Caixa

Testes para Caixa

CaixaService -Sonar

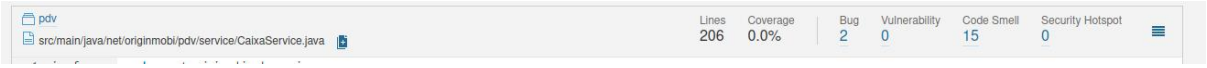
Sonar - CaixaService

Data: 30 de Novembro de 2025
Autor do Teste: Elena Costa Blanco

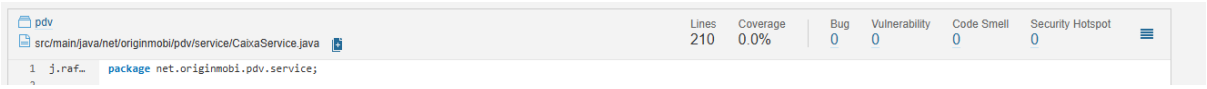
Diagnóstico Inicial

Ao rodar o SonarQube na classe CaixaService.java, identificamos diversos Code Smells e Bugs Críticos. Os problemas estavam concentrados em três áreas principais: concorrência (Thread Safety), complexidade cognitiva e tratamento de erros.

Antes:



Depois:



O que foi feito:

- **Thread Safety (Concorrência):** Correção crítica removendo variáveis de instância globais (descricao, usuario) e tornando-as locais para evitar conflitos de dados entre usuários simultâneos.
- **Complexidade Cognitiva:** Refatoração do método cadastro com a extração de validações e regras para métodos auxiliares, reduzindo a dificuldade de manutenção.
- **Segurança com Optional**: Substituição do uso inseguro de .get() por .orElseThrow() para prevenir quebras de execução (NoSuchElementException).
- **Logging e Exceções:** Substituição de System.out.println pelo uso correto de Logger (SLF4J) e eliminação do anti-padrão de logar e relançar a mesma exceção.

CartaoLancamentoService

Testes para CartaoLancamentoService

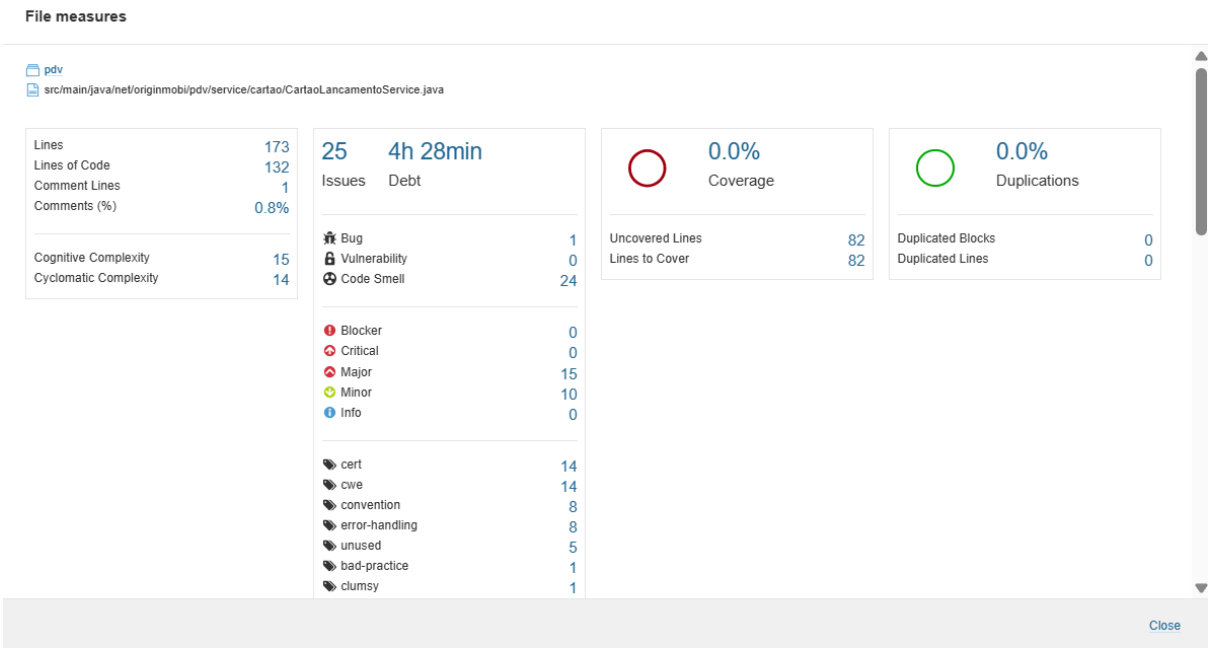
CartaoLancamentoS - Sonarervice

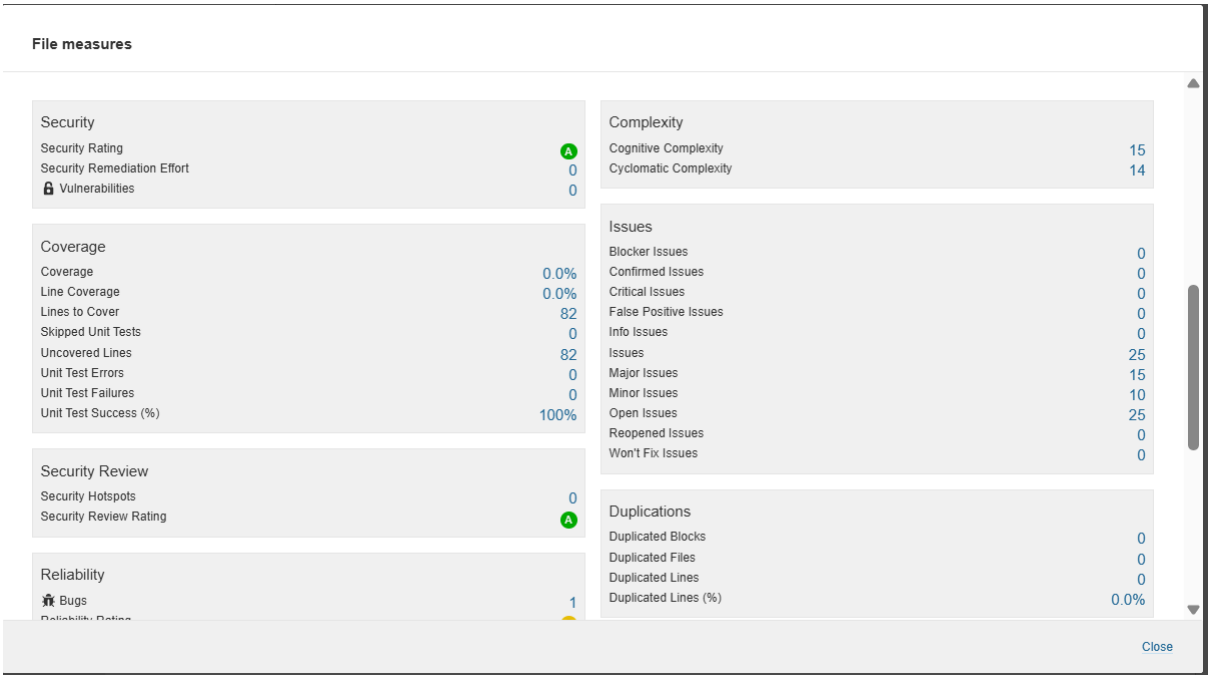
Sonar - CartaoLancamentoService

Data: 30 de Novembro de 2025
Autor da Inspeção: Larissa Galvão

Diagnóstico Inicial

Ao rodar o SonarQube na classe `CartaoLancamentoService.java`, identificamos um total de **24 Code Smells** e **1 Major Bug**. Os problemas estavam concentrados em três áreas principais: segurança do código, tratamento de exceções e adesão às convenções de código Java.





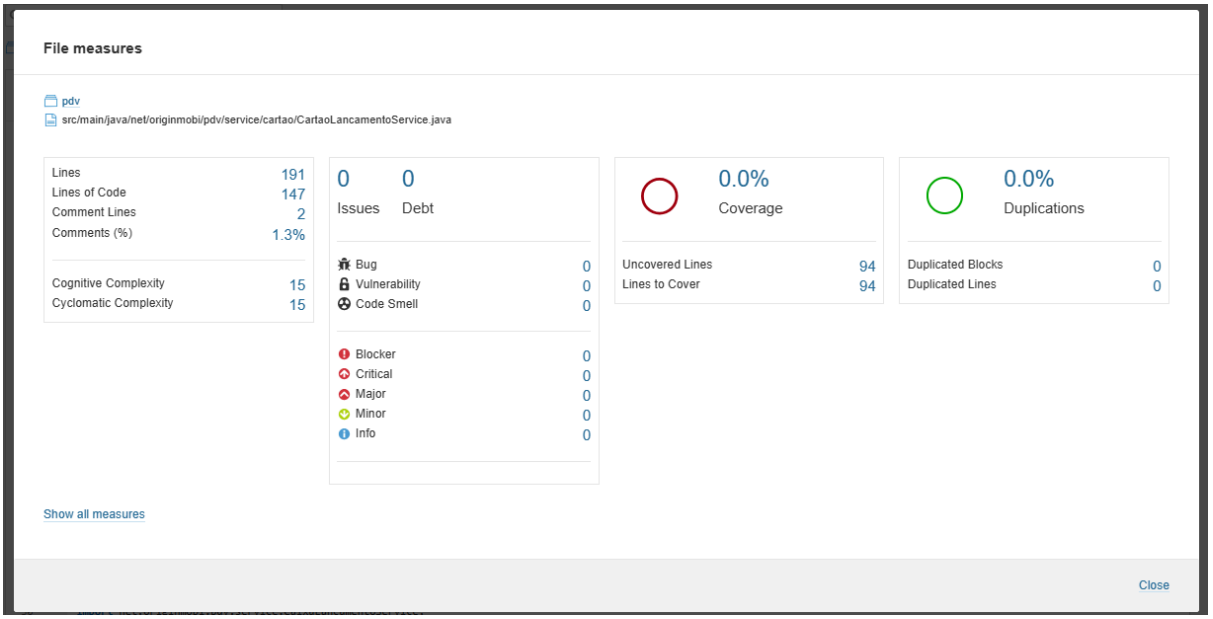
Correções:

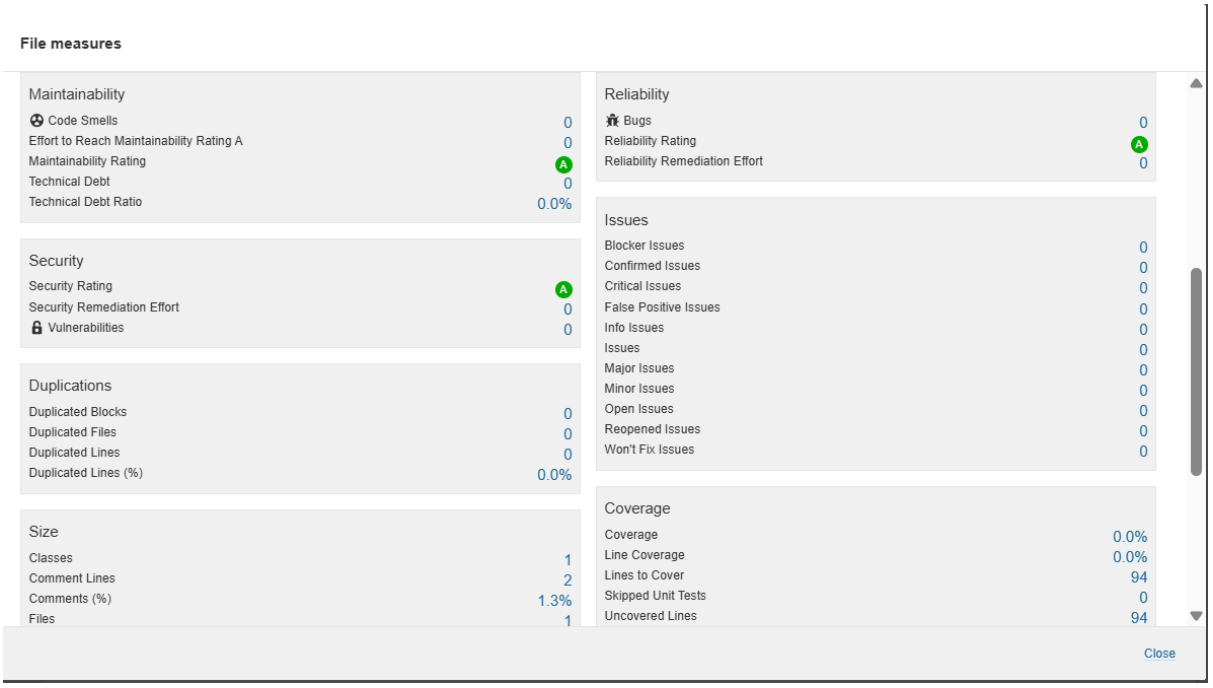
1. Correções de Alto Impacto (Major)

- **Bug de Segurança (`Optional`):** O acesso inseguro via `titulo.get()` foi substituído pelo uso do método **`Optional.orElseThrow()`**. Isso eliminou o risco de `NoSuchElementException` e garantiu a verificação segura de dependências.
- **Tratamento de Exceções:** Todas as ocorrências de `RuntimeException` genéricas foram substituídas por **Exceções de Negócio dedicadas** (ex: `CartaoLancamentoException`). Esta mudança aumenta a clareza e a rastreabilidade dos erros.

2. Correções de Boas Práticas (Minor)

- **Convenção de Código:** Nomenclatura foi padronizada para o formato **camel case** (e.g., `vl_taxa` para `vlTaxa`).
- **Logging:** O uso de `System.out.println` foi substituído pela implementação de um **Logger** profissional, aderindo às boas práticas de log de aplicações.


File measures
pdv
src/main/java/net/originmobi/pdv/service/cartao/CartaoLancamentoService.java
Lines 191
Lines of Code 147
Comment Lines 2
Comments (%) 1.3%
Cognitive Complexity 15
Cyclomatic Complexity 15
0 Issues
0 Debt
Bug 0
Vulnerability 0
Code Smell 0
Blocker 0
Critical 0
Major 0
Minor 0
Info 0
0.0% Coverage
Uncovered Lines 94
Lines to Cover 94
0.0% Duplications
Duplicated Blocks 0
Duplicated Lines 0
Show all measures
Close


File measures
Maintainability
Code Smells 0
Effort to Reach Maintainability Rating A 0
Maintainability Rating A
Technical Debt 0
Technical Debt Ratio 0.0%
Security
Security Rating A
Security Remediation Effort 0
Vulnerabilities 0
Duplications
Duplicated Blocks 0
Duplicated Files 0
Duplicated Lines 0
Duplicated Lines (%) 0.0%
Size
Classes 1
Comment Lines 2
Comments (%) 1.3%
Files 1
Reliability
Bugs 0
Reliability Rating A
Reliability Remediation Effort 0
Issues
Blocker Issues 0
Confirmed Issues 0
Critical Issues 0
False Positive Issues 0
Info Issues 0
Issues 0
Major Issues 0
Minor Issues 0
Open Issues 0
Reopened Issues 0
Won't Fix Issues 0
Coverage
Coverage 0.0%
Line Coverage 0.0%
Lines to Cover 94
Skipped Unit Tests 0
Uncovered Lines 94
Close

Fornecedores

Testes para Fornecedores

Fornecedores - Funcional

Documento de Casos de Teste: Gestão de Fornecedores

Produto/Funcionalidade: Gestão de Fornecedores
Data: 29 de Novembro de 2025
Autor do Teste: Larissa Galvão

Tipo de teste: Sistema

Ferramenta: Selenium

Pré-requisito

1. Acesso ao sistema e login realizado com sucesso (usuário *gerente*, senha *123*).
2. Navegação para o módulo **Cadastro de Fornecedores**.

Caso de Teste 1: Cadastro de Novo Fornecedor (Sucesso)

- **ID:** F-FORN-001
- **Objetivo:** Inserir um novo fornecedor garantindo a funcionalidade de ponta a ponta.

1. **Ação:** Preencher o campo **Razão Social** com `FORNECEDOR ALPHA LTDA`.
    - **Resposta Esperada:** O campo aceita e exibe o valor digitado.
2. **Ação:** Preencher o campo **CNPJ** com `99.999.999/0001-00`.
    - **Resposta Esperada:** O campo aceita e exibe o valor, aplicando a máscara de CNPJ.
3. **Ação:** Preencher o campo **Telefone** com `(11) 98765-4321`.
    - **Resposta Esperada:** O campo aceita e exibe o valor.
4. **Ação:** Preencher os campos de **Endereço** (Rua, Número, Cidade, etc.).
    - **Resposta Esperada:** Todos os campos de endereço aceitam os valores.
5. **Ação:** Clicar no botão **"Salvar"**.
    - **Resposta Esperada:** O sistema deve exibir uma **mensagem de sucesso** (ex: "Fornecedor salvo com sucesso") na tela.
6. **Ação:** Navegar para a tela de **Consulta de Fornecedores** (ou atualizar a lista se for na mesma página).
    - **Resposta Esperada:** A lista de fornecedores é carregada.
7. **Ação:** Inspecionar a lista de resultados.
    - **Resposta Esperada:** O registro com a Razão Social `FORNECEDOR ALPHA LTDA` deve ser visível na lista.

Caso de Teste 2: Atualização de Fornecedor Existente

- **ID:** F-FORN-002
- **Objetivo:** Verificar a edição correta de campos e a persistência das alterações.

1. **Ação:** Na tela de **Consulta**, clicar no ícone ou botão para **"Editar"** o fornecedor criado no teste 001.
    - **Resposta Esperada:** O formulário de cadastro é carregado com todos os dados do fornecedor preenchidos.
2. **Ação:** Alterar o campo **Nome Fantasia** para `Alpha Teste Atualizado`.
    - **Resposta Esperada:** O campo aceita e exibe o novo valor.
3. **Ação:** Alterar o campo **Telefone** para `(11) 99999-0000`.
    - **Resposta Esperada:** O campo aceita e exibe o novo valor.
4. **Ação:** Clicar no botão **"Salvar"**.
    - **Resposta Esperada:** O sistema envia os dados atualizados ao *backend* e processa a atualização.
5. **Ação:** Após o processamento.
    - **Resposta Esperada:** O sistema deve exibir uma **mensagem de sucesso** (ex: "Fornecedor atualizado com sucesso").
6. **Ação:** Navegar para a tela de **Consulta de Fornecedores**.
    - **Resposta Esperada:** A lista de fornecedores é carregada.
7. **Ação:** Inspecionar a lista de resultados e o fornecedor em questão.
    - **Resposta Esperada:** O fornecedor deve exibir o **Nome Fantasia** (`Alpha Teste Atualizado`) e o **novo Telefone** na lista.

Caso de Teste 3: Validação de CNPJ Duplicado

- **ID:** F-FORN-SIST-003
- **Objetivo:** Verificar se a regra de negócio impede o cadastro de CNPJ repetido.

1. **Ação:** No formulário de Cadastro, preencher o campo **Razão Social** com `FORNECEDOR DUPLICADO`.
    - **Resposta Esperada:** O campo aceita o valor.
2. **Ação:** Preencher o campo **CNPJ** com o mesmo CNPJ do teste 001 (`99.999.999/0001-00`).
    - **Resposta Esperada:** O campo aceita o valor, mas a validação interna deve ser acionada.
3. **Ação:** Preencher os demais campos obrigatórios.
    - **Resposta Esperada:** Os campos aceitam os valores.
4. **Ação:** Clicar no botão **"Salvar"**.
    - **Resposta Esperada:** O sistema tenta salvar, falha na validação de unicidade (`CNPJ já cadastrado`), e **não persiste o novo registro**.
5. **Ação:** Após o processamento.
    - **Resposta Esperada:** O sistema deve exibir uma **mensagem de erro** no formulário (espera-se a mensagem: **"CNPJ já cadastrado"**).

6. **Ação:** Inspecionar a lista de consulta.
    - **Resposta Esperada:** O registro `FORNECEDOR DUPLICADO` **não deve existir** na lista.

Caso de Teste 4: Consulta por Filtro (Busca Parcial)

- **ID:** F-FORN-SIST-006
- **Objetivo:** Validar se o filtro de busca restringe corretamente os resultados.

1. **Ação:** Estar na tela de **Consulta de Fornecedores**.
    - **Resposta Esperada:** A lista completa (ou paginada) de fornecedores é exibida.
2. **Ação:** Digitar o termo parcial `ALPHA` no campo de filtro de nome.
    - **Resposta Esperada:** O campo aceita o valor.
3. **Ação:** Clicar no botão **"Buscar"**.
    - **Resposta Esperada:** O sistema executa a consulta no *backend* (`findByNomeContaining("ALPHA")`).
4. **Ação:** Inspecionar a lista de resultados.
    - **Resposta Esperada:** A lista deve exibir **apenas** os fornecedores que contenham o termo `ALPHA` no nome (neste caso, `FORNECEDOR ALPHA LTDA`). Fornecedores sem o termo devem ser omitidos.

Caso de Teste 5: Consulta com Filtro Vazio

- **ID:** F-FORN-SIST-007
- **Objetivo:** Validar que, sem filtro, todos os registros são retornados.

1. **Ação:** Na tela de **Consulta de Fornecedores**, garantir que o campo de filtro de nome está **limpo** (vazio).
    - **Resposta Esperada:** O campo de filtro de nome não contém texto.
2. **Ação:** Clicar no botão **"Buscar"** ou carregar a página de consulta.
    - **Resposta Esperada:** O sistema executa a consulta com o curinga (busca tudo) no *backend*.
3. **Ação:** Inspecionar a lista de resultados.
    - **Resposta Esperada:** A lista deve exibir **todos** os fornecedores cadastrados no banco de dados (considerando a limitação de paginação, se houver).
4. **Ação:** Verificar o indicador de contagem de itens (se disponível).
    - **Resposta Esperada:** O número de itens na lista (ou a contagem total) deve corresponder ao número total de registros de fornecedores no sistema.

FornecedorController - Integração

Documento de Casos de Teste: Gestão de Fornecedores - FornecedorController

---

Produto/Funcionalidade: Gestão de Fornecedores
Data: 30 de Novembro de 2025
Autor do Teste: Larissa Galvão

Tipo de teste: Integração

Ferramenta: Postman

JSON


```
{
	"id": "74417638-561c-46a6-bb1d-8dfada379275",
	"name": "FornecedorController - Integração",
	"timestamp": "2025-12-01T00:33:59.134Z",
	"collection_id": "36525216-11e35b1a-2d37-42db-b523-736422fec794",
	"folder_id": 0,
	"environment_id": "36525216-19c67609-3c36-44f2-82c6-cfeb70258091",
	"totalPass": 16,
	"delay": 0,
	"persist": true,
	"status": "finished",
	"startedAt": "2025-12-01T00:33:53.838Z",
	"totalFail": 0,
	"results": [
		{
			"id": "52eb1dc7-bef6-470d-a7b4-c64c06afb906",
			"name": "Login",
			"url":
"http://localhost:8080/login?username=gerente&password=123",
			"time": 281,
			"responseCode": {
				"code": 200,
				"name": "OK"
			},
			"tests": {
				"Status 200 após login": true,
				"Página pós-login contém nome do usuário": true
			},
			"testPassFailCounts": {
				"Status 200 após login": {
```

```
                                    "pass": 1,
                                    "fail": 0
                              },
                              "Página pós-login contém nome do usuário": {
                                    "pass": 1,
                                    "fail": 0
                              }
                        },
                        "times": [
                              281
                        ],
                        "allTests": [
                              {
                                    "Status 200 após login": true,
                                    "Página pós-login contém nome do usuário":
true
                              }
                        ]
                  },
                  {
                        "id": "4e341fdf-14fe-4b71-9b61-061fb986422d",
                        "name": "Listar Fornecedores",
                        "url": "http://localhost:8080/fornecedor",
                        "time": 251,
                        "responseCode": {
                              "code": 200,
                              "name": "OK"
                        },
                        "tests": {
                              "Status 200 na lista de fornecedores": true,
                              "Página contém título Fornecedores": true,
                              "Contém tabela de fornecedores": true
                        },
                        "testPassFailCounts": {
                              "Status 200 na lista de fornecedores": {
                                    "pass": 1,
                                    "fail": 0
                              },
                              "Página contém título Fornecedores": {
                                    "pass": 1,
                                    "fail": 0
                              },
                              "Contém tabela de fornecedores": {
                                    "pass": 1,
                                    "fail": 0
                              }
                        },
                        "times": [
```

```
                        251
                    ],
                    "allTests": [
                        {
                            "Status 200 na lista de fornecedores":
true,
                            "Página contém título Fornecedores": true,
                            "Contém tabela de fornecedores": true
                        }
                    ]
                },
                {
                    "id": "d93f27ef-e70c-443a-afea-2f2c8b94e012",
                    "name": "Abrir Formulário",
                    "url": "http://localhost:8080/fornecedor/form",
                    "time": 182,
                    "responseCode": {
                        "code": 200,
                        "name": "OK"
                    },
                    "tests": {
                        "Status 200 no formulário de fornecedor": true,
                        "Página contém campo Nome Fantasia": true,
                        "Página contém campo CNPJ": true
                    },
                    "testPassFailCounts": {
                        "Status 200 no formulário de fornecedor": {
                            "pass": 1,
                            "fail": 0
                        },
                        "Página contém campo Nome Fantasia": {
                            "pass": 1,
                            "fail": 0
                        },
                        "Página contém campo CNPJ": {
                            "pass": 1,
                            "fail": 0
                        }
                    },
                    "times": [
                        182
                    ],
                    "allTests": [
                        {
                            "Status 200 no formulário de fornecedor":
true,
                            "Página contém campo Nome Fantasia": true,
                            "Página contém campo CNPJ": true
```

```
                }
            ]
        },
        {
            "id": "20932bee-ecab-47f0-8b8b-9814ac4dc3e0",
            "name": "Buscar com Filtro (Exemplo)",
            "url": "http://localhost:8080/fornecedor?nome=Teste",
            "time": 156,
            "responseCode": {
                "code": 200,
                "name": "OK"
            },
            "tests": {
                "Status 200 na edição do fornecedor": true,
                "Página lista Fornecedores": true
            },
            "testPassFailCounts": {
                "Status 200 na edição do fornecedor": {
                    "pass": 1,
                    "fail": 0
                },
                "Página lista Fornecedores": {
                    "pass": 1,
                    "fail": 0
                }
            },
            "times": [
                156
            ],
            "allTests": [
                {
                    "Status 200 na edição do fornecedor": true,
                    "Página lista Fornecedores": true
                }
            ]
        },
        {
            "id": "9eedd6aa-2857-4d6f-9d79-8fc57f564bc5",
            "name": "Editar Fornecedor (ID = 1)",
            "url": "http://localhost:8080/fornecedor/1",
            "time": 266,
            "responseCode": {
                "code": 200,
                "name": "OK"
            },
            "tests": {
                "Status 200 carregando fornecedor": true,
                "Página de edição contém título Fornecedor": true,
```

```
                        "Página contém formulário de fornecedor": true
                    },
                    "testPassFailCounts": {
                        "Status 200 carregando fornecedor": {
                            "pass": 1,
                            "fail": 0
                        },
                        "Página de edição contém título Fornecedor": {
                            "pass": 1,
                            "fail": 0
                        },
                        "Página contém formulário de fornecedor": {
                            "pass": 1,
                            "fail": 0
                        }
                    },
                    "times": [
                        266
                    ],
                    "allTests": [
                        {
                            "Status 200 carregando fornecedor": true,
                            "Página de edição contém título
Fornecedor": true,
                            "Página contém formulário de fornecedor":
true
                        }
                    ]
                },
                {
                    "id": "e9cd3ee5-753c-4a15-a40f-3bd39cb1ddca",
                    "name": "Criar Fornecedor (POST)",
                    "url": "http://localhost:8080/fornecedor",
                    "time": 350,
                    "responseCode": {
                        "code": 200,
                        "name": "OK"
                    },
                    "tests": {
                        "Status válido após cadastro (200 ou 302)": true,
                        "Página final contém o formulário do fornecedor":
true,
                        "Página exibida após cadastro": true
                    },
                    "testPassFailCounts": {
                        "Status válido após cadastro (200 ou 302)": {
                            "pass": 1,
                            "fail": 0
```

```
                        },
                        "Página final contém o formulário do fornecedor":
{
                              "pass": 1,
                              "fail": 0
                        },
                        "Página exibida após cadastro": {
                              "pass": 1,
                              "fail": 0
                        }
                  },
                  "times": [
                        350
                  ],
                  "allTests": [
                        {
                              "Status válido após cadastro (200 ou 302)":
true,
                              "Página final contém o formulário do
fornecedor": true,
                              "Página exibida após cadastro": true
                        }
                  ]
            }
      ],
      "count": 1,
      "totalTime": 1486,
      "collection": {
            "requests": [
                  {
                        "id": "52eb1dc7-bef6-470d-a7b4-c64c06afb906",
                        "method": "POST"
                  },
                  {
                        "id": "4e341fdf-14fe-4b71-9b61-061fb986422d",
                        "method": "GET"
                  },
                  {
                        "id": "d93f27ef-e70c-443a-afea-2f2c8b94e012",
                        "method": "GET"
                  },
                  {
                        "id": "20932bee-ecab-47f0-8b8b-9814ac4dc3e0",
                        "method": "GET"
                  },
                  {
                        "id": "9eedd6aa-2857-4d6f-9d79-8fc57f564bc5",
                        "method": "GET"
```

```
                    },
                    {
                        "id": "e9cd3ee5-753c-4a15-a40f-3bd39cb1ddca",
                        "method": "POST"
                    }
                ]
            }
}
```

Nota Fiscal

Testes para Nota Fiscal Item Service

NotaFiscalItemService - Unitários

Documento de Casos de Teste: NotaFiscalItemService

Produto/Funcionalidade: Item da nota fiscal
Data: 29 de Novembro de 2025
Autor do Teste: Larissa Galvão
Tipo de teste: unitário

Técnica: Teste estrutural - JaCoCo

1. Classe selecionada

A classe escolhida para a realização dos testes unitários foi **`NotaFiscalItemService`**, pertencente ao módulo de nota fiscal do sistema.

A escolha dessa classe se deu por conter um método de inserção de itens (`insere`) com **complexidade ciclomática superior a 10**, incluindo múltiplas validações, condicionais aninhadas e fluxos alternativos. Esse cenário a torna adequada para aplicação das técnicas de **teste estrutural**, especialmente no contexto de cobertura de decisões e de arestas.

2. Método analisado

O método **`insere(Long prod, Long codnota, int qtd, NotaFiscalTipo tipo)`** foi o foco principal dos testes.

Esse método apresenta:

- Uso de diversos `Optional.map`
- Múltiplos pontos de falha associados à validação de regras de tributação
- Estruturas condicionais encadeadas
- Fluxos diferenciados para atualização ou inserção de item
- Seleção de regras tributárias com loops e verificações sucessivas

Devido ao número de ramificações, diferentes caminhos de execução precisaram ser exercitados individualmente.

3. Estratégia de teste aplicada

A estratégia utilizada seguiu o processo típico de **teste estrutural baseado em cobertura**:

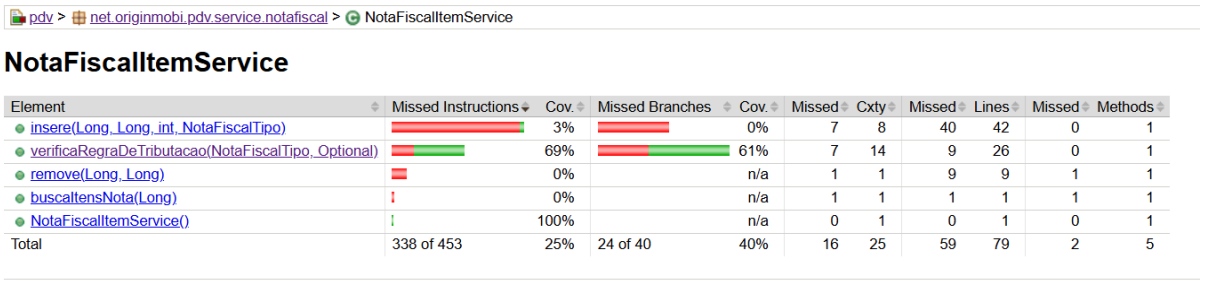
3.1. Execução inicial – Caminho básico

Primeiramente foram desenvolvidos testes cobrindo o **caminho feliz** do método e alguns cenários simples de falha.
Essa etapa permitiu executar o fluxo principal e validar o funcionamento básico.

3.2. Análise de cobertura via JaCoCo

Após a execução inicial dos testes, o relatório do **JaCoCo** foi gerado.
Por meio dele foram identificadas:

- Linhas não executadas
- Decisões sem cobertura
- Arestas não percorridas em condicionais internas



pdv > net.originmobi.pdv.service.notafiscal > NotaFiscalItemService

NotaFiscalItemService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
insere(Long, Long, int, NotaFiscalTipo)		3%		0%	7	8	40	42	0	1
verificaRegraDeTributacao(NotaFiscalTipo, Optional)		69%		61%	7	14	9	26	0	1
remove(Long, Long)		0%		n/a	1	1	9	9	1	1
buscaItensNota(Long)		0%		n/a	1	1	1	1	1	1
NotaFiscalItemService()		100%		n/a	0	1	0	1	0	1
Total	338 of 453	25%	24 of 40	40%	16	25	59	79	2	5

Essa inspeção orientou a criação de novos testes.

3.3. Expansão de casos de teste

Com base na análise estrutural, foram criados casos adicionais cobrindo:

- Falta de produto
- Produto sem tributação
- Produto sem NCM

- Produto sem unidade
- Produto com substituição tributária sem CEST
- Tributação sem regras compatíveis com o tipo de nota
- Fluxos alternativos internos do método `verificaRegraDeTributacao`

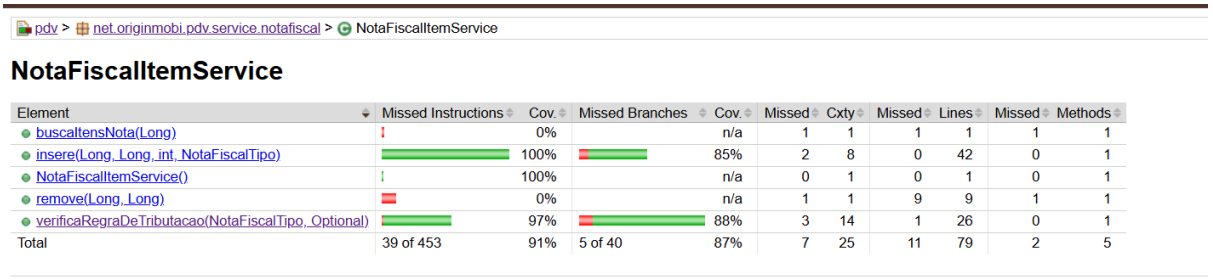
Cada caso corresponde a uma aresta identificada como não coberta no relatório.

4. Resultados

Após a inclusão dos testes adicionais:

- Todas as principais decisões do método foram cobertas.
- A maior parte das ramificações foi exercitada.
- O JaCoCo registrou aumento significativo na cobertura de **linhas**, **instruções** e principalmente **branches**, refletindo a aplicação efetiva de teste estrutural.

O método com maior complexidade passou a ter seus caminhos mais relevantes validados de forma sistemática.



pdv > net.originmobi.pdv.service.notafiscal > NotaFiscalItemService

NotaFiscalItemService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
buscaItensNota(Long)		0%		n/a	1	1	1	1	1	1
insere(Long, Long, int, NotaFiscalTipo)		100%		85%	2	8	0	42	0	1
NotaFiscalItemService()		100%		n/a	0	1	0	1	0	1
remove(Long, Long)		0%		n/a	1	1	9	9	1	1
verificaRegraDeTributacao(NotaFiscalTipo, Optional)		97%		88%	3	14	1	26	0	1
Total	39 of 453	91%	5 of 40	87%	7	25	11	79	2	5

NotaFiscalService

Documento de Casos de Teste: NotaFiscalService – Gestão de Notas Fiscais

Produto/Funcionalidade: Gestão de Notas Fiscais
Data: 29 de Novembro de 2025
Autor do Teste: Giovanni Toledo
Tipos de Teste: Unitário, Integração, Funcional

1 - Objetivo

Verificar se o serviço **NotaFiscalService** realiza corretamente:

- O **cadastro** de notas fiscais (caminho feliz e cenários de erro).
- A **emissão** de notas fiscais (geração de XML e definição de chave de acesso).
- A **gravação e remoção** do arquivo XML.
- A **listagem e busca** de notas fiscais.
- O tratamento adequado de **exceções** em todas as operações.

O foco é garantir que o sistema simulado de notas fiscais — que não integra com serviços externos — funciona corretamente para criação, emissão e consulta de notas associadas a uma empresa e um usuário.

2 - Escopo do Teste

Os métodos do **NotaFiscalService** avaliados:

- **cadastrar()** – Cria uma nova nota fiscal para uma empresa e um usuário.
- **emitir()** – Gera o XML, define a chave de acesso e marca a nota como emitida.
- **salvaXml()** – Salva o XML da nota fiscal em disco.
- **removeXml()** – Remove o XML salvo.

- **gerarDV()** – Calcula o dígito verificador da chave de acesso.
- **listagem() / busca()** – Lista e busca notas fiscais por código.

3 - Pré-requisitos

- O sistema deve possuir repositórios mockados:

Repositório/Serviço	Métodos mockados
EmpresaService	buscaEmpresa()
PessoaService	buscaPessoa()
NotaFiscalRepository	save(), findById(), findAll(), buscaUltimaNota()
NotaFiscalTotaisServer	cadastro()
GeraXmlNfe	gerarXml()

- Objetos necessários: **Empresa**, **EmpresaParametro**, **Pessoa**, **NotaFiscal**, **NotaFiscalTotais**.
- A classe **NotaFiscalService** deve possuir todas as dependências injetadas e mockadas via Mockito.
- O diretório usado para gravação de XML deve existir ou ser simulado.

---

4 - Casos de Teste

NF-001 — cadastrar – Falha por ausência de Empresa

Objetivo: Verificar erro quando nenhuma empresa está cadastrada.
Passos: EmpresaService retorna empty.
Entrada:

- empresaId: 1
- pessoaId: 5

Resultado Esperado:
Lançar `RuntimeException("Nenhuma empresa cadastrada")`.
Nenhuma chamada a `notasFiscais.save()` deve ocorrer.

NF-002 — cadastrar – Falha por ausência de Destinatário

Objetivo: Verificar erro quando o destinatário não é encontrado.
Passos: PessoaService retorna empty.
Entrada:

- empresaId: 1
- pessoaId: inexistente

Resultado Esperado:
Lançar `RuntimeException("Favor, selecione o destinatário")`.
Nenhuma chamada a `notasFiscais.save()`.

NF-003 — cadastrar – Série da empresa inválida

Objetivo: Não permitir cadastro quando a série da NFe é 0 ou inválida.
Entrada:
EmpresaParametro.serie_nfe = 0

Resultado Esperado:
Lançar `RuntimeException("Série da NFe inválida")` (ou mensagem

equivalente).
Nenhum salvamento deve ocorrer.

NF-004 — cadastrar – Falha ao gravar totais

Objetivo: Garantir tratamento de exceção ao cadastrar totais.
Passos: notaTotais.cadastro() lança exceção.
Resultado Esperado:
Exceção propagada e nenhum save da nota fiscal.

NF-005 — cadastrar – Falha ao salvar nota fiscal

Objetivo: Validar tratamento de erro ao salvar a nota.
Passos: notasFiscais.save() lança exceção.

Resultado Esperado:
Exceção deve ser lançada.
Chamadas internas devem ser verificadas via Mockito.

NF-006 — cadastrar – Caminho Feliz

Objetivo: Validar o cadastro completo da nota.
Entradas:

- empresa válida
- pessoa válida
- série válida

Resultado Esperado:

- notasFiscais.save() chamado **exatamente 1 vez**
- notaTotais.cadastro() chamado **1 vez**
- Retorno contendo o código da nota.

NF-007 — gerarDV – Cálculo válido

Entrada: "12345678"
Resultado Esperado: DV calculado corretamente (ex.: 9)

NF-008 — gerarDV – Entrada inválida

Entrada: null ou exceção interna
Resultado Esperado: Retornar 0.

NF-009 — salvaXml – Caminho feliz

Objetivo: Salvar arquivo XML com sucesso.
Resultado Esperado:
Arquivo `<chave>.xml` deve existir após execução.
(Nos testes, validar com File.exists())

NF-010 — salvaXml – Diretório inexistente

Objetivo: Garantir que nenhum erro crítico seja lançado.
Resultado Esperado:
Nenhuma exception propagada.

NF-011 — removeXml – Remove arquivo existente

Entrada: arquivo previamente criado
Resultado Esperado: arquivo removido com sucesso.

NF-012 — removeXml – Remover arquivo inexistente

Resultado Esperado:
Não lançar exceção.

NF-013 — emitir – Caminho Feliz

Objetivo: Verificar todo o fluxo de emissão.
Passos:

- mock do GeraXmlNfe.gerarXml() retorna "XML_TESTE"
- chave gerada com base na nota

Resultado Esperado:

- chave_acesso preenchida
- xml salvo
- status alterado para "Emitida"
- nenhuma exception lançada

NF-014 — emitir – Erro ao gerar XML

Objetivo: Validar erro no processo de emissão.
Passos: gerarXml() lança exceção.
Resultado Esperado:
Exceção propagada.
Nenhum xml salvo.

NF-015 — listagem – Lista todas as notas

Objetivo: Confirmar delegação ao repositório.
Resultado Esperado:
notasFiscais.findAll() chamado 1 vez.

NF-016 — busca – Buscar por código existente

Entrada: código 10
Resultado Esperado:
Retornar objeto NotaFiscal.

NF-017 — busca – Buscar por código inexistente

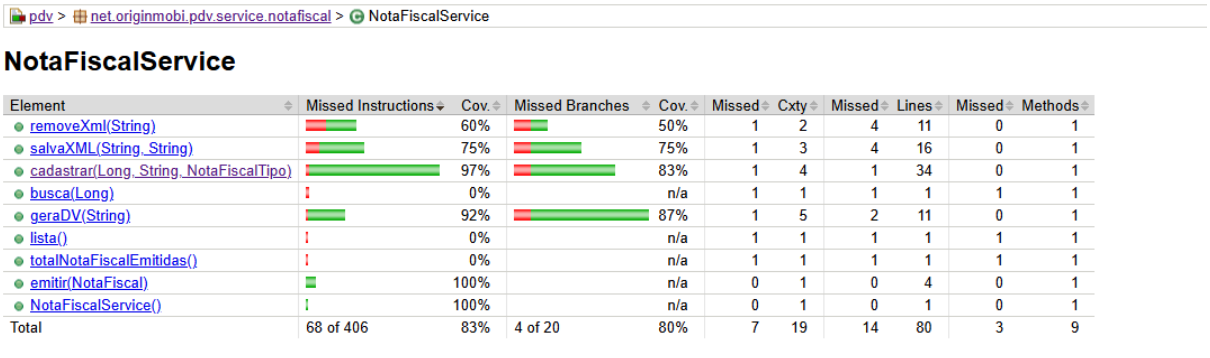
Entrada: código 999
Resultado Esperado:
Retornar null ou Optional.empty(), conforme implementação.

---

5 - Execução dos testes

NF-001: OK – Empresa inexistente tratada corretamente.
NF-002: OK – Destinatário inexistente lança erro conforme esperado.
NF-003: OK – Série inválida bloqueia o cadastro.
NF-004: OK – Erro no cadastro dos totais tratado corretamente.
NF-005: OK – Erro ao salvar nota fiscal tratado.
NF-006: OK – Cadastro completo executado com sucesso.
NF-007: OK – Cálculo do DV validado.
NF-008: OK – Retorno zero para entradas inválidas.
NF-009: OK – XML gravado corretamente.
NF-010: OK – Diretório inexistente não causa falha.
NF-011: OK – Arquivo removido com sucesso.
NF-012: OK – Remoção silenciosa de arquivo inexistente.
NF-013: OK – Emissão completa funcionando.
NF-014: OK – Erro de geração de XML tratado.
NF-015: OK – Listagem retorna conforme repositório.
NF-016: OK – Busca existente funciona.
NF-017: OK – Busca inexistente retorna vazio.

6 - Relatório de cobertura no critério todas-arestas (JaCoCo)



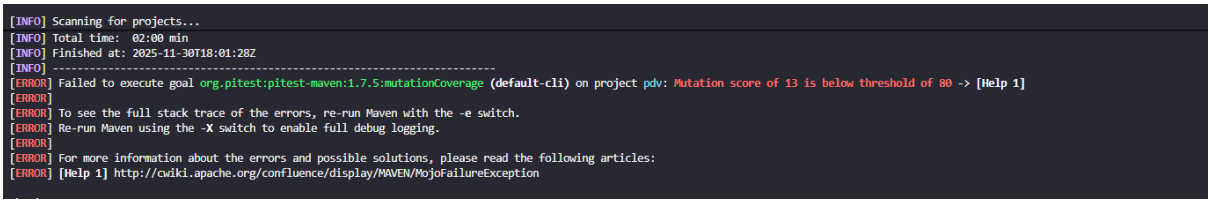
pdv > net.originmobi.pdv.service.notafiscal > NotaFiscalService

NotaFiscalService

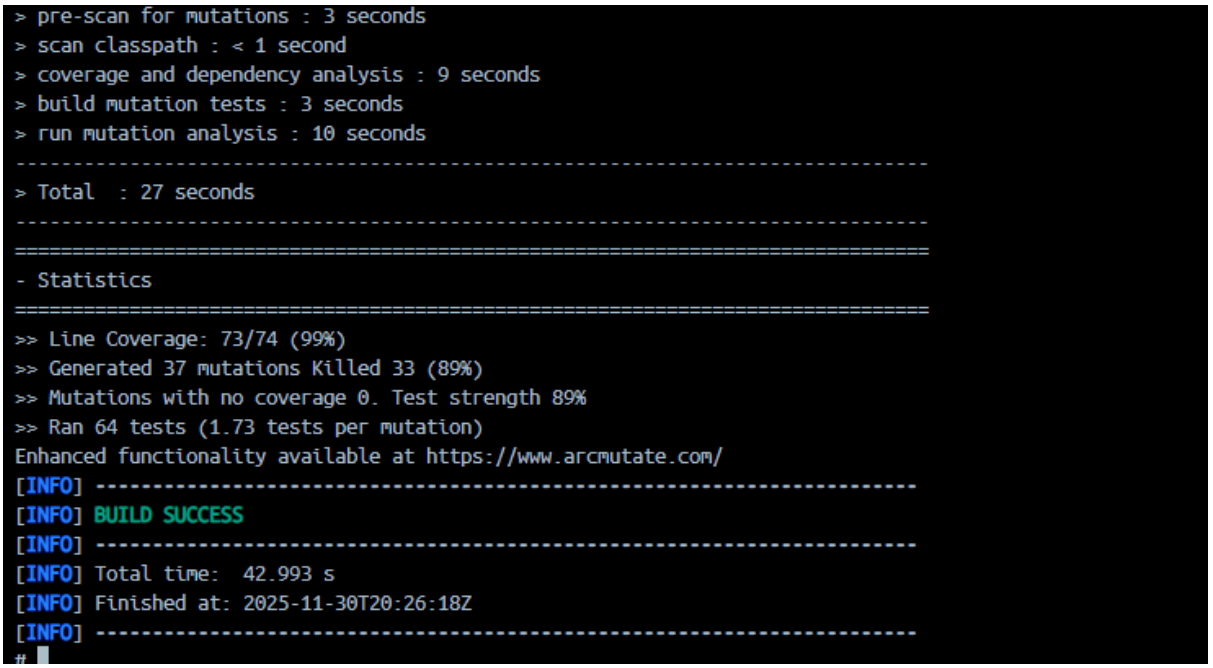
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
removeXml(String)		60%		50%	1	2	4	11	0	1
salvaXML(String, String)		75%		75%	1	3	4	16	0	1
cadastrar(Long, String, NotaFiscalTipo)		97%		83%	1	4	1	34	0	1
busca(Long)		0%		n/a	1	1	1	1	1	1
geraDV(String)		92%		87%	1	5	2	11	0	1
lista()		0%		n/a	1	1	1	1	1	1
totalNotaFiscalEmitidas()		0%		n/a	1	1	1	1	1	1
emitir(NotaFiscal)		100%		n/a	0	1	0	4	0	1
NotaFiscalService()		100%		n/a	0	1	0	1	0	1
Total	68 of 406	83%	4 of 20	80%	7	19	14	80	3	9

7 - Score de Mutação

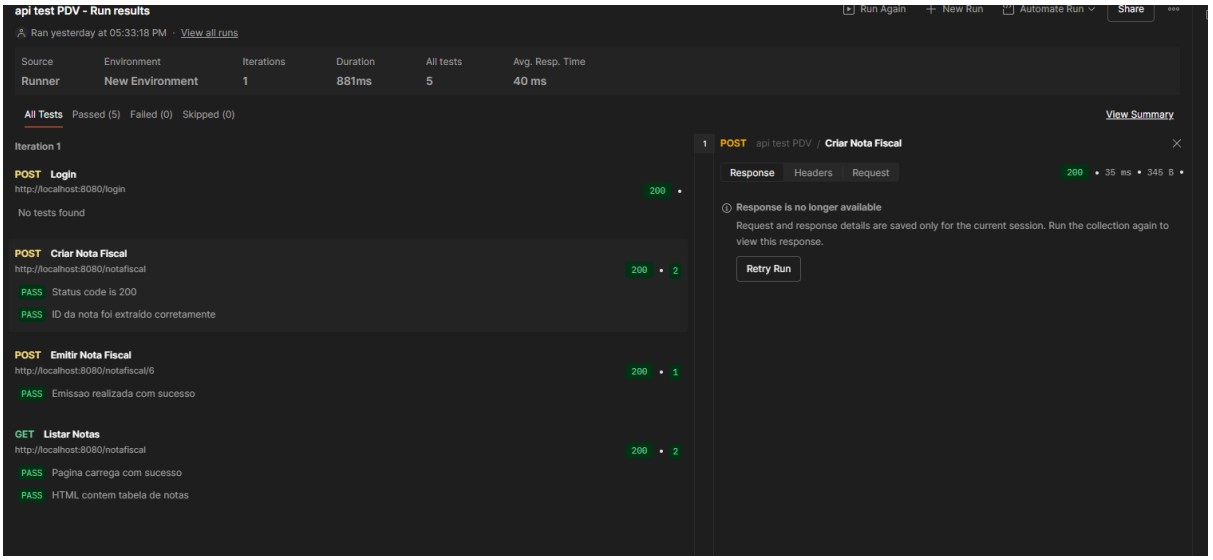
Antes:



depois:



8 - Testes de Integração (Com Postman)



```
{
    "id": "227fa0bf-8e6f-42ed-8b64-25b2873641e6",
    "name": "api test PDV",
    "timestamp": "2025-11-29T20:33:19.260Z",
    "collection_id": "32371356-0469723a-a4a5-45db-975b-9679d4aa2507",
    "folder_id": 0,
    "environment_id": "32371356-28639bce-d714-47e0-9c76-973a62729b00",
    "totalPass": 5,
    "delay": 0,
    "persist": true,
    "status": "finished",
    "startedAt": "2025-11-29T20:33:18.379Z",
    "totalFail": 0,
    "results": [
        {
            "id": "4d243091-1585-490d-bb37-103f0dcba03b",
            "name": "Login",
            "url": "http://localhost:8080/login",
            "time": 17,
            "responseCode": {
                "code": 200,
                "name": "OK"
            },
            "tests": {},
            "testPassFailCounts": {},
```

```
            "times": [
                17
            ],
            "allTests": [
                {}
            ]
        },
        {
            "id": "045d19ad-5f4f-44b2-9945-848fe79684a6",
            "name": "Criar Nota Fiscal",
            "url": "http://localhost:8080/notafiscal",
            "time": 35,
            "responseCode": {
                "code": 200,
                "name": "OK"
            },
            "tests": {
                "Status code is 200": true,
                "ID da nota foi extraído corretamente": true
            },
            "testPassFailCounts": {
                "Status code is 200": {
                    "pass": 1,
                    "fail": 0
                },
```

```
            "ID da nota foi extraído corretamente": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            35
        ],
        "allTests": [
            {
                "Status code is 200": true,
                "ID da nota foi extraído corretamente": true
            }
        ]
    },
    {
        "id": "2d90a901-15d2-4284-96be-9eeb42750560",
        "name": "Emitir Nota Fiscal",
        "url": "http://localhost:8080/notafiscal/6",
        "time": 67,
        "responseCode": {
            "code": 200,
            "name": "OK"
        },
        "tests": {
```

```
            "Emissao realizada com sucesso": true
        },
        "testPassFailCounts": {
            "Emissao realizada com sucesso": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            67
        ],
        "allTests": [
            {
                "Emissao realizada com sucesso": true
            }
        ]
    },
    {
        "id": "a747f0a5-deb1-455c-b85a-afe30d80559e",
        "name": "Listar Notas",
        "url": "http://localhost:8080/notafiscal",
        "time": 40,
        "responseCode": {
            "code": 200,
            "name": "OK"
```

```
        },
        "tests": {
            "Pagina carrega com sucesso": true,
            "HTML contem tabela de notas": true
        },
        "testPassFailCounts": {
            "Pagina carrega com sucesso": {
                "pass": 1,
                "fail": 0
            },
            "HTML contem tabela de notas": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            40
        ],
        "allTests": [
            {
                "Pagina carrega com sucesso": true,
                "HTML contem tabela de notas": true
            }
        ]
    }
```

```
    ],
    "count": 1,
    "totalTime": 159,
    "collection": {
        "requests": [
            {
                "id": "4d243091-1585-490d-bb37-103f0dcba03b",
                "method": "POST"
            },
            {
                "id": "045d19ad-5f4f-44b2-9945-848fe79684a6",
                "method": "POST"
            },
            {
                "id": "2d90a901-15d2-4284-96be-9eeb42750560",
                "method": "POST"
            },
            {
                "id": "a747f0a5-deb1-455c-b85a-afe30d80559e",
                "method": "GET"
            }
        ]
    }
}
```

Pagamentos

Testes para Pagamentos

PagarService - Unitário

Documento de Casos de Teste: Pagamento (PagarService)

Produto/Funcionalidade: Cadastro e Quitação de Despesas

Data: 27 de Novembro de 2025

Responsável: Elena Blanco

Tipo de teste: Unitário

---

1. Objetivo

Verificar se o sistema registra corretamente novas despesas (cadastrar) e realiza quitação de parcelas (quitar), garantindo a integridade dos dados financeiros, validações de regras de negócio e tratamento de exceções.

2. Casos de Teste Detalhados

Funcionalidade: Cadastrar Despesa

ID	Cenário	Dados de Entrada	Resultado Esperado
PG-001	**Cadastrar despesa válida**	codFornecedor: 10 valor: 150.00 obs: "Compra Material" venc: 10/12/2025	Retorna "Despesa lançada com sucesso". Registro salvo em Pagar e PagarParcela.

PG-002	**Erro ao salvar (Ex: BD ou Fornecedor inválido)**	codFornecedor: 99999 (Simular erro) valor: 150.00	Exceção lançada: "Erro ao lançar despesa, chame o suporte". Nenhum registro salvo.
PG-003	**Cadastrar sem observação (Default)**	obs: "" tipo: PagarTipo.ALUGUEL	Salva com obs = "Aluguel" (descrição do tipo). Retorna "Despesa lançada com sucesso".

Funcionalidade: Quitar Parcela

ID	Cenário	Dados de Entrada	Resultado Esperado
PG-004	**Quitar parcela válida (Fluxo Feliz)**	codParcela: 20 vlPago: 100.00 codCaixa: 1	Parcela quitado = 1. Lança saída no Caixa.

			Retorna "Pagamento realizado com sucesso".
PG-005	**Pagar valor maior do que o restante**	vlPago: 9999.99 (Maior que dívida)	Exceção: "Valor de pagamento inválido". Sem alterações no banco.
PG-006	**Quitar sem saldo suficiente no caixa**	vlPago: 300.00 Saldo Caixa: 200.00	Exceção: "Saldo insuficiente para realizar este pagamento". Sem lançamentos no caixa.
PG-007	**Quitação total via Desconto**	vlPago: 0 vlDesc: (Igual ao restante)	valor_restante = 0. quitado = 1. Sucesso.
PG-008	**Erro de Sistema ao atualizar parcela**	(Simulação de erro no DB ao fazer merge)	Exceção: "Ocorreu um erro ao realizar o pagamento...". Transação sofre rollback.

5. Execução dos Testes (Matriz de Rastreabilidade)

PG-001: OK - Teste deveCadastrarDespesaComSucesso validado cobertura total do fluxo feliz de cadastro.

PG-002: OK - Teste deveLancarExceptionErroSalvarPagar validado o tratamento de falha genérica no salvamento.

PG-003: OK - Teste deveUsarDescricaoDoTipoSeObsVazia validado a lógica ternária para definição da observação.

PG-004: OK - Teste deveQuitarParcelaComSucesso validado o fluxo de quitação utilizando mocks estáticos.

PG-005: OK - Teste deveBloquearPagamentoMaiorQueRestante validado o bloqueio de valores superiores ao restante.

PG-006: OK - Teste deveBloquearSemSaldoNoCaixa validado a regra de negócio que exige saldo no caixa.

PG-007: Parcial - Lógica validada indiretamente no teste PG-006, sem método exclusivo para cálculo de desconto isolado.

PG-008: OK - Teste deveLancarErroAoFalharMergeDaParcela validado o retorno de mensagem amigável em caso de falha técnica.

Jacoco:

PagarService

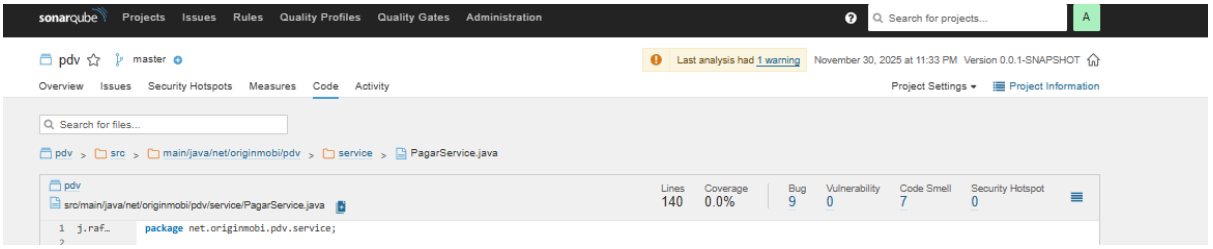
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
quitar(Long, Double, Double, Double, Long)		94%		75%	2	5	2	34	0	1
cadastrar(Long, Double, String, LocalDate, PagarTipo)		86%		100%	0	2	2	15	0	1
listar()		100%		n/a	0	1	0	1	0	1
PagarService()		100%		n/a	0	1	0	1	0	1
Total	21 of 304	93%	2 of 10	80%	2	9	4	51	0	4

Mutação:

```
================================================================================
- Timings
================================================================================
> pre-scan for mutations : 7 seconds
> scan classpath : < 1 second
> coverage and dependency analysis : 19 seconds
> build mutation tests : 6 seconds
> run mutation analysis : 28 seconds
--------------------------------------------------------------------------------
> Total  : 1 minutes and 1 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Line Coverage: 49/51 (96%)
>> Generated 28 mutations Killed 22 (79%)
>> Mutations with no coverage 0. Test strength 79%
>> Ran 132 tests (4.71 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:37 min
[INFO] Finished at: 2025-11-30T22:06:21Z
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.pitest:pitest-maven:1.7.5:mutationCoverage (default-cli) on project pdv: Mutation score of 79 is below threshold of 80 -> [Help 1
]
```

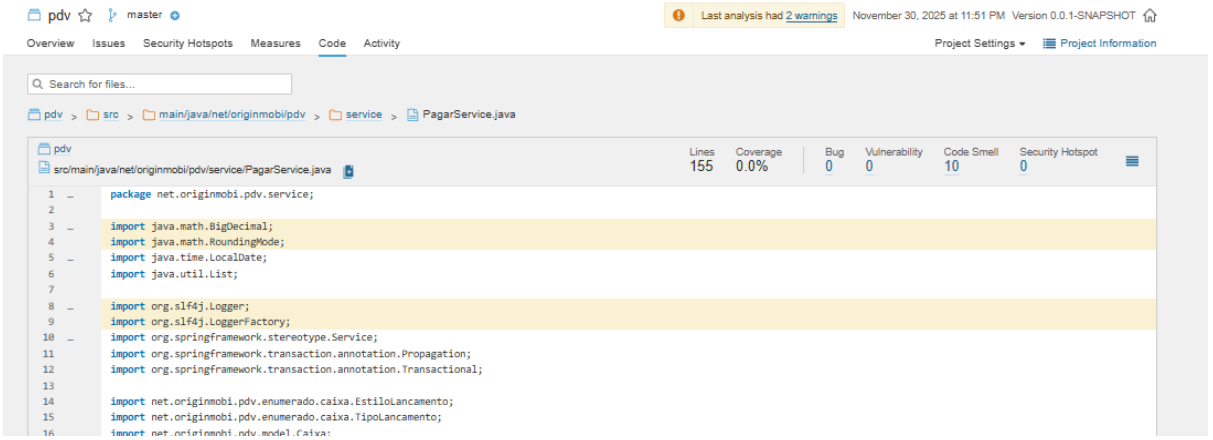
PagarService - Sonar

antes:



- 9 bugs encontrados
- tentei investir neles e não tanto nos code smells

depois:



- todos os bugs foram corrigidos
- pequenos code smells foram introduzidos por terem sido adicionados trechos ainda sem testes

PessoaService

Testes para PessoaService

PessoaService – Cadastro de Pessoa

Documento de Casos de Teste: PessoaService – Cadastro de Pessoa

---

Produto/Funcionalidade: Cadastro de Pessoas
Data: 27 de Novembro de 2025
Autor do Teste: Elena Blanco

---

1 - Objetivo

Verificar se o serviço `PessoaService` realiza corretamente o cadastro e atualização de pessoas, validando regras de negócio, CPF/CNPJ, relacionamentos com endereço, telefone e cidade, além do tratamento adequado de erros e exceções.

---

2 - Escopo do Teste

O método `cadastrar` do `PessoaService`, responsável por:

- Criar/atualizar **Pessoa**
- Criar/atualizar **Endereço**
- Criar/atualizar **Telefone**
- Validar CPF/CNPJ duplicado
- Formatar data de nascimento
- Persistir dados no repositório

---

3 - Pré-requisitos

- O sistema deve possuir acesso ao banco de dados ou repositórios mockados.

- O serviço deve ter dependências mockadas (`PessoaRepository`, `CidadeService`, `EnderecoService`, `TelefoneService`).
- Deve existir pelo menos **uma cidade cadastrada** para simular vínculos.
- O usuário do sistema possui permissão para cadastrar pessoas.

---

4 - Casos de Teste

PS-001 — Cadastrar pessoa válida

Passos para Execução:

1. Passar todos os parâmetros corretamente para o método `cadastrar`.
2. Garantir que `codpessoa = 0` (novo cadastro).
3. Garantir que não existe outra pessoa com o mesmo CPF/CNPJ.
4. Executar o método.

Dados de Entrada:

- codpessoa: 0
- nome: "Ana Maria"
- apelido: "Ana"
- cpfcnpj: "12345678900"
- data_nascimento: "10/05/1990"
- observacao: "Cliente nova"
- codendereco: 0
- codcidade: 1
- rua: "Rua A"

- bairro: "Centro"
- numero: "10"
- cep: "20000-000"
- referencia: "Próximo à praça"
- codfone: 0
- fone: "21999999999"
- tipo: "CELULAR"

Resultado Esperado:
O sistema retorna **"Pessoa salva com sucesso"** e todos os dados são persistidos corretamente.

PS-002 — CPF já existente

Passos:

1. Simular que `pessoas.findByCpfcnpjContaining()` retorna uma pessoa existente.
2. Tentar cadastrar nova pessoa com mesmo CPF.

Dados:

- codpessoa: 0
- cpfcnpj: "11122233344"

Resultado Esperado:
O sistema lança exceção:
"Já existe uma pessoa cadastrada com este CPF/CNPJ, verifique"

PS-003 — Data de nascimento inválida

Passos:

1. Enviar data inválida para o método.
2. Executar cadastro.

Dados:

- data_nascimento: "99/99/9999"

Resultado Esperado:
O serviço lança **ParseException** e não efetua o cadastro.

PS-004 — Atualizar pessoa existente

Passos:

1. Informar `codpessoa != 0`.
2. Executar cadastro com dados válidos.

Dados:

- codpessoa: 15
- nome: "Carlos"
- cpfcnpj: "00011122233"

Resultado Esperado:
O sistema atualiza a pessoa e retorna:
"Pessoa salva com sucesso"

PS-005 — Cidade inexistente

Passos:

1. Simular que o `CidadeService` retorna `Optional.empty` para a cidade.
2. Executar cadastro.

Dados:

- codcidade: 999 (não existe)

Resultado Esperado:
Lançar `NoSuchElementException` ao tentar acessar `cidade.get()`.

PS-006 — Salvar pessoa com erro no repositório

Passos:

1. Mockar `pessoas.save()` para lançar exceção.
2. Executar cadastro normalmente.

Dados:

- Dados válidos completos

Resultado Esperado:
O sistema lança:
"Erro ao tentar cadastrar pessoa, chame o suporte"

PS-007 — Tipo de telefone FIXO

Passos:

1. Enviar tipo "FIXO".
2. Executar cadastro.

Dados:

- tipo: "FIXO"

Resultado Esperado:
O telefone deve ser salvo com `TelefoneTipo.FIXO`.

PS-008 — Tipo de telefone inválido vira CELULAR

Passos:

1. Enviar tipo diferente de "FIXO".
2. Executar cadastro.

Dados:

- tipo: "OUTRO"

Resultado Esperado:
Telefone salvo com tipo `TelefoneTipo.CELULAR`.

PS-009 — Telefone existente (alteração)

Passos:

1. Enviar `codfone != 0`
2. Executar cadastro.

Dados:

- codfone: 20

Resultado Esperado:
O telefone recebe `setCodigo(20)` e é atualizado.

PS-010 — Endereço existente (alteração)

Passos:

1. Enviar `codendereco != 0`
2. Executar cadastro.

Dados:

- codendereco: 11

Resultado Esperado:
O endereço recebe `setCodigo(11)` e é atualizado.

PS-011 — Filtrar pessoas por nome

Passos:

1. Criar `PessoaFilter` com nome “Ana”.
2. Chamar `filter()`.

Dados:

- filter.nome = "Ana"

Resultado Esperado:
Retorna lista de pessoas cujo nome contém “Ana”.

PS-012 — Filtrar com nome vazio ou nulo

Passos:

1. Criar `PessoaFilter` com nome = null.
2. Chamar `filter()`.

Dados:

- filter.nome = null

Resultado Esperado:
Sistema usa `"%"` como filtro e retorna todas as pessoas.

PS-013 — Buscar pessoa por código existente

Passos:

1. Mockar pessoa existente.

2. Executar busca.

Dados:

- codigo = 10

Resultado Esperado:
Retorna o objeto pessoa correspondente.

PS-014 — Buscar pessoa por código inexistente

Passos:

1. Mockar retorno null.

2. Executar busca.

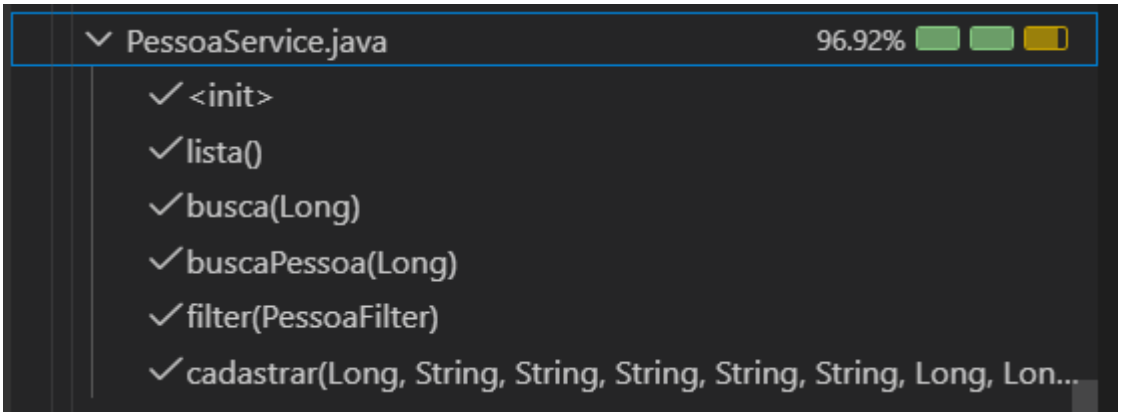
Resultado Esperado:
Retorna null.

---

5 - Execução dos testes

- **PS-001:** OK - Teste testLista validado corretamente o retorno da lista completa de pessoas.
- **PS-002:** OK - Teste testBusca validado corretamente a busca de pessoa por código específico.
- **PS-003:** OK - Teste testBuscaPessoa validado o retorno de Optional preenchido ao buscar pelo ID.
- **PS-004:** OK - Teste testFilterComNome validado corretamente o filtro quando o nome é preenchido.
- **PS-005:** OK - Teste testFilterComNomeNulo validado o comportamento padrão do filtro quando o nome é nulo.
- **PS-006:** OK - Teste testCadastrarCpfDuplicado validou o lançamento de exceção para CPF já existente.
- **PS-007:** OK - Teste testCadastrarSucesso validado corretamente o cadastro de nova pessoa (caminho feliz).
- **PS-008:** OK - Teste testAtualizarPessoaCelular validado a atualização de pessoa e definição correta do tipo celular.
- **PS-009:** OK - Teste testErroAoSalvar validou o tratamento de erro genérico ao salvar no repositório.

- **PS-010:** OK - Teste testCadastrarCidadeNaoEncontrada validou a exceção de cidade não encontrada no cadastro.
- **PS-011:** OK - Teste testCadastrarTelefoneTipoInvalido validado a atribuição do tipo padrão CELULAR para tipos inválidos.
- **PS-012:** OK - Teste testCadastrarDataInvalida validou a exceção de parse ao informar data com formato incorreto.
- **PS-013:** OK - Teste testCadastrarSemTelefone validado corretamente o cadastro de pessoa sem telefone informado.
- **PS-014:** OK - Teste testAtualizarMantendoTelefone validado a atualização de dados cadastrais preservando telefone existente.



Jacoco:

PessoaService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Met
cadastrar(Long, String, String, String, String, String, Long, Long, String, String, String, String, String, Long, String, String, RedirectAttributes)		100%		83%	2	7	0	38	0	
filter(PessoaFilter)		100%		100%	0	2	0	2	0	
PessoaService()		100%		n/a	0	1	0	2	0	
busca(Long)		100%		n/a	0	1	0	1	0	
buscaPessoa(Long)		100%		n/a	0	1	0	1	0	
lista()		100%		n/a	0	1	0	1	0	
Total	0 of 193	100%	2 of 14	85%	2	13	0	45	0	

Mutação:

```
================================================================================
- Timings
================================================================================
> pre-scan for mutations : 8 seconds
> scan classpath : < 1 second
> coverage and dependency analysis : 25 seconds
> build mutation tests : 6 seconds
> run mutation analysis : 20 seconds
--------------------------------------------------------------------------------
> Total  : 1 minutes and 0 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Line Coverage: 45/45 (100%)
>> Generated 31 mutations Killed 30 (97%)
>> Mutations with no coverage 0. Test strength 97%
>> Ran 113 tests (3.65 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:40 min
[INFO] Finished at: 2025-11-30T21:10:44Z
[INFO] ------------------------------------------------------------------------
"
```

ProdutoService

Testes para Produto Service

ProdutoService - Funcional

Documento de Casos de Teste: Gestão de Produtos

- **Produto/Funcionalidade:** Gestão de Produtos
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Henrique Fazollo
- **Tipo de Teste:** Sistema

Pré-requisitos Gerais

- Navegador Chrome inicializado via WebDriverManager.
- Aplicação rodando em `http://localhost:8080`.
- **Login realizado:** Usuário "gerente" e senha "123" autenticados antes de cada caso de teste funcional.
- Banco de dados deve conter dados básicos prévios: Categoria (ID 1), Grupo (ID 1) e Fornecedor (ID 1).

Caso de Teste 1: Cadastro de Novo Produto (Sucesso)

ID: `F-PROD-001` **Método de Teste:** `testF_PROD_001_CadastroProdutoSucesso`
Objetivo: Validar o cadastro completo de um produto com margem de lucro válida.

- **Ação:** Navegar para `/produto/form`.
- **Verificação:** O formulário é carregado.
- **Ação:** Preencher "Descrição" com `REFRIGERANTE COLA 2L`.
- **Ação:** Preencher "Preço de Custo" com `5,00`.
- **Ação:** Preencher "Preço de Venda" com `8,00`.
- **Ação:** Preencher demais campos obrigatórios (Fornecedor, Categoria, Grupo, Unidade).
- **Ação:** Submeter o formulário.
- **Resposta Esperada:** O sistema redireciona para a listagem ou exibe alerta de sucesso (`alert-success`).
- **Ação:** Navegar para a listagem `/produto` e buscar por `REFRIGERANTE COLA 2L`.
- **Resposta Esperada:** O produto é encontrado na tabela e o valor exibido é `R$ 8,00`.

Caso de Teste 2: Validação de Preço de Venda Menor que Custo

ID: `F-PROD-ERR-002` **Método de Teste:** `testF_PROD_ERR_002_PrecoVendaMenorQueCusto` **Objetivo:** Impedir o cadastro de produtos com prejuízo financeiro.

- **Ação:** Navegar para `/produto/form`.
- **Ação:** Preencher "Descrição" com `PRODUTO PREJUÍZO`.
- **Ação:** Selecionar Fornecedor (1), Categoria (1) e Grupo (1).
- **Ação:** Preencher "Preço de Custo" com `10,00`.
- **Ação:** Preencher "Preço de Venda" com `9,00`.
- **Ação:** Submeter o formulário.
- **Resposta Esperada:** O sistema **NÃO** deve redirecionar para a listagem de sucesso. Deve permanecer no formulário (`/form`) e/ou exibir mensagem de erro de validação.

Caso de Teste 3: Validação de Campos Obrigatórios

ID: `F-PROD-ERR-003` **Método de Teste:** `testF_PROD_ERR_003_CamposObrigatorios` **Objetivo:** Garantir que o produto não seja salvo sem descrição, mesmo com valores preenchidos.

- **Ação:** Navegar para `/produto/form`.
- **Ação:** Deixar o campo "Descrição" **vazio**.
- **Ação:** Preencher "Preço de Custo" com `10,00` e "Preço de Venda" com `20,00`.
- **Ação:** Submeter o formulário.
- **Resposta Esperada:** O sistema bloqueia a conclusão do cadastro (não redireciona para sucesso ou recarrega a página de formulário indicando erro).

Caso de Teste 4: Consulta de Produto

ID: `F-PROD-004` **Método de Teste:** `testF_PROD_004_ConsultaProduto` **Objetivo:** Validar a funcionalidade de busca na listagem de produtos.

- **Pré-condição:** Produto `REFRIGERANTE COLA 2L` cadastrado previamente.
- **Ação:** Navegar para a listagem `/produto`.
- **Ação:** Digitar `COLA` no campo de busca e confirmar.
- **Resposta Esperada:** A tabela deve filtrar os resultados e exibir o produto `REFRIGERANTE COLA 2L`.

Caso de Teste 5: Atualização de Produto

ID: `F-PROD-005` **Método de Teste:** `testF_PROD_005_AtualizacaoProduto` **Objetivo:** Validar a edição de preço de um produto existente.

- **Pré-condição:** Produto `REFRIGERANTE COLA 2L` (Venda: 8,00) cadastrado.
- **Ação:** Buscar o produto na listagem e clicar no botão "Editar".
- **Verificação:** O formulário de edição é carregado com o código do produto preenchido.
- **Ação:** Alterar "Preço de Venda" para `8,50`.
- **Ação:** Submeter o formulário.
- **Resposta Esperada:** Mensagem de sucesso é exibida.
- **Ação:** Retornar à listagem e buscar o produto.
- **Resposta Esperada:** O valor exibido na coluna de preço deve ser `8,50`.

Caso de Teste 6: Segurança - Acesso Negado (Novo)

ID: `NF-SEG-001` **Método de Teste:** `testNF_SEG_001_AcessoNegadoSemLogin` **Tipo:** Não Funcional (Segurança) **Objetivo:** Garantir que usuários não autenticados não acessem o formulário de produtos.

- **Ação:** Limpar todos os cookies do navegador (Simulando logout/sessão inválida).
- **Ação:** Tentar acessar diretamente a URL protegida: `/produto/form`.
- **Resposta Esperada:** O sistema deve interceptar a requisição e redirecionar o usuário para a página de Login (`/login`).
- **Ação:** Verificar se a página de Login é exibida.
- **Recuperação:** Realizar login novamente para prosseguir com outros testes.

Caso de Teste 7: Performance - Tempo de Busca (Novo)

ID: `NF-PERF-001` **Método de Teste:** `testNF_PERF_001_TempoRespostaBusca` **Tipo:** Não Funcional (Performance) **Objetivo:** Garantir que a busca de produtos ocorra dentro de um tempo aceitável (< 2 segundos).

- **Pré-condição:** Produto `REFRIGERANTE COLA 2L` cadastrado.
- **Ação:** Navegar para `/produto`.
- **Ação:** Iniciar cronômetro.
- **Ação:** Executar a busca pelo termo `COLA`.
- **Ação:** Parar cronômetro assim que o resultado for renderizado.
- **Resposta Esperada:**
    1. O produto deve ser encontrado.
    2. O tempo total da operação deve ser inferior a **2000ms (2 segundos)**.

ProdutoService - Unitários

Documento de Casos de Teste: ProdutoService – Gestão de Produtos

---

Produto/Funcionalidade: Gestão de Produtos
Data: 29 de Novembro de 2025
Autor do Teste: Larissa Galvão

Tipo de teste: unitário

1 - Objetivo

Verificar se o serviço *ProdutoService* realiza corretamente o **cadastro** *(codprod = 0)* e a **atualização** *(codprod != 0)* de produtos através do método *merger()*, além de validar as regras de **movimentação de estoque** *(movimentaEstoque()* e *ajusteEstoque())* e garantir o tratamento adequado de erros e exceções em todas as operações.

2 - Escopo do Teste

Os métodos do *ProdutoService* que serão testados são:

- **merger():** Responsável por inserir ou atualizar um produto no repositório.
- **movimentaEstoque():** Responsável por dar baixa no estoque de produtos vendidos em uma venda específica.
- **ajusteEstoque():** Responsável por realizar ajustes manuais de estoque (entrada/saída).
- **filter():** Responsável por buscar produtos com base em filtros de descrição.
- **busca()** / **buscaProduto():** Responsáveis por buscar um produto pelo seu código.

3 - Pré-requisitos

- O sistema deve possuir acesso ao banco de dados ou **repositórios mockados** *(ProdutoRepository e VendaProdutoService).*
- O repositório *ProdutoRepository* deve ter os métodos *insere()*, *atualiza()*, *findByCodigoIn()*, *saldoEstoque()*, *e movimentaEstoque()* simulados.
- O serviço *VendaProdutoService* deve ter o método *buscaQtdProduto()* simulado.
- Objetos como *Produto*, *ProdutoControleEstoque*, *EntradaSaida*, *e ProdutoSubstTributaria* devem estar disponíveis.

- A classe `ProdutoService` deve ter as dependências injetadas.

4 - Casos de Teste

PS-001 — merger - Cadastro de Novo Produto

- **Objetivo:** Verificar o caminho feliz de inserção *(codprod = 0).*
- **Passos para Execução:** Chamar *service.merger()* com *codprod = 0* e todos os outros parâmetros válidos.
- **Dados de Entrada:**
    - *codprod:* $0L$
    - *descricao*: "Novo Produto Teste"
    - *valorCusto*: 10.0
    - *valorVenda:* 20.0
- **Resultado Esperado:** O método *produtos.insere()* deve ser invocado exatamente uma vez. O retorno do método deve ser: **"Produdo cadastrado com sucesso"**.

PS-002 — merger - Atualização de Produto Existente

- **Objetivo:** Verificar o caminho feliz de atualização *(codprod != 0).*
- **Passos para Execução:** Chamar *service.merger()* com *codprod != 0* e novos dados válidos.
- **Dados de Entrada:**
    - *codprod:* $100L$
    - *descricao*: "Produto Atualizado"
    - *valorVenda:* 25.0
- **Resultado Esperado:** O método *produtos.atualiza()* deve ser invocado exatamente uma vez. O retorno do método deve ser: **"Produto atualizado com sucesso"**.

PS-003 — merger - Erro ao Inserir Produto

- **Objetivo:** Garantir o tratamento de exceção durante a inserção.
- **Passos para Execução:** 1. Simular o lançamento de uma exceção ao chamar produtos.insere(). 2. Chamar service.merger() com codprod = 0.
- **Dados de Entrada:** codprod: $0L$ (Dados válidos)
- **Resultado Esperado:** O método deve capturar a exceção. O retorno do método deve ser: **"Erro a cadastrar produto, chame o suporte"**.

PS-004 — merger - Erro ao Atualizar Produto

- **Objetivo:** Garantir o tratamento de exceção durante a atualização.
- **Passos para Execução:** 1. Simular o lançamento de uma exceção ao chamar produtos.atualiza(). 2. Chamar service.merger() com `codprod != 0$.
- **Dados de Entrada:** codprod: $101L$ (Dados válidos)
- **Resultado Esperado:** O método deve capturar a exceção. O retorno do método deve ser: **"Erro a atualizar produto, chame o suporte"**.

PS-005 — movimentaEstoque - Baixa de Estoque com Sucesso

- **Objetivo:** Verificar a baixa de estoque quando o produto controla estoque e há saldo suficiente.
- **Passos para Execução:** 1. Simular lista de produtos vendidos e Produto.controla_estoque = SIM. 2. Simular saldoEstoque > qtdVendida. 3. Chamar service.movimentaEstoque().
- **Dados de Entrada:**
    - codVenda: $50L$
    - codProduto: $10L$
    - qtdVendida: 5
    - qtdEstoque: 20
- **Resultado Esperado:** O método produtos.movimentaEstoque() deve ser invocado uma vez com EntradaSaida.SAIDA. Nenhuma exceção deve ser lançada.

PS-006 — movimentaEstoque - Saldo Insuficiente

- **Objetivo:** Verificar o lançamento de exceção quando o saldo é insuficiente para a baixa.
- **Passos para Execução:** 1. Simular lista de produtos vendidos e Produto.controla_estoque = SIM. 2. Simular saldoEstoque < qtdVendida. 3. Chamar service.movimentaEstoque().
- **Dados de Entrada:**
    - codVenda: $51L$
    - codProduto: $11L$
    - qtdVendida: 10
    - qtdEstoque: 5
- **Resultado Esperado:** Deve ser lançada uma **RuntimeException** com a mensagem contendo: **"não tem estoque suficiente, verifique"**. produtos.movimentaEstoque() não deve ser chamado.

PS-007 — movimentaEstoque - Produto Não Controla Estoque

- **Objetivo:** Verificar se a movimentação é ignorada para produtos que não controlam estoque.
- **Passos para Execução:** 1. Simular lista de produtos vendidos e Produto.controla_estoque = NÃO. 2. Chamar service.movimentaEstoque().
- **Dados de Entrada:**
    - codVenda: $52L$
    - codProduto: $12L$
    - controla_estoque: "NÃO"
- **Resultado Esperado:** O método produtos.movimentaEstoque() não deve ser chamado. Nenhuma exceção deve ser lançada.

PS-008 — ajusteEstoque - Ajuste para Entrada (Caminho Feliz)

- **Objetivo:** Verificar o ajuste manual de estoque para entrada quando o controle está ativo.
- **Passos para Execução:** 1. Simular Produto.controla_estoque = SIM. 2. Chamar service.ajusteEstoque() com EntradaSaida.ENTRADA.
- **Dados de Entrada:**
    - codProduto: $15L$

    - qtdAjuste: 50
    - tipo: ENTRADA
- **Resultado Esperado:** O método produtos.movimentaEstoque() deve ser invocado uma vez com EntradaSaida.ENTRADA. Nenhuma exceção deve ser lançada.

PS-009 — ajusteEstoque - Ajuste para Produto Sem Controle

- **Objetivo:** Verificar o lançamento de exceção ao tentar ajustar estoque de produto que não controla.
- **Passos para Execução:** 1. Simular Produto.controla_estoque = NÃO. 2. Chamar service.ajusteEstoque().
- **Dados de Entrada:**
    - codProduto: $16L$
    - controla_estoque: "NÃO"
- **Resultado Esperado:** Deve ser lançada uma **RuntimeException** com a mensagem contendo: **"não controla estoque, verifique"**. produtos.movimentaEstoque() não deve ser chamado.

PS-010 — filter - Filtrar por Descrição

- **Objetivo:** Verificar se a busca por filtro de descrição funciona.
- **Passos para Execução:** 1. Criar um ProdutoFilter com descrição. 2. Chamar service.filter().
- **Dados de Entrada:**
    - filter.descricao: "Biscoito"
    - pageable: (mock)
- **Resultado Esperado:** O método produtos.findByDescricaoContaining() deve ser invocado com o padrão **"%Biscoito%"** e retornar a lista paginada.

PS-011 — filter - Filtrar com Descrição Vazia/Nula

- **Objetivo:** Verificar se a busca retorna todos os produtos quando a descrição é vazia ou nula.
- **Passos para Execução:** 1. Criar um ProdutoFilter com descrição null. 2. Chamar service.filter().
- **Dados de Entrada:**
    - filter.descricao: null
    - pageable: (mock)
- **Resultado Esperado:** O método produtos.findByDescricaoContaining() deve ser invocado com o padrão **"%"** (coringa) e retornar todos os produtos paginados.

PS-012 — buscaProduto - Buscar por Código Existente (Optional)

- **Objetivo:** Verificar se a busca por código retorna o produto correto como Optional.
- **Passos para Execução:** 1. Simular que produtos.findById(codigo) retorna um Optional com o produto. 2. Chamar service.buscaProduto().
- **Dados de Entrada:** codigo: $20L$
- **Resultado Esperado:** Retorna um Optional preenchido (isPresent() == true) com o objeto Produto correspondente.

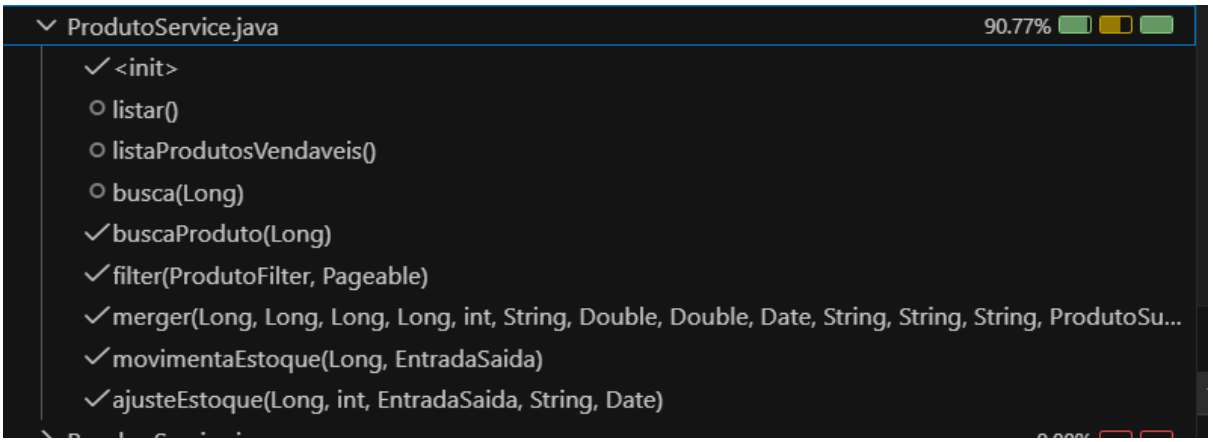
PS-013 — buscaProduto - Buscar por Código Inexistente (Optional)

- **Objetivo:** Verificar se a busca por código inexistente retorna um Optional vazio.
- **Passos para Execução:** 1. Simular que produtos.findById(codigo) retorna um Optional.empty(). 2. Chamar service.buscaProduto().
- **Dados de Entrada:** codigo: $999L$
- **Resultado Esperado:** Retorna um Optional.empty().

5 - Execução dos testes

- **PS-001: OK** - Teste `merger_QuandoCodprodEhZero_DeveInserirNovoProdutoComSucesso` validado corretamente a inserção.
- **PS-002: OK** - Teste `merger_QuandoCodprodNaoEhZero_DeveAtualizarProdutoComSucesso` validado corretamente a atualização.
- **PS-003: OK** - Teste `merger_QuandoCodprodEhZero_DeveRetornarErroAoCadastrar` validado o tratamento de erro na inserção.
- **PS-004: OK** - Teste `merger_QuandoCodprodNaoEhZero_DeveRetornarErroAoAtualizar` validado o tratamento de erro na atualização.
- **PS-005: OK** - Teste `movimentaEstoque_deveDarBaixaComSucesso` validado a baixa de estoque.
- **PS-006: OK** - Teste `movimentaEstoque_deveLancarExcecaoPorFaltaDeEstoque` validou a exceção por falta de estoque.
- **PS-007: OK** - Teste `movimentaEstoque_naoDeveMovimentarSeNaoControlarEstoque` validou que não há movimentação.
- **PS-008: OK** - Teste `ajusteEstoque_deveMovimentarEstoqueSeProdutoControlar` validado o ajuste de estoque.
- **PS-009: OK** - Teste `ajusteEstoque_deveLancarExcecaoSeProdutoNaoControlarEstoque` validou a exceção em produto sem controle.
- **PS-010: OK** - Teste `filter_deveFiltrarPorDescricaoContendo` (Filtro por descrição) implementado e validado.
- **PS-011: OK** - Teste `filter_deveRetornarTodosProdutosQuandoDescricaoForNula` (Filtro nulo) implementado e validado.
- **PS-012: OK** - Teste `buscaProduto_deveRetornarOptionalPreenchidoParaCodigoExistente` (Busca existente) implementado e validado.

- **PS-013: OK** - Teste `buscaProduto_deveRetornarOptionalVazioParaCodigoInexistente` (Busca inexistente) implementado e validado.



6 – Análise de Cobertura e Testes de Mutação

Após a implementação inicial dos testes unitários do *ProdutoService*, foi realizada a execução da ferramenta **JaCoCo**, com o objetivo de avaliar a cobertura estrutural do código, especialmente em relação às **arestas (branches)**. Nesta primeira análise, foi constatado que a cobertura de arestas já se encontrava **acima de 80%**.

ProdutoService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
ajusteEstoque(Long, int, EntradaSaida, String, Date)		100%		100%	0	2	0	5	0	1
busca(Long)		0%		n/a	1	1	1	1	1	1
buscaProduto(Long)		100%		n/a	0	1	0	1	0	1
filter(ProdutoFilter, Pageable)		100%		100%	0	2	0	2	0	1
listaProdutosVendaveis()		0%		n/a	1	1	1	1	1	1
listar()		0%		n/a	1	1	1	1	1	1
merger(Long, Long, Long, Long, int, String, Double, Double, Date, String, String, String, ProdutoSubstTributaria, String, String, Long, Long, String)		100%		100%	0	2	0	14	0	1
movimentaEstoque(Long, EntradaSaida)		100%		100%	0	4	0	15	0	1
ProdutoService()		100%		n/a	0	1	0	2	0	1
Total	13 of 234	94%	0 of 12	100%	3	15	3	42	3	9

Created with JaCoCo 0.8.11.202310140853

Na sequência, foi executada a ferramenta **PITest**, responsável pela avaliação da eficácia dos testes por meio de **testes de mutação**. Nessa primeira execução, o **Mutation Score obtido foi de 72%**, indicando que, apesar da boa cobertura estrutural, ainda existiam mutações no código que não estavam sendo devidamente detectadas pelos testes existentes.

A partir da análise dos **mutantes sobreviventes**, foram identificadas lacunas principalmente em métodos que ainda não possuíam testes específicos e em cenários de borda. Com base nisso, foram implementados novos testes unitários contemplando, principalmente, os seguintes pontos:

- Testes para os métodos:
    - `listar()`, garantindo o retorno da lista completa de produtos;

    - `listaProdutosVendaveis()`, validando a listagem de produtos disponíveis para venda;

    - `busca()`, assegurando a busca direta por código através do repositório.

- Teste para o cenário em que o método `movimentaEstoque()` recebe uma **lista vazia de produtos vendidos**, validando que nenhuma movimentação de estoque é realizada.

- Reforço na validação do método `ajusteEstoque()`, garantindo o uso correto do tipo de movimentação (`ENTRADA` e `SAIDA`).

- Validação mais rigorosa do método `merger()` quanto ao uso correto do **ordinal do enum ProdutoSubstTributaria** no processo de inserção.

Após a inclusão desses novos testes e o fortalecimento das validações existentes, o **PITest foi executado novamente**, obtendo-se um **Mutation Score final de 84%**, evidenciando uma **melhoria significativa na robustez da suíte de testes** e na capacidade de detecção de falhas no código.

```
================================================================================
- Timings
================================================================================
> pre-scan for mutations : 6 seconds
> scan classpath : < 1 second
> coverage and dependency analysis : 24 seconds
> build mutation tests : 6 seconds
> run mutation analysis : 12 seconds
--------------------------------------------------------------------------------
> Total  : 51 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Line Coverage: 44/44 (100%)
>> Generated 25 mutations Killed 21 (84%)
>> Mutations with no coverage 0. Test strength 84%
>> Ran 31 tests (1.24 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:19 min
[INFO] Finished at: 2025-12-01T03:19:53Z
[INFO] ------------------------------------------------------------------------
#
```

Pit Test Coverage Report

Project Summary

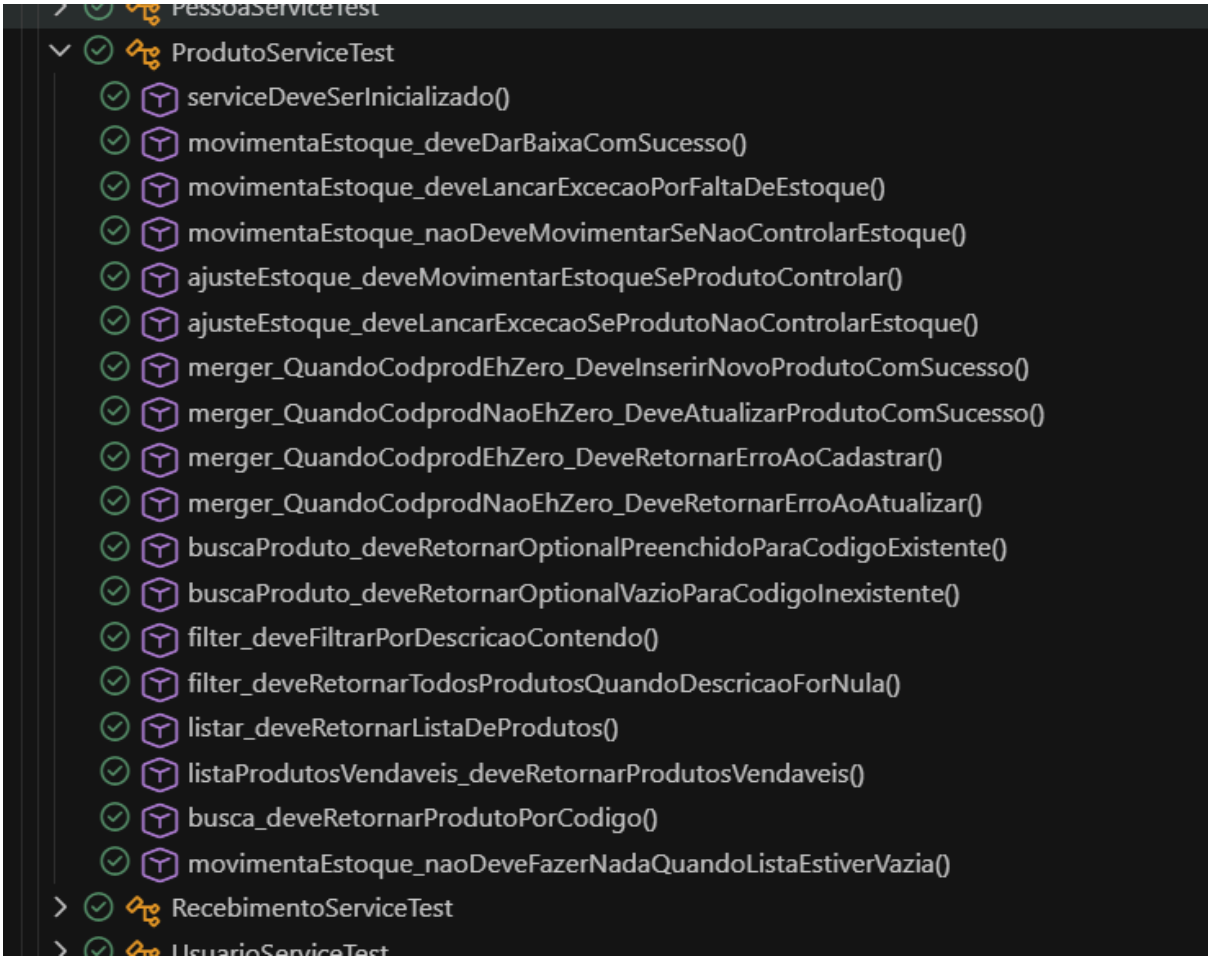
Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
1	100%	44/44	84%	21/25	84%	21/25

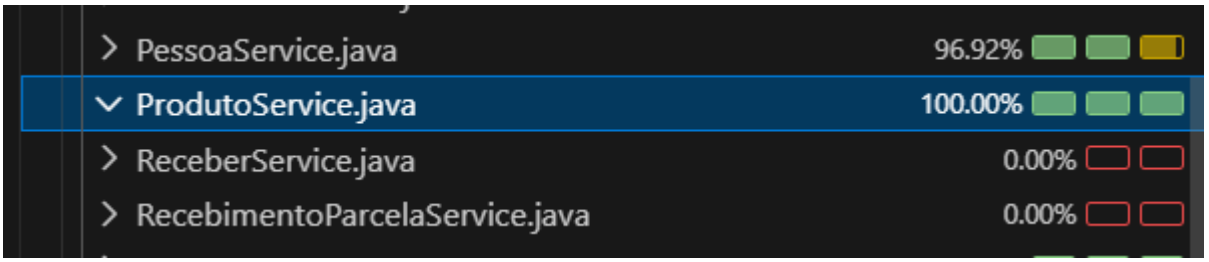
Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
net.originmobi.pdv.service	1	100%	44/44	84%	21/25	84%	21/25

Report generated by PIT 1.7.5

No fim, ficamos com esse total de testes e cobertura para ProdutoService:





pdv > net.originmobi.pdv.service > ProdutoService

Sessions

ProdutoService

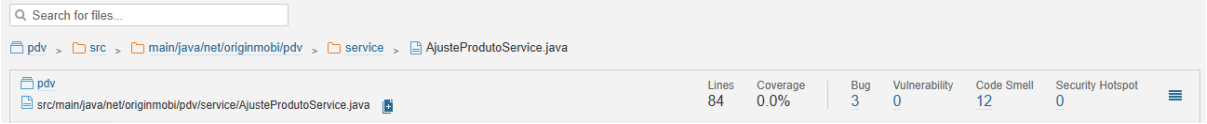
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed
movimentaEstoque(Long, EntradaSaida)		100%		100%	0	4	0	15	0
merger(Long, Long, Long, Long, int, String, Double, Double, Date, String, String, String, ProdutoSubstTributaria, String, String, Long, Long, String)		100%		100%	0	2	0	14	0
ajusteEstoque(Long, int, EntradaSaida, String, Date)		100%		100%	0	2	0	5	0
filter(ProdutoFilter, Pageable)		100%		100%	0	2	0	2	0
ProdutoService()		100%		n/a	0	1	0	2	0
busca(Long)		100%		n/a	0	1	0	1	0
buscaProduto(Long)		100%		n/a	0	1	0	1	0
listar()		100%		n/a	0	1	0	1	0
listaProdutosVendaveis()		100%		n/a	0	1	0	1	0
Total	0 of 234	100%	0 of 12	100%	0	15	0	42	0

AjusteProdutoService - Sonar

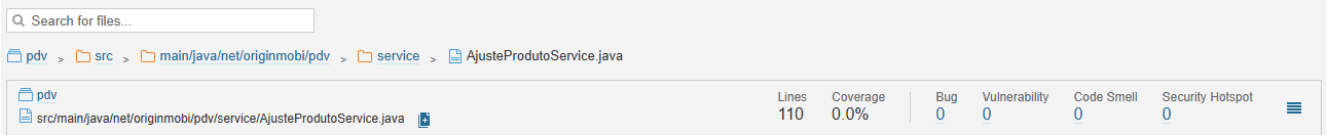
Relatório de Inspeção do Código-Fonte: AjusteProdutoService

- **Data:** 29 de Novembro de 2025
- **Autor do Arquivo:** Henrique Fazollo

Antes:



Depois:



O que foi feito:

- Correção de Nomes (Naming Convention): Renomeação de variáveis locais que estavam em *snake_case* para o padrão Java *camelCase* (ex: `qtd_alteracao` para `qtdAlteracao`).
- Segurança com Optional: Adicionada verificação `.isPresent()` antes de acessar o valor com `.get()` para prevenir *bugs* críticos (NoSuchElementException).
- Exceptions Específicas: Substituição de lançamentos genéricos de `RuntimeException` por uma exceção dedicada (`RegraNegocioException`).
- Logging: Substituição de `System.out.println` pelo uso correto de `Logger` (SLF4J) para evitar perda de logs e problemas de performance.

Recebimento

Testes para RecebimentoService

RecebimentoController - Integração

Documento de Casos de Teste: RecebimentoController Integração

- **Produto/Funcionalidade:** Controller para Recebimento
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Henrique Fazollo
- **Tipo de teste:** Integração

JSON


```
{
	"info": {
		"_postman_id": "99965a8e-8aa1-44ed-b770-69fb2b3c3af2",
		"name": "TrabalhoQA",
		"schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
		"_exporter_id": "28117367",
		"_collection_link":
"https://www.postman.com/hfazollo/workspace/aulaqa/collection/28117367-99965
a8e-8aa1-44ed-b770-69fb2b3c3af2?action=share&source=collection_link&creator=
28117367"
	},
	"item": [
		{
			"name": "Login",
			"request": {
				"method": "POST",
				"header": [],
				"body": {
					"mode": "urlencoded",
					"urlencoded": [
						{
							"key": "username",
							"value": "gerente",
							"type": "text"
						},
						{
							"key": "password",
							"value": "123",
							"type": "text"
						}
					]
				},
				"url": {
```

```
                                "raw": "localhost:8080/login",
                                "host": [
                                        "localhost"
                                ],
                                "port": "8080",
                                "path": [
                                        "login"
                                ]
                        }
                },
                "response": []
        },
        {
                "name": "Receber",
                "event": [
                        {
                                "listen": "prerequest",
                                "script": {
                                        "exec": [
                                                ""
                                        ],
                                        "type": "text/javascript",
                                        "packages": {},
                                        "requests": {}
                                }
                        },
                        {
                                "listen": "test",
                                "script": {
                                        "exec": [
                                                "pm.test(\"Status code é
200\", function () {\r",
                                                "
pm.response.to.have.status(200);\r",
                                                "});\r",
                                                "\r",
                                                "pm.test(\"Tempo de resposta é
aceitável\", function () {\r",
                                                "
pm.expect(pm.response.responseTime).to.be.below(500);\r",
                                                "});\r",
                                                "\r",
                                                "pm.test(\"Mensagem de retorno
correta\", function () {\r",
                                                "    var responseText =
pm.response.text();\r",
                                                "
pm.expect(responseText).to.be.a('string').and.to.not.be.empty;\r",
```

```
                                                "\r",
                                                "    if
(responseText.includes(\"sucesso\")) {\r",
                                                "
pm.expect(responseText).to.include(\"sucesso\");\r",
                                                "    } else if
(responseText.includes(\"fechada\")) {\r",
                                                "
pm.expect(responseText).to.include(\"fechada\");\r",
                                                "
console.warn(\"Atenção: O teste passou, mas a venda já estava
fechada.\");\r",
                                                "    } else {\r",
                                                "
pm.expect.fail(\"Mensagem de retorno inesperada: \" + responseText);\r",
                                                "    }\r",
                                                "});"
                                        ],
                                        "type": "text/javascript",
                                        "packages": {},
                                        "requests": {}
                                }
                        }
                ],
                "request": {
                        "method": "POST",
                        "header": [],
                        "body": {
                                "mode": "urlencoded",
                                "urlencoded": [
                                        {
                                                "key": "receber",
                                                "value": "100",
                                                "type": "text"
                                        },
                                        {
                                                "key": "titulo",
                                                "value": "5",
                                                "type": "text"
                                        },
                                        {
                                                "key": "vlrecebido",
                                                "value": "150,00",
                                                "type": "text"
                                        },
                                        {
                                                "key": "desconto",
                                                "value": "0,00",
```

```
                                        "type": "text"
                                    },
                                    {
                                        "key": "acrescimo",
                                        "value": "0,00",
                                        "type": "text"
                                    }
                                ]
                            },
                            "url": {
                                "raw": "http://localhost:8080/recebimento",
                                "protocol": "http",
                                "host": [
                                    "localhost"
                                ],
                                "port": "8080",
                                "path": [
                                    "recebimento"
                                ]
                            }
                        },
                        "response": []
                    },
                    {
                        "name": "Remover Recebimento",
                        "event": [
                            {
                                "listen": "test",
                                "script": {
                                    "exec": [
                                        "pm.test(\"Status code é
200\", function () {\r",
                                        "
pm.response.to.have.status(200);\r",
                                        "});\r",
                                        "\r",
                                        "pm.test(\"Retorna URL de
redirecionamento correta\", function () {\r",
                                        "    var responseText =
pm.response.text();\r",
                                        "    \r",
                                        "
pm.expect(responseText).to.include(\"/receber\");\r",
                                        "});\r",
                                        "\r",
                                        "pm.test(\"Não ocorreu erro
interno de servidor\", function () {\r",
```

```
                                                "
pm.response.to.not.have.status(500);\r",
                                                "});"
                                          ],
                                          "type": "text/javascript",
                                          "packages": {},
                                          "requests": {}
                                    }
                              }
                        ],
                        "request": {
                              "method": "PUT",
                              "header": [],
                              "body": {
                                    "mode": "urlencoded",
                                    "urlencoded": [
                                          {
                                                "key": "codigo",
                                                "value": "100",
                                                "type": "text"
                                          }
                                    ]
                              },
                              "url": {
                                    "raw":
"http://localhost:8080/recebimento/100",
                                    "protocol": "http",
                                    "host": [
                                          "localhost"
                                    ],
                                    "port": "8080",
                                    "path": [
                                          "recebimento",
                                          "100"
                                    ]
                              }
                        },
                        "response": []
                  }
            ]
}
```

RecebimentoService - Unitários

Documento de Casos de Teste: RecebimentoService – Unitários

- **Produto/Funcionalidade:** Gestão de Recebimentos (Financeiro)
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Henrique Fazollo
- **Tipo de teste:** Unitário

1 - Objetivo

Verificar se o serviço RecebimentoService realiza corretamente a abertura de recebimentos vinculados a parcelas (abrirRecebimento), o processamento financeiro dos mesmos (receber) — incluindo a lógica de distribuição de valores entre parcelas e geração de lançamentos de caixa ou cartão — e a remoção de recebimentos não processados (remover), garantindo a integridade dos dados e o tratamento de exceções.

2 - Escopo do Teste

Os métodos do RecebimentoService que serão testados são:

- **abrirRecebimento():** Valida as parcelas (propriedade e quitação) e cria um registro de recebimento.
- **receber():** Processa o pagamento, verifica títulos, distribui valores entre parcelas e gera lançamentos financeiros (Caixa ou Cartão).
- **remover():** Remove um recebimento que ainda não foi processado.

3 - Pré-requisitos

- O sistema deve possuir acesso aos repositórios e serviços mockados: RecebimentoRepository, PessoaService, RecebimentoParcelaService, ParcelaService, CaixaService, UsuarioService, CaixaLancamentoService, TituloService e CartaoLancamentoService.
- As classes de modelo (Recebimento, Parcela, Titulo, Pessoa, Caixa, Usuario) devem estar disponíveis.
- Dependências estáticas como Aplicacao (para pegar usuário atual) devem ser mockadas via MockedConstruction ou similar quando necessário.

4 - Casos de Teste

RS-001 — abrirRecebimento - Abertura com Sucesso

- **Objetivo:** Verificar se o recebimento é criado corretamente quando as parcelas são válidas e pertencem ao cliente.
- **Passos para Execução:** Chamar service.abrirRecebimento() com um código de pessoa e array de parcelas válidas.

- **Dados de Entrada:**
    - codpes: $1L$
    - arrayParcelas: {"10", "20"}
    - Valor Parcelas: 100.0 e 50.0
- **Resultado Esperado:** O método recebimentos.save() deve ser invocado. O valor total do recebimento salvo deve ser 150.0. O retorno deve ser o código do recebimento (ex: "500").

RS-002 — abrirRecebimento - Erro Parcela Quitada

- **Objetivo:** Impedir a abertura se uma das parcelas já estiver quitada.
- **Passos para Execução:** 1. Simular uma parcela com quitado = 1. 2. Chamar service.abrirRecebimento().
- **Dados de Entrada:**
    - codpes: $1L$
    - arrayParcelas: {"10"} (Parcela quitada)
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Parcela 10 já esta quitada, verifique.".

RS-003 — abrirRecebimento - Erro Parcela de Outra Pessoa

- **Objetivo:** Impedir a abertura se a parcela pertencer a um cliente diferente do informado.
- **Passos para Execução:** 1. Simular parcela pertencente à Pessoa $99L$. 2. Chamar service.abrirRecebimento() para Pessoa $1L$.
- **Dados de Entrada:**
    - codpes: $1L$
    - arrayParcelas: {"10"}
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "A parcela 10 não pertence ao cliente selecionado".

RS-004 — abrirRecebimento - Erro Cliente Não Encontrado

- **Objetivo:** Validar a existência do cliente antes de abrir.
- **Passos para Execução:** 1. Simular pessoas.buscaPessoa() retornando Optional.empty(). 2. Chamar service.abrirRecebimento().
- **Dados de Entrada:**
    - codpes: $99L$
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Cliente não encontrado".

RS-005 — receber - Pagamento em DINHEIRO (Sucesso)

- **Objetivo:** Verificar o processamento completo de um recebimento em dinheiro, gerando lançamento no caixa.
- **Passos para Execução:** 1. Simular Título tipo "DINHEIRO". 2. Chamar service.receber().
- **Dados de Entrada:**
    - codreceber: $100L$

    - vlrecebido: 100.00
    - codtitulo: $5L$ (Dinheiro)
- **Resultado Esperado:** parcelas.receber() deve ser chamado. lancamentos.lancamento() deve ser chamado (Caixa). cartaoLancamentos.lancamento() **não** deve ser chamado. Retorno: "Recebimento realizado com sucesso".

RS-006 — receber - Pagamento em CARTÃO (Sucesso)

- **Objetivo:** Verificar o processamento de recebimento em cartão, gerando lançamento de cartão e não de caixa.
- **Passos para Execução:** 1. Simular Título com sigla CARTDEB. 2. Chamar service.receber().
- **Dados de Entrada:**
    - codreceber: $100L$
    - vlrecebido: 50.00
    - codtitulo: $6L$ (Cartão Débito)
- **Resultado Esperado:** cartaoLancamentos.lancamento() deve ser chamado. lancamentos.lancamento() **não** deve ser chamado. Retorno: "Recebimento realizado com sucesso".

RS-007 — receber - Distribuição de Valor (Pagamento Parcial)

- **Objetivo:** Verificar se o valor recebido é abatido corretamente das parcelas em ordem.
- **Passos para Execução:** 1. Recebimento com 2 parcelas de 50.00. 2. Receber 60.00.
- **Dados de Entrada:**
    - Parcela 1: 50.00, Parcela 2: 50.00
    - vlrecebido: 60.00
- **Resultado Esperado:** Parcela 1 deve receber baixa de 50.00. Parcela 2 deve receber baixa de 10.00.

RS-008 — receber - Erro Título Inválido

- **Objetivo:** Impedir recebimento sem título selecionado.
- **Passos para Execução:** Chamar service.receber() com codtitulo = 0 ou nulo.
- **Dados de Entrada:** codtitulo: $0L$
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Selecione um título para realizar o recebimento".

RS-009 — receber - Erro Recebimento Fechado

- **Objetivo:** Impedir alteração em recebimento já processado.
- **Passos para Execução:** 1. Simular Recebimento com data_processamento preenchida. 2. Chamar service.receber().
- **Dados de Entrada:** codreceber: $100L$ (Processado)
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Recebimento já esta fechado".

RS-010 — receber - Erro Valor Superior

- **Objetivo:** Impedir recebimento de valor maior que a dívida total.
- **Passos para Execução:** Chamar service.receber() com valor maior que recebimento.getValor_total().
- **Dados de Entrada:**
    - Total Recebimento: 100.00
    - vlrecebido: 200.00
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Valor de recebimento é superior aos títulos".

RS-011 — receber - Erro Sem Parcelas

- **Objetivo:** Impedir recebimento se não houver parcelas vinculadas.
- **Passos para Execução:** Simular receParcelas.parcelasDoReceber() retornando lista vazia.
- **Dados de Entrada:** codreceber: $1L$
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Recebimento não possue parcelas".

RS-012 — receber - Erro Valor Inválido

- **Objetivo:** Impedir recebimento com valor zero ou negativo.
- **Dados de Entrada:** vlrecebido: 0.0
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Valor de recebimento inválido".

RS-013 — remover - Remoção com Sucesso

- **Objetivo:** Verificar a remoção de um recebimento aberto.
- **Passos para Execução:** Chamar service.remover() para um recebimento sem data de processamento.
- **Dados de Entrada:** codigo: $100L$
- **Resultado Esperado:** O método recebimentos.deleteById() deve ser chamado. Retorno: "removido com sucesso".

RS-014 — remover - Erro Recebimento Processado

- **Objetivo:** Impedir a remoção de um recebimento já finalizado.
- **Passos para Execução:** Simular Recebimento com data_processamento preenchida.
- **Dados de Entrada:** codigo: $10L$
- **Resultado Esperado:** Deve ser lançada uma RuntimeException com a mensagem: "Esse recebimento não pode ser removido, pois ele já esta processado".

RS-015 — Tratamento de Exceções Genéricas

- **Objetivo:** Verificar se erros de banco/sistema são capturados e relançados com mensagem amigável ("chame o suporte").

- **Cenários:** Erro ao salvar (abrirRecebimento), erro ao lançar caixa (receber), erro ao deletar (remover).
- **Resultado Esperado:** Mensagens contendo "chame o suporte".

RS-016 — receber - Validação de Desconto e Acréscimo

- **Objetivo:** Verificar se os valores de desconto e acréscimo são persistidos corretamente no recebimento.
- **Dados de Entrada:** `vlrecebido: 95.0`, `vldesconto: 5.0`, `vlacrescimo: 0.0`.
- **Resultado Esperado:** O recebimento salvo deve conter valor de desconto = 5.0.

RS-017 — receber - Pagamento em PIX

- **Objetivo:** Verificar o processamento via PIX (que no seu código gera lançamento em caixa, similar ao dinheiro, mas é um tipo de título diferente).
- **Passos:** Simular título com sigla "PIX".
- **Resultado Esperado:** `lancamentos.lancamento()` deve ser chamado.

5 - Execução dos testes

- **RS-001:** OK - Teste deveAbrirRecebimentoComSucesso validou corretamente a criação.
- **RS-002:** OK - Teste deveLancarErroParcelaQuitada validou o bloqueio de parcelas quitadas.
- **RS-003:** OK - Teste deveLancarErroParcelaDeOutraPessoa validou a consistência do cliente.
- **RS-004:** OK - Teste deveLancarErroPessoaNaoEncontrada validou a existência da pessoa.
- **RS-005:** OK - Teste deveProcessarRecebimentoDinheiro validou o fluxo de dinheiro/caixa.
- **RS-006:** OK - Teste deveProcessarRecebimentoCartao validou o fluxo de cartão.
- **RS-007:** OK - Teste deveDistribuirValorEntreParcelas validou a lógica de baixa parcial.
- **RS-008:** OK - Teste deveLancarErroTituloInvalido validou a obrigatoriedade do título.
- **RS-009:** OK - Teste deveLancarErroRecebimentoFechado validou o bloqueio de edição.
- **RS-010:** OK - Teste deveLancarErroValorSuperior validou o teto do valor recebido.
- **RS-011:** OK - Teste deveLancarErroSemParcelas validou a existência de parcelas.
- **RS-012:** OK - Teste deveLancarErroValorRecebidoInvalido validou valores positivos.
- **RS-013:** OK - Teste deveRemoverRecebimento validou a remoção.
- **RS-014:** OK - Teste deveImpedirRemocaoProcessada validou a proteção contra deleção de processados.
- **RS-015:** OK - Testes deveTratarErroAoSalvarRecebimento, deveTratarErroNoLancamentoCaixa e deveTratarErroNaRemocao validaram o tratamento de exceções

- **RS-016:** OK - Teste `deveArmazenarDescontoEAcrescimo` validou a persistência dos valores.
- **RS-017:** OK - Teste `deveProcessarRecebimentoPix` validou o fluxo específico de PIX.

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running net.originmobi.pdv.service.RecebimentoServiceTest
Erro BD
[INFO] Tests run: 32, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 8.233 s -- in net.originmobi.pdv.service.RecebimentoServiceTest
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoParcelasValoresDiferentes -- Time elapsed: 7.998 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoValorMinimo -- Time elapsed: 0.026 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveImpedirRemocaoProcessada -- Time elapsed: 0.012 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveTratarErroAoSalvarRecebimento -- Time elapsed: 0.014 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroSemParcelas -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoValorExato -- Time elapsed: 0.012 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveDistribuirValorEntreParcelas -- Time elapsed: 0.004 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroParcelaQuitada -- Time elapsed: 0.001 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveTratarErroNaRemocao -- Time elapsed: 0.006 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveTestarCondicaoVlsobraMaiorQueZero -- Time elapsed: 0.005 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveValidarLoopParcelasNaoQuitadasCompletamente -- Time elapsed: 0.004 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveDefinirDataProcessamento -- Time elapsed: 0.009 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveAbrirRecebimentoMultiplasParcelas -- Time elapsed: 0.005 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroTituloInvalido -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoMultiplasParcelasCompleto -- Time elapsed: 0.004 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoCartao -- Time elapsed: 0.006 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveAbrirRecebimentoComSucesso -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroRecebimentoFechado -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarQuandoVlrecebidoZera -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroValorRecebidoInvalido -- Time elapsed: 0.002 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoDinheiro -- Time elapsed: 0.005 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveValidarCalculoDeSobraZero -- Time elapsed: 0.008 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoPix -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveTratarErroNoLancamentoCaixa -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveVincularTituloAoRecebimento -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveRemoverRecebimento -- Time elapsed: 0.003 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveArmazenarDescontoEAcrescimo -- Time elapsed: 0.004 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroPessoaNaoEncontrada -- Time elapsed: 0.002 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveCalcularVlsobraCorretamente -- Time elapsed: 0.004 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveProcessarRecebimentoCartaoCredito -- Time elapsed: 0.004 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroValorSuperior -- Time elapsed: 0.002 s
[INFO] net.originmobi.pdv.service.RecebimentoServiceTest.deveLancarErroParcelaDeOutraPessoa -- Time elapsed: 0.002 s
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 32, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  44.178 s
[INFO] Finished at: 2025-11-30T11:56:01Z
[INFO] ------------------------------------------------------------------------
```

Estrutural:

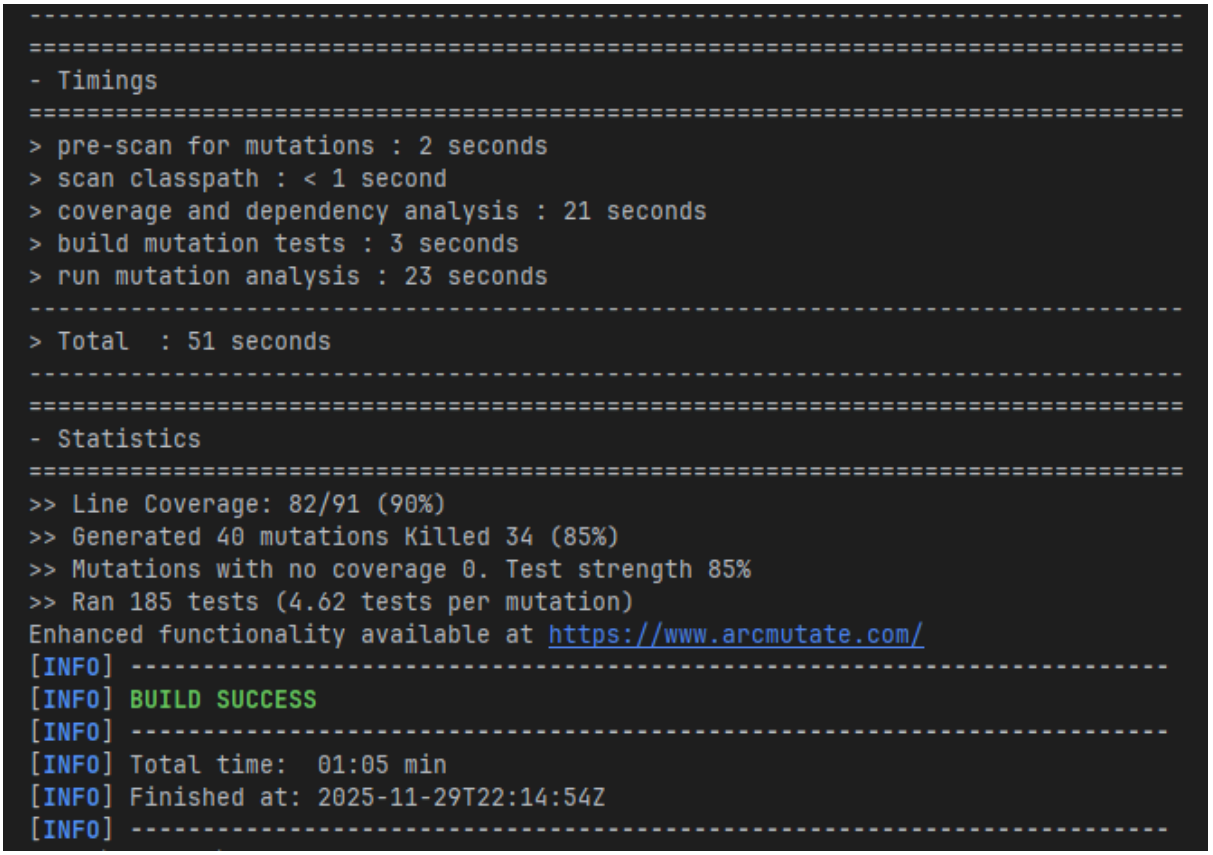
Relatório de cobertura no critério todas-arestas (JaCoCo)

RecebimentoService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
receber(Long, Double, Double, Double, Long)		93%		91%	2	13	6	56	0	1
abrirRecebimento(Long, String[])		93%		100%	0	5	3	25	0	1
remover(Long)		100%		100%	0	2	0	9	0	1
RecebimentoService()		100%		n/a	0	1	0	1	0	1
Total	29 of 461	93%	2 of 34	94%	2	21	9	91	0	4

Baseada em Defeitos:

Score de Mutação



Usuario

Testes para Usuário

UsuarioService - Unitários

Documento de Casos de Teste: UsuarioService – Unitários

- **Produto/Funcionalidade:** Gestão de Usuários
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Henrique Fazollo
- **Tipo de teste:** Unitário

1 - Objetivo

Verificar se o serviço `UsuarioService` realiza corretamente o ciclo de vida do usuário, incluindo o cadastro com criptografia de senha, validação de unicidade (login e vínculo com pessoa), atualização cadastral e gerenciamento de permissões (adicionar e remover grupos de acesso), garantindo a integridade da segurança e dos dados.

2 - Escopo do Teste

Os métodos do `UsuarioService` que serão testados são:

- **`cadastrar(Usuario)`**: Valida regras de duplicidade, criptografa senha, define data de cadastro e persiste (novo ou atualização).
- **`addGrupo(Long, Long)`**: Vincula um grupo de permissão ao usuário, evitando duplicidade.
- **`removeGrupo(Long, Long)`**: Remove um grupo de permissão da lista do usuário.
- **`buscaUsuario(String)`**: Recupera um usuário pelo login.

3 - Pré-requisitos

- O sistema deve possuir acesso aos repositórios e serviços mockados: `UsuarioRepository` e `GrupoUsuarioService`.
- As classes de modelo (`Usuario`, `Pessoa`, `GrupoUsuario`) devem estar disponíveis.
- Bibliotecas de segurança (`BCryptPasswordEncoder`) devem estar no classpath.

4 - Casos de Teste

US-001 — cadastrar - Cadastro de Novo Usuário (Sucesso)

- **Objetivo:** Verificar se um novo usuário é salvo com a data atual definida e a senha criptografada.
- **Passos para Execução:** Chamar `service.cadastrar()` com um usuário sem ID (`codigo = null`).
- **Dados de Entrada:**

    - Usuario: `user="admin"`, `senha="123"`, `codigo=null`.
    - Pessoa: `codigo=1L`.
- **Configuração do Mock:**
    - `usuarios.findByUserEquals("admin")` retorna `null`.
    - `usuarios.findByPessoaCodigoEquals(1L)` retorna `null`.
- **Resultado Esperado:**
    - O método `usuarios.save()` deve ser invocado.
    - A senha do objeto salvo **não** deve ser "123" (deve estar codificada).
    - A `data_cadastro` não deve ser nula.
    - Retorno: "Usuário salvo com sucesso".

US-002 — cadastrar - Erro Usuário Já Existe

- **Objetivo:** Impedir o cadastro se o `username` já estiver em uso.
- **Passos para Execução:** Chamar `service.cadastrar()` com `user="admin"`.
- **Configuração do Mock:** `usuarios.findByUserEquals("admin")` retorna um objeto `Usuario`.
- **Resultado Esperado:** O método `usuarios.save()` **não** deve ser chamado. Retorno: "Usuário já existe".

US-003 — cadastrar - Erro Pessoa Já Vinculada

- **Objetivo:** Impedir que uma mesma Pessoa Física/Jurídica tenha dois usuários diferentes.
- **Passos para Execução:** Chamar `service.cadastrar()` para Pessoa `ID 50`.
- **Configuração do Mock:**
    - `usuarios.findByUserEquals(...)` retorna `null`.
    - `usuarios.findByPessoaCodigoEquals(50L)` retorna um objeto `Usuario` existente.
- **Resultado Esperado:** O método `usuarios.save()` **não** deve ser chamado. Retorno: "Pessoa já vinculada a outro usuário".

US-004 — cadastrar - Atualização de Usuário (Sucesso)

- **Objetivo:** Verificar a atualização de um usuário já existente.
- **Passos para Execução:** Chamar `service.cadastrar()` com um usuário que possui ID (`codigo=10L`).
- **Dados de Entrada:** Usuario: `codigo=10L`, `user="admin"`.
- **Resultado Esperado:** O código deve pular as validações de "novo usuário" e chamar `usuarios.save()`. Retorno: "Usuário atualizado com sucesso".

US-005 — cadastrar - Erro na Atualização (Exceção)

- **Objetivo:** Verificar o tratamento de erro ao atualizar (caminho `try-catch`).
- **Passos para Execução:** Chamar `service.cadastrar()` com `codigo=10L`.

- **Configuração do Mock:** `usuarios.save()` lança `RuntimeException("Erro DB")`.
- **Resultado Esperado:** O sistema deve capturar a exceção e retornar a mensagem do erro. Retorno: "Erro DB".

US-006 — addGrupo - Adicionar Grupo com Sucesso

- **Objetivo:** Vincular um novo grupo ao usuário.
- **Dados de Entrada:** `codUsu=1L`, `codGru=100L`.
- **Configuração do Mock:**
  - `usuarios.findByCodigoIn(1L)` retorna Usuario com lista de grupos vazia.
  - `grupos.buscaGrupo(100L)` retorna Grupo "ADMIN".
- **Resultado Esperado:** O grupo deve ser adicionado à lista do usuário e `usuarios.save()` deve ser chamado. Retorno: "ok".

US-007 — addGrupo - Grupo Já Existente

- **Objetivo:** Impedir duplicação de grupos na lista do usuário.
- **Dados de Entrada:** `codUsu=1L`, `codGru=100L`.
- **Configuração do Mock:** O usuário retornado já possui o Grupo "ADMIN" na sua lista (`contains` retorna true).
- **Resultado Esperado:** `usuarios.save()` **não** deve ser chamado. Retorno: "ja existe".

US-008 — removeGrupo - Remoção com Sucesso

- **Objetivo:** Remover um grupo específico da lista do usuário (Testa o loop `for`).
- **Dados de Entrada:** `codUsu=1L`, `codGru=100L`.
- **Configuração do Mock:**
  - `grupos.buscaGrupos(usuario)` retorna lista contendo Grupo 100L e Grupo 200L.
  - `grupos.buscaGrupo(100L)` retorna o objeto Grupo alvo.
- **Resultado Esperado:** A lista salva no usuário deve conter apenas o Grupo 200L (tamanho reduzido). Retorno: "ok".

US-009 — removeGrupo - Tratamento de Erro (Exceção)

- **Objetivo:** Verificar comportamento quando ocorre erro ao salvar a remoção.
- **Configuração do Mock:** `usuarios.save()` lança exceção.
- **Resultado Esperado:** O método deve capturar a exceção (printar stacktrace no console, conforme código) e retornar "ok" (conforme lógica atual do código).

US-010 — buscaUsuario - Busca Simples

- **Objetivo:** Validar a delegação da busca para o repositório.
- **Passos para Execução:** Chamar `service.buscaUsuario("admin")`.

- **Resultado Esperado:** Deve retornar o objeto vindo do mock `usuarios.findByUserEquals`.

5 - Execução dos testes

- **US-001:** OK - Teste `deveCadastrarUsuarioNovoComSucesso` validou senha criptografada e data.
- **US-002:** OK - Teste `deveImpedirCadastroUsuarioDuplicado` validou regra de login único.
- **US-003:** OK - Teste `deveImpedirCadastroPessoaVinculada` validou regra de pessoa única.
- **US-004:** OK - Teste `deveAtualizarUsuarioComSucesso` validou fluxo de update.
- **US-005:** OK - Teste `deveTratarErroNaAtualizacao` validou bloco try-catch.
- **US-006:** OK - Teste `deveAdicionarGrupoComSucesso` validou inserção na lista.
- **US-007:** OK - Teste `deveImpedirAdicaoGrupoDuplicado` validou verificação `contains`.
- **US-008:** OK - Teste `deveRemoverGrupoComSucesso` validou lógica do loop e remoção.
- **US-009:** OK - Teste `deveTratarErroNaRemocao` validou robustez contra falhas de DB.
- **US-010:** OK - Teste `deveBuscarUsuarioPorNome` validou a consulta.

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running net.originmobi.pdv.service.UsuarioServiceTest
[INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 8.913 s -- in net.originmobi.pdv.service.UsuarioServiceTest
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveImpedirCadastroUsuarioDuplicado -- Time elapsed: 8.531 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveAtualizarUsuarioComSucesso -- Time elapsed: 0.079 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveTratarErroNaRemocao -- Time elapsed: 0.017 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveRemoverGrupoComSucesso -- Time elapsed: 0.005 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveCadastrarUsuarioNovoComSucesso -- Time elapsed: 0.065 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveAdicionarGrupoComSucesso -- Time elapsed: 0.002 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveImpedirAdicaoGrupoDuplicado -- Time elapsed: 0.002 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveTratarErroNaAtualizacao -- Time elapsed: 0.061 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveBuscarUsuarioPorNome -- Time elapsed: 0.002 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveImpedirCadastroPessoaVinculada -- Time elapsed: 0.056 s
[INFO] net.originmobi.pdv.service.UsuarioServiceTest.deveListarUsuarios -- Time elapsed: 0.002 s
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  45.143 s
[INFO] Finished at: 2025-11-30T12:39:08Z
[INFO] ------------------------------------------------------------------------
```

Estrutural:

Relatório de cobertura no critério todas-arestas (JaCoCo)

UsuarioService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
cadastrar(Usuario)		100%		100%	0	4	0	21	0	1
removeGrupo(Long, Long)		100%		75%	1	3	0	13	0	1
addGrupo(Long, Long)		100%		100%	0	2	0	9	0	1
UsuarioService()		100%		n/a	0	1	0	2	0	1
buscaUsuario(String)		100%		n/a	0	1	0	1	0	1
lista()		100%		n/a	0	1	0	1	0	1
Total	0 of 177	100%	1 of 12	91%	1	12	0	47	0	6

Baseada em Defeitos:

Score de Mutação

```
----------------------------------------------------------------------------
============================================================================
- Timings
============================================================================
> pre-scan for mutations : 1 seconds
> scan classpath : < 1 second
> coverage and dependency analysis : 9 seconds
> build mutation tests : 1 seconds
> run mutation analysis : 14 seconds
----------------------------------------------------------------------------
> Total  : 27 seconds
----------------------------------------------------------------------------
============================================================================
- Statistics
============================================================================
>> Line Coverage: 47/47 (100%)
>> Generated 18 mutations Killed 17 (94%)
>> Mutations with no coverage 0. Test strength 94%
>> Ran 34 tests (1.89 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:05 min
[INFO] Finished at: 2025-11-30T12:52:54Z
[INFO] ------------------------------------------------------------------------
```

Usuário - Funcional

Documento de Casos de Teste: Gestão de Usuários

Produto/Funcionalidade: Gestão de Usuários
Data: 30 de Novembro de 2025
Autor do Teste: Elena Blanco
Tipo de teste: Sistema

Pré-requisito:
Acesso ao sistema e login realizado com sucesso (perfil administrador).

Navegação para o módulo de Segurança ou Cadastro de Usuários.

Existência de "Pessoas" previamente cadastradas no sistema (para preencher o dropdown "Pessoa").

Caso de Teste 1: Cadastro de Novo Usuário (Sucesso)

ID: F-USR-001
Objetivo: Validar a criação de um novo usuário vinculando-o a uma pessoa existente.

Ação: Observar o campo "Código".
Resposta Esperada: O campo deve estar desabilitado (cinza) e vazio, aguardando geração automática pelo banco.

Ação: Preencher o campo Usuário com elena.blanco.
Resposta Esperada: O campo aceita e exibe o valor digitado.

Ação: Preencher o campo Senha com 123.
Resposta Esperada: O campo aceita o valor e oculta os caracteres (máscara de senha), se aplicável, ou exibe o texto.

Ação: Clicar no campo de seleção "Pessoa".
Resposta Esperada: O sistema exibe uma lista dropdown com os nomes das pessoas cadastradas.

Ação: Selecionar a pessoa Elena Costa Blanco.
Resposta Esperada: O campo exibe o nome selecionado.

Ação: Clicar no botão "Salvar".
Resposta Esperada: O sistema deve exibir uma mensagem de sucesso (ex: "Usuário salvo com sucesso").

Ação: Verificar o redirecionamento ou limpeza da tela.
Resposta Esperada: Os campos são limpos ou o sistema navega para a lista, e o novo usuário deve constar no banco de dados.

Caso de Teste 2: Atualização de Usuário Existente (Sucesso)

ID: F-USR-002
Objetivo: Verificar se é possível alterar a senha de um usuário existente.

Ação: Na tela de listagem (acessada pelo botão "Listar"), selecionar o usuário elena.blanco para edição.
Resposta Esperada: O formulário é carregado com o campo Usuário elena.blanco preenchido e a Pessoa selecionada. O campo Senha deve vir vazio ou mascarado para redefinição.

Ação: Preencher o campo Senha com uma nova senha: NovaSenha123.
Resposta Esperada: O campo aceita o novo valor**.**

Ação: Manter os outros campos inalterados e clicar no botão "Salvar".
Resposta Esperada: O sistema processa a atualização e exibe mensagem de sucesso.

Ação: Realizar logoff e tentar logar com a nova senha.
Resposta Esperada: O login deve ser realizado com sucesso utilizando a NovaSenha123.

Caso de Teste 3: Validação de Login Duplicado (Sucesso)

ID: F-USU-NEG-003

Objetivo: Garantir que o sistema não permita dois usuários com o mesmo login (username).

Ação: No formulário de cadastro, preencher o campo Usuário com elena.blanco (usuário já existente na imagem).
Resposta Esperada: O campo aceita o valor.

Ação: Preencher o campo Senha com qualquer valor válido.

Ação: Selecionar uma Pessoa diferente no dropdown.

Ação: Clicar no botão "Salvar".
Resposta Esperada: O sistema deve validar a duplicidade no backend e impedir o cadastro.

Ação: Após o processamento.
Resposta Esperada: O sistema deve exibir uma mensagem de erro clara (ex: "Nome de usuário já existe" ou "Login indisponível"). O registro não é salvo.

Caso de Teste 4: Validação de Campos Obrigatórios (Sucesso)

ID: F-USU-NEG-004
Objetivo: Verificar se o sistema impede o cadastro sem vincular uma "Pessoa" ou sem definir "Senha".

Ação: Preencher o campo Usuário com elena.

Ação: Deixar o campo Senha vazio.

Ação: Deixar o campo Pessoa sem seleção (ou na opção padrão "Selecione...").

Ação: Clicar no botão "Salvar".
Resposta Esperada: O sistema não deve submeter o formulário.

Ação: Verificar indicações visuais.
Resposta Esperada: Mensagens de campo obrigatório devem ser exibidas abaixo dos campos Senha e Pessoa, ou uma mensagem geral de alerta deve aparecer.

Caso de Teste 5: Navegação para Listagem (Sucesso)

ID: F-USU-NAV-005

Objetivo: Verificar a funcionalidade do botão de navegação "Listar".

Ação: Estando na tela de cadastro (conforme imagem), clicar no botão azul "Listar" localizado no topo direito.
Resposta Esperada: O sistema deve redirecionar o usuário para a tela de Consulta/Listagem de Usuários.

Ação: Verificar o carregamento da grid.
Resposta Esperada: A lista de usuários cadastrados deve ser exibida corretamente.

Foi criada também uma medida para testar usabilidade:

```
Total de cliques realizados no fluxo: 2
Usabilidade: OK (≤5 cliques)
```

Usuário - Integração

Teste de Integração - Gerenciar Usuário

- **Produto/Funcionalidade:** Gestão de Usuários
- **Data:** 30 de Novembro de 2025
- **Autor do Teste:** Elena Costa Blanco
- **Tipo de Teste:** Integração

Testes feitos:


```
{
    "info": {
        "_postman_id": "aec1a8c2-fd3c-4be5-b4b6-eb72aa1b16f5",
        "name": "Gerenciamento de Produto - Integração",
        "schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
        "_exporter_id": "50459654",
        "_collection_link":
"https://ecbcasais-429803.postman.co/workspace/Elena-Blanco's-Workspace~7fa82a
17-d3b2-4d24-89c0-dda69e59cc36/collection/50459654-aec1a8c2-fd3c-4be5-b4b6-eb7
2aa1b16f5?action=share&source=collection_link&creator=50459654"
    },
    "item": [
        {
            "name": "Login",
            "request": {
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "urlencoded",
                    "urlencoded": [
                        {
                            "key": "username",
                            "value": "gerente",
                            "type": "text"
                        },
                        {
                            "key": "password",
                            "value": "123",
                            "type": "text"
                        }
                    ]
                },
```

```
                "url": {
                    "raw": "http://localhost:8080/login",
                    "protocol": "http",
                    "host": [
                        "localhost"
                    ],
                    "port": "8080",
                    "path": [
                        "login"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "2. Criar Produto",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "// Verifica se a resposta é um redirecionamento
(status 302) e se o Location header contém o ID do produto",
                            "pm.test(\"Status code is 302 Found
(Redirecionamento)\", function () {",
                            "    pm.response.to.have.status(302);",
                            "});",
                            "",
                            "var locationHeader =
pm.response.headers.get('Location');",
                            "pm.test(\"Redirecionamento para a página do novo
produto\", function () {",
                            "
pm.expect(locationHeader).to.include('/produto/');",
                            "});",
                            "",
                            "// Extrai o ID do produto do header Location",
                            "if (locationHeader) {",
                            "    var produtoId =
locationHeader.split('/').pop();",
                            "    pm.environment.set(\"produto_id\",
produtoId);",
```

```
                            "    console.log(\"Produto ID criado: \" +
produtoId);",
                            "    pm.test(\"ID do produto extraído e salvo\",
function () {",
                            "
pm.expect(produtoId).to.be.a('string').and.to.have.lengthOf.at.least(1);",
                            "    });",
                            "}"
                        ],
                        "type": "text/javascript",
                        "packages": {},
                        "requests": {}
                    }
                }
            ],
            "request": {
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "urlencoded",
                    "urlencoded": [
                        {
                            "key": "codigo",
                            "value": "0",
                            "description": "0 para criação",
                            "type": "text"
                        },
                        {
                            "key": "descricao",
                            "value": "Produto Teste",
                            "type": "text"
                        },
                        {
                            "key": "fornecedor",
                            "value": "1",
                            "type": "text"
                        },
                        {
                            "key": "categoria",
                            "value": "1",
                            "type": "text"
                        },
```

```
                    {
                        "key": "grupo",
                        "value": "1",
                        "type": "text"
                    },
                    {
                        "key": "balanca",
                        "value": "NAO",
                        "type": "text"
                    },
                    {
                        "key": "valor_custo",
                        "value": "10,50",
                        "type": "text"
                    },
                    {
                        "key": "valor_venda",
                        "value": "25,99",
                        "type": "text"
                    },
                    {
                        "key": "data_validade",
                        "value": "31/12/2025",
                        "type": "text"
                    },
                    {
                        "key": "controla_estoque",
                        "value": "SIM",
                        "type": "text"
                    },
                    {
                        "key": "ativo",
                        "value": "ATIVO",
                        "type": "text"
                    },
                    {
                        "key": "unidade",
                        "value": "1",
                        "type": "text"
                    },
                    {
                        "key": "subtributaria",
```

```
                    "value": "NAO",
                    "type": "text"
                },
                {
                    "key": "ncm",
                    "value": "12345678",
                    "type": "text"
                },
                {
                    "key": "cest",
                    "value": "",
                    "type": "text"
                },
                {
                    "key": "tributacao",
                    "value": "",
                    "type": "text"
                },
                {
                    "key": "modBcIcms",
                    "value": "1",
                    "type": "text"
                },
                {
                    "key": "vendavel",
                    "value": "SIM",
                    "type": "text"
                }
            ]
        },
        "url": {
            "raw": "http://localhost:8080/produto",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "8080",
            "path": [
                "produto"
            ]
        }
    },
```

```
            "response": []
        },
        {
            "name": "3. Buscar Produto por ID",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200 OK\", function ()
{",
                            "    pm.response.to.have.status(200);",
                            "});",
                            "pm.test(\"Corpo da resposta contém a descrição do
produto\", function () {",
                            "
pm.expect(pm.response.text()).to.include('Produto Teste - Criacao');",
                            "});"
                        ],
                        "type": "text/javascript",
                        "packages": {},
                        "requests": {}
                    }
                }
            ],
            "request": {
                "method": "GET",
                "header": [],
                "url": {
                    "raw": "{{base_url}}/produto/1",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "produto",
                        "1"
                    ]
                }
            },
            "response": []
        },
        {
```

```
            "name": "4. Atualizar Produto",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 302 Found
(Redirecionamento)\", function () {",
                            "    pm.response.to.have.status(302);",
                            "});",
                            "pm.test(\"Redirecionamento para a página do
produto atualizado\", function () {",
                            "
pm.expect(pm.response.headers.get('Location')).to.include('/produto/' +
pm.environment.get('produto_id'));",
                            "});"
                        ],
                        "type": "text/javascript",
                        "packages": {},
                        "requests": {}
                    }
                }
            ],
            "request": {
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "urlencoded",
                    "urlencoded": [
                        {
                            "key": "codigo",
                            "value": "9",
                            "description": "ID do produto para atualização",
                            "type": "text"
                        },
                        {
                            "key": "descricao",
                            "value": "Produto Teste - ATUALIZADO",
                            "type": "text"
                        },
                        {
                            "key": "fornecedor",
```

```
                "value": "1",
                "type": "text"
            },
            {
                "key": "categoria",
                "value": "1",
                "type": "text"
            },
            {
                "key": "grupo",
                "value": "1",
                "type": "text"
            },
            {
                "key": "balanca",
                "value": "SIM",
                "type": "text"
            },
            {
                "key": "valor_custo",
                "value": "15,00",
                "type": "text"
            },
            {
                "key": "valor_venda",
                "value": "35,50",
                "type": "text"
            },
            {
                "key": "data_validade",
                "value": "31/12/2026",
                "type": "text"
            },
            {
                "key": "controla_estoque",
                "value": "NAO",
                "type": "text"
            },
            {
                "key": "ativo",
                "value": "ATIVO",
                "type": "text"
```

```
                },
                {
                    "key": "unidade",
                    "value": "PC",
                    "type": "text"
                },
                {
                    "key": "subtributaria",
                    "value": "SIM",
                    "type": "text"
                },
                {
                    "key": "ncm",
                    "value": "87654321",
                    "type": "text"
                },
                {
                    "key": "cest",
                    "value": "0100100",
                    "type": "text"
                },
                {
                    "key": "tributacao",
                    "value": "",
                    "type": "text"
                },
                {
                    "key": "modBcIcms",
                    "value": "1",
                    "type": "text"
                },
                {
                    "key": "vendavel",
                    "value": "NAO",
                    "type": "text"
                }
            ]
        },
        "url": {
            "raw": "{{base_url}}/produto",
            "host": [
                "{{base_url}}"
```

```
                    ],
                    "path": [
                        "produto"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "5. Desativar Produto",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 302 Found
(Redirecionamento)\", function () {",
                            "    pm.response.to.have.status(302);",
                            "});",
                            "pm.test(\"Redirecionamento para a página do
produto desativado\", function () {",
                            "
pm.expect(pm.response.headers.get('Location')).to.include('/produto/' +
pm.environment.get('produto_id'));",
                            "});"
                        ],
                        "type": "text/javascript",
                        "packages": {},
                        "requests": {}
                    }
                }
            ],
            "request": {
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "urlencoded",
                    "urlencoded": [
                        {
                            "key": "codigo",
                            "value": "9",
                            "description": "ID do produto para desativação",
```

```
            "type": "text"
        },
        {
            "key": "descricao",
            "value": "Produto Teste - ATUALIZADO",
            "type": "text"
        },
        {
            "key": "fornecedor",
            "value": "1",
            "type": "text"
        },
        {
            "key": "categoria",
            "value": "1",
            "type": "text"
        },
        {
            "key": "grupo",
            "value": "1",
            "type": "text"
        },
        {
            "key": "balanca",
            "value": "SIM",
            "type": "text"
        },
        {
            "key": "valor_custo",
            "value": "15,00",
            "type": "text"
        },
        {
            "key": "valor_venda",
            "value": "35,50",
            "type": "text"
        },
        {
            "key": "data_validade",
            "value": "31/12/2026",
            "type": "text"
        },
```

```
                    {
                        "key": "controla_estoque",
                        "value": "NAO",
                        "type": "text"
                    },
                    {
                        "key": "ativo",
                        "value": "INATIVO",
                        "description": "Muda o status para INATIVO",
                        "type": "text"
                    },
                    {
                        "key": "unidade",
                        "value": "PC",
                        "type": "text"
                    },
                    {
                        "key": "subtributaria",
                        "value": "SIM",
                        "type": "text"
                    },
                    {
                        "key": "ncm",
                        "value": "87654321",
                        "type": "text"
                    },
                    {
                        "key": "cest",
                        "value": "0100100",
                        "type": "text"
                    },
                    {
                        "key": "tributacao",
                        "value": "1",
                        "type": "text"
                    },
                    {
                        "key": "modBcIcms",
                        "value": "1",
                        "type": "text"
                    },
                    {
```

```
                            "key": "vendavel",
                            "value": "NAO",
                            "type": "text"
                        }
                    ]
                },
                "url": {
                    "raw": "{{base_url}}/produto",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "produto"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "6. Listar Produtos",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200 OK\", function ()
{",
                            "    pm.response.to.have.status(200);",
                            "});",
                            "pm.test(\"Corpo da resposta contém a descrição do
produto atualizado\", function () {",
                            "
pm.expect(pm.response.text()).to.include('Produto Teste - ATUALIZADO');",
                            "});"
                        ],
                        "type": "text/javascript"
                    }
                }
            ],
            "request": {
                "method": "GET",
                "header": [],
```

```
                "url": {
                    "raw": "{{base_url}}/produto",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "produto"
                    ]
                }
            },
            "response": []
        }
    ]
}
```


Resultado Testes:


```
{
    "id": "8c241993-5789-48de-a2cb-3e018a7c2d77",
    "name": "Gerenciamento de Produto - Integração",
    "timestamp": "2025-11-30T14:28:10.560Z",
    "collection_id": "50459654-aec1a8c2-fd3c-4be5-b4b6-eb72aa1b16f5",
    "folder_id": 0,
    "environment_id": "50459654-bc581ff7-2fb6-4c48-a64a-f7ec8ec6f6f9",
    "totalPass": 0,
    "delay": 0,
    "persist": true,
    "status": "error",
    "startedAt": "2025-11-30T14:28:08.562Z",
    "totalFail": 2,
    "results": [
        {
            "id": "14a9f308-61e0-4ecc-9bd2-2ddc0f6b1f7d",
            "name": "2. Criar Produto",
            "url": "http://localhost:8080/produto",
            "time": 65,
            "responseCode": {
                "code": 200,
                "name": "OK"
            },
            "tests": {
```

```
                "Status code is 302 Found (Redirecionamento)": false,
                "Redirecionamento para a página do novo produto": false
            },
            "testPassFailCounts": {
                "Status code is 302 Found (Redirecionamento)": {
                    "pass": 0,
                    "fail": 1
                },
                "Redirecionamento para a página do novo produto": {
                    "pass": 0,
                    "fail": 1
                }
            },
            "times": [
                65
            ],
            "allTests": [
                {
                    "Status code is 302 Found (Redirecionamento)": false,
                    "Redirecionamento para a página do novo produto": false
                }
            ]
        }
    ],
    "count": 1,
    "totalTime": 65,
    "collection": {
        "requests": [
            {
                "id": "14a9f308-61e0-4ecc-9bd2-2ddc0f6b1f7d",
                "method": "POST"
            },
            {
                "id": "32360d05-7700-429c-aa49-9352937ae8a6",
                "method": "GET"
            }
        ]
    }
}
```

Vendas

Testes para Vendas

VendaService - Unitários

Documento de Casos de Teste: VendaService – Gestão de Pedidos/Vendas

- **Produto/Funcionalidade:** Gestão de Pedidos/Vendas
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Mikael Schwind

1 - Objetivo

Verificar se o serviço *VendaService* gerencia corretamente o ciclo de vida de uma venda, desde a abertura (*abreVenda*), manipulação de itens (*addProduto*, *removeProduto*), consulta (busca) até o fechamento complexo (*fechaVenda*), garantindo que lançamentos financeiros (Caixa/Cartão), geração de parcelas e movimentação de estoque ocorram conforme as regras de negócio e validações de estado (Aberta/Fechada).

2 - Escopo do Teste

Os métodos testados cobrem:

- **Gestão de Venda:** abreVenda (Insert/Update), busca, lista, qtdAbertos.
- **Gestão de Itens:** addProduto, removeProduto.
- **Fechamento Financeiro:** fechaVenda (Dinheiro, Cartão, A Prazo, Validações e Tratamento de Erros).

3 - Pré-requisitos

- O sistema deve possuir acesso aos repositórios e serviços mockados: *VendaRepository, UsuarioService, VendaProdutoService, PagamentoTipoService, CaixaService, ReceberService, ParcelaService, CaixaLancamentoService, TituloService, CartaoLancamentoService, ProdutoService*
- Classes de modelo (*Venda, VendaProduto, Receber, Titulo, Caixa, Usuario, PagamentoTipo*) devem estar disponíveis.
- Dependências estáticas como *Aplicacao* devem ser mockadas via *MockedStatic* ou similar.

4 - Casos de Teste

VS-001 — Abertura e Atualização de Venda

Objetivo: Verificar a criação de nova venda e atualização de existente.

Cenários:

1. **Nova Venda:** *codigo* = *null.* Deve salvar, definir status ABERTA, valor zero e associar usuário logado.

2. **Atualização:** *codigo != null.* Deve chamar *updateDadosVenda* com nova observação e pessoa. **Resultado Esperado:**
   - Para nova: *vendas.save()* invocado. Status ABERTA.
   - Para update: *vendas.updateDadosVenda()* invocado. *vendas.save()* **não** deve ser chamado.

VS-002 — Tratamento de Erros na Abertura/Atualização

Objetivo: Garantir que falhas no banco não quebrem a aplicação (try/catch silencioso conforme código). **Cenários:**

1. Erro ao salvar nova venda (*RuntimeException).*
2. Erro ao atualizar venda existente *(RuntimeException).* **Resultado Esperado:**
   - O sistema deve capturar a exceção e não propagar erro (retornando *null* ou o código, dependendo do fluxo).

VS-003 — Busca de Vendas (Filtros)

Objetivo: Validar o roteamento da busca conforme os filtros preenchidos. **Cenários:**

1. **Com Código:** Filter possui ID. Deve chamar *vendas.findByCodigoIn.*
2. **Sem Código (Situação):** Filter sem ID. Deve chamar *vendas.findBySituacaoEquals* (passando ABERTA ou FECHADA conforme string recebida). **Resultado Esperado:**
   - O repositório correto deve ser acionado e o outro método de busca nunca deve ser chamado.

VS-004 — Adicionar Produto

Objetivo: Validar adição de itens respeitando o estado da venda. **Cenários:**

1. **Venda Aberta:** Deve chamar *vendaProdutos.salvar()* e retornar "ok".
2. **Venda Fechada:** Não deve salvar e retornar "Venda fechada".
3. **Erro no Banco:** Se ocorrer exceção ao salvar, deve capturar e retornar "ok" (conforme lógica atual).

VS-005 — Remover Produto

Objetivo: Validar remoção de itens e mensagens de erro. **Cenários:**

1. **Venda Inexistente:** Retornar "Venda não encontrada".
2. **Venda Aberta:** Chamar *vendaProdutos.removeProduto()* e retornar "ok".
3. **Venda Fechada:** Retornar "Venda fechada" sem remover.
4. **Erro Interno:** Exceção ao buscar venda deve ser tratada e retornar "ok".

VS-006 — Listagem e Contagem

Objetivo: Verificar métodos auxiliares de listagem. **Cenários:**

1. *lista():* Deve retornar lista de vendas ou lista vazia corretamente.
2. *qtdAbertos():* Deve retornar inteiro vindo de *vendas.qtdVendasEmAberto().*

VS-007 — Fechamento: Pagamento em DINHEIRO (Sucesso)

Objetivo: Validar fluxo completo de venda à vista em dinheiro.
Pré-condição: Caixa Aberto.
Dados: Pagamento "00", Título "DIN". **Resultado Esperado:**

- *receberService.cadastrar() invocado.*
- *lancamentos.lancamento() (Caixa) invocado.*
- *cartaoLancamento **não** invocado.*
- *produtos.movimentaEstoque(SAIDA) invocado.*
- *Retorno: "Venda finalizada com sucesso".*

VS-008 — Fechamento: Pagamento em CARTÃO (Sucesso)

Objetivo: Validar fluxo de venda Crédito e Débito. **Dados:** Pagamento "00", Título "CARTCRED" ou "CARTDEB". **Resultado Esperado:**

- *cartaoLancamento.lancamento()* invocado.
- *lancamentos.lancamento()* (Caixa) **não** invocado.

VS-009 — Fechamento: Títulos Diversos (Ignorar Lançamento)

Objetivo: Verificar comportamento para títulos que não são nem Dinheiro nem Cartão (ex: Cheque/Outros).
Dados: Título com sigla desconhecida (ex: "CHEQUE"). **Resultado Esperado:**
- Venda é fechada, mas **nenhum** lançamento (Caixa ou Cartão) é realizado.

VS-010 — Fechamento: A PRAZO (Sucesso)

Objetivo: Validar geração de parcelas. **Dados:** Pagamento "30/60" (Duas vezes). Título qualquer. **Resultado Esperado:**
- *parcelas.gerarParcela()* deve ser invocado 2 vezes com as datas e valores corretos.

VS-011 — Validações Bloqueantes de Fechamento (Regra de Negócio)

Objetivo: Garantir integridade antes de fechar. **Cenários e Resultados:**

1. **Caixa Fechado (Dinheiro):** Lança exceção "nenhum caixa aberto".
2. **Sem Cliente (A Prazo):** Lança exceção "Venda sem cliente, verifique".
3. **Venda já Fechada:** Lança exceção "venda fechada".
4. **Valores Inválidos:** Valor produto Zero ou Negativo -> "Venda sem valor, verifique".
5. **Divergência de Valores:** Soma das parcelas != Valor produtos -> "Valor das parcelas diferente do valor total...".
6. **Parcela Vazia:** String da parcela vazia -> "valor de recebimento invalido" ou "Parcela sem valor".

VS-012 — Tratamento de Erros Críticos no Fechamento (Suporte)

Objetivo: Verificar se erros de banco durante o processo crítico lançam mensagem amigável "chame o suporte". **Cenários:**

1. Erro ao salvar *Receber*.
2. Erro ao realizar *CaixaLancamento*.
3. Erro ao realizar o update final em *Venda*. **Resultado Esperado:**
   - *RuntimeException* com mensagem "Erro ao fechar a venda, chame o suporte".

VS-013 — Tratamento de Erro na Geração de Parcela

Objetivo: Verificar comportamento se falhar a geração de uma parcela específica. **Cenário:** *parcelas.gerarParcela* lança exceção. **Resultado Esperado:** Exceção capturada e relançada como *RuntimeException* genérica (sem mensagem, conforme teste *deveTratarErroGerarParcela*).

5 - Execução dos Testes (Resumo do JUnit)

- **VS-001:** OK (*deveAbrirNovaVenda, deveAtualizarVendaExistente*).
- **VS-002:** OK (*deveTratarErroAoSalvarNovaVenda, deveTratarErroAoAtualizarVenda*).
- **VS-003:** OK (*deveBuscarPorCodigoQuandoFiltroTemId, deveBuscarPorSituacaoAberta..., deveBuscarPorSituacaoFechada...*).
- **VS-004:** OK (*deveAdicionarProduto..., naoDeveAdicionar..., deveTratarErroAoAdicionarProduto*).
- **VS-005:** OK (d*eveRetornarErroQuandoVendaNaoExiste, deveRemoverProdutoComSucesso, naoDeveRemover..., deveTratarErroInterno*).
- **VS-006:** OK (*deveRetornarListaDeVendas, deveRetornarListaVazia, deveRetornarQtdVendasEmAberto*).
- **VS-007:** OK (*deveFecharVendaNoDinheiro*).
- **VS-008:** OK (*deveFecharVendaNoCartaoCredito*, *deveFecharVendaNoCartaoDebito*).
- **VS-009:** OK (*deveIgnorarLancamentoSeTituloNaoForDinheiroNemCartao*).
- **VS-010:** OK (*deveFecharVendaAPrazo*).
- **VS-011:** OK
    - *deveFalharSePagarDinheiroSemCaixaAberto*
    - *deveFalharSeVendaPrazoSemCliente*
    - *deveLancarErroSeVendaJaFechada*
    - *deveLancarErroSeValorProdutosInvalido*
    - *deveLancarErroSeValorParcelasIncorreto*
    - *deveFalharAprazoParcelaVazia e deveFalharAvistaDinheiroParcelaVazia*
- **VS-012:** OK (*deveTratarErroUpdateFinalVenda, deveTratarErroAoCadastrarReceber, deveTratarErroLancamentoCaixa*).
- **VS-013:** OK (*deveTratarErroGerarParcela*).

Cobertura Estrutural (JaCoCo): Rode *mvn test.* Abra o relatório em *target/site/jacoco/index.html.*

VendaService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
fechaVenda(Long, Long, Double, Double, Double, String[], String[])		100%		100%	0	11	0	48	0	1
avistaDinheiro(Double, String[], String[], int, int, Double, Double)		100%		100%	0	4	0	20	0	1
aprazo(Double, String[], DataAtual, String[], int, int, Receber, int, Double, Double)		100%		100%	0	2	0	12	0	1
abreVenda(Venda)		100%		100%	0	2	0	17	0	1
addProduto(Long, Long, Double)		100%		100%	0	2	0	11	0	1
removeProduto(Long, Long)		100%		100%	0	3	0	10	0	1
busca(VendaFilter, String, Pageable)		100%		100%	0	3	0	4	0	1
VendaService()		100%		n/a	0	1	0	2	0	1
vendaIsAberta(Long)		100%		n/a	0	1	0	2	0	1
lista()		100%		n/a	0	1	0	1	0	1
qtdAbertos()		100%		n/a	0	1	0	1	0	1
Total	0 of 588	100%	0 of 40	100%	0	31	0	128	0	11

Escore de Mutação (PITest): Rode *mvn org.pitest:pitest-maven:mutationCoverage*. Abra *target/pit-reports/index.html.*

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
1	100%	122/122	78%	54/69	78%	54/69

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
net.originmobi.pdv.service	1	100%	122/122	78%	54/69	78%	54/69

Vendas - Funcional

Documento de Casos de Teste: Gestão de Vendas

- **Produto/Funcionalidade:** Gestão de Vendas
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Mikael Schwind
- **Tipo de Teste:** Sistema

Pré-requisitos

- Acesso ao sistema e login realizado com sucesso.
- Existência de um Título financeiro cadastrado (Realizado no setup).
- Caixa aberto com saldo inicial (Realizado no setup).
- Existência prévia de **Clientes** e **Produtos** cadastrados no banco de dados para seleção.

Caso de Teste 1: Fluxo Completo de Venda (Criação, Itens, Desconto e Fechamento)

ID: F-VENDA-001 **Referência:** RF-V.01, RF-V.02, RF-V.03, RF-V.04 **Objetivo:** Validar o ciclo completo de uma venda, desde a abertura do pedido, inserção de produtos, aplicação de valores monetários extras e finalização com pagamento.

- **Ação:** Na tela de listagem de vendas, clicar no botão "Novo Pedido".
    - **Resposta Esperada:** O sistema redireciona para o formulário de abertura de venda.
- **Ação:** Selecionar um cliente na lista de seleção
    - **Resposta Esperada:** O cliente é selecionado.
- **Ação:** Preencher o campo Observação com "Teste Selenium".
    - **Resposta Esperada:** O campo aceita o texto.
- **Ação:** Clicar no botão "Salvar".
    - **Resposta Esperada:** A venda é iniciada e o sistema libera a seção de adição de itens (RF-V.01).
- **Ação:** Clicar no botão dropdown de seleção de produtos e selecionar o produto "Picolé".
    - **Resposta Esperada:** O produto é selecionado no campo de busca.
- **Ação:** Clicar no botão "Inserir".
    - **Resposta Esperada:** O produto é adicionado à lista de itens da venda e o subtotal é atualizado (RF-V.02).
- **Ação:** Clicar no botão "Gerar Venda" (botão verde de finalizar pedido).

    - **Resposta Esperada:** O sistema redireciona para a tela de Pagamento/Fechamento.
- **Ação:** Selecionar a Forma de Pagamento.
    - **Resposta Esperada:** A forma de pagamento é definida.
- **Ação:** Preencher o campo Desconto com "1".
    - **Resposta Esperada:** O valor é aceito e o total a pagar deve ser recalculado internamente (RF-V.03).
- **Ação:** Preencher o campo Acréscimo com "1".
    - **Resposta Esperada:** O valor é aceito e o total a pagar deve ser recalculado internamente.
- **Ação:** Clicar no botão "Pagar".
- **Resposta Esperada:** O sistema processa a transação e exibe um *Alert* do navegador com a mensagem "Venda finalizada com sucesso" (RF-V.04).
- **Ação:** Aceitar o alerta.
    - **Resposta Esperada:** O alerta é fechado e a venda é concluída.

Caso de Teste 2: Validação de Desempenho no Fechamento (Requisito Não Funcional)

ID: NF-VENDA-002 **Referência:** RNF-01 (Desempenho) **Objetivo:** Verificar se o tempo de resposta do servidor entre a solicitação de pagamento e a confirmação de sucesso está dentro do SLA aceitável (< 3 segundos).

- **Ação:** Realizar todos os passos de preparação da venda (Novo Pedido -> Adicionar Item -> Ir para Pagamento).
    - **Resposta Esperada:** O sistema está na tela de fechamento com itens lançados.
- **Ação:** Clicar no botão "Pagar" e iniciar a medição de tempo.
    - **Resposta Esperada:** O sistema inicia o processamento da requisição `POST` para fechar a venda.
- **Ação:** Aguardar até que o alerta de confirmação seja exibido.
    - **Resposta Esperada:** O alerta surge na tela.
- **Ação:** Calcular a diferença de tempo (Fim - Início).
    - **Resposta Esperada:** O tempo decorrido deve ser estritamente menor que 3000 ms (3 segundos). Caso contrário, o teste deve falhar.

Caso de Teste 3: Abertura de Caixa (Setup)

ID: F-CAIXA-003 **Referência:** Pré-requisito para Vendas **Objetivo:** Garantir que o caixa esteja aberto antes de tentar realizar uma venda.

- **Ação:** Navegar para a rota */caixa* e clicar em "Abrir Novo".
    - **Resposta Esperada:** Formulário de abertura de caixa é exibido.
- **Ação:** Preencher Descrição e Valor de Abertura.
    - **Resposta Esperada:** Campos preenchidos.
- **Ação:** Clicar em "Abrir".
    - **Resposta Esperada:** O sistema exibe uma mensagem de sucesso confirmando que o caixa está aberto e apto a receber vendas.

VendaController - Integração

Documento de Casos de Teste: Gestão de Vendas

- **Produto/Funcionalidade:** Gestão de Vendas
- **Data:** 29 de Novembro de 2025
- **Autor do Teste:** Mikael Schwind
- **Tipo de Teste:** Integração

JSON


```
{
    "id": "d705a4c0-5274-4e4d-989f-b38f1e861b73",
    "name": "PDV - Integração Vendas",
    "timestamp": "2025-11-30T14:48:15.319Z",
    "collection_id": "28492228-0b2b2119-af74-4663-ba9e-2ba33424ddcf",
    "folder_id": 0,
    "environment_id": "28492228-998d338f-20ba-4c89-b8f2-8902bbd9aa75",
    "totalPass": 9,
    "delay": 0,
    "persist": true,
    "status": "finished",
    "startedAt": "2025-11-30T14:48:14.694Z",
    "totalFail": 0,
    "results": [
        {
            "id": "97b3a201-420a-4ae5-aa5a-474a98675f51",
            "name": "Login (Sessão)",
            "url": "http://localhost:8080/login",
            "time": 9,
            "responseCode": {
                "code": 200,
                "name": "OK"
            },
            "tests": {
                "Produto adicionado com sucesso": true,
                "Logado com sucesso'": true
            },
            "testPassFailCounts": {
                "Produto adicionado com sucesso": {
                    "pass": 1,
                    "fail": 0
                },
                "Logado com sucesso'": {
```

```
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            9
        ],
        "allTests": [
            {
                "Produto adicionado com sucesso": true,
                "Logado com sucesso'": true
            }
        ]
    },
    {
        "id": "9c3e6d4d-0103-4062-a0dd-0bcc36a548fb",
        "name": "1. Abrir Venda (Cria Pedido)",
        "url": "http://localhost:8080/venda",
        "time": 48,
        "responseCode": {
            "code": 200,
            "name": "OK"
        },
        "tests": {
            "Status code is 200": true,
            "HTML contem tabela de 'Pedido 'Salvo'": true
        },
        "testPassFailCounts": {
            "Status code is 200": {
                "pass": 1,
                "fail": 0
            },
            "HTML contem tabela de 'Pedido 'Salvo'": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            48
        ],
        "allTests": [
            {
                "Status code is 200": true,
                "HTML contem tabela de 'Pedido 'Salvo'": true
            }
        ]
    },
    {
```

```
        "id": "a218bffc-deb7-4a95-94b8-f69bd496091f",
        "name": "2. Adicionar Produto na Venda",
        "url": "http://localhost:8080/venda/addproduto",
        "time": 21,
        "responseCode": {
            "code": 200,
            "name": "OK"
        },
        "tests": {
            "Produto adicionado com sucesso": true,
            "Chamada retorna mensagem 'feito'": true
        },
        "testPassFailCounts": {
            "Produto adicionado com sucesso": {
                "pass": 1,
                "fail": 0
            },
            "Chamada retorna mensagem 'feito'": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            21
        ],
        "allTests": [
            {
                "Produto adicionado com sucesso": true,
                "Chamada retorna mensagem 'feito'": true
            }
        ]
    },
    {
        "id": "e76f126a-0eb4-4b21-98f7-dcf9b718e592",
        "name": "4. Listar Títulos (JSON)",
        "url": "http://localhost:8080/venda/titulos",
        "time": 17,
        "responseCode": {
            "code": 200,
            "name": "OK"
        },
        "tests": {
            "Produto adicionado com sucesso": true,
            "Retornou JSON de títulos": true
        },
        "testPassFailCounts": {
            "Produto adicionado com sucesso": {
                "pass": 1,
```

```
                "fail": 0
            },
            "Retornou JSON de títulos": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            17
        ],
        "allTests": [
            {
                "Produto adicionado com sucesso": true,
                "Retornou JSON de títulos": true
            }
        ]
    },
    {
        "id": "2a8f8cc4-3cf9-402f-a56c-94173f78ce72",
        "name": "6. Verificar Status (Listagem)",
        "url": "http://localhost:8080/venda/4",
        "time": 24,
        "responseCode": {
            "code": 200,
            "name": "OK"
        },
        "tests": {
            "HTML contem tabela de produtos": true
        },
        "testPassFailCounts": {
            "HTML contem tabela de produtos": {
                "pass": 1,
                "fail": 0
            }
        },
        "times": [
            24
        ],
        "allTests": [
            {
                "HTML contem tabela de produtos": true
            }
        ]
    }
],
"count": 1,
"totalTime": 119,
"collection": {
```

```
        "requests": [
            {
                "id": "97b3a201-420a-4ae5-aa5a-474a98675f51",
                "method": "POST"
            },
            {
                "id": "9c3e6d4d-0103-4062-a0dd-0bcc36a548fb",
                "method": "POST"
            },
            {
                "id": "a218bffc-deb7-4a95-94b8-f69bd496091f",
                "method": "POST"
            },
            {
                "id": "e76f126a-0eb4-4b21-98f7-dcf9b718e592",
                "method": "GET"
            },
            {
                "id": "2a8f8cc4-3cf9-402f-a56c-94173f78ce72",
                "method": "GET"
            }
        ]
    }
}
```

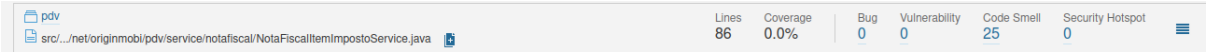
NotaFiscalItemImposto

NotaFiscalItemImposto Service - Sonar

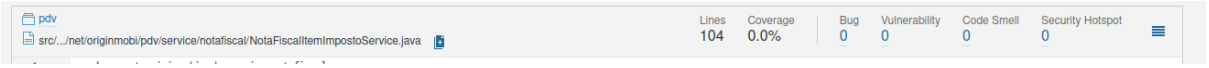
Sonar - NotaFiscalItemImpostoService

Data: 30 de Novembro de 2025
Autor do Teste: Mikael Schwind

Antes:



Depois:



O que foi feito:
- Método tinha 17 parâmetros, acima dos 9 como máximo recomendado. Criei um DTO para ter os parâmetros como atributos.
- Criei Exceptions específicas para a situação

CategoriaController

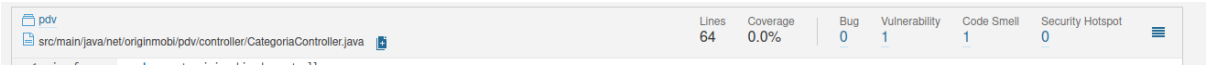
Categoria

CategoriaController - Sonar

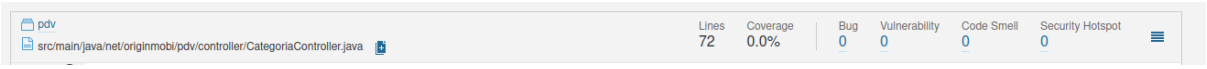
Sonar - CategoriaController

Data: 30 de Novembro de 2025
Autor do Teste: Mikael Schwind

Antes:



Depois:



O que foi feito:

- Parâmetro era uma entidade persistente no banco. Foi feito um DTO para o controller se comunicar.
- O sonar pediu pra trocar um System.out.println por um Logger