Sistema de Criação de Estoque

Allejandro S. Dos Santos
Ivanor Meira Lima Neto
Guilherme Dantas Pinto
Klayvert de A. Araújo

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2023

Sumário

1 INTRODUÇÃO

O gerenciamento eficaz de estoques é um desafio constante para empresas e organizações de diversos setores. A necessidade de controlar o fluxo de produtos, garantir a disponibilidade de mercadoriars e otimizar o espaço de armazenamento é um deseafio constante que impacta diretamente a eficiência operacional.

Nesse contexto, o Sistema de Criação o de Armazenagem de Estoques visa solucionar os desafios enfrentados na admnistração de estoques, otimizar o controle e a gestão, especialmente em ambientes onde o armazenamento de produtos é uma parte crítica das operações diárias, como centros de distribuição, armazéns e empresas de logística. A falta de um sistema eficiente de gerenciamento de estoque pode resultar em desperdício de recursos, perdas financeiras e, em última instância, em insatisfação do cliente devido a atrasos ou falta de produtos.

1.1 Objetivos:

1. **Desenvolver sistema de gerenciamento de estoque eficiente e versátil:** Criar uma solução eficaz para os desafios enfrentados no gerenciamento de estoque. Isso envolve a automação de operações, como alocação de espaço, rastreamento de produtos e movimentação eficiente.

2. **Otimizar a Eficiência Operacional:** Busca-se melhorar a eficiência operacional no gerenciamento de estoque, reduzindo o tempo e os recursos necessários para tarefas de rotina;

2. **Melhorar a Precisão do Rastreamento:** Fornecer uma solução precisa para rastrear produtos dentro do estoque, minimizando perdas e erros;

3. **Aumentar a Capacidade de Armazenamento:** Otimizar o uso do espaço e expandir a capacidade de armazenamento;

4. **Reduzir Custos Operacionais:** Ao melhorar a eficiência, busca-se a redução dos custos operacionais do gerenciamento de estoque;

2 MODELAGEM DO PROBLEMA

2.1 Produto:

- A classe **Produto** representa os produtos que são armazenados no estoque.
- Possui atributos **id**, **nome**, **tipoDeArmazenagem**, **descricao** e **posicao** que descrevem o produto.
- A classe **Produto** tem um relacionamento com a classe **Posicao** para rastrear sua localização no estoque.

2.2 Posicao:

- A classe **Posicao** descreve a posição tridimensional de um produto no estoque.
- Classe responsável por permitir definir e recuperar as coordenadas de posição do produto no estoque.
- Possui os atributos **andar**, **apartamento**, e **predio**.
- Outras classes, como **Produto** e **Estoque**, usam a classe **Posicao** para manter o controle da localização de produtos no estoque.

2.3 Estoque:

- A classe **Estoque** representa o estoque geral onde os produtos são armazenados.
- Possui os atributos **predios**, uma lista de objetos **Predio**, representado um prédio no estoque, **produtosNoEstoque**, uma lista de todos os produtos atualmente armazenados no estoque, **dimensão**, representado pela classe **DimensaoEstoque**, definindo o tamanho e a capacidade do estoque.
- A classe **Estoque** mantém uma lista de produtos armazenados no estoque, estabelecendo um relacionamento de agregação entre o estoque e os produtos.
- Possui métodos para adicionar, remover, mover, alocar e fazer inventário de Produtos e métodos para verificar se o estoque ainda tem capacidade para armazenar produtos e retonar a quantidade total de produtos

2.4 Predio:

- A classe **Predio** representa um edifício onde os produtos podem ser armazenados.
- Possui atributos como **predioID**, que indica o prédio, e uma matriz de produtos chamada **matrizDeProdutos**, que organiza os produtos em estrutura de andares e apartamentos
- Possui métodos para adicionar e remover produtos, bem como para verificar a disponibilidade do prédio para receber novos produtos.

2.5 Dimensao:

- A classe abstrata **Dimensao** fornece uma estrutura comum para representar as dimensões de diferentes partes do estoque, como **Estoque** e **Predio**.
- Ela encapsula informações sobre o números de apartamentos, andares e prédios, permitindo calular a capacidade máxima de armazenamento. **S**eus atributos são **numApartamentos** e **numAndares**.
- Subclasses, como **DimensaoEstoque** e **DimensaoPredio**, estendem **Dimensao** para fornecer implementações específicas para as dimensões de diferentes partes do estoque.

2.6 Menu:

- A classe **Menu** fornece uma interface para interagir com o sistema de gerenciamento de estoque e produtos, permitindo que os usuários escolham entre opções como dimensionar o estoque, adcionar, retirar, mover produtos e fazer inventário.
- A classe possui como métodos **printMenu**, usado para exibir o menu principal na tela, **inputString** método usando para obter entradas de texto do usuário realizando a validação da entrada, **inputInt**, método usado para obter entradas numéricas do usuário, realizando a validação da entrada.

2.7 Princípios da Orientação a Objetos Aplicados no Projeto:

Abstração:

- A classe **Armazenamento** é uma abstração genérica que representa o conceito comum entre **Predio** e **Estoque**. Ela define métodos como **adicionarProduto**, **removerProduto** e **moverProduto**, que são implementados de forma específica nas classes derivadas.
- A classe **Dimensao** é uma classe abstrada que representa as dimensões de uma entidade no sistema, como **Estoque** e **Predio**.

Encapsulamento:

- Os atributos das classes são protegidos com modificadores de acesso privado e acessados por meio de métodos **getters** e **setters**, garantindo a segurança dos dados.

Herança:

- A classe **Produto** herda atributos e métodos da classe **Posicao**, aproveitando os atributos e métodos relacionados à posição. Essa herança facilita a modelagem de produtos com posição.
- As classes **DimensaoEstoque** e **DimensaoPredio** herdam da classe abstrata Dimensao. Isso permite que compartilhem características comuns relacionadas à dimensão.
- As classe **Estoque** estende **Armazenamento**, herdando suas características e métodos. Isso representa uma forma de herança na hierarquia de classes.
- A classe **Predio** herda de **Armazenamento**. Essa herança permite que compartilhe funcionalidades comuns de armazenamento, como adicionar e remover produtos.

Polimorfismo:

- Métodos como **adicionarProduto** e **removerProduto** são polomórficos, pois são aplicados tanto na classe **Predio** quanto na classe **Estoque**, permitindo que o mesmo método seja utilizado em contextos diferentes.
- O método **inputInt** na classe **Menu** usa sobrecarga, aceitando diferentes tipos de parâmetros (um para cada tipo de mensagem) para obter uma entrada do usuário. Isso é um exemplo de polimorfismo de sobrecarga.
- Na classe **Estoque**, o método **fazerInventarioDeProdutos** itera sobre a lista de produtos, independentemente do tipo específico de produto. Isso é polimorfismo de lista, onde objetos de classes diferentes são tratados de maneira uniforme.
- O método auxiliar **encontrarProdutoPorID** na classe **Main** aceita um objeto da classe **Estoque**, mas funciona para qualquer classe que tenha uma lista de produtos. Isso ilustra polimorfismo de método auxiliar.

3 FERRAMENTAS UTILIZADAS

- **IDE (Ambiente de Desenvolvimento Integrado):** Para escrever, compilar e depurar o código-fonte, foi adotada a IDE IntelliJ IDEA, devido à sua riqueza de recursos, simplificando a criação, edição e organização do código de forma intuitiva, além de facilitar a integração de bibiliotecas e o gerenciamento do projeto.
- **Java:** O desenvolvimento deste sistema foi realizado em um ambiente de programação Java.
- **Git e GitHub:** Para controle de versão do código-fonte, foi utilizado o sistema de controle de versão Git, juntamente com a plataforma GitHub para hospedar o código-fonte. Isso permitiu que a equipe de desenvolvimento colaborasse de forma eficaz, rastreasse alterações no código e garantisse a integridade do projeto.
- **UML:** Para a criação de diagramas de pacotes e classes, ferramentas de modelagem UML, como o Lucidchart, foram empregadas.

3.1 Estrutura de Pacote

- **Pacote Armazenamento:** Este pacote contém as classes relacionadas ao armazenamento físico de produtos, como a classe Estoque. Ele lida com a adição, retirada e movimentação de produtos no estoque.
- **Pacote Dimensoes:** Este pacote contém classes relacionadas à representação de dimensões de objetos, incluindo Dimensao (classe abstrata) e DimensaoEstoque. Essas classes são usadas para definir o tamanho de estoques e outros componentes no sistema.
- **Pacote Posicao:** Este pacote abriga a classe Posicao, que é usada para representar a posição de produtos no estoque.
- **Pacote Produto:** Este pacote contém a classe Produto, que representa os produtos a serem gerenciados no sistema. Essa classe inclui informações como ID, nome, tipo de armazenagem e descrição.
- **Pacote Frontend:** Este pacote contém a classe Menu, que fornece uma interface interativa com o sistema de gerenciamento de estoque, permitindo que os usuários escolham entre opções como dimensionar, adcionar, retirar, mover e fazer inventário de produtos.

4 RESULTADOS E CONSIDERAÇÕES

Ao longo do desenvolvimento deste projeto, pudemos alcançar resultados significativos que impactam positivamente o gerenciamento de estoques em diferentes organizações. A solução proposta proporcionou uma abordagem eficiente e versátil para o controle de produtos em estoque, e as considerações finais abrangem tanto os resultados alcançados quanto as lições aprendidas no processo.

4.1 Resultados Alcançados:

- **Melhoria na Eficiência Operacional:** A automação das operações de gerenciamento de estoque, incluindo a alocação de produtos e a movimentação de mercadorias, contribui para uma maior organização operacional.
- **Precisão no Rastreamento de Produtos:** O sistema proporcionou rastreabilidade completa das operações de entrada, saída e movimentação de produtos, minimizando erros e perdas, garantindo maior visibilidade e confiabilidade das informações.
- **Melhoria na Capacidade de Armazenamento:** A otimização de espaço gerada pelo sistema de criação de estoques aumenta a capacidade geral de armazenamento sem a necessidade de expansões físicas.
- **Redução dos Custos Operacionais:** A automação e a eficiência melhorada resultam em uma redução significativa nos custos operacionais do gerenciamento de estoque. Menos mão de obra humana foi necessária para operações diárias, economizando recursos financeiros.

4.2 Dificuldades e Desafios

- **Complexidade Tecnológica:** O gerenciamento de estoques com dimensionamento variável inicial dos estoques é um domínio complexo com diversas nuances, compreender completamente esses detalhes foi um desafio.
- **Implementação de Funcionalidades Complexas:** A implementação de funcionalidades avançadas, como rastreamento de produtos e relatórios de inventário, exigiu um aprofundamento considerável no conhecimento da linguagem Java e do paradigma de Orientação a Objetos.
- **Testes e Depuração:** Garantir a estabilidade e o funcionamento confiável do sistema, especialmente em situações de alta demanda, foi um desafio que exigiu uma série de testes e depurações;

- **Gerenciamento de Múltiplos Pacotes e Classes:** Foi necessário manter uma estrutura organizada e garantir a comunicação eficiente entre as diferentes partes do sistema.

4.3 Considerações Finais

Este projeto proporcionou uma oportunidade valiosa para aprofundar nosso entendimento da linguagem de programação Java e do paradigma de Orientação a Objetos. A aplicação prática desses conceitos nos desafiou a pensar de forma mais abstrata e estruturada, o que se traduziu em um código mais modular e flexível. Aprender a projetar sistemas orientados a objetos e aplicar herança, encapsulamento e polimorfismo foi fundamental para o sucesso do projeto.

Em retrospectiva, a jornada de desenvolvimento nos ensinou a importância da resolução de problemas complexos, do trabalho em equipe e da adaptabilidade. À medida que avançamos em nossa carreira na programação, os princípios adquiridos nesse projeto continuarão a influenciar positivamente nosso trabalho.