TRABALHO PRÁTICO

Sistema PDV

Apresentando sistema:

- ERP Sistema integrado de gestão empresarial
- Spring Boot
- Principais funcionalidades
 - Cadastro produtos/clientes/fornecedor
 - Controle de estoque

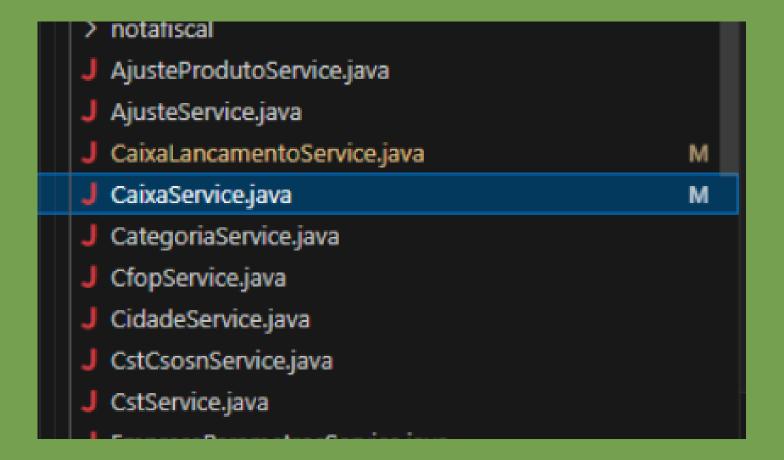




Relatório de inspeção

- Ferramenta utilizada: Sonarlint
- Classe escolhida: CaixaLancamentoService
- e CaixaService
- Principais Desafios: Presença de um super metodo
- Solução: Foi necessário a quebra em métodos menores

J AjusteProdutoService.java	Ø 18	gservice
	19	public class CaixaLancamen
J AjusteService.java	20	
J CaixaLancamentoService.java 5	21	@Autowired
J CaixaService.java	22	private CaixaLancament
J CategoriaService.java	23	
J CfopService.java	24	@Autowired
J CidadeService.java	25	UsuarioService usuario
	26	
J CstCsosnService.java	27	private Timestamp data
J CstService.java	28	
J EmpresaParametrosService.java	29	public CaixaLancamento
J EmpresaService.java	30	}
J EnderecoService.java	31	
J EstadoService.java	32	@Transactional(readOnl
	33	public String lancamen
J FornecedorService.java	24	dotallanattual co



Teste unitários

- Ferramenta utilizada: Junit5 e Mockito
- Classe escolhida: CaixaLancamentoService e CaixaService

```
@Test
void testLancamento semCaixaAberto lancaExcecao() {
   CaixaLancamento lancamento = new CaixaLancamento();
   lancamento.setCaixa(null);
   RuntimeException thrown = assertThrows(
        RuntimeException.class,
        () -> caixaLancamentoService.lancamento(lancamento),
        "Esperava lançar RuntimeException"
    assertTrue(thrown.getMessage().contains("Nenhum caixa aberto"));
@Test
void testLancamento saldoInsuficiente lancaExcecao() {
   Caixa caixa = new Caixa();
   caixa.setValor total(100.0);
   CaixaLancamento lancamento = new CaixaLancamento();
   lancamento.setCaixa(caixa);
   lancamento.setEstilo(EstiloLancamento.SAIDA);
   lancamento.setValor(200.0);
   String resultado = caixaLancamentoService.lancamento(lancamento);
    assertEquals("Saldo insuficiente para realizar esta operação", resultado);
```

Coverage Summary for Class: CaixaLancamentoService (net.originmobi.pdv.service)

Class Class, % Method, % Line, % CaixaLancamentoService 100% (1/1) 100% (7/7) 82,8% (24/29)

Current scope: all classes | net.originmobi.pdv.service

Coverage Summary for Class: CaixaLancamentoService (net.originmobi.pdv.service)

 Class
 Class, %
 Method, %
 Line, %

 CaixaLancamentoService
 100% (1/1)
 100% (7/7)
 82,8% (24/29)

```
1 package net.originmobi.pdv.service;
    import java.sql.Timestamp;
  4 import java.util.List;
5 import java.util.Optional;
    import org.springframework.stereotype.Service;
   import org.springframework.transaction.annotation.Propagation;
    import org.springframework.transaction.annotation.Transactional;
 11 import net.originmobi.pdv.enumerado.caixa.EstiloLancamento;
12 import net.originmobi.pdv.enumerado.caixa.TipoLancamento;
13 import net.originmobi.pdv.model.Caixa;
 14 import net.originmobi.pdv.model.CaixaLancamento;
 15 import net.originmobi.pdv.repository.CaixaLancamentoRepository;
 18 public class CaixaLancamentoService (
         private final CaixaLancamentoRepository caixaLancamento;
         final UsuarioService usuarios;
        private Timestamp dataHoraAtual;
public CaixaLancamentoService (CaixaLancamentoRepository caixaLancamento, UsuarioService usuarios) {

this.caixaLancamento = caixaLancamento*
             this.caixaLancamento = caixaLancamento;
this.usuarios = usuarios;
       @Transactional(readOnly = false, propagation = Propagation.REQUIRED)
public String lancamento(CaixaLancamento lancamento) {
    dataHoraAtual = new Timestamp(System.currentTimeMillis());
             lancamento.setData_cadastro(dataHoraAtual);
             if (!isCaixaValido(lancamento)) {
                   throw new RuntimeException("Nenhum caixa aberto");
             if (isSaidaComSaldoInsuficiente(lancamento)) {
    return "Saldo insuficiente para realizar esta operação";
             ajustarValorParaSaida(lancamento);
                    caixaLancamento.save(lancamento);
              } catch (Exception e) {
   throw new RuntimeException("Erro ao realizar lançamento, chame o suporte", e);
             return "Lançamento realizado com sucesso";
         private boolean isCaixaValido(CaixaLancamento lancamento) {
             return lancamento.getCaixa().isPresent() && !lancamento.getCaixa().map(Caixa::getData_fechamento).isPresent();
         private boolean isSaidaComSaldoInsuficiente(CaixaLancamento lancamento) (
             if (lancamento.getEstilo().equals(EstiloLancamento.SAIDA)) {
   Optional<Double> vlTotalCaixa = lancamento.getCaixa().map(Caixa::getValor_total);
   return lancamento.getValor() > vlTotalCaixa.orElse(0.0);
```

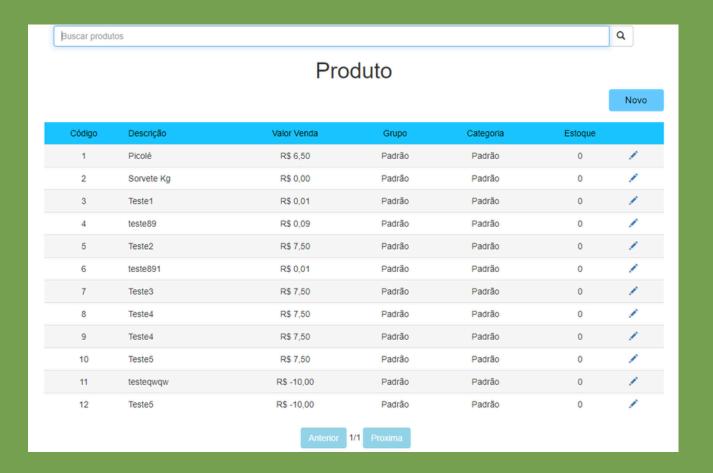
Teste de sistema

- Ferramenta utilizada: Selenium e Pytest
- Funcionalidades testadas:

Login – Verificar Autenticidade

Cadastro produto – Requisitos funcionais

```
def test_cadastro_produto_vazio(driver):
   driver.get("http://localhost:8080/login")
   time.sleep(2)
   username_input = driver.find_element(By.ID, "user")
   password_input = driver.find_element(By.ID, "password")
   username_input.send_keys("gerente")
   password_input.send_keys("123")
   time.sleep(1)
   login_button = driver.find_element(By.ID, "btn-login")
   login_button.click()
   time.sleep(2)
   driver.get("http://localhost:8080/produto/form")
   login_button = driver.find_element(By.XPATH, '//*[@id="form_produto"]/input[2]')
   login_button.click()
   time.sleep(2)
   error_message = driver.find_element(By.XPATH, ' // *[ @ id = "descricao-error"]').text
   assert error_message == "Este campo é requerido."
def test_cadastro_sem_preco(driver):
       driver.get("http://localhost:8080/login")
       username_input = driver.find_element(By.ID, "user")
       password_input = driver.find_element(By.ID, "password")
       username_input.send_keys("gerente")
       password_input.send_keys("123")
       time.sleep(1)
       login_button = driver.find_element(By.ID, "btn-login")
       login_button.click()
       time.sleep(2)
       driver.get("http://localhost:8080/produto/form")
       time.sleep(2)
       login_button = driver.find_element(By.XPATH, '//*[@id="form_produto"]/input[2]')
       login_button.click()
       time.sleep(2)
       name_input = driver.find_element(By.ID, "descricao")
       name_input.send_keys("Teste2")
       login_button = driver.find_element(By.XPATH, '//*[@id="form_produto"]/input[2]')
       login_button.click()
       error_message = driver.find_element(By.ID, 'valorVenda-error').text
       assert error_message == "Este campo é requerido."
```





Atributos de Qualidade - ISO/IEC 25010

Adequação Funcional

Nota Ideal: 5

O sistema deve garantir que todas as funcionalidades sejam de fato funcionais e de forma apropriada conforme o previsto.

Eficiência

Nota Ideal: 3.5

O sistema deve ter uma resposta rápida as operações criticas, porém, não deverá ter uma capacidade de usuários simultaneos enorme.

Compatibilidade

Nota Ideal: 3.5

O sistema deve ter uma compatibilidade com outros sistemas de pagamentos e financeiros.

O sistema deve operar bem

em ambientes onde outros aplicativos estão em uso, sem causar conflitos significativos.

Usabilidade

Nota Ideal: 4

O sistema deve garantir ter uma interface intuitiva de facil aprendizado e com bastante proteção a erros de usuários.

Atributos de Qualidade – ISO/IEC 25010

Confiabilidade

Nota Ideal: 4

O sistema deve ter pouquissimas falhas e uma alta disponbilidade para não prejudicar o funcionamento do estabelecimento.

Segurança

Nota Ideal: 4

O sistema deve ter uma forte autenticidade, não permitir que usuários desconhecidos logarem. Também possuir não deve possuir muitas vulnerabilidades.

Manutenibilidad e

Nota Ideal: 4

O sistema deve ser divido em bastantes módulos e uma alta testabilidade por se tratar de um sistema com operações financeiras.

Portabilidade

Nota Ideal: 4

O sistema deve ter uma alta adaptatividade ao diversos sistemas operacionais e uma fácil e rápida instalação.