Trabalho Prático 2: Módulo de vendas

Professor: Saulo Henrique Cabral Silva

Alunes: Paloma Tavares e Rebeka Góes

Curso: Técnico Integrado em Informática - 3° ano

Disciplina: Tópicos em Desenvolvimento de Software

Introdução

A proposta do projeto consiste na criação de um software com interface gráfica para gerenciar a venda dos produtos ofertados pela Mercearia do Seu José. Com o objetivo de rever conceitos básicos de programação e foco na manipulação de tabelas, bem como a prática de programação utilizando a API Swing.

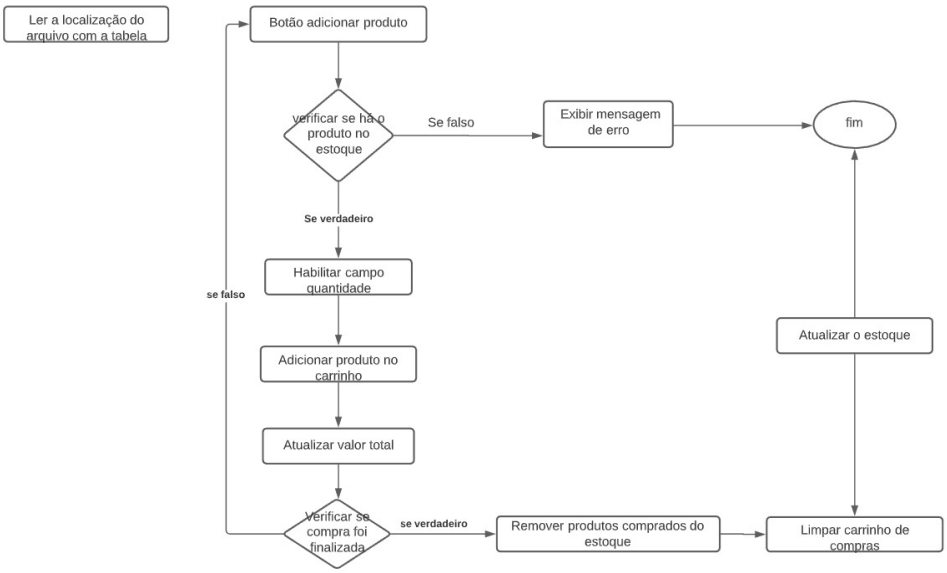
A Mercearia do Seu José

Seu José resolveu abrir uma mercearia para suprir as necessidades dos moradores do seu bairro na cidade de Ouro Branco, num cenário pós Covid-19. Inicialmente ele anotou numa planilha Excel todos os produtos que ele vendia, no entanto, com o passar dos dias, ele percebeu que a consulta dos produtos na tabela Excel, estava levando mais tempo do que ele havia planejado, e concluir uma pequena compra dos clientes estava demandando muito tempo e esforço.

Descrição da situação

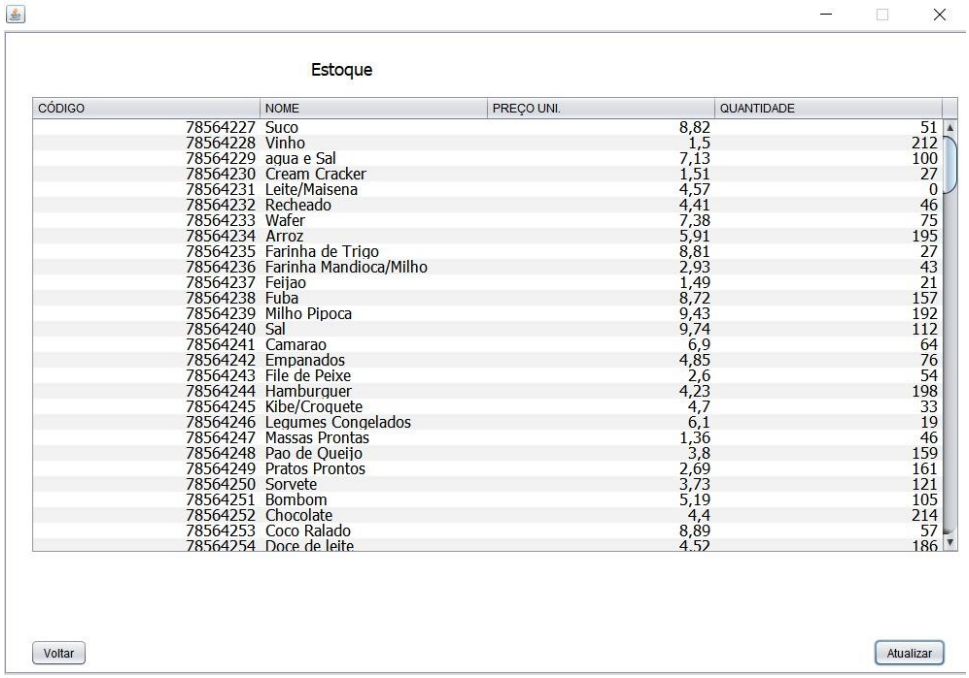
Neste programa, iremos receber o arquivo com a planilha de produtos de Seu José e, em seguida, ofereceremos para o operador do caixa um software com interface gráfica para gerenciar a venda dos produtos ofertados pela Mercearia do Seu José.

Interdependência dos Módulos



Interface gráfica

Esta é a tela principal, ela possui 4 campos de texto, “Código”, “Nome”, “Quantidade” e “Preço unitário” para adicionar o produto desejado, além de uma tabela com todos os produtos já inseridos na compra. O botão “Remover” retira da tabela da compra o produto desejado, caso o cliente desista de comprar. O botão “Finalizar” tem a funcionalidade de finalizar a compra, somando todos os itens. O botão “Estoque” abre a tela que possui uma tabela com todos os produtos existentes na Mercearia.



Esta é tela que possui a tabela de estoque, nela estão as informações “Código”, “Nome”, “Preço unitário” e “Quantidade” de todos os produtos existentes na Mercearia. As colunas “Código” e “Nome” são bloqueadas por senha, mas as colunas “Preço unitário” e “Quantidade” podem ser modificadas pelo operador do caixa por meio do botão “Atualizar”. O botão “Voltar” leva o usuário de volta à tela principal.

Código Fonte

Na classe “Produto” o usuário informa as informações do produto, para que ele seja adicionado na compra.

```
package vendasInterface;


public class Produto {
    private int codigo;
    private String nome;
    private double preco;
    private int quantidade;

    public Produto(int codigo, String nome, double preco, int quantidade) {
        this.codigo = codigo;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
    }

    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }


    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    @Override
    public String toString() {
        return this.codigo+";"+this.nome+";"+this.preco+";"+this.quantidade;
    }

}
```

Na classe “ModeloTabelaCompra”, os produtos são adicionados na tabela da compra total; é calculado o valor parcial e total da compra; pode ser deletado um produto não mais desejado na compra; e ocorre a verificação de senha para alteração das colunas “Nome” e “Preço unitário” na tabela de estoque.

```
package vendasInterface;

import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.AbstractTableModel;


public class ModeloTabelaCompra extends AbstractTableModel{

    private Vector<Produto> carrinhoCompra;
    private PainelCompraGUI painel;

    public ModeloTabelaCompra(PainelCompraGUI painel){
        this.carrinhoCompra = new Vector<>();
        this.painel = painel;
    }

    @Override
    public int getRowCount() {
        return carrinhoCompra.size();
    }

    @Override
    public int getColumnCount() {
        return 4;
    }

    @Override
    public Object getValueAt(int linha, int coluna) {
        Produto temp = carrinhoCompra.get(linha);

        switch(coluna){
            case 0: return temp.getNome();
            case 1: return temp.getPreco();
            case 2: return temp.getQuantidade();
            case 3: return temp.getQuantidade() * temp.getPreco();
            default :return null;
        }
    }

    public void addNovoProduto(Produto vendido){
        this.carrinhoCompra.add(vendido);
    }

    public void removeProdutoCarrinho(int indice){
        this.carrinhoCompra.remove(indice);

    }


    @Override
    public String getColumnName(int coluna){
        switch(coluna){
            case 0: return"NOME";
            case 1: return"PREÇO UNI.";
            case 2: return"QUANTIDADE";
            case 3: return"PREÇO PARC";
            default: return null;
        }

    }

    @Override
    public boolean isCellEditable(int linha, int coluna) {
        if(coluna == 2){
            return true;
        }else{
            return false;
        }
    }
```

```java
@Override
public void setValueAt(Object novoValor, int linha, int coluna) {
    if(coluna ==2){
        //requisitando senha do gerente para permitir a modicação de quantidades
        String senha = JOptionPane.showInputDialog(null, "Informe a senha do gerente",
            "Operação restrita", JOptionPane.INFORMATION_MESSAGE);

        if(senha != null && senha.equals("ifmg")){
           carrinhoCompra.get(linha).setQuantidade((int)novoValor);

        // a tabela e o valor são atualizados
        this.painel.atualizaQuantidades();
        }
    }
}

@Override
public Class<?> getColumnClass(int coluna) {
    switch(coluna){
        case 0: return String.class ;
        case 1: return Double.class;
        case 2: return Integer.class;
        case 3: return Double.class;
        default: return null;
    }
}
```

```java
public double calculaPrecoParcialCompra(){
    double valor = 0.0;
    //recalculando o valor do carrinho com os atuais produtos
    for(Produto p : carrinhoCompra){
        valor += p.getQuantidade() * p.getPreco();
    }
    return valor;

}
 public Vector<Produto> produtosCarrinho() {
    return carrinhoCompra;
}

public void limpaCarrinho(){
    this.carrinhoCompra.clear();
}


}
```

Na classe “ModeloTabelaEstoque” ocorre o retorno das informações dos produtos para a tabela “Estoque” que é apresentada para o usuário, além da adição e exclusão de produtos na mesma.

```java
package vendasInterface;

import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.AbstractTableModel;


public class ModeloTabelaEstoque extends AbstractTableModel{
    private Vector<Produto> produtos;
    private EstoqueGUI painel;

    public ModeloTabelaEstoque(EstoqueGUI painel){
        this.produtos = new Vector<>();
        this.painel = painel;
    }

    @Override
    public int getRowCount() {
        return produtos.size();
    }

    @Override
    public int getColumnCount() {
        return 4;
    }

    @Override
    public Object getValueAt(int linha, int coluna) {
        Produto temp = produtos.get(linha);

        switch(coluna){
            case 0: return temp.getCodigo();
            case 1: return temp.getNome();
            case 2: return temp.getPreco();
            case 3: return temp.getQuantidade();
            default :return null;
        }
    }

    public void limpaEstoque(){
        this.produtos.clear();
    }

}
```

```java
public void addNovoProduto(Produto novo){
    this.produtos.add(novo);
}


@Override
public String getColumnName(int coluna){
    switch(coluna){
        case 0: return"CÓDIGO";
        case 1: return"NOME";
        case 2: return"PREÇO UNI.";
        case 3: return"QUANTIDADE";
        default: return null;
    }

}


@Override
public Class<?> getColumnClass(int coluna) {
    switch(coluna){
        case 0: return Integer.class;
        case 1: return String.class ;
        case 2: return Double.class;
        case 3: return Integer.class;
        case 4: return Double.class;
        default: return null;
    }
}


public Vector<Produto> produtos() {
    for(Produto prod1 : produtos){
        produtos().add(prod1);
    }
    return produtos;
}
```

Na classe “FakeBD” é importada a tabela do Seu José, banco de dados, para o programa; é efetuada a leitura dos produtos e atualização da tabela Excel.

```java
package vendasInterface;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;


public class FakeBD extends FramePrincipal{

    private static Vector<Produto>produtos;
    private static String localArquivo(){
    String localArquivo =FramePrincipal.perguntaLocalTabela();
    return localArquivo;
    }
    String local;
```

```java
// leitura das informacoes do arquivo excel
private static void cargaArquivo(){
    // ajuste na criacao de vetor de produtos static
    if(produtos == null){
        produtos = new Vector<>();
    }else{
        produtos.clear();
    }


    try {

        File arquivoCsv = new File(localArquivo()) ;
        FileReader marcaLeitura = new FileReader(arquivoCsv);
        BufferedReader bufLeitura =  new BufferedReader(marcaLeitura);

        //ler cada linha
        //le primeira linha (cabecalho)---descartar
        bufLeitura.readLine();
        //le primeiro produto
        String linha = bufLeitura.readLine();

        while(linha != null){
            //linhas seguintes ate o fim do arquivo
            // separar as informacoes
            String infos[] = linha.split(";");
            int cod = Integer.parseInt(infos[0]);
            String nome = infos[1];
            double preco = Double.parseDouble(infos[2]);
            int quant = Integer.parseInt(infos[3]);

            //adicao dos produtos em um vetor dinamico
            produtos.add(new Produto(cod, nome, preco, quant));
            //le a proxima linha
            linha = bufLeitura.readLine();

        }
        // libera o arquivo para outros processos
        bufLeitura.close();
```

```java
        } catch (FileNotFoundException ex) {
            System.out.println("Arquivo não existe");
        }catch(IOException e){
            System.err.println("Arquivo corrompido");
        }
    }

    public static Produto consultaProdutoCod(int cod){
        //carrega o arquivo caso ele nao tenha sido carregado antes
        if(produtos == null){
            cargaArquivo();
        }
        for(Produto prod1 : produtos){
           if(prod1.getCodigo() == cod){
               return prod1;
           }
        }
        //nao existe produto com o codigo especificado no parametro
        return null;
    }

     public static Vector<Produto> estoque(){
        Vector<Produto> prod = new Vector<>();
        if(produtos == null){
            cargaArquivo();
        }
        for(Produto prod1 : produtos){
           prod.add(prod1);
         }
         return prod;
     }
```

```java
    public static void atualizaArquivo(){

        File arquivo = new File(localArquivo());

        try{
           FileWriter escritor  = new FileWriter(arquivo);

           BufferedWriter bufEscrita = new BufferedWriter(escritor);

           for(int i = 0; i< produtos.size(); i++){
               bufEscrita.write(produtos.get(i)+"\n");
           }

           //bufEscrita.flush();//descarrega arquivo
           bufEscrita.close();
        }catch(IOException ex){
            System.err.println("dispositivo com falha");
        }
    }

}
```

Problemas Encontrados

Tivemos problemas relacionados à localização do arquivo, que estava sendo pedida 3 vezes pelo programa. Também tivemos problemas relacionados ao estoque, que não estava sendo exibido na tela.

Conclusão

A atividade foi útil para relembrar aspectos sobre a utilização da biblioteca Java Swing entre outras ferramentas do Java, além do aprendizado com relação à manipulação de tabelas, e a ampliação dos conhecimentos gerais em Java e relacionadas à banco de dados.

Referências bibliográficas

https://sauloifmg.com.br/roteirotp.pdf

http://sauloifmg.com.br/topicosSoftware.html