

# Relatório de Testes JUnit e Code Coverage

## 1- Introdução:

- Objetivo: Este documento relata os testes realizados na classe TesteAplicacao localizado no diretório src → test → java → JUnit utilizando o último para validar as funcionalidades do sistema.
- Contexto: Os testes foram realizados nas ações de cadastrar e exibir clientes, vendedores e produtos, além de operações específicas como salvar, deletar e atualizar os mesmos.

## 2 - Ambiente de Testes:

Todas as dependências e plugins para testes estão presentes no arquivo de configuração do projeto **pom.xml** do Apache Maven.

**Versão do JUnit:** 7.14.0

### Outras Ferramentas Utilizadas:

- Plugin Surefire para execução de testes JUnit: v3.2.2
- Dependência do Cucumber: v7.14.0
- JaCoCo Maven: v0.8.6

## 3- Casos de Teste:

- ☒ Cadastrar Novo Cliente: Testa o método **cadastrarNovoCliente** da classe **aplicacao**.

**Passos:** Obtém o tamanho da lista de clientes antes do cadastro, executa o método de cadastro de novo cliente, obtém o tamanho da lista de clientes depois do cadastro.

**Resultado Esperado:** O tamanho da lista de clientes deve aumentar em 1.

- ☒ Exibir Clientes: Testa o método **exibirClientes** da classe **aplicacao**.

**Passos:** Executa o método de exibição de clientes presente na classe citada, que está localizado no diretório src → main → aplicacao.

**Resultado Esperado:** O método deve ser executado sem gerar exceções.

- ☒ Cadastrar Novo Vendedor: Testa o método **cadastrarNovoVendedor** da classe **aplicacao**.

**Passos:** Obtém o tamanho da lista de vendedores antes do cadastro, executa o método de cadastro de novo vendedor, obtém o tamanho da lista de vendedores depois do cadastro.

**Resultado Esperado:** O tamanho da lista de vendedores não deve ser alterado.

- ☑ Exibir Vendedores: Testa o método `exibirVendedores` da classe `aplicacao`.

**Passos:** Executa o método de exibição de vendedores. Classe localizada em src → main → aplicacao.

**Resultado Esperado:** O método deve ser executado sem gerar exceções.

- ☑ Cadastrar Novo Produto: Testa o método `cadastrarNovoProduto` da classe `aplicacao`.

**Passos:** Obtém o tamanho da lista de produtos antes do cadastro, executa o método de cadastro de novo produto, Obtém o tamanho da lista de produtos depois do cadastro.

**Resultado Esperado:** O tamanho da lista de produtos deve aumentar em 1.

- ☑ Exibir Produtos: Testa o método `exibirProdutos` da classe `aplicacao`.

**Passos:** Executa o método de exibição de produtos. Classe localizada em src → main → aplicacao.

**Resultado Esperado:** O método deve ser executado sem gerar exceções.

- ☑ Cadastrar Novo Cliente com Informações Inválidas: Testa o método `cadastrarNovoClienteComInformacoesInvalidas` da classe `aplicacao`.

**Passos:** Obtém o tamanho da lista de clientes antes do cadastro inválido, executa o método de cadastro de novo cliente com informações inválidas e obtém o tamanho da lista de clientes depois do cadastro inválido.

**Resultado Esperado:** O tamanho da lista de clientes não deve ser alterado.

- ☑ Get Clientes: Testa o método `getClientes` da classe `ClienteDAO`, é necessário a existência de clientes no banco de dados.

**Passos:** Executa o método de obtenção de clientes. Classe localizada em src → main → dao.

**Resultado Esperado:** A lista de clientes não deve ser nula.

- ☑ Save Produto: Testa o método `saveProduto` da classe `ProdutoDAO`.

**Passos:** Obtém o tamanho da lista de produtos antes do salvamento, executa o método de salvamento de novo produto e obtém o tamanho da lista de produtos depois do salvamento.

**Resultado Esperado:** O tamanho da lista de produtos deve aumentar em 1.

- ☑ Save Produto Com Nome Nulo: Testa o método `saveProduto` da classe `ProdutoDAO` com nome nulo.

**Passos:** Obtém o tamanho da lista de produtos antes do salvamento inválido, executa o método de salvamento de novo produto com nome nulo e obtém o tamanho da lista de produtos depois do salvamento inválido.

**Resultado Esperado:** O tamanho da lista de produtos não deve ser alterado.

- ☑ Salvar Produto com Preço Zero: Testa o método `saveProduto` da classe `ProdutoDAO` com preço zero.

**Passos:**

Obtém o tamanho da lista de produtos antes do salvamento inválido, executa o método de salvamento de novo produto com preço zero e obtém o tamanho da lista de produtos depois do salvamento inválido.

**Resultado Esperado:** O tamanho da lista de produtos não deve ser alterado.

- ☑ Get Produtos: Testa o método `getProdutos` da classe `ProdutoDAO`, é necessário a existência de produtos no banco de dados.





**Passos:** Executa o método de obtenção de produtos. Classe localizada em `src → main → dao`.

**Resultado Esperado:** A lista de produtos não deve ser nula.

#### 4- Evidências:

Capturas de Tela:

*[\\*CLIQUE NOS ÍCONES PARA ACESSAR A MÍDIA!!!](#)*

- Jacoco Code Coverage Geral:  `CODE COVERAGE.png`
- Site Jacoco:  `Site Jacoco .png`
- Vídeo Navegando pelo Jacoco da aplicação: [Vídeo Classes Jacoco.mp4](#)
- Testes JUnit :  `Testes JUnit.png`
- Testes JUnit (VS CODE):  `Testes JUnit VS CODE.png`
- Vídeo acessando Code Metrics: [Java Code Metrics.mp4](#)

Importante: O Java Code Metrics não exibiu nenhuma sugestão de alteração no código, sendo possível concluir que não há alterações ou melhorias a serem feitas no momento.

#### 5- Análise de Resultados:

Observando os resultados apresentados é notório que as funcionalidades testadas estão em conformidade com as especificações fornecidas do nosso sistema fornecida pelo nosso grupo.

**Problemas Identificados:** Algumas exceções não foram tratadas em alguns métodos, como `cadastrarNovoClienteComInformacoesInvalidas`, o que pode resultar em comportamento inesperado. Para a solução, a equipe está trabalhando para implementar o tratamento adequado para exceções.

#### 6- Conclusão:

Os testes no geral foram positivos, mostrando que as funcionalidades estão dentro dos conformes. O sistema conseguiu lidar bem com o cadastro de clientes,

vendedores e produtos, e também foi positivo ao recuperar, mostrar e deletar os dados.

Encontramos alguns pontos que precisam de atenção como por exemplo: pontos no código que não lidam muito bem com situações inesperadas, o que pode impactar a segurança e a estabilidade do sistema.

## **7- Aprovação:**

Testes concluídos em 28/11/2023 e revisados nos dias 29 e 30/11/2023.

Responsáveis pelos Testes: Jean Carlos, Almo Contim, Sâmeck Zanela e Gabriel Moura.