Plano de teste para o Sistema de Reserva de Voos

1. INTRODUÇÃO

Este plano de teste descreve estratégias, processos, fluxos de trabalho e metodologias para garantir que o sistema de reserva de voos da Turkish Airlines atenda aos requisitos funcionais e não funcionais definidos.

1.1 ESCOPO

1.1.1 No escopo:

- Cadastro e login de usuários.
- Gerenciamento de voos e reservas.
- Aprovação de alterações de assentos.
- Integração com sistemas de pagamento.
- Gerenciamento de itinerários e detalhes de voos.

1.1.2 Fora do escopo:

- Testes de desempenho em escala massiva.
- Funcionalidades além do MVP.

1.2 OBJETIVOS DE QUALIDADE

- Garantir conformidade com os requisitos.
- Assegurar especificações de qualidade.
- Identificar e corrigir defeitos antes da operação.

1.3 PAPÉIS E RESPONSABILIDADES

- Implementação de testes unitários e de integração.

- Medição de atributos de qualidade.

- Projetar e melhorar casos de teste.

- Alcançar cobertura de código satisfatória.

2. METODOLOGIA DE TESTE

2.1 Visão Geral:

Metodologia Ágil, permitindo adaptações rápidas.

2.2 Fases de Teste:

- Teste Unitário: Verificação de unidades individuais.

- Teste de Integração: Interação entre módulos.

- Teste de Sistema: Validação do sistema como um todo.

- Teste de Aceitação: Confirmação de requisitos do cliente.

2.3 Triagem de Erros:

- Definição de resolução e prioridade dos erros.

2.4 Critérios de Suspensão e Requisitos de Retomada:

- Suspensão por erros críticos e retomada após correção.

2.5 Completude do Teste:

- Execução de 100% dos casos de teste.

- Correção de todos os erros críticos.

- Cobertura de código e score de mutação satisfatórios.

3. ENTREGÁVEIS DE TESTE

- Plano de Teste

- Casos de Teste

- Relatórios de Erros

- Métricas de Teste

- Documentação de Estratégia de Teste

4. NECESSIDADES DE RECURSOS E AMBIENTE

4.1 Ferramentas de Teste:

- Repositório de código: GitHub

- Comunicação: Discord, WhatsApp, reuniões presenciais

- Automação de teste: JUnit, Mockito, Selenium

- Gerenciamento de casos de teste: Testlink

- Integração contínua: Jenkins

4.2 Ambiente de Teste:

- Backend: Java com Servlets e JSP

- Frontend: HTML/CSS, JavaScript

- Banco de Dados: MySQL

- Servidores: Tomcat

- Requisitos de hardware: Processadores i5 ou superiores, 8GB de RAM.

5. TERMOS / ACRÔNIMOS

- API: Application Program Interface

- MVP: Minimum Viable Product

- Ágil: Metodologia de desenvolvimento ágil

- JUnit: Framework para testes unitários em Java

- Selenium: Ferramenta para testes de browser end-to-end

- Mockito: Biblioteca para criação de mocks em testes de unidade

- Testlink: Ferramenta para gerenciamento de casos de teste