APRESENTAÇÃO 2 - PDV

Elena Blanco, Giovanni Toledo, Henrique Fazollo,
Larissa Galvão, Mikael Schwind

OBJETIVOS

Melhorar testes unitários

Criar testes de integração (Postman)

Criar testes de sistema (Selenium)

Aplicar teste estrutural e mutação

Avaliar qualidade com ISO 25010

Análise de código com Sonar

DIVISÃO DAS TAREFAS

Cada pessoa ficou responsável por:

- 1 Classe Service e Controller
- Criar os casos de testes e implementar
- Documentar os resultados
- Correções do Sonar
- Obs: Os casos de testes e as evidências completas estão organizados em documento específico por classe e funcionalidade testada.

TESTES UNITÁRIOS

- Criação e aumento dos testes unitários
- Isolamento das dependências

- Uso de:
  - JUnit
  - Mockito
  - JaCoCo (cobertura estrutural)
  - PiTest

TESTES UNITÁRIOS

Exemplo de metodo testado:

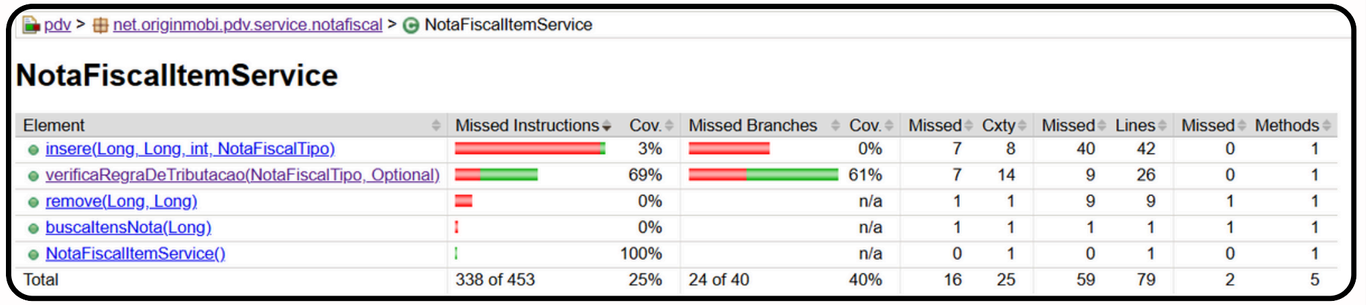
ProdutoService.movimentaEstoque()

Caso de Teste: PS-006 - movimentaEstoque - Saldo Insuficiente

```java
//Simular o saldo insuficiente para o produto vendido
@Test
void movimentaEstoque_deveLancarExcecaoPorFaltaDeEstoque() {
    Long codVendaTeste = 51L;
    Long codProdutoTeste = 11L;
    int qtdVendida = 10;
    int qtdEstoque = 5;

    List<Object[]> produtosVendidos = Collections.singletonList(
        new Object[]{codProdutoTeste, qtdVendida}
    );
    when(vendaProdutos.buscaQtdProduto(codVendaTeste)).thenReturn(produtosVendidos);

    Produto produto = new Produto();
    produto.setControla_estoque(ProdutoControleEstoque.SIM);
    when(produtos.findByCodigoIn(codProdutoTeste)).thenReturn(produto);

    when(produtos.saldoEstoque(codProdutoTeste)).thenReturn(qtdEstoque);

    assertThrows(expectedType: RuntimeException.class, () -> {
        service.movimentaEstoque(codVendaTeste, EntradaSaida.SAIDA);
    });

    verify(produtos, never()).movimentaEstoque(any(), any(), anyInt(), any(), any());
}
```

TESTE ESTRUTURAL COM JACOCO

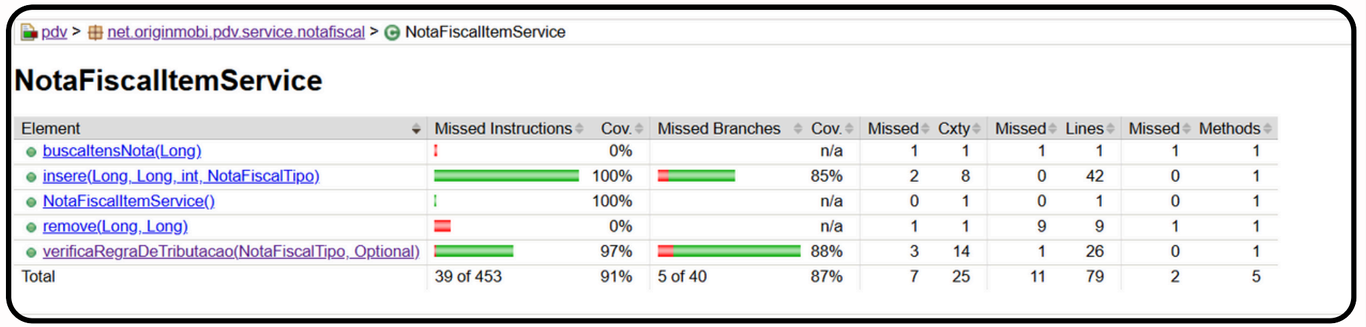
1. NotaFiscalItemService.insere()
2. Critério: todas as arestas
3. Antes: 3%
4. Mínima exigida: 80%
5. Resultado obtido: 100%



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



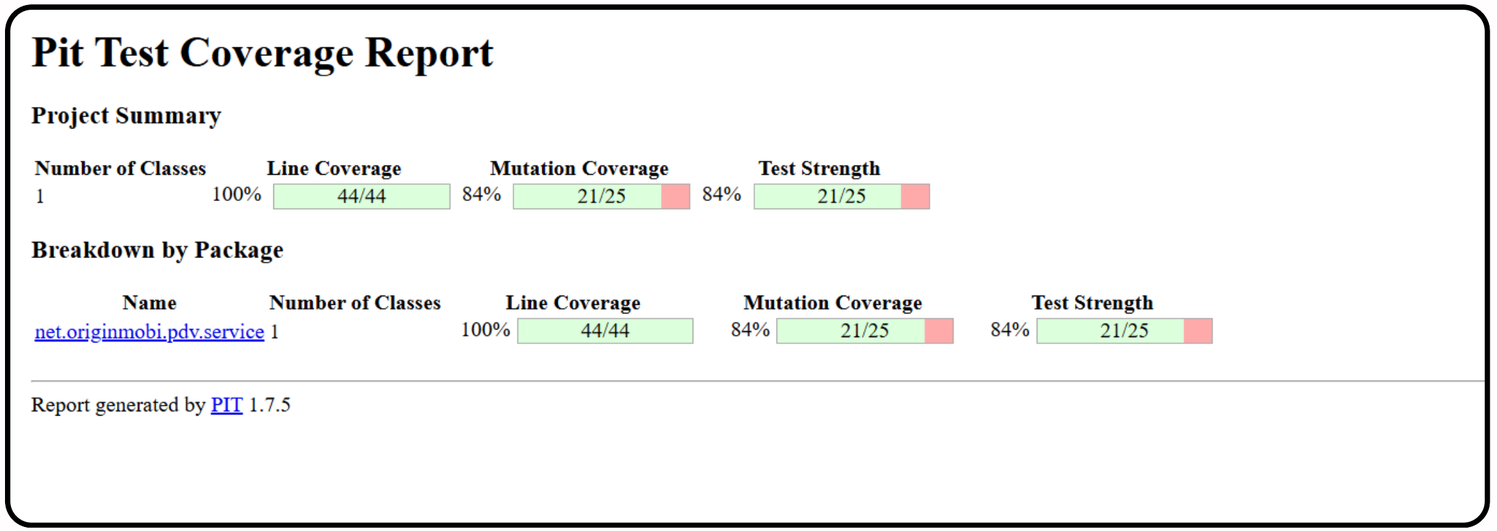
pdv > net.originmobi.pdv.service.notafiscal > NotaFiscalItemService

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

TESTE DE MUTAÇÃO

1. Objetivo: medir a qualidade da suíte de testes.
2. Serviço Avaliado: ProdutoService
3. Escore mínimo: 80%
4. Resultado obtido: 84%



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

TESTES DE INTEGRAÇÃO

- Testes em endpoints
- Verificação de:
  - Status HTTP
  - Retorno dos dados
- Fluxos testados: fornecedores, usuários, venda, etc
- Dificuldades:
  - Retorno sempre em html entao sempre dava 200;
  - Token da sessão as vezes se perdia;

```
{
    "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
        }
    ]
}
```

```
{
    "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
        }
    },
```

TESTES DE SISTEMA

Objetivo: Validação do Sistema Completo na Perspectiva do Usuário

- Requisitos de Sistema Baseado no Readme e PDV
- Fizemos os casos de teste
- Implementação com Selenium

TESTES DE SISTEMA - REQUISITOS

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).

TESTES DE SISTEMA - CASOS + SELENIUM

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.

```java
@Test  Elena-Blanco
@DisplayName("RF-S.01 - Deve criar um usuário e atribuir um grupo de permissão")
public void testCriarUsuarioEAtribuirGrupo() throws InterruptedException {
    driver.get(BASE_URL + "/usuario/form");

    try {
        Select selectPessoa = new Select(driver.findElement(By.id("pessoa")));
        selectPessoa.selectByIndex(1);

        driver.findElement(By.id("userName")).sendKeys( ...keysToSend: "vendedor02");
        driver.findElement(By.id("password")).sendKeys( ...keysToSend: "123456");

        clicar(By.cssSelector("button[type='submit']"));

        wait.until(ExpectedConditions.urlContains( fraction: "/usuario"));

        driver.get(BASE_URL + "/usuario");

        WebElement botaoEditar = driver.findElement(
                By.xpath( xpathExpression: "//td[contains(text(),'vendedor02')]/..//a[contains(@href, 'editar')]")
        );
        clicar(botaoEditar);

        Select selectGrupo = new Select(driver.findElement(By.id("grupo")));
        selectGrupo.selectByVisibleText("Vendedor");

        clicar(By.id("btn-add-grupo"));

        Thread.sleep( millis: 1000);

        WebElement tabelaGrupos = wait.until(
                ExpectedConditions.visibilityOfElementLocated(By.id("tabela-grupos-usuario"))
        );
        assertTrue(tabelaGrupos.getText().contains("Vendedor"));
        avaliarUsabilidade();

    } catch (Exception e) {
        System.out.println("Erro ao testar grupo: " + e.getMessage());
    }
}
```

ISO 25010

1. Funcionalidade

Métrica: Cobertura de Implementação de Requisitos Funcionais (CIRF)

Escala: ≥95% ok | 90–94% médio | <90% ruim

Justificativa: Funções críticas

2. Desempenho

Métrica: Tempo Médio de Resposta

Escala: ≤300 ok | 301–800 médio | >800 ruim

Justificativa: O PDV exige respostas rápidas no atendimento.

3. Compatibilidade

Métrica: Funcionalidades Compatíveis Entre Navegadores (PFN)

Escala: 100% ok | ≥95% médio | <95% ruim

Justificativa: Diferentes navegadores.

4. Usabilidade

Métrica: Nº médio de cliques

Escala: ≤5 ok | 6–8 médio | >8 ruim

Justificativa: Operações no caixa precisam ser rápidas.

ISO 25010

5. Confiabilidade

Métrica: Taxa de Falhas em Operações Críticas (TFOC)

Escala: ≤0,5% ok | 0,6–2% médio | >2% ruim

Justificativa: Falhas impactam o caixa.

6. Segurança

Métrica: Taxa de Controle de Acesso Correto (TCAC)

Escala: ≥99% ok | 95–98% médio | <95% ruim

Justificativa: Controle de acesso por perfil.

7. Manutenibilidade

Métrica: Complexidade Ciclomática Média (CCM)

Escala: ≤10 ok | 11–20 médio | >20 ruim

Justificativa: Manutenções precisar ser fáceis.

8. Portabilidade

Métrica: Esforço de Implantação

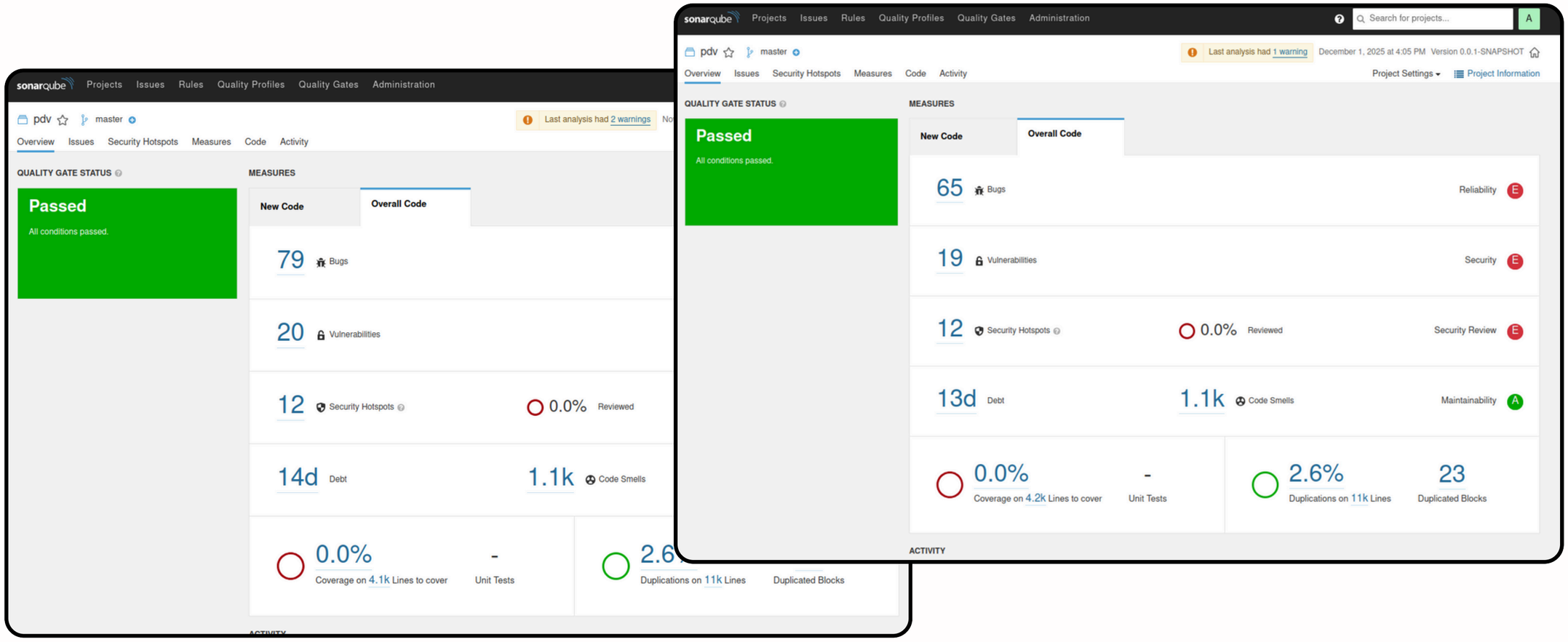
Escala: 0 ajustes ok | 1–2 médio | muda código ruim

Justificativa: Sistema roda em diferentes lojas.

INSPEÇÃO DE CÓDIGO COM SONAR

- Quantidade de bugs resolvidos: 15
- Code smells: 77
- Vulnerabilidades 1
- Dificuldade: rodar o sonar compativel com o java 8

INSPEÇÃO DE CÓDIGO COM SONAR



CORREÇÕES APLICADAS

As correções envolveram refatorações que não influenciaram em funcionamento, mas melhoraram a estrutura. Algumas foram:

- Criação de Exceções específicas;
- Múltiplos parâmetros em um método transformados em DTO;
- Nomes de variáveis para ficar de acordo com o padrão do Sonar;

As alterações em detalhes se encontram no relatório de testes.

DESAFIOS

- Compatibilidade com Java8;
- Aumentar a cobertura do teste de mutação;
- Testes Funcionais atrapalhando;
- Rodar o Sonar;
- Selenium enfrentava muitos erros de perda de conexão com o web-driver;

OBRIGADO!