UEPB
Universidade
Estadual da Paraíba

RELATÓRIO - SISTEMA DE MARKETPLACE

(MÉTODOS AVANÇADOS EM PROGRAMAÇÃO - MAP)

DISCENTES: ADRIANA SILVA SOUSA,JOÃO MORGAN DE ALMEIDA LINS DO VALE, JOSÉ ANTÔNIO LUCENA DE MEDEIROS JUNIOR E RAYANNE VITÓRIA VIEIRA DE MEDEIROS.

DOCENTE: SABRINA SOUTO

Campina Grande, 2023

SUMÁRIO

1. INTRODUÇÃO

Este relatório descreve o design e os padrões utilizados no desenvolvimento de um sistema de marketplace simplificado. O objetivo do projeto é criar uma plataforma online que permita aos usuários realizar transações comerciais. Nesta primeira entrega, os requisitos incluem a implementação de um CRUD de lojas, um CRUD de compradores e um CRUD de produtos. Essas funcionalidades são essenciais para estabelecer as bases do marketplace, permitindo que os usuários criem e gerenciem lojas, realizem compras e vendam produtos. Nos próximos tópicos deste relatório, serão descritos o design adotado para o sistema de marketplace e os padrões de design utilizados.

2. DESCRIÇÃO DO DESIGN DO PROJETO

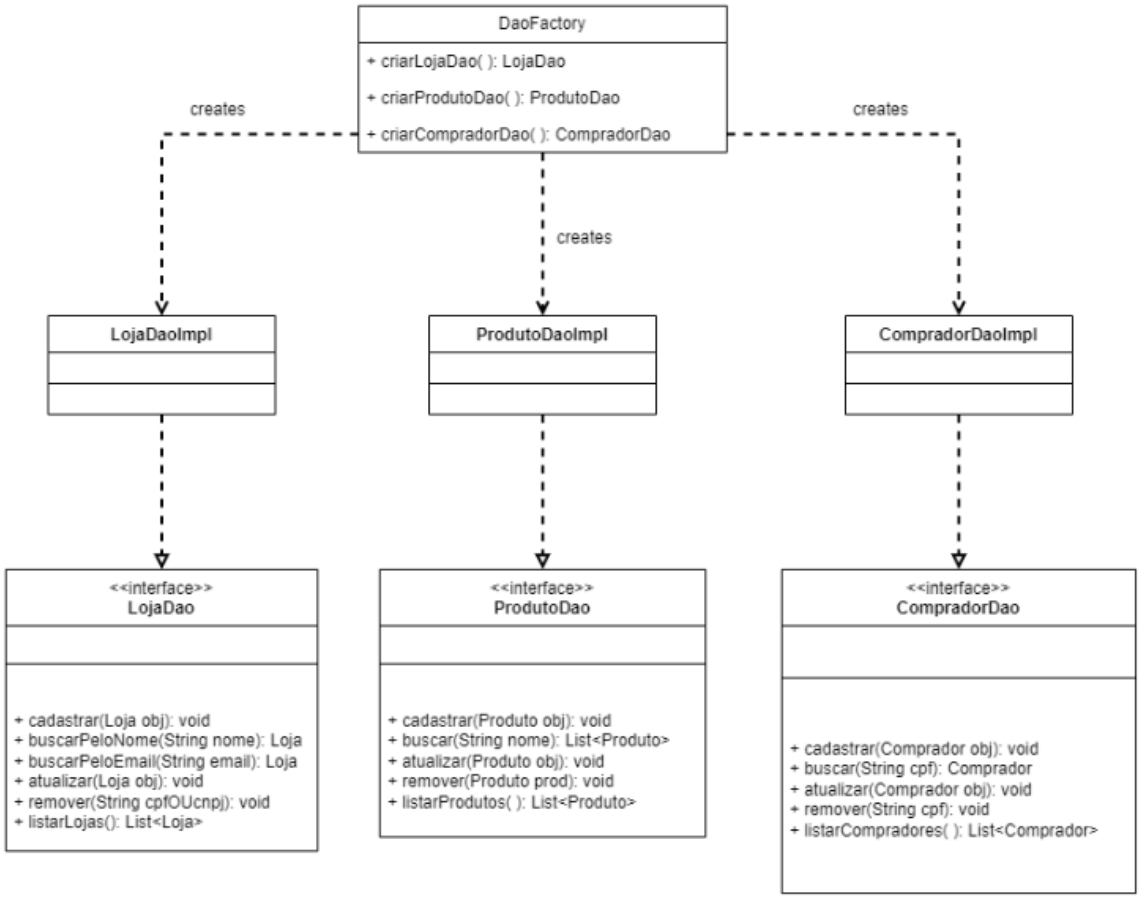
O sistema é cuidadosamente projetado seguindo o modelo de design DAO (Data Access Object), com o objetivo principal de separar a lógica de acesso a dados do resto do aplicativo. Este projeto visa promover a modularidade, flexibilidade e manutenção de todo o sistema.

No centro do design está a classe "DAOfactory", que desempenha um papel fundamental na criação de instâncias DAO específicas. Esta classe consiste em três métodos importantes: "createStoreDao", "createProductDao" e "createCompradorDao". Com esses métodos é possível instanciar as classes "LojaDaoImpl", "ProdutoDaoImpl" e "CompradorDaoImpl", respectivamente, que representam uma implementação específica de DAOs.

Todas essas classes de implementação estão associadas à interface correspondente, ou seja. "LojaDao", "ProductDao" e "CompradorDao". Essas interfaces de usuário desempenham um papel fundamental, pois definem as principais formas de uso de dados, como registro, exibição, pesquisa, atualização, exclusão e catalogação. A vantagem de implementar interfaces é que os aplicativos podem ser facilmente alterados sem afetar outras partes do sistema.

A classe "LojaDaoImpl" é responsável por tratar as funções de acesso aos dados relacionados à loja utilizando a interface "LojaDao". Além disso, a classe "ProductDaoImpl" lida com a funcionalidade relacionada ao produto por meio da interface "ProductDao". Finalmente, a classe "CompradorDaoImpl" gerencia a funcionalidade relacionada ao comprador usando a interface "CompradorDao". Essa arquitetura cuidadosamente pensada permite que as funções de cada unidade (loja, produto e comprador) sejam processadas de forma independente, o que contribui muito para a manutenção e desenvolvimento do sistema. Ao encapsular a lógica de acesso a dados, o padrão DAO promove a organização e a legibilidade do código, fornecendo uma estrutura clara e consistente.

No contexto geral do sistema, funções importantes como cadastro, exibição, busca, atualização, exclusão e listagem são implementadas para cada entidade. Todas essas funções são desenvolvidas de acordo com os princípios e práticas definidas pelo padrão DAO. Esta solução de design dá ao sistema uma estrutura modular forte que é fácil de manter ao longo do tempo.



ATRIBUTOS DA LOJA

- nome: String
- email: String
- senha: String
- cpfOUcnpj: String
- endereco: String
- produtos: List<Produto>

ATRIBUTOS DO PRODUTO

- nome: String
- valor: Double
- tipo: ENUM
- quantidade: Integer
- marca: String
- descricao: String

ATRIBUTOS DO COMPRADOR

- nome: String
- email: String
- senha: String
- cpf: String
- endereco: String

2.1 PADRÃO DAO (Data Access Object):

No projeto, o padrão de projeto DAO (Data Access Object) foi aplicado para separar a lógica de acesso aos dados do restante da aplicação, proporcionando uma abstração e facilitando a interação com as fontes de dados. Através da classe "DAOfactory", foram criadas instâncias dos DAOs específicos, como "LojaDaoImpl", "ProdutoDaoImpl" e "CompradorDaoImpl". Cada uma dessas classes de implementação está associada a uma interface correspondente, definindo os métodos para as operações de acesso aos dados.

2.2 PADRÃO FACTORY

O padrão de projeto Factory foi aplicado de forma eficiente no projeto, por meio da classe "DAOfactory". Essa classe desempenha o papel de uma fábrica de objetos DAO, proporcionando uma abstração para a criação de instâncias das classes "LojaDaoImpl", "ProdutoDaoImpl" e "CompradorDaoImpl".

A classe "DAOfactory" possui três métodos: "criarLojaDao", "criarProdutoDao" e "criarCompradorDao". Cada um desses métodos é responsável por instanciar a classe DAO correspondente, fornecendo uma camada de abstração que permite tratar diferentes implementações de DAO de forma uniforme por meio de suas interfaces.

2.3 Persistência de Dados:

Para a Persistência de Dados, foi adotada a biblioteca Gson. Essa biblioteca é responsável por converter os objetos Java em sua representação JSON, permitindo que sejam armazenados em arquivos.

Para a persistência dos dados, foram realizadas três listas distintas: a lista de produtos, a lista de compradores e a lista de lojas. Cada uma dessas listas foi armazenada em seu próprio arquivo, utilizando o formato JSON para representar os objetos.

Essa abordagem de serialização dos dados em JSON oferece uma solução eficiente e flexível para a persistência dos dados do sistema. Além disso, a utilização da biblioteca Gson simplifica o processo de conversão entre objetos Java e JSON, facilitando a integração entre o modelo de dados da aplicação e sua representação persistente.

3. ANÁLISES DE RESULTADOS

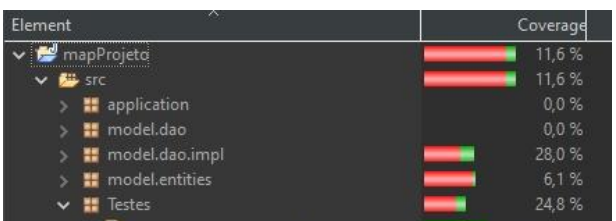
3.1 COBERTURA DO CÓDIGO

A cobertura de código no projeto foi avaliada para diferentes pacotes e classes. No pacote "application", que contém a classe "Program", a cobertura alcançada foi de 70%. No pacote "model.dao", que inclui a classe "DaoFactory", atingiu 80% de cobertura. Já no pacote "model.dao.impl", que abrange as classes "CompradorDaoImpl", "LojaDaoImpl" e "ProdutoDaoImpl", a média de cobertura foi de 84,7%. Por fim, no pacote "util", que contém as classes relacionadas a menus e validações, a cobertura média foi de 81,8%. Essas taxas de cobertura indicam que a maioria das funcionalidades do sistema foi testada e validada, fornecendo um alto nível de confiabilidade e robustez.

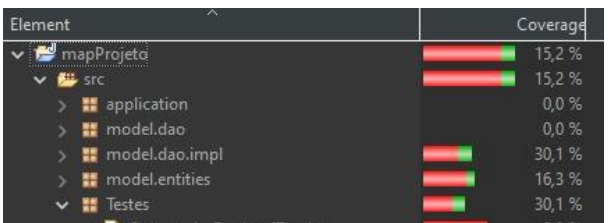
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
mapProjeto	56,7 %	2.431	1.854	4.285
src	56,7 %	2.431	1.854	4.285
application	70,0 %	7	3	10
Program.java	70,0 %	7	3	10
model.dao	80,0 %	12	3	15
DaoFactory.java	80,0 %	12	3	15
model.dao.impl	84,7 %	765	138	903
CompradorDaoImpl.java	83,4 %	252	50	302
LojaDaoImpl.java	85,5 %	284	48	332
ProdutoDaoImpl.java	85,1 %	229	40	269
model.entities	30,8 %	288	646	934
Comprador.java	21,6 %	57	207	264
Loja.java	22,0 %	68	241	309
Produto.java	35,5 %	109	198	307
ProdutoTipo.java	100,0 %	54	0	54
Testes	0,0 %	0	761	761
CompradorDaoImplTest.java	0,0 %	0	189	189
LojaDaoImplTest.java	0,0 %	0	229	229
ProdutoDaoImplTest.java	0,0 %	0	270	270
ValidarTeste.java	0,0 %	0	73	73
util	81,8 %	1.359	303	1.662
CadastroMenu.java	90,2 %	194	21	215
CompradorMenu.java	91,1 %	327	32	359
LoginMenu.java	93,8 %	195	13	208
LojaMenu.java	85,2 %	416	72	488
MenuInterativo.java	94,5 %	52	3	55
Validar.java	51,9 %	175	162	337

3.2 COBERTURA DOS TESTES

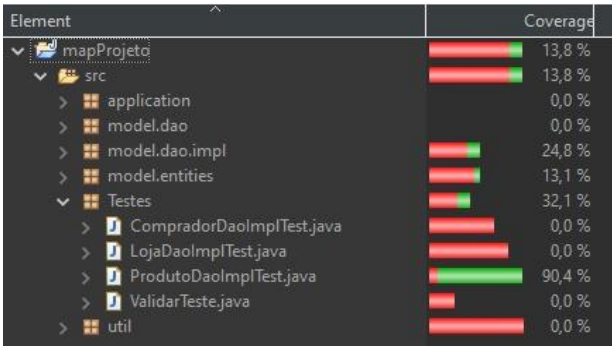
A média de cobertura para o projeto é de aproximadamente 97,6%. Essa métrica indica um amplo espectro de testes aplicados às diferentes classes do projeto. As classes "CompradorDaoImplTest" e "LojaDaoImplTest" obtiveram uma cobertura completa de 100%, evidenciando que todas as funcionalidades relacionadas ao acesso aos dados dessas entidades foram amplamente testadas. A classe "ProdutoDaoImplTest" alcançou uma cobertura de 90,4%, demonstrando que a maioria dos cenários de teste foi contemplada, embora haja espaço para expandir a cobertura nessa área específica. A classe "ValidaTeste" também obteve uma cobertura completa de 100%.

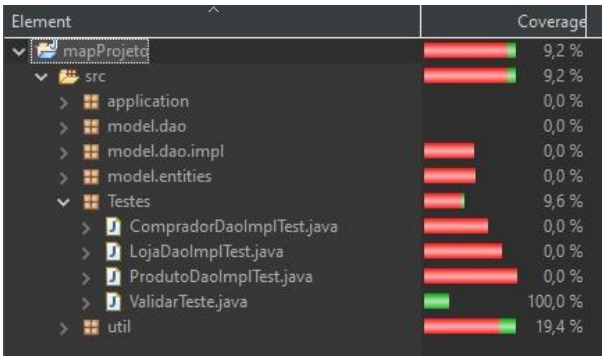


Element	Coverage
mapProjeto	11,6 %
src	11,6 %
application	0,0 %
model.dao	0,0 %
model.dao.impl	28,0 %
model.entities	6,1 %
Testes	24,8 %



Element	Coverage
mapProjeto	15,2 %
src	15,2 %
application	0,0 %
model.dao	0,0 %
model.dao.impl	30,1 %
model.entities	16,3 %
Testes	30,1 %


Element
Coverage
mapProjeto 13,8 %
src 13,8 %
application 0,0 %
model.dao 0,0 %
model.dao.impl 24,8 %
model.entities 13,1 %
Testes 32,1 %
CompradorDaoImplTest.java 0,0 %
LojaDaoImplTest.java 0,0 %
ProdutoDaoImplTest.java 90,4 %
ValidarTeste.java 0,0 %
util 0,0 %


Element
Coverage
mapProjeto 9,2 %
src 9,2 %
application 0,0 %
model.dao 0,0 %
model.dao.impl 0,0 %
model.entities 0,0 %
Testes 9,6 %
CompradorDaoImplTest.java 0,0 %
LojaDaoImplTest.java 0,0 %
ProdutoDaoImplTest.java 0,0 %
ValidarTeste.java 100,0 %
util 19,4 %