Plano de Teste
PDV Caminho Feliz

Registro de Mudanças

Versão	Data de Mudança	Por	Descrição
1.0	07/08/2021	Vânia	Tradução do documento
2.0	04/11/2024	Grupo Caminho Feliz	Especificação do projeto e seu escopo
3.0	06/11/2024	Grupo Caminho Feliz	Atualização da documentação seguindo recomendações da professora Vânia

1 Introdução

Este Plano de Teste define o escopo, a abordagem e o cronograma para validar o sistema de ERP web para ponto de venda (PDV), desenvolvido em Java com Spring Framework. Disponível no repositorio abaixo: https://github.com/ovvesley/uff-qualidade-teste-pdv

O plano identifica os itens a serem testados, os recursos a serem testados, os tipos de teste a serem realizados, o pessoal responsável pelo teste, os recursos e o cronograma necessários para concluir o teste e os riscos associados ao plano.

1.1 Escopo

1.1.1 No escopo

Todos do site PDV que foram definidos abaixo devem ser testados.

Mais módulos a definir.

Nome do Módulo	Papéis aplicáveis	Descrição
Gerenciar Caixa	Gerente	**Gerente:** O módulo "Gerenciar Caixa" permite que o gerente controle suprimentos, retiradas, lançamentos e saldo do caixa, além de visualizar o usuário e a data de abertura. Ele garante o monitoramento diário do fluxo financeiro.
Gerenciar grupo de usuários	Gerente	**Gerente:** O módulo “Gerenciar grupo de usuários”permite que o gerente visualize todos os grupos de usuário, edite atributos dos grupos, adicione e exclua permissões do grupo, apague grupos, visualize todos os grupos que estão associados a determinado usuário e relacione usuários com grupos. Garante o gerenciamento dos grupos de usuários.
Adicionar empresas	Gerente	**Gerente:** O módulo “adicionar empresas” permite que o gerente adicione empresas fornecedoras de produtos
Gerenciar Pagamentos de Despesas	Gerente	**Gerente:** O módulo de gerenciar pagamentos de despesas permite com que, dado um valor em caixa e um despesa, o gerente consiga, usando o valor em caixa, diminuir das despesas registradas

1.1.2 Critérios para a definição do escopo

No contexto deste plano de teste, adotamos o critério de escopo baseado na **complexidade** das classes e na **relevância** dos módulos dentro do sistema. A escolha das classes foi feita com base na complexidade mínima considerada média, garantindo que o código a ser testado tivesse um nível suficiente de dificuldade para representar desafios reais.

Além disso, a seleção dos módulos seguiu a lógica dos **casos de uso** do sistema, focando em testar as funcionalidades em um ambiente realista, dentro dos cenários em que o sistema seria efetivamente utilizado.

Dessa forma, buscamos assegurar que os testes realizados fossem representativos das situações práticas que o software enfrentará em produção.

1.1.3 Fora do escopo

Esses recursos não serão testados porque não estão incluídos nas especificações de requisitos do software

- Interfaces de hardware
- Interfaces de software
- Lógica de banco de dados
- Interfaces de comunicação
- Segurança e desempenho do site

1.2 Objetivos de Qualidade

O objetivo deste plano de testes é assegurar que o sistema de ERP PDV opere com precisão, integridade e estabilidade em ambiente de produção, atendendo aos requisitos de funcionalidade e usabilidade. Serão validadas a exatidão das operações essenciais, como cadastro de pedidos e controle de caixa, a integridade dos dados, evitando inconsistências ou perdas, garantindo uma experiência eficiente para o usuário.

1.3 Papéis e Responsabilidades

Para este projeto, todos os quatro alunos de graduação irão compartilhar as responsabilidades, assumindo um pouco de cada papel. Eles irão:

- **Gerenciar o projeto de teste**, definindo direções e organizando recursos.
- **Criar e revisar os casos de teste**, escolhendo técnicas e ferramentas adequadas.
- **Implementar e executar os testes**, configurando e rodando a suíte de testes no ambiente de teste.
- **Manter o ambiente de teste e garantir a qualidade**, verificando a conformidade com os requisitos.

2 Metodologia de Teste

2.1 Fases de Teste

No projeto Caminho Feliz, serão conduzidas as seguintes fases de teste:

- **Teste de unidade** (módulos são testados individualmente)
- **Teste de integração** (módulos de software individuais são combinados e testados como um grupo)
- Teste de **sistema**: conduzir teste de alguns casos de uso em um sistema **completo** e **integrado** para avaliar a conformidade do sistema com seus requisitos especificados.

2.2 Critérios de Suspensão e Requisitos de Retomada

Se os membros da equipe relatarem que há **40%** dos casos de teste com **falha**, suspenda o teste até que a equipe de desenvolvimento corrija todos os casos com falha.

2.3 Completude do Teste

- Especifica os critérios que denotam a conclusão **bem-sucedida** de uma fase de teste
- A taxa de 100% de **execução** dos testes é **obrigatória**, a menos que um motivo claro seja fornecido.
- A taxa de **100%** de **aprovação** dos testes é **obrigatória**.
- Dos códigos de teste executados, deve possuir **mais de 80% de cobertura.**

2.4 Atividades do projeto, estimativas e cronograma

Tarefa	**Membros**	**Estimativa de esforço**
Criar especificação de testes	Wesley	3 horas
Executar testes unitários	Todos os Membros	4 horas
Executar testes de integração	Todos os Membros	12 horas
Criar o relatório de testes	Todos os Membros	5 horas
Entregar e revisar os testes	Entrega feita por Clara Lino, Revisão feita por todos os membros	3 horas
Total		27 horas

3 Entregáveis de Teste

Entregáveis de Teste

Para cada item do escopo definido e para cada classe estabelecida, além dos arquivos de teste na pasta padrão do JUnit, será criada uma pasta adicional chamada `entrega1` **ou** `entrega2`, conforme a etapa. Nessa pasta, serão armazenadas evidências dos testes executados em formato de imagem, mostrando o resultado de cada caso de teste e a cobertura de código da respectiva classe. Essa estrutura

facilitará a verificação e avaliação dos resultados de cada entrega, permitindo uma rápida inspeção visual dos testes e da cobertura alcançada.

Pasta da Entrega 1: https://github.com/ovvesley/uff-qualidade-teste-pdv/tree/master/entrega01

Entrega 1

1. **Descrição do Escopo do Sistema**
    - Definição dos módulos/componentes que serão testados, documentando o escopo no Plano de Teste.
2. **Plano de Teste**
    - Documento do Plano de Teste conforme template, incluindo ferramentas utilizadas, artefatos gerados e métodos.
3. **Casos de Teste Unitários**
    - Casos de teste projetados para pelo menos uma classe para cada membro do grupo.
4. **Casos de Teste Manuais**
    - Casos de teste manuais para pelo menos uma funcionalidade para cada membro, utilizando Testlink ou uma alternativa (texto/planilha).

Pasta da Entrega 2:

Entrega 2

1. **Casos de Teste Unitários (Expandidos)**
    - Revisão e expansão dos casos unitários, com isolamento de dependências.
2. **Casos de Teste de Integração**
    - Implementação de testes de integração cobrindo interações entre módulos.
3. **Atributos de Qualidade da ISO 25010**
    - Avaliação e medida dos atributos de qualidade conforme a norma, justificando as escolhas.
4. **Casos de Teste de Sistema**
    - Implementação de testes de sistema considerando requisitos funcionais e um atributo de qualidade.
5. **Conjunto de Casos de Teste Melhorado**
    - Conjunto de testes aprimorado com técnicas:
        - **Funcional**
        - **Estrutural** (80% de cobertura no critério todas-arestas)
        - **Baseada em Defeitos** (80% de escore de mutação).
6. **Relatório de Inspeção do Código-Fonte**
    - Utilização de uma ferramenta como SonarQube, com captura de tela dos resultados e solução de problemas em uma classe para cada membro.

Após Conclusão dos Testes

- **Relatório Final de Testes:** Resultado dos testes com detalhes dos casos, falhas e status final.

- **Prints das Correções e Análise de Qualidade:** Evidência das correções e da inspeção de código para as classes de cada membro.

4 Necessidades de Recursos e Ambiente

4.1 Ferramentas de Teste

Servidor: Necessário um servidor local com Docker instalado para gerenciar o banco de dados MySQL e o servidor Web.
Ferramenta de Teste: Utilizar uma ferramenta de teste automatizada que seja compatível com Docker, capaz de gerar relatórios no formato predefinido e executar testes automaticamente.
Rede: Configure uma rede local (LAN) com conexão de internet de pelo menos 5 Mb/s.
Computador: Pelo menos 4 máquinas locais com suporte a Docker, com 2GB de RAM e CPU de 3.4 GHz ou superior.

4.2 Ambiente de Teste

Para executar o sistema de forma simples e rápida no ambiente de teste, basta utilizar uma máquina local com Docker e seguir os passos abaixo. A configuração inclui clonagem do repositório, preparação com Maven e execução via Docker Compose. Com isso, o sistema e suas dependências serão automaticamente configurados e estarão acessíveis na porta 8080.

Etapas de Configuração

1. **Instalação do Docker:**
   - Certifique-se de que a máquina de teste possui o Docker e Docker Compose instalados. O Docker permitirá o uso de contêineres para configurar e gerenciar o ambiente de teste de maneira simplificada, isolando as dependências do sistema.
2. **Clonagem do Repositório:**
   - Acesse a máquina de teste e, no terminal, execute o comando `git clone https://github.com/ovvesley/uff-qualidade-teste-pdv`
   - Isso irá copiar o código-fonte e todos os arquivos necessários do sistema para a máquina local.

Exemplo de comando:

```
git clone
https://github.com/ovvesley/uff-qualidade-teste-pdv
```

- Navegue até o diretório clonado usando `cd uff-qualidade-teste-pdv` para acessar os arquivos do projeto.

3. **Preparação das Dependências com Maven:**
   - O Docker Compose está configurado para construir o projeto usando Maven, que será executado dentro do próprio contêiner.
   - Esse processo de build irá baixar e configurar todas as dependências do sistema automaticamente.
   - **Observação:** A configuração com Maven permite que o sistema esteja sempre atualizado com as bibliotecas e frameworks necessários, sem necessidade de instalação manual.
4. **Execução com Docker Compose:**

No diretório do projeto, execute o comando:

```
docker-compose up
```

   - Esse comando fará o seguinte:
     - **Construção dos Contêineres**: Cria contêineres para o servidor de aplicação e para o banco de dados MySQL.
     - **Configuração Automática**: Configura as variáveis de ambiente e demais parâmetros necessários, como as credenciais e configurações do banco de dados.
     - **Inicialização do Sistema**: Inicializa o sistema e disponibiliza a aplicação para acesso.
5. **Acesso ao Sistema:**
   - Após a execução do Docker Compose, o sistema estará acessível na porta `8080` da máquina local.
   - Para verificar se o sistema está funcionando, abra um navegador e acesse `http://localhost:8080`.

5 Termos / Acrônimos

TERMO / ACRÔNIMO	DEFINIÇÃO
API	*Application Program Interface*
SUT	*Software Under Test*
ERP	Enterprise Resource Planning
JUnit	Ferramenta para execução de testes unitários, mencionada nas entregas de teste.
Maven	Ferramenta de build que gerencia dependências e facilita a configuração de projetos Java, usada na preparação do ambiente de teste.

TERMO / ACRÔNIMO	DEFINIÇÃO
Docker	Ferramenta de contêinerização que permite configurar o ambiente de teste de forma isolada e replicável.
Git	Sistema de versionamento usado para o controle de versões do código-fonte e testes.
Testlink	Ferramenta de gerenciamento de testes manuais, mencionada para organizar casos de teste.
ISO 25010	Padrão para avaliar e medir atributos de qualidade de software, como confiabilidade e eficiência, usado para orientar a avaliação dos testes.