Relatório

Relatório – Entrega 2

Disciplina: Qualidade e Teste de Software
Professor(a): Vânia
Sistema Avaliado: PDV
Equipe: Elena Blanco, Giovanni Toledo, Henrique Fazollo, Larissa Galvão, Mikael Schwind

1. Objetivo da Entrega

O principal propósito desta etapa foi elevar o patamar de maturidade do sistema PDV, transcendendo a validação funcional básica para garantir robustez, manutenibilidade e confiabilidade. Os objetivos específicos traçados e executados nesta entrega foram:

- **Aumento e Melhoria da Qualidade dos Testes Unitários:** Expandir a cobertura de código focando não apenas em linhas, mas na cobertura de arestas (*branches*), e aplicar testes de mutação para assegurar que os casos de teste sejam capazes de revelar falhas reais na lógica de negócio.
- **Aplicação de Testes de Integração:** Validar a comunicação e o contrato de dados entre os componentes do sistema (especialmente entre Controllers e Services), garantindo a integridade das transações internas.
- **Execução de Testes de Sistema:** Simular o comportamento do usuário final através de automação na interface gráfica, assegurando que as jornadas críticas (como vendas e cadastros) funcionem corretamente de ponta a ponta.
- **Avaliação da Qualidade (ISO 25010):** Analisar o software sob a ótica de requisitos não funcionais essenciais, mensurando atributos como desempenho, usabilidade, segurança e manutenibilidade.
- **Inspeção de Código-Fonte Automatizada:** Utilizar ferramentas de análise estática para identificar e corrigir dívidas técnicas, vulnerabilidades, *bugs* e *code smells*, promovendo a padronização e a saúde do código a longo prazo.

2. Visão Geral do Sistema

Este é um sistema completo, feito para ajudar a gerenciar todo o dia a dia de uma loja ou comércio.

Ele serve para organizar e controlar todas as operações de um negócio, desde as vendas até o estoque e as finanças.

Principais Funcionalidades Identificadas:

- Módulo completo para registrar vendas, buscar produtos e gerenciar comandas.
- Caixa: Controle de caixa com funcionalidades de abertura, fechamento, lançamento de suprimentos, sangrias e transferências.

- Estoque: Gerenciamento de produtos, incluindo cadastro de categorias, fornecedores, grupos e ajustes de estoque.
- Financeiro: Controle de contas a pagar e a receber, com gerenciamento de títulos e tipos de pagamento.
- Cadastros Gerais: Gerenciamento de clientes (pessoas), empresas, bancos, máquinas de cartão e tributações.
- Segurança: Sistema de login com controle de acesso baseado em usuários e grupos de permissões.
- Nota Fiscal: Funcionalidades para emissão de Nota Fiscal Eletrônica (NFe), incluindo cadastro de destinatário e produtos.

3. Organização da Equipe

Integrante	Classes Trabalhadas	Testes/Ferramentas
Mikael	VendaService	Unitários (JaCoCo), Mutação(PIT)
	VendaController	Integração (Postman)
	Categoria Controller	Sonar
	NotaFiscalItemImpostoService	Sonar
	Gestão de Vendas	Funcionais (Selenium)
Larissa	ProdutoService	Unitários (JaCoCo), Mutação(PIT)
	NotaFiscalItemService	Unitários (JaCoCo)
	FornecedorController	Integração (Postman)
	Gestão de Fornecedores	Funcionais (Selenium)
	CartãoLancamentoService	Sonar
Elena	PessoaService, PagarService	Unitários (JaCoCo), Mutação(PIT)
	Gestão de Usuários	Integração (Postman), Funcionais (Selenium)
	CaixaService	Sonar
Henrique	RecebimentoService	Unitários (JaCoCo), Mutação (PIT)
	RecebimentoController	Integração (Postman)

	UsuarioService	Unitários (JaCoCo), Mutação (PIT)
	AjusteProdutoService	Sonar
	ProdutoService	Funcionais (Selenium)
Giovanni	NotaFiscalService	Unitários (JaCoCo), Mutação(PIT), Integração (Postman), Funcionais (Selenium)

4. Documento de Casos de Teste

Os casos de teste desenvolvidos para esta entrega foram organizados em um **documento único**, contendo **abas separadas por classe ou funcionalidade testada**.

Em cada aba, estão descritos:

- A **classificação do teste** (unitário, integração ou sistema);
- Os **casos de teste** referentes àquela classe ou funcionalidade;
- As **evidências de execução**, incluindo:
    - Respostas em formato JSON dos testes de integração no Postman;
    - Prints dos relatórios de cobertura do JaCoCo;
    - Prints dos relatórios de mutação gerados pelo PIT;

Essa organização permitiu centralizar os testes de forma rastreável, facilitando a validação dos resultados e a inspeção do processo de qualidade aplicado ao sistema.

5. Teste Estrutural – JaCoCo

A cobertura estrutural do código foi avaliada minuciosamente por meio da ferramenta **JaCoCo**, utilizando como métrica principal o critério de **todas as arestas** (*branch coverage*).

Essa abordagem de teste de caixa-branca foi fundamental para garantir que não apenas as linhas de instrução fossem executadas, mas que todos os caminhos lógicos de decisão (fluxos de verdadeiro/falso em condicionais) dentro dos métodos fossem devidamente percorridos e validados.

Embora os relatórios analíticos detalhados por classe estejam disponíveis no

documento de testes em anexo, esta seção apresenta o resultado consolidado da cobertura. O foco foi assegurar que a lógica de negócio crítica, residente nos Services e Controllers, atingisse níveis seguros de confiabilidade, minimizando o risco de falhas em cenários alternativos.

pdv

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
net.originmobi.pdv.controller		2%		0%	222	251	687	716	165	194	0	28
net.originmobi.pdv.model		29%		0%	548	786	957	1,397	545	783	0	49
net.originmobi.pdv.service		47%		45%	194	328	517	972	115	191	0	38
net.originmobi.pdv.xml.nfe		0%		0%	32	33	487	488	16	17	3	4
net.originmobi.pdv.service.cartao		1%		0%	17	19	90	92	7	9	0	2
net.originmobi.pdv.service.notafiscal		70%		86%	18	66	86	243	10	33	0	8
net.originmobi.pdv.model.cartao		2%		n/a	45	47	79	83	45	47	0	2
net.originmobi.pdv.dto		0%		n/a	42	42	63	63	42	42	2	2
net.originmobi.pdv.relatorios		3%		n/a	5	7	40	42	5	7	1	3
net.originmobi.pdv.filter		21%		n/a	33	42	45	57	33	42	9	12
net.originmobi.pdv.security		84%		0%	9	13	20	84	6	10	1	3
net.originmobi.pdv.enumerado		70%		n/a	2	7	4	14	2	7	2	7
net.originmobi.pdv.utilitarios		47%		n/a	5	8	17	29	5	8	2	3
net.originmobi.pdv.singleton		0%		0%	4	4	7	7	3	3	1	1
net.originmobi.pdv		41%		n/a	2	4	4	6	2	4	0	1
net.originmobi.pdv.exceptions		0%		n/a	2	2	4	4	2	2	1	1
net.originmobi.pdv.enumerado.cartao		95%		n/a	1	4	1	8	1	4	0	2
net.originmobi.pdv.enumerado.ajuste		91%		n/a	1	3	1	6	1	3	0	1
net.originmobi.pdv.enumerado.caixa		100%		n/a	0	3	0	6	0	3	0	3
net.originmobi.pdv.enumerado.produto		100%		n/a	0	4	0	8	0	4	0	4
net.originmobi.pdv.enumerado.notafiscal		100%		n/a	0	1	0	2	0	1	0	1
Total	11,729 of 16,830	30%	339 of 520	34%	1,182	1,674	3,109	4,327	1,005	1,414	22	175

6. Teste Baseado em Defeitos – Mutação

Para validar a real eficácia dos testes unitários, utilizou-se a técnica de testes de mutação com a ferramenta **PIT**. O objetivo foi verificar se a suíte de testes é robusta o suficiente para "matar" mutantes (falhas artificiais inseridas no código).

- **Escore Obtido:** A equipe alcançou um índice de mutação de aproximadamente **80%**.
- **Escopo da Análise:** Vale ressaltar que essa métrica foi focada especificamente nas **classes alteradas ou criadas pela equipe** nesta entrega, e não sobre todo o código legado do projeto. Esse resultado demonstra que as novas implementações possuem uma rede de segurança de testes altamente sensível a falhas.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
155	24%	993/4221	18%	325/1799	82%	325/394

Breakdown by Package

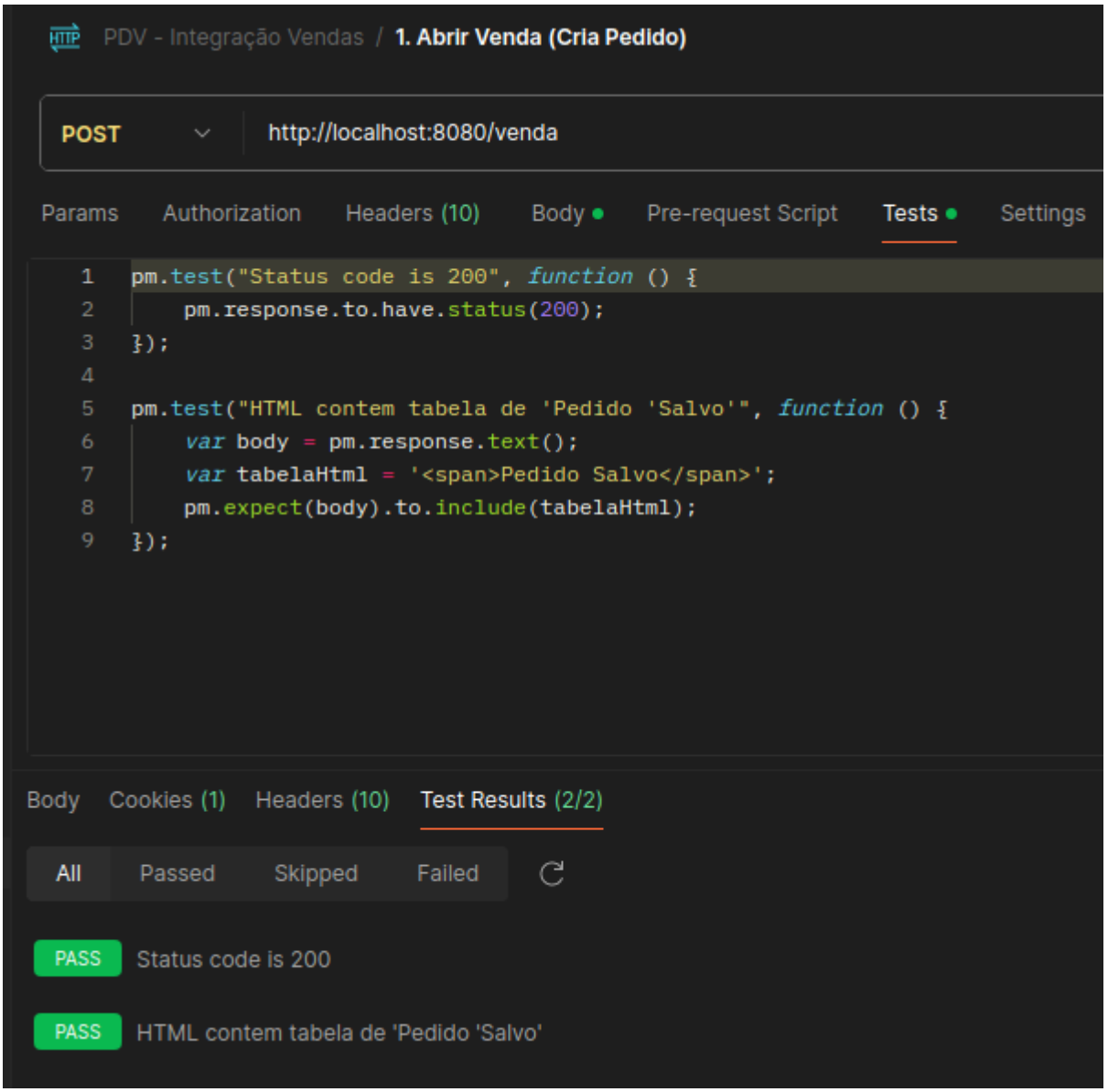
Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
net.originmobi.pdv	1	0%	0/6	0%	0/3	0%	0/0
net.originmobi.pdv.controller	28	0%	0/716	0%	0/274	0%	0/0
net.originmobi.pdv.dto	2	0%	0/63	0%	0/20	0%	0/0
net.originmobi.pdv.enumerado.ajuste	1	0%	0/6	0%	0/1	0%	0/0
net.originmobi.pdv.enumerado.cartao	1	0%	0/6	0%	0/1	0%	0/0
net.originmobi.pdv.filter	12	21%	12/57	19%	3/16	100%	3/3
net.originmobi.pdv.model	49	29%	402/1397	21%	78/364	84%	78/93
net.originmobi.pdv.model.cartao	2	0%	0/83	0%	0/25	0%	0/0
net.originmobi.pdv.relatorios	3	0%	0/42	0%	0/14	0%	0/0
net.originmobi.pdv.security	2	0%	0/21	0%	0/5	0%	0/0
net.originmobi.pdv.service	38	43%	415/972	35%	181/520	83%	181/217
net.originmobi.pdv.service.cartao	2	0%	0/92	0%	0/25	0%	0/0
net.originmobi.pdv.service.notafiscal	8	62%	152/244	48%	61/126	79%	61/77
net.originmobi.pdv.singleton	1	0%	0/7	0%	0/3	0%	0/0
net.originmobi.pdv.utilitarios	2	48%	12/25	25%	2/8	50%	2/4
net.originmobi.pdv.xml.nfe	3	0%	0/484	0%	0/394	0%	0/0

Report generated by PIT 1.7.5

7. Testes de Integração – Postman

Foram testados os endpoints dos Controllers de acordo com o Service que cada um dos alunos trabalhou nos testes unitários e de estrutura.

- Sendo um MVC utilizando Thymeleaf, os retornos das API's eram com status 200 (OK) e com um HTML na resposta. Dessa forma, os testes focaram em verificar status e conteúdos do html



8. Testes de Sistema – Selenium

Os testes de sistema simularam a interação de um usuário real (operador de caixa/gerente), seguindo os Requisitos Funcionais mapeados, com a interface gráfica da aplicação, garantindo que as principais jornadas de venda funcionem conforme esperado.

A automação focou na validação "ponta a ponta" dos fluxos mais críticos para o negócio, cobrindo:

- **Frente de Caixa (RF-V.01 a RF-V.04):** Validação completa do ciclo de vendas, desde a abertura da venda e adição de itens (por código ou busca) até a finalização com cálculo de troco e subtotal.
- **Integração com Estoque (RF-E.01):** Verificação da baixa automática na quantidade de produtos imediatamente após a concretização da venda.

- **Segurança e Acesso (RF-S.02):** Testes de tentativa de acesso a módulos administrativos (como Financeiro) por usuários com perfil de Vendedor, garantindo o bloqueio adequado.
- **Cadastros Base (RF-C.02 e RF-C.04):** Garantia de que novos produtos e fornecedores são registrados corretamente e refletidos nas operações de venda.

9. Requisito Não Funcional

A validação do sistema não se limitou apenas ao "o que" o sistema faz, mas também ao "como" ele se comporta. Foram avaliados atributos de qualidade essenciais para a operação em um ambiente comercial real:

- **Desempenho:** O foco foi garantir que o tempo de resposta das operações críticas (como lançar um item ou finalizar uma venda) seja mínimo. Em um PDV, a agilidade é vital para evitar filas e demora no atendimento.
- **Usabilidade:** Foi analisada a eficiência da interface, verificando se as tarefas comuns exigem uma quantidade excessiva de cliques ou navegação complexa. O objetivo é assegurar que o operador consiga utilizar o sistema de forma intuitiva e rápida.
- **Segurança:** A integridade do sistema foi testada através da verificação rigorosa de perfis de acesso. Garantiu-se que apenas usuários com as devidas permissões (ex: Gerentes) possam realizar operações sensíveis, como sangrias de caixa ou visualização de relatórios financeiros.

10. Avaliação da Qualidade – ISO 25010

Para garantir que o sistema atenda aos padrões de mercado, a avaliação da qualidade seguiu as diretrizes da norma **ISO/IEC 25010** (veja ISO 25010). Foram selecionados atributos críticos para o contexto de um Ponto de Venda (PDV), definindo-se métricas quantitativas e escalas de aceitação rigorosas. A tabela a seguir detalha os critérios utilizados e os resultados esperados para cada característica de qualidade:

Atributo	Métrica	Escala	Justificativa
Funcionalidade	CIRF = (Requisitos Implementados / Requisitos Especificados) × 100	≥95% ok | 90–94% médio | <90% ruim	Sistema possui funções críticas como vendas, caixa e estoque.

Desempenho	TMR = (Soma dos tempos de resposta) / (Quantidade de requisições)	≤300 ok | 301–800 médio | >800 ruim	O PDV exige respostas rápidas no atendimento.
Compatibilidade	PFN = (Funcionalidades compatíveis / Funcionalidades testadas) × 100	100% ok | ≥95% médio | <95% ruim	Sistema web usado em diferentes navegadores.
Usabilidade	NA = (Total de cliques registrados) / (Total de tarefas analisadas)	≤5 ok | 6–8 médio | >8 ruim	Operações no caixa precisam ser rápidas.
Confiabilidade	TFOC = (Operações com falha / Total de operações realizadas) × 100	≤0,5% ok | 0,6–2% médio | >2% ruim	Falhas impactam diretamente as vendas e o caixa.
Segurança	TCAC = (Funcionalidades com permissão correta / Funcionalidades analisadas) × 100	≥99% ok | 95–98% médio | <95% ruim	Controle de acesso por perfil (gerente, atendente).
Manutenibilidade	CCM = (Soma das complexidades ciclomáticas	≤10 ok | 11–20 médio | >20 ruim	Alta complexidade dificulta manutenção.

	das classes) / (Número de classes analisadas)		
Portabilidade	Classificação baseada na quantidade de ajustes necessários para rodar em outro ambiente (ex.: Windows → Linux).	0 ajustes ok | 1–2 médio | muda código ruim	Sistema roda em diferentes ambientes com baixo esforço.

11. Inspeção de Código-Fonte – Sonar

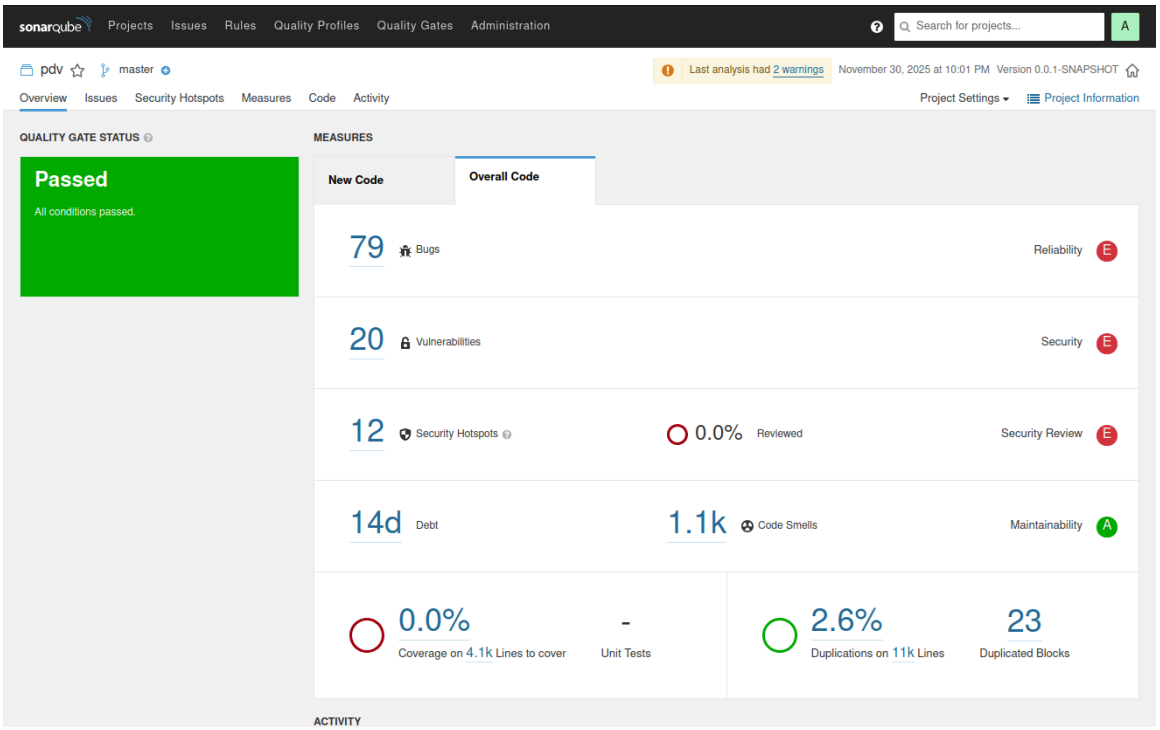
A análise estática do código foi realizada utilizando a ferramenta SonarQube, com o objetivo de identificar dívidas técnicas, falhas de lógica e possíveis brechas de segurança. A inspeção automatizada permitiu um diagnóstico preciso da saúde do código, orientando as refatorações necessárias.

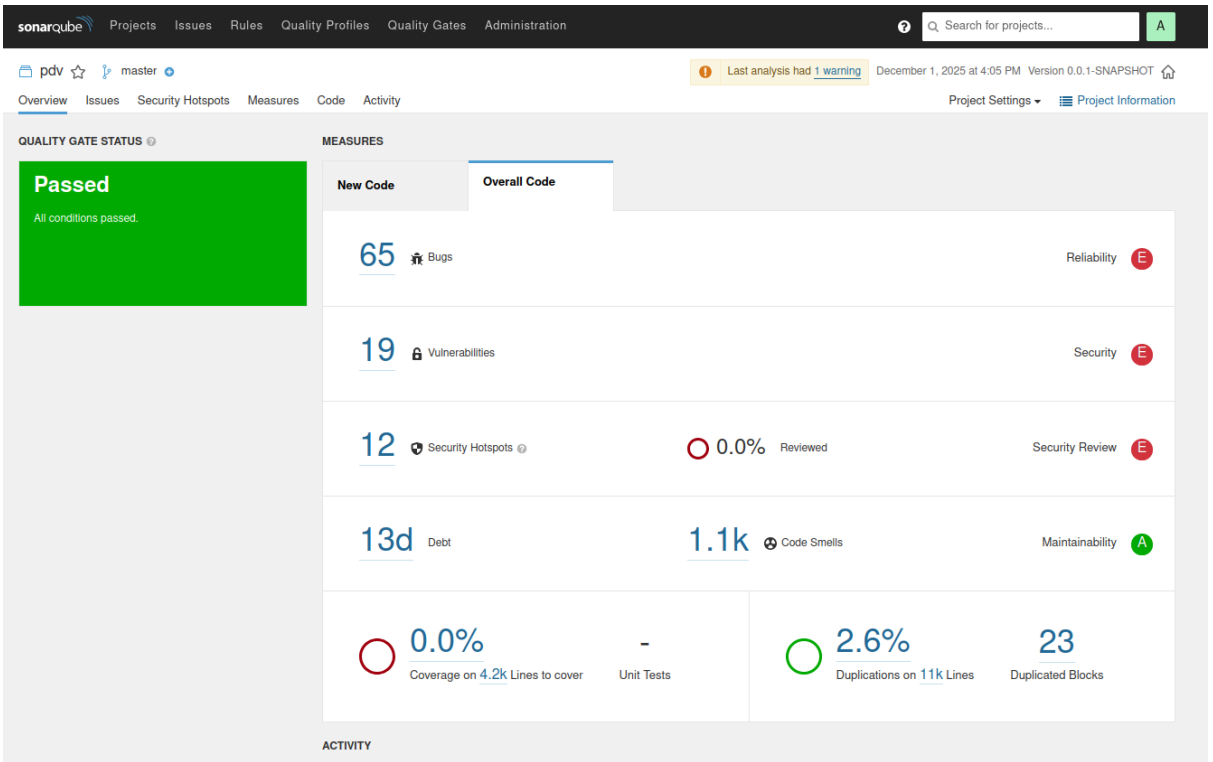
Os resultados obtidos na análise foram:

- **Quantidade de Bugs:**
  - Foram identificados defeitos que poderiam alterar o comportamento esperado do sistema ou gerar erros em tempo de execução.
- **Code Smells (Mau Cheiro de Código):**
  - A maior parte dos apontamentos referia-se à manutenibilidade, incluindo violações de convenção de nomenclatura de variáveis, métodos com excesso de parâmetros e falta de tratamento específico de exceções.
- **Vulnerabilidades:**
  - Foram mapeados pontos de atenção relacionados à segurança da aplicação.

Essa inspeção serviu como base para a etapa de correções, onde priorizamos a resolução dos *code smells* para adequar o projeto aos padrões de qualidade exigidos, melhorando a

legibilidade e facilitando futuras manutenções





12. Correções Aplicadas

As correções realizadas focaram em refatorações estratégicas que, embora não alterassem o comportamento funcional do sistema, elevaram significativamente a qualidade estrutural do código. As principais melhorias incluíram:

- **Criação de Exceções Específicas:** Substituição de exceções genéricas por tipos personalizados, o que facilitou o tratamento de erros e tornou o diagnóstico de falhas mais preciso.
- **Adoção de DTOs:** Métodos que recebiam múltiplos parâmetros isolados foram refatorados para utilizar *Data Transfer Objects* (DTOs), aumentando a organização e a flexibilidade das assinaturas.
- **Padronização de Nomenclatura:** Ajuste nos nomes de variáveis e métodos para aderir estritamente às convenções apontadas pelo Sonar, melhorando a legibilidade e a manutenção do código.
-

As alterações em detalhes se encontram no relatório de testes, que pode ser acessado aqui.

13. Conclusão

A execução da segunda entrega consolidou a maturidade do sistema PDV através de uma estratégia abrangente de testes e análise estática. Os principais ganhos foram:

1. Elevação da Qualidade

A combinação de testes estruturais (JaCoCo) e de mutação (PIT), atingindo escores próximos a **80%**, validou profundamente a lógica de negócio. A conformidade com a **ISO 25010** garantiu que o software atende aos requisitos não funcionais essenciais.

2. Confiabilidade e Segurança

Testes de integração (Postman) e de sistema (Selenium) asseguraram o funcionamento correto de jornadas críticas (como vendas e fechamento de caixa) e validaram os controles de acesso, garantindo a integridade e segurança dos dados.

3. Manutenibilidade e Evolução

A análise automatizada via **Sonar** e as refatorações realizadas (remoção de *code smells*, padronização e tratamentos de exceção) reduziram a dívida técnica, resultando em um código mais limpo, legível e resiliente.

Em suma: O PDV evoluiu de um sistema funcional para um software auditado, com menor densidade de defeitos e garantias rastreáveis de funcionamento e facilidade de manutenção.

Requisitos funcionais

📄 Requisitos Funcionais (RFs) do Sistema PDV/ERP

I. Módulo de Cadastro (Produtos, Clientes, Fornecedores)

Este módulo trata da manutenção dos dados mestres do sistema.

RF-C.01: Cadastro de Cliente

O sistema deve permitir o registro de um novo cliente, exigindo obrigatoriamente Nome, e permitindo o registro de CPF/CNPJ, endereço e telefone.

RF-C.02: Cadastro de Produto

O sistema deve permitir o cadastro de um novo produto, incluindo Descrição, Preço de Custo, Preço de Venda, Unidade de Medida e vínculo obrigatório a uma Categoria.

RF-C.03: Validação de Produto

O sistema deve garantir que o Preço de Venda de um produto seja sempre maior que o Preço de Custo.

RF-C.04: Cadastro de Fornecedor

O sistema deve permitir o cadastro de fornecedores com razão social, CNPJ e telefone de contato.

RF-C.05: Consulta e Manutenção

O sistema deve permitir a consulta, edição e exclusão de Clientes, Produtos e Fornecedores cadastrados, respeitando a integridade referencial (ex: não excluir produto em estoque).

II. Módulo de Vendas (PDV e Comandas)

Este módulo trata da interface de frente de caixa e gerenciamento de pedidos.

RF-V.01: Início de Venda

O sistema deve permitir a abertura de uma nova venda ou comanda, vinculada a um cliente (opcional) e a um usuário operador.

RF-V.02: Adição de Itens

O sistema deve permitir a adição de produtos à venda/comanda, seja por código de barras ou busca por descrição, exibindo o subtotal em tempo real.

RF-V.03: Aplicação de Descontos

O sistema deve permitir a aplicação de desconto percentual ou em valor absoluto no item ou no total da venda.

RF-V.04: Finalização e Pagamento

O sistema deve exibir a tela de pagamento, permitindo o registro do valor total, o cálculo de troco (para pagamento em dinheiro) e o registro do pagamento em múltiplas formas.

III. Módulo de Controle de Estoque

Este módulo trata da gestão e movimentação do inventário.

RF-E.01: Baixa Automática

Ao finalizar uma venda (RF-V.04), o sistema deve automaticamente dar baixa na quantidade de produtos vendidos no estoque.

RF-E.02: Movimentação Manual (Ajuste)

O sistema deve fornecer uma funcionalidade para registrar entradas ou saídas manuais (ajustes) de estoque por produto, exigindo a justificativa da operação.

RF-E.03: Alerta de Estoque Mínimo

O sistema deve exibir um alerta ou notificação quando a quantidade de um produto cair abaixo do limite de estoque mínimo configurado em seu cadastro.

IV. Módulo Financeiro

Este módulo abrange o fluxo de caixa e o controle de contas.

RF-F.01: Registro de Fluxo de Caixa

Todas as transações de venda e pagamentos de despesas devem ser registradas automaticamente no fluxo de caixa com data, tipo (Entrada/Saída) e valor.

RF-F.02: Contas a Pagar (Despesas)

O sistema deve permitir o lançamento de contas a pagar (ex: aluguel, fornecedor), com data de vencimento e status (pendente/pago).

RF-F.03: Contas a Receber (Crédito)

O sistema deve gerar automaticamente contas a receber no caso de vendas parceladas ou a crédito, com a data de vencimento de cada parcela.

RF-F.04: Formas de Pagamento

O sistema deve permitir que um usuário autorizado cadastre novas formas de pagamento, definindo o tipo (dinheiro, crédito, débito, voucher, etc.).

V. Módulo de Usuários

Este módulo trata da segurança e da extração de informações.

RF-S.01: Permissões por Grupo

O sistema deve permitir a criação e edição de Grupos de Usuários (ex: Gerente, Vendedor), onde cada grupo possui um conjunto específico de permissões de acesso às funcionalidades (CRUD).

RF-S.02: Acesso Restrito

O sistema deve verificar o grupo do usuário logado e bloquear o acesso a páginas ou ações para as quais ele não possui permissão (ex: apenas o Gerente pode acessar o módulo Financeiro).

VI. Módulo de Nota Fiscal (NF-e)

Este módulo trata da emissão, criação e exibição de notas fiscais eletrônicas.

RF-NF.01: Criação de Nota Fiscal

O sistema deve permitir a criação de uma nova Nota Fiscal, exigindo obrigatoriamente a identificação da empresa emitente, do usuário responsável, da lista de itens e os respectivos valores financeiros (subtotal, descontos e total).

RF-NF.02: Emissão da Nota Fiscal

O sistema deve permitir a emissão de uma Nota Fiscal previamente criada, alterando seu status de "Criada" para "Emitida", registrando automaticamente a data e hora da emissão e gerando um número sequencial interno para a nota.

RF-NF.03: Listagem de Notas Fiscais

O sistema deve permitir a listagem de todas as Notas Fiscais existentes, possibilitando a filtragem dos registros por empresa, usuário responsável, status (Criada ou Emitida) e por período (data inicial e final).

RF-NF.04: Consulta de Nota Fiscal

O sistema deve permitir a consulta de uma Nota Fiscal específica através de seu identificador interno, exibindo em detalhes a empresa, usuário, itens, status, totais financeiros e as datas de criação e emissão.

RF-NF.05: Cancelamento de Nota Fiscal

O sistema deve permitir o cancelamento de uma Nota Fiscal somente se ela estiver com o status "Criada" ou "Emitida" (desde que não bloqueada internamente), atualizando o status para "Cancelada" e registrando a data, hora e o usuário responsável pela operação.

ISO 25010

1. Funcionalidade (Adequação Funcional)

Métrica:
Cobertura de Implementação de Requisitos Funcionais (CIRF)

CIRF = (Requisitos Implementados / Requisitos Especificados) × 100

Escala:

- **≥ 95%** — Aceitável
- **90–94%** — Tolerável
- **< 90%** — Inaceitável

Justificativa:
O ERP possui muitos requisitos funcionais críticos (vendas, caixa, estoque, pagamentos). A métrica CIRF permite medir objetivamente o quanto do comportamento esperado realmente foi implementado e validado.

---

2. Eficiência de Desempenho

Métrica:
Tempo Médio de Resposta (TMR) por endpoint REST (em ms)

TMR = (Soma dos tempos de resposta) / (Quantidade de requisições)

Escala:

- **≤ 300 ms** — Aceitável
- **301–800 ms** — Tolerável
- **> 800 ms** — Inaceitável

Justificativa:
Um PDV depende de operações rápidas, especialmente em vendas. O TMR é uma métrica comum em sistemas web e permite análise direta de desempenho.

---

3. Compatibilidade

Métrica:
Percentual de Funcionalidades Compatíveis Entre Navegadores (PFN)

PFN = (Funcionalidades compatíveis / Funcionalidades testadas) × 100

Escala:

- **100%** — Aceitável
- **≥ 95%** — Tolerável
- **< 95%** — Inaceitável

Justificativa:
O sistema é web (Thymeleaf + Spring), rodado em estabelecimentos comerciais. Compatibilidade evita falhas por ambiente de uso.

---

4. Usabilidade

Métrica:
Número de Ações Necessárias (cliques) para realizar operações críticas
Ex.: abrir comanda, registrar venda, fechar caixa.

NA = (Total de cliques registrados) / (Total de tarefas analisadas)

Escala:

- **≤ 5 cliques** — Aceitável
- **6–8 cliques** — Tolerável
- **> 8 cliques** — Inaceitável

Justificativa:
PDV exige agilidade; tarefas com muitos passos prejudicam atendimento.

---

5. Confiabilidade

Métrica:
Taxa de Falhas em Operações Críticas (TFOC)

TFOC = (Operações com falha / Total de operações realizadas) × 100

Escala:

- **≤ 0,5%** — Aceitável
- **0,6–2%** — Tolerável
- **> 2%** — Inaceitável

Justificativa:
Falhas em vendas, estoque ou fluxo de caixa têm impacto direto no negócio e precisam ser minimizadas.

---

6. Segurança

Métrica:
Taxa de Controle de Acesso Correto (TCAC)

TCAC = (Funcionalidades com permissão correta / Funcionalidades analisadas) × 100

Escala:

- **≥ 99%** — Aceitável
- **95–98%** — Tolerável
- **< 95%** — Inaceitável

Justificativa:
O sistema possui controle de permissões por grupos (gerente, atendente etc.). É essencial garantir que usuários não acessem funções que não deveriam.

---

7. Manutenibilidade

Métrica:
Complexidade Ciclomática Média (CCM) das classes

CCM = (Soma das complexidades ciclomáticas das classes) / (Número de classes analisadas)

Escala:

- **≤ 10** — Aceitável
- **11–20** — Tolerável
- **> 20** — Inaceitável

Justificativa:
O sistema tem muitas regras de negócio. A complexidade ciclomática é uma medida objetiva que indica o custo de manutenção.

---

8. Portabilidade

Métrica:
Esforço de Implantação (EI)
Classificação baseada na quantidade de ajustes necessários para rodar em outro ambiente (ex.: Windows → Linux).

Escala:

- **0 ajustes** (apenas configurar `.properties`) — Aceitável
- **1–2 ajustes menores** (ex.: caminhos de arquivos) — Tolerável
- **Exige mudança de código** — Inaceitável

Justificativa:
O sistema roda com Spring Boot, MySQL e Flyway. A implantação deve ser simples em qualquer servidor compatível.