ..... 4

6. Critérios de paragem ..... 4

7. Taxa de Cobertura de Testes ..... 5

8. Riscos e Mitigações ..... 5

# 1. Introdução

Este documento apresenta um plano de testes para a Tap&Send, baseado no documento dos requisitos funcionais (RF-001 a RF—013) e não funcionais (RNF-001 e RNF-002). O plano foi elaborado com foco nos componentes core da plataforma, uma vez que estes têm impacto diretamente na usabilidade, segurança e eficiência do sistema, estes componentes incluem funcionalidades de alta prioridade, como criar encomenda, rastrear encomendas em tempo real, atualização do estado das encomendas e gestão da plataforma por parte do administrador.

Por fim, este plano tem por objetivo garantir que o software seja robusto, confiável, detalhando o que será testado (funcionalidades, fluxos e cenários) durante a fase de desenvolvimento, priorizando a deteção de falhas.

# 2. Objetivos dos testes

Os testes visam:

- Verificar a conformidade com os requisitos, garantindo que os componentes core funcionem conforme descrito (ex.: validação de dados em registo, rastreamento preciso em tempo real).
- Identificar defeitos em fluxos principais e secundários, como erros de validação, falhas de integração com GPS ou notificações não enviadas.
- Avaliar a usabilidade e desempenho não funcional, como layout intuitivo e design responsivo.
- Assegurar segurança e robustez, testando restrições como autenticação prévia e dependências externas (internet/GPS).
- Maximizar a cobertura para reduzir riscos em produção, focando em componentes core que representam 80% da funcionalidade crítica.

# 3. Estratégias e Abordagem de Testes

## 3.1. Estratégias:

- **Testes Unitários**: Verificam funções isoladas, como validação de email em RF-001.

- **Testes de Integração:** Avaliam interações entre módulos, ex.: integração de RF-003 com RF-009 para atribuição de transportadores.
- **Testes de Sistema:** Cobrem fluxos end-to-end, como criar um pedido, atribuí-lo e rastreá-lo (RF-003, RF-009, RF-004).

### 3.2. Abordagens

De entre as várias abordagens de testes no contexto do nosso projeto, iremos utilizar as seguintes:

- Black-Box, baseando nos requisitos, testando inputs e outputs;
- White-Box, foco na análise do código.

## 4. Condições para testes

As condições definem o ambiente e pré-requisitos para execução dos testes, garantindo consistência e escalabilidade do sistema.

Sendo assim segue-se algumas condições que melhor asseguram a qualidade dos testes:

- **Ambiente:** Ambiente de teste isolado com servidores, base de dados de teste, no nosso caso SQL Server e dispositivos reais (desktop, mobile, tablet). Dados de teste fictícios para privacidade. Para suportar CI/CD, o ambiente será integrado a runners do GitHub, permitindo execução automática de testes em múltiplos ambientes.

- **Pipelines de CI/CD e Testes Automáticos**: Os testes serão executados automaticamente via pipelines no GitHub Actions, configurados em arquivos.yml no diretório .github/workflows.

- **Ferramenta:** a equipa de desenvolvimento utilizará o GitHub para padronizar testes automáticos, reduzindo duplicação de código e facilitando a manutenção de pipelines. Isso inclui automação de e execução de testes em pull requests para feedback rápido

## 6. Critérios de paragem

Os testes param quando:

- **Critérios de Saída Geral:** Cobertura de requisitos >95% para core; taxa de defeitos críticos resolvidos =100%; defeitos médios/baixos <5% abertos.

- **Por Fase:** Unitários: 90% código coberto, todos testes passados. Integração: Sem falhas em interações. Sistema: Fluxos end-to-end sem erros. Aceitação: Aprovação de stakeholders.
- **Critérios Específicos por Componente:** Ex.: Para RF-004, paragem quando atualizações em tempo real funcionam em 99% dos cenários, incluindo falhas de GPS.
- **Métricas de Paragem:** Taxa de falha <2%; tempo de execução dentro de limites (ex.: resposta <2s). Se critérios não atendidos, ciclo de fix e reteste até conformidade.

## 7. Taxa de Cobertura de Testes

Embora a taxa de cobertura num contexto real deva ser mais elevada, decidimos adotar uma métrica mais baixa de forma a cumprir com as restantes funcionalidades do projeto:

- Cobertura de Requisitos: 60% para core.
- Cobertura de Código: 55-40%.
- Métricas: Defeitos por componente; tempo de resolução; satisfação em testes de usabilidade.

## 8. Riscos e Mitigações

- **Risco:** Dependências externas (GPS) falharem → Mitigação: Simulações e testes fallback.
- **Risco:** Atrasos em fases → Mitigação: Priorização core, revisões semanais.
- **Risco:** Falhas em pipelines CI/CD → Mitigação: Monitoramento contínuo.