Plano de Teste para o Sistema PDV

1. Introdução

Este plano de teste tem como objetivo definir a estratégia e os recursos necessários para testar o Sistema PDV (Ponto de Venda), disponível no repositório GitHub: https://github.com/repo-software-testing-courses/pdv. O plano abrange testes unitários, de integração e de interface web, utilizando ferramentas como JUnit, Mockito e Selenium para garantir a qualidade e a confiabilidade do sistema.

---

2. Objetivos do Plano de Teste

- **Validar** todas as funcionalidades críticas do sistema PDV.
- **Identificar e corrigir** defeitos antes da implantação em produção.
- **Assegurar** que o sistema atenda aos requisitos funcionais e não funcionais.
- **Garantir** a usabilidade e a responsividade da interface web.

---

3. Escopo

3.1 Funcionalidades a Serem Testadas

- Processamento de vendas com múltiplos itens.
- Aplicação de descontos em itens específicos.
- Processamento de devoluções de produtos.
- Atualização do estoque após vendas e devoluções.
- Autenticação e autorização de usuários.
- Emissão de notas fiscais.
- Interface web para operadores e administradores.

3.2 Funcionalidades Fora do Escopo

- Integração com sistemas externos não implementados.
- Funcionalidades futuras ou em fase de planejamento.

---

4. Estratégia de Teste

4.1 Testes Unitários

- **Ferramenta**: JUnit
- **Descrição**: Desenvolver testes unitários para validar a funcionalidade de métodos e classes individuais.
- **Abordagem**:
    - Testar métodos de cálculo de preços e totais.
    - Verificar a aplicação correta de descontos.
    - Testar a atualização do estoque após operações.

4.2 Testes de Integração

- **Ferramentas**: JUnit e Mockito
- **Descrição**: Validar a interação entre diferentes módulos do sistema.
- **Abordagem**:
    - Utilizar Mockito para simular dependências e serviços externos.
    - Testar o fluxo completo de uma venda, incluindo a atualização do estoque e emissão de nota fiscal.
    - Verificar a autenticação e autorização de usuários.

4.3 Testes de Interface Web

- **Ferramenta**: Selenium WebDriver
- **Descrição**: Automatizar testes da interface web para garantir a usabilidade e funcionalidade do sistema.
- **Abordagem**:
    - Simular ações do usuário, como login, processamento de vendas e devoluções.
    - Verificar a responsividade em diferentes navegadores (Chrome, Firefox, Edge).
    - Testar validações de formulários e mensagens de erro.

4.4 Testes de Regressão

- **Descrição**: Garantir que novas alterações não introduzam defeitos em funcionalidades existentes.
- **Abordagem**:
    - Reexecutar testes automatizados após cada alteração significativa no código.
    - Atualizar os casos de teste conforme necessário.

---

5. Ambiente de Teste

5.1 Hardware

- Servidores ou máquinas virtuais para hospedar o ambiente de teste.
- Computadores para execução dos testes de interface web.

5.2 Software

- **Back-end**: Java, Spring Framework.
- **Front-end**: HTML, CSS, JavaScript.
- **Banco de Dados**: MySQL
- **Navegadores**: Últimas versões do Chrome, Firefox e Edge.

5.3 Ferramentas

- **JUnit**: Para testes unitários.
- **Mockito**: Para criação de mocks em testes de integração.
- **Selenium WebDriver**: Para testes automatizados da interface web.
- **Git**: Controle de versão.
- **Maven**: Gerenciamento de dependências e builds.

---

6. Cronograma de Teste

Atividade	Início	Término
Planejamento dos testes	08/11/2024	10/11/2024
Desenvolvimento de testes unitários	03/11/2024	04/11/2024
Desenvolvimento de testes de integração	18/11/2024	24/11/2024
Desenvolvimento de testes de interface web	25/11/2024	01/12/2024
Execução de testes e registro de defeitos	02/12/2024	08/12/2024
Reteste e testes de regressão	09/12/2024	12/12/2024
Relatório final de testes	04/01/2024	10/12/2024

---

7. Recursos Necessários

7.1 Equipe

- **Analista de Testes Sênior**: Responsável pelo planejamento e coordenação dos testes.
- **Dois Engenheiros de Teste**: Desenvolvimento e execução dos casos de teste.
- **Desenvolvedores**: Suporte para correção de defeitos identificados.

7.2 Ferramentas e Licenças

- Licenças para ferramentas de automação, se necessário.
- Acesso ao repositório do código-fonte.
- Ambiente configurado para execução dos testes.

---

8. Riscos e Mitigações

Risco	Mitigação
Atrasos no desenvolvimento que impactem os testes	Realizar reuniões semanais para alinhamento.
Ambiente de teste instável ou indisponível	Configurar ambientes alternativos ou virtuais.
Falta de familiaridade com ferramentas	Treinamento prévio da equipe em JUnit, Mockito e Selenium.

---

9. Critérios de Aceitação

- **Critérios de Entrada**:
    - Código-fonte estável disponível no repositório.
    - Ambientes de teste configurados.
    - Casos de teste aprovados pelo analista de testes.
- **Critérios de Saída**:
    - Todos os casos de teste planejados foram executados.
    - Defeitos críticos e de alta prioridade foram corrigidos e retestados.
    - Documentação atualizada com resultados e evidências dos testes.

---

10. Entregáveis

- **Casos de Teste**: Documentação detalhada dos casos de teste unitários, de integração e de interface web.
- **Scripts de Teste**: Código dos testes automatizados em JUnit, Mockito e Selenium.

- **Relatórios de Defeitos**: Registro de todos os defeitos encontrados, com detalhes e status.
- **Relatório Final de Testes**: Sumário dos resultados, métricas e recomendações.