Sistema de estoque para Farmácias - Extorage

Francis Tsigbey N., Joshua Kook Ho P., Luigi Muller S. Linhares, Tarlison S. L. Brito

1Bacharelado em Ciência da Computação – Universidade Federal de Roraima (UFRR)  
Boa Vista, Roraima – RR – Brasil

sluigimuller@gmail.com, britotarlison@gmail.com, tsigbeyfrancis@gmail.com, inpincible@hotmail.com

**Resumo.** Este resumo irá mostrar o software desenvolvido para a disciplina de programação orientada a objetos. Este software é para o gerenciamento de um sistema de estoque de uma farmácia, sendo desenvolvido em Java e com os paradigmas da programação orientada a objetos nele é possível adicionar, remover, produtos, adicionar quem são os fornecedores além de gerar relatórios das operações que ocorrem com os produtos.

# 1. Informações gerais

O sistema, Extorage, servirá de grande ajuda para gerenciamento de estoques de farmácias, pois o sistema sempre gera um relatório automático a cada movimentação que ocorre com os produtos, seja de entrada ou saída, além de ser possível acessa-lo depois para verificar cada movimentação que ocorreu no estoque.

Com isso há uma segurança maior no gerenciamento e evita qualquer tipo de problema além de fácil manuseamento dos produtos sendo separados em categorias, alimentos, medicamentos e produtos de higiene.

# 2. Funcionalidades

Extorage é um software Java desenvolvido para farmácias que precisam de fácil gerenciamento nos produtos do seu estoque. O sistema é capaz de adicionar novos produtos e assim que inseridos são categorizados em Medicamentos, Alimentos e produtos de Higiene, facilitando assim a organização dos produtos.

A cada operação realizada no estoque seja ela de credenciamento do produto, reposição de estoque, e as retiradas de produtos, é gerado um relatório dessas operações para que que haja um controle maior sobre o que está acontecendo no estoque da empresa.

É possível também cadastrar os fornecedores, assim a empresa fica também a par em relação a quem fornece os produtos e quais são eles.

Agora em questão ao desenvolvimento da aplicação, ele foi desenvolvido na linguagem Java e usado as IDE’s NetBeans e Eclipse. A codificação foi baseada no paradigma da programação orientada a objetos.

Semanalmente foi apresentado sprints para ser avaliado como estava o andamento do software, nele apresentamos a ideia e o que seria feito para que o software fosse considerado pronto, os requisitos funcionais.

Como requisitos funcionais, foi apresentado cinco. O primeiro foi ‘Registrar a entrada de produtos’ que funciona para receber produtos de um fornecedor e colocá-los no sistema e sempre sendo possível obter o quanto do produto está no estoque e este requisito foi cumprido, pois tanto em linha de comando quanto na interface gráfica temos a opção de cadastrar produto e essa opção faz com que o produto seja cadastrado, armazenado e ainda é gerado um relatório automático desta operação salvando a quantidade que entrou no estoque.

Há também o requisito ‘Alterar o estoque’ que permite ao usuário administrador do sistema a alteração dos dados relativos ao estoque de produtos (Quantidade, descrição, etc) que também foi cumprido sendo possível fazer a alteração de quantidade, repor ou retirar do estoque, além de ser possível a alteração de nome, preço e quantidade mínima do produto.

Já o terceiro requisito, trata-se do ‘Registro de saída de produtos’ que se trata da retirada de uma quantidade especificada pelo usuário e sempre atento para verificar se a quantidade solicitada seja menor que a quantidade existente, evitando que seja retirada uma quantidade que não existe, outro requisito também cumprido já que qualquer alteração de retirada que seja maior que a quantidade de produtos existentes irá mostrar um erro, pois não é possível retirar aquilo que não se tem no estoque.

O penúltimo requisito funcional, ‘Geração de um relatório’ também é atendido, pois, cada alteração no estoque gera um relatório daquela operação contendo a data e hora em que ocorreu aquilo, seja na entrada ou retirada do estoque, além de ser possível conferir esses relatórios depois.

E por fim o ultimo requisito funcional especificado e que foi cumprido é o ‘Gerenciar Fornecedor’ que é possível fazer sem problemas no sistema é pedido os dados do fornecedor como razão social, CNPJ e qual o produto que ele está fornecendo a empresa.

Mas ainda há também os requisitos não funcionais que apesar de não serem, como já diz o nome, necessários para o funcionamento do software todos eles foram cumpridos. Como a interface gráfica que foi feita para o usuário poder mexer no sistema com mais facilidade do que seria por linha de comando já que visualmente fica tudo mais fácil e acessível. A interface foi feita com a biblioteca do javax.swing e otimizada para que o uso ficasse intuitivo dependendo de quais operações iam ser realizadas ali.

Temos também o requisito não funcional ‘Salvar os arquivos de dados’ que seria manter os arquivos salvos por meio de serialização ou banco de dados para firmar uma segurança ao dono de ter algo como backup do que aconteceu ali, e foi feito a serialização para isso e quando o software é fechado e reaberto os dados continuam lá. Na codificação vimos o que era possível e o que não era possível de ser feito e uma dessas coisas foi usar banco de dados. Como a equipe não tinha experiencia com banco de dados a saída foi usar a serialização.

E apenas um requisito não funcional não foi cumprido, o ‘Aviso de estoque baixo’ não foi feito, pois, a equipe não achou nada necessário já que quando é feito o gerenciamento do estoque é possível ver a quantidade mínima exigida para o produto e quantos existem ainda daquele determinado produto, além de que há uma mensagem de alerta avisando se foi tentado retirar mais produtos do estoque do que estão disponíveis lá.

Em relação a codificação nós temos as classes Dados, Produtos, Fornecedor, Relatório, Entrada, Saída sendo as classes Entrada e Saída classes que herdam de relatório que é uma classe abstrata (usando herança nesse ponto, pois duas classes estão herdando da classe Relatório). Tudo isso no pacote sistemaDeEstoque.

A classe Dados é usada para controlar os dados que irão ser usados pela interface gráfica, a interface importa a classe e a usa para manipular os dados em relação aos produtos, fornecedores e relatórios, além de possuir métodos de serialização para cada um desses dados já que isso será usado no pacote das telas do software.

Já a classe Produtos usado para representar o produto armazenado no estoque, e os seus atributos como ID, validade, lote, quantidade, nome, preço e categoria. Possui um método para cadastro desses produtos em que é possível preencher quase todas as informações sobre o produto, os outros métodos separados são getters e setters para mostrar os dados como o atributo ‘categoria’ e alterar esses dados, usando assim o conceito de encapsulamento (vale ressaltar que os atributos estão declarados como private) deixando que seja possível alterar os atributos das classes apenas por meio de métodos e assim ‘escondendo’ do usuário o que ocorre ali, impedindo que mudanças fora do esperado.

Nesta classe também há um Override do método toString() para que quando fosse necessário imprimir as informações do produto bastasse apenas usar System.out.println do produto para facilitar as coisas e ter reaproveitamento de código.

A classe Relatorio é uma classe abstrata que servirá de classe mãe para as classes Entrada e Saida, estas que darão Override no método gerar relatório já que cada uma das classes tem um jeito especifico de lidar com a criação do relatório (obviamente já que um é de Entrada e o outro de Saida, mas não deixam de ser um relatório).

E com isso foi possível a utilização de polimorfismo, já que podemos usar por exemplo, em linha de código, “**Relatorio relatorioNovo = new Entrada()**” e utilizar isso ao nosso favor para fazer métodos que utilizem parâmetros do tipo Relatorio sem nos preocupar se é de Entrada ou Saida já que funcionara para ambos e assim teremos um reaproveitamento de código também, já que não foi necessário fazer métodos pra cada tipo de Relatorio.

E a classe Fornecedor está encapsulada com os atributos e os métodos do fornecedor e para seu gerenciamento. Já a classe principal, a classe SistemaDeEstoque é a classe em que as outras classes são instanciadas e a lógica do sistema é criada para que tudo seja feito como foi pedido, incluindo serialização e deserialização dos objetos.

Em suma o software está seguindo os paradigmas da programação orientada a objetos e funcionando de acordo com os requisitos funcionais, foi um desenvolvimento cheio de obstáculos, mas com um grande êxito no final e funcionalidade total.

# 3. Design Patterns

O sistema usa o seguinte design pattern:

**3.1 Iterator**

Problema: temos que percorrer os elementos de um objeto sequencialmente sem expor sua representação.

Solução: usamos uma estrutura de Java para percorrer sequencialmente os elementos de um objeto da classe ArrayList. Na imagem abaixo especificamos a classe do elemento (Produto), uma variável para armazená-lo pelo loop (prod) e o objeto que contém os elementos (Dados.dados).

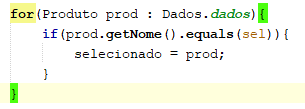


Figura 1. Estrutura do for

Consequência: A interface é simplificada, facilitando o percurso pelo objeto. A forma como o objeto é percorrido pode ser alterado, nesse exemplo a iteração é sequencial, suficiente para o que desejamos.

**Referências**

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (2000) “Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos”, Editora Bookman.