Universidade Federal do Cariri

Antonio Nunes de Oliveira Filho e José Vitor Dias dos Santos

PROGRAMAÇÃO ORIENTADA À OBJETOS

Documentação do Projeto - Linkin Park Store

Juazeiro do Norte - CE

2024

UNIVERSIDADE FEDERAL DO CARIRI

Antonio Nunes de Oliveira Filho e José Vitor Dias dos Santos

PROGRAMAÇÃO ORIENTADA À OBJETOS

Documentação do Projeto - Linkin Park Store

Projeto apresentado no curso de Ciência da Computação da Universidade Federal do Cariri, como requisito na disciplina de POO - Programação Orientada à Objetos.

Orientadora: Paola Accioly.

Juazeiro do Norte - CE

2024

UNIVERSIDADE
FEDERAL DO CARIRI

Controle da Versão

Data	Versão	Descrição	Autor
21/09/2024	1.0	Criação do documento	Antonio Nunes e José Vitor
06/11/2024	2.0	Atualização do documento com as finalizações do projeto	Antonio Nunes e José Vitor

UNIVERSIDADE
FEDERAL DO CARIRI

Índice Analítico

UNIVERSIDADE FEDERAL DO CARIRI

Documentação do Projeto - Linkin Park Store

1. Descrição do Sistema: Loja Online do Linkin Park

Nosso projeto, a Linkin Park Store, é uma loja online do Linkin Park, uma famosa e histórica banda de rock internacional dos anos 2000, que voltou à ativa recentemente. Nessa loja online, estarão disponíveis à venda os mais diversos produtos da banda, como discos, álbuns, camisetas, pôsteres, bandeiras e muito mais, de forma acessível e facilitada para os fãs mais entusiastas da banda. Ficamos motivados a escolher este tema pelo apreço que temos pelo legado da banda e por suas músicas.

O gerente da loja é o responsável por cadastrar todos os produtos, inserindo dados como nome, preço, categoria, estoque e descrição, sendo todos obrigatórios. Além disso, o mesmo pode remover e atualizar os produtos, podendo alterar dados como preço e estoque de cada produto. Este também tem as opções de listar produtos, filtrar produtos por categoria e ordenar produtos por preço, além de poder visualizar as últimas vendas realizadas na loja, ou a última venda, de forma individual.

Por outro lado, o cliente deve cadastrar-se no sistema, fazer login, e a partir daí, poderá visualizar os produtos da loja e montar seu carrinho de compras com as opções disponíveis, que como dito, são previamente cadastradas pelo gerente. O mesmo pode visualizar os produtos de uma forma geral, com uma ordenação por preço ou filtrá-los por categoria, como por exemplo, apenas “pôsteres”. Além disso, pode alterar seu carrinho de compras a qualquer momento, removendo produtos anteriormente adicionados ou adicionando novos, e finalizar sua compra no checkout da loja.

2. Backlog do Projeto

Funcionalidade	Responsável
CRUD Cliente	Antonio Nunes e José Vitor
CRUD Produto	Antonio Nunes e José Vitor
CRUD Carrinho	Antonio Nunes e José Vitor
CRUD Pedido	Antonio Nunes e José Vitor

Cadastrar Cliente e Fazer Login	Antonio Nunes e José Vitor
Cadastrar Gerente e Fazer Login	Antonio Nunes e José Vitor
Cadastrar e Remover Produto (Gerente)	Antonio Nunes e José Vitor
Atualizar Preço e Estoque dos Produtos (Gerente)	Antonio Nunes e José Vitor
Visualizar Produtos (Cliente e Gerente)	Antonio Nunes e José Vitor
Adicionar Produto ao Carrinho (Cliente)	Antonio Nunes e José Vitor
Remover Produto do Carrinho (Cliente)	Antonio Nunes e José Vitor
Filtrar Produtos por Categoria (Cliente e Gerente)	Antonio Nunes e José Vitor
Ordenar Produtos por Preço (Cliente e Gerente)	Antonio Nunes e José Vitor
Visualizar Carrinho (Cliente)	Antonio Nunes e José Vitor
Finalizar Compra (Cliente)	Antonio Nunes e José Vitor
Visualizar Última Venda (Gerente)	Antonio Nunes e José Vitor
Visualizar Todas Vendas (Gerente)	Antonio Nunes e José Vitor

3. Arquitetura do Sistema

3.1. Diagrama de Casos de Uso

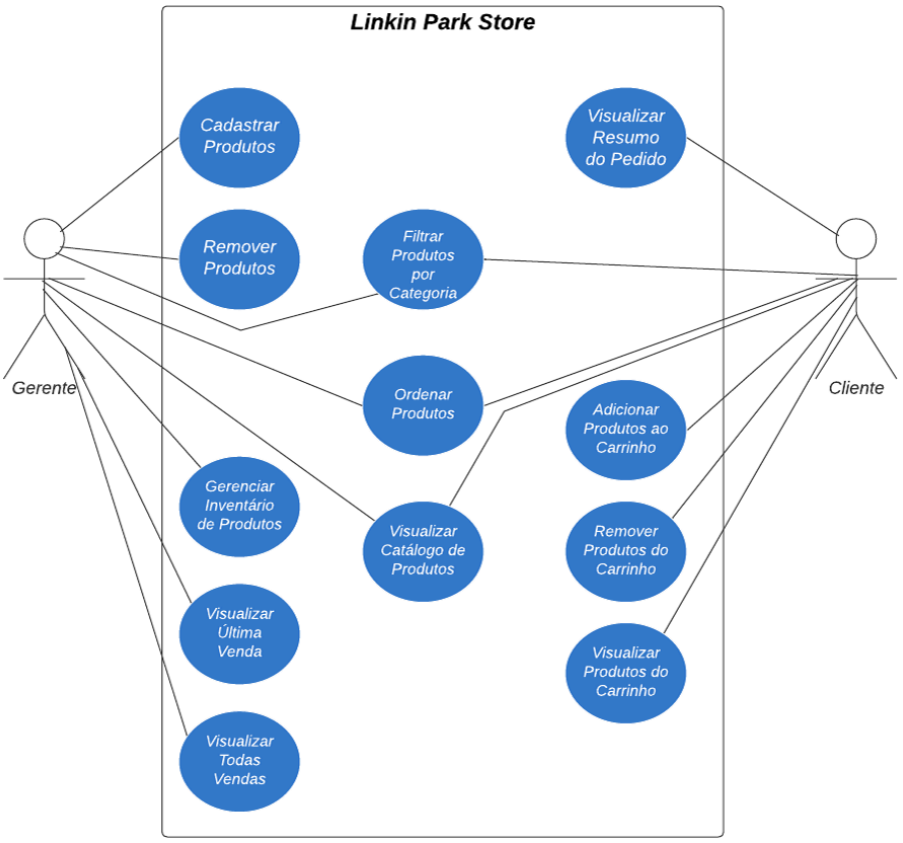

Figura 1: diagrama de casos de uso

Aqui estão as descrições dos casos de uso, divididos de acordo com as funcionalidades específicas para o **Gerente** e para o **Cliente**, além das funcionalidades compartilhadas:

Casos de Uso para o Gerente:

1. **Cadastrar Produtos**:
   - Permite ao Gerente adicionar novos produtos à loja. O Gerente pode especificar informações detalhadas, como nome do produto, descrição, preço, categoria (ex.: álbuns, camisetas, pôsteres) e quantidade em estoque.
2. **Remover Produtos**:

    - Permite ao Gerente excluir produtos que não estão mais disponíveis na loja. Essa funcionalidade ajuda a manter o catálogo atualizado, removendo itens descontinuados ou fora de estoque.
3. **Gerenciar Inventário de Produtos**:
    - Permite ao Gerente monitorar e atualizar as informações de estoque e preços dos produtos. Essa funcionalidade é essencial para garantir que o inventário esteja sempre correto e que os preços reflitam as condições de venda.
4. **Visualizar Última Venda**:
    - Permite ao Gerente visualizar detalhes da última compra realizada na loja. Esse recurso inclui informações como ID do pedido, nome do cliente que efetuou a compra, produtos vendidos e valor total, auxiliando o Gerente no controle de vendas e planejamento de estoque.
5. **Visualizar Todas Vendas**:
    - Permite ao Gerente visualizar detalhes de todas as compras realizadas na loja. Esse recurso inclui informações como ID do pedido, nome do cliente que efetuou a compra, produtos vendidos e valor total, auxiliando o Gerente no controle de vendas e planejamento de estoque.

Casos de Uso para o Cliente:

1. **Visualizar Resumo do Pedido**:
    - Exibe para o Cliente um resumo do pedido antes de finalizar a compra. O resumo inclui os produtos selecionados, suas quantidades, preços individuais e o valor total, permitindo que o Cliente revise a compra antes de concluir.
2. **Adicionar Produtos ao Carrinho**:
    - Permite ao Cliente selecionar produtos e adicioná-los ao carrinho de compras, preparando-os para uma possível compra. Cada item adicionado ao carrinho inclui informações como preço e quantidade.
3. **Remover Produtos do Carrinho**:
    - Oferece ao Cliente a opção de retirar produtos previamente adicionados ao carrinho. Isso possibilita ajustes antes de finalizar a compra, como remoção de itens indesejados ou ajuste de quantidades.
4. **Visualizar Produtos do Carrinho**:

- Permite ao Cliente ver um resumo dos produtos que foram adicionados ao carrinho, incluindo os preços e a quantidade de cada item. Esse recurso facilita a verificação da seleção de produtos antes de prosseguir para o checkout.

Casos de Uso Compartilhados (Gerente e Cliente):

1. **Filtrar Produtos por Categoria**:
   - Disponível para ambos, permite que os usuários filtrem os produtos de acordo com categorias específicas (ex.: álbuns, roupas, acessórios). Essa funcionalidade facilita a navegação e ajuda os usuários a encontrar rapidamente o que procuram.
2. **Ordenar Produtos**:
   - Essa funcionalidade permite a ordenação dos produtos pelo critério de preço. Disponível para o Gerente e para o Cliente, ajuda ambos a organizar a visualização do catálogo conforme desejado.
3. **Visualizar Catálogo de Produtos**:
   - Exibe a lista completa de produtos disponíveis na loja. Para o Cliente, é uma maneira de ver todos os produtos para selecionar o que deseja. Para o Gerente, é uma forma de revisar o catálogo e gerenciar os itens exibidos aos clientes.

3.2. Diagrama de Classes da UML

UML ← Diagrama de Classes da UML.

Aqui estão as descrições do diagrama de classes da UML, divididos de acordo com os tipos de relacionamentos, além das funcionalidades compartilhadas:

Relações de Associação:

1. **CarrinhoCRUD ↔ Carrinho**
   - **Tipo**: Associação
   - **Descrição**: A classe CarrinhoCRUD contém uma instância da classe Carrinho, indicando que CarrinhoCRUD pode operar diretamente sobre um objeto Carrinho.
2. **ClienteCRUD ↔ Cliente**
   - **Tipo**: Associação
   - **Descrição**: A classe ClienteCRUD possui uma lista de Cliente, permitindo gerenciar a entidade Cliente.

3. **PedidoCRUD ↔ Pedido**
    - **Tipo**: Associação
    - **Descrição**: A classe PedidoCRUD mantém uma lista de objetos Pedido para gerenciamento.
4. **ProdutoCRUD ↔ Produto**
    - **Tipo**: Associação
    - **Descrição**: A classe ProdutoCRUD armazena e manipula uma lista de objetos Produto.

Relações de Agregação:

1. **Carrinho ↔ ItemCarrinho**
    - **Tipo**: Agregação
    - **Descrição**: O Carrinho contém uma lista de ItemCarrinho, mas esses itens podem existir fora do contexto do Carrinho.
2. **Pedido ↔ Carrinho**
    - **Tipo**: Agregação
    - **Descrição**: Um Pedido utiliza a lista de itens do Carrinho, mas o Carrinho pode existir independentemente de um Pedido.
3. **FachadaLoja ↔ ClienteCRUD, ProdutoCRUD, CarrinhoCRUD, PedidoCRUD**
    - **Tipo**: Agregação
    - **Descrição**: A FachadaLoja contém referências a objetos dos CRUDs (ClienteCRUD, ProdutoCRUD, CarrinhoCRUD, PedidoCRUD), mas eles podem existir independentemente da FachadaLoja.

Relações de Herança:

1. **Cliente → Pessoa**
    - **Tipo**: Herança
    - **Descrição**: Cliente herda de Pessoa, indicando que um Cliente é uma especialização de Pessoa.
2. **Gerente → Pessoa**
    - **Tipo**: Herança
    - **Descrição**: Gerente herda de Pessoa, indicando que um Gerente é uma especialização de Pessoa.
3. **Carrinho → Compra**
    - **Tipo**: Herança
    - **Descrição**: Carrinho é um tipo de Compra, representando uma relação de herança entre as duas classes.

4. **Pedido → Compra**
    - **Tipo**: Herança
    - **Descrição**: Pedido é um tipo de Compra, representando uma relação de herança entre as duas classes
5. **ItemCarrinho → ProdutoBase**
    - **Tipo**: Herança
    - **Descrição**: ItemCarrinho é um tipo de ProdutoBase, representando uma relação de herança entre as duas classes
6. **Produto → ProdutoBase**
    - **Tipo**: Herança
    - **Descrição**: Produto é um tipo de ProdutoBase, representando uma relação de herança entre as duas classes

Relações Extras:

1. **Cliente — nome, id**
    - **Descrição**: Os dados que são pedidos pelo cliente e tem relação com essa classe.
2. **Gerente — nome, id**
    - **Descrição**: Os dados que são pedidos pelo gerente e tem relação com essa classe.
3. **Produto — nome, id, quantidadeestoque, preço, categoria, descrição**
    - **Descrição**: Os dados que são pedidos na hora de cadastrar os produtos e que tem relação com essa classe.

4. Pacotes

O sistema foi dividido em três pacotes: o pacote de dados, o pacote de negócios e o pacote de interface.

- **Pacote de dados:**

  Esse pacote cuida da persistência e da manipulação de dados. Ele é responsável por interagir com a base de dados ou qualquer outra fonte de dados que a aplicação utilize.

  As **classes** que estão contidas neste pacote são: CarrinhoCRUD, ProdutoCRUD, ClienteCRUD, PedidoCRUD.

- **Pacote de negócios:**

Esse pacote é onde a lógica de negócio da aplicação é implementada. Ele contém classes e métodos que representam e manipulam as regras de negócio do sistema.

As **classes** que estão contidas neste pacote são: Carrinho, Cliente, Compra, FachadaLoja, Gerente, ItemCarrinho, Pedido, Pessoa, Produto e ProdutoBase.

- **Pacote de interface:**

  Esse pacote é responsável pela interação do usuário com o sistema. Pode conter classes relacionadas às formas de entrada e saída de dados. É o pacote que contém os prints do sistema.

  A **classe** que está contida neste pacote é: Main.

5. Considerações Adicionais

Nosso sistema tem a fachadaLoja, que é a classe que contém todos os métodos dos CRUDs e métodos básicos, e retorna strings para o main, que é onde está localizado todos os prints, e assim é feita a divisão. A implementação em questão centraliza todas as operações do CRUD, as quais são empregadas na função principal (main). Ela também assume responsabilidades anteriormente delegadas à função principal, com o intuito de reduzir a complexidade e evitar a sobrecarga de funções dentro do bloco principal. Esse design permite que o código da função principal permaneça mais organizado e legível, contendo apenas as interações essenciais com o usuário, como exibição de informações e prompts, sem expor diretamente a lógica de processamento e as chamadas de funções. As interações no bloco principal se restringem, portanto, às chamadas que utilizam a fachada da loja, mantendo a abstração e ocultando a complexidade das operações internas.

A herança está contida na classe Pessoa, que seria a superclasse em que as subclasses Cliente e Gerente herdam algumas características de Pessoa. Além disso, ainda temos a superclasse ProdutoBase que tem como herdeiros as subclasses Produto e ItemCarrinho. Ademais, ainda temos a superclasse Compra que tem como herdeiros as seguintes subclasses: Pedido e Carrinho.

A classe abstrata Pessoa é usada como uma base comum para outras classes, como Cliente e Gerente, que herdam dela. A herança permite que essas classes compartilhem atributos e métodos comuns, enquanto o polimorfismo possibilita que objetos de Cliente e Gerente sejam tratados como objetos do tipo Pessoa, mas ainda executem seus próprios métodos específicos. Com isso, métodos como exibirDados() podem ser chamados de forma genérica em objetos do tipo Pessoa, proporcionando flexibilidade e reutilização de código.

```java
package pacoteDeNegocios;

public class Gerente extends Pessoa {
    private final String senha;

    public Gerente(String nome, String senha) {
        super(nome);
        this.senha = senha;
    }

    public boolean validarSenha(String senha) {
        return this.senha.equals(senha);
    }

    @Override
    public String exibirDados() {
        return "Nome do gerente: " + nome + "\nID do gerente: " + id;
    }
}
```

Figura 2: Exemplo de ocultação de informação

Figura 3: Exemplo de interface

```
case 2 -> {
        System.out.print(s:"Digite o nome do produto que deseja adicionar: ");
        String nomeProduto = scanner.nextLine();
        int quantidade = -1;
        boolean quantidadeValida = false;

        while (!quantidadeValida) {
            try {
                System.out.print(s:"Digite a quantidade: ");
                quantidade = scanner.nextInt();
                scanner.nextLine();
                quantidadeValida = true;
            } catch (InputMismatchException e) {
                System.out.println(x:"Quantidade inválida. Por favor, digite um número inteiro.");
                scanner.nextLine();
            }
        }

        System.out.println(fachadaLoja.adicionarItem(nomeProduto, quantidade));
}
```

Figura 4: Exemplo de tratamento e lançamento de exceções