

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Algoritmos e Estrutura de Dados 1

Projeto Final - Grupo 9

Professor: Roberto Ferrari Junior

Autores

Matheus Goulart Ranzani - BCC 021 - 800278

Rodrigo Pavão Coffani Nunes - BCC 021 - 800345

Vinícius Matheus Romualdo Santos - BCC 021 - 801258

São Carlos, 23 abril de 2022

Sumário

Sumário	2
1 Equipe	3
2 Nome do projeto	3
3 Aplicação	3
4 Imagens da execução	3
5 Teaser	6
6 Explicação sobre como desenvolver um projeto como este	6
7 Link para executável	6
8 Conclusões	6
9 Autorizações	6
10 Código Fonte	7
10.1 Estoque.hpp	7
10.2 frmMenu.h	10

1 Equipe

1. Matheus Goulart Ranzani - BCC 021 - 800278
2. Rodrigo Pavão Coffani Nunes - BCC 021 - 800345
3. Vinícius Matheus Romualdo Santos - BCC 021 - 801258

2 Nome do projeto

O nome do projeto é: Controle de Estoque - Supermercado.

3 Aplicação

Este projeto utiliza o conceito de Lista Cadastral em sua construção. A aplicação tem como objetivo ser um sistema para supermercados, focado no controle de estoque.

Através da aplicação, é possível cadastrar, alterar a quantidade, remover e exibir informações sobre produtos. Além de também ser possível exibir o estoque completo.

4 Imagens do projeto executando

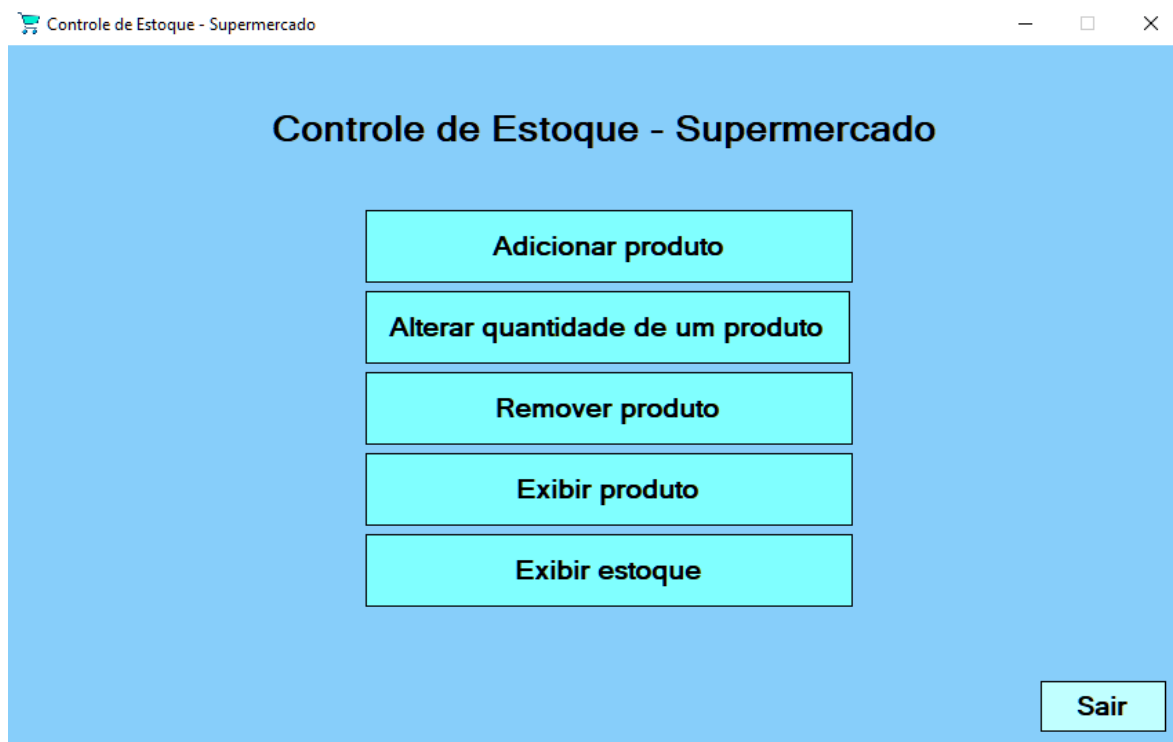


Imagem 1: Menu

Controle de Estoque - Supermercado

Adicionar produto

Nome:

Preço:

Quantidade:

Imagem 2: Tela de cadastramento de produtos

Controle de Estoque - Supermercado

Adicionar produto

Nome:

Preço:

Quantidade:

Controle de Estoque - Supermercado


 Quantidade de produto inválida!

Imagem 3: Erro apresentado ao inserir um número inválido

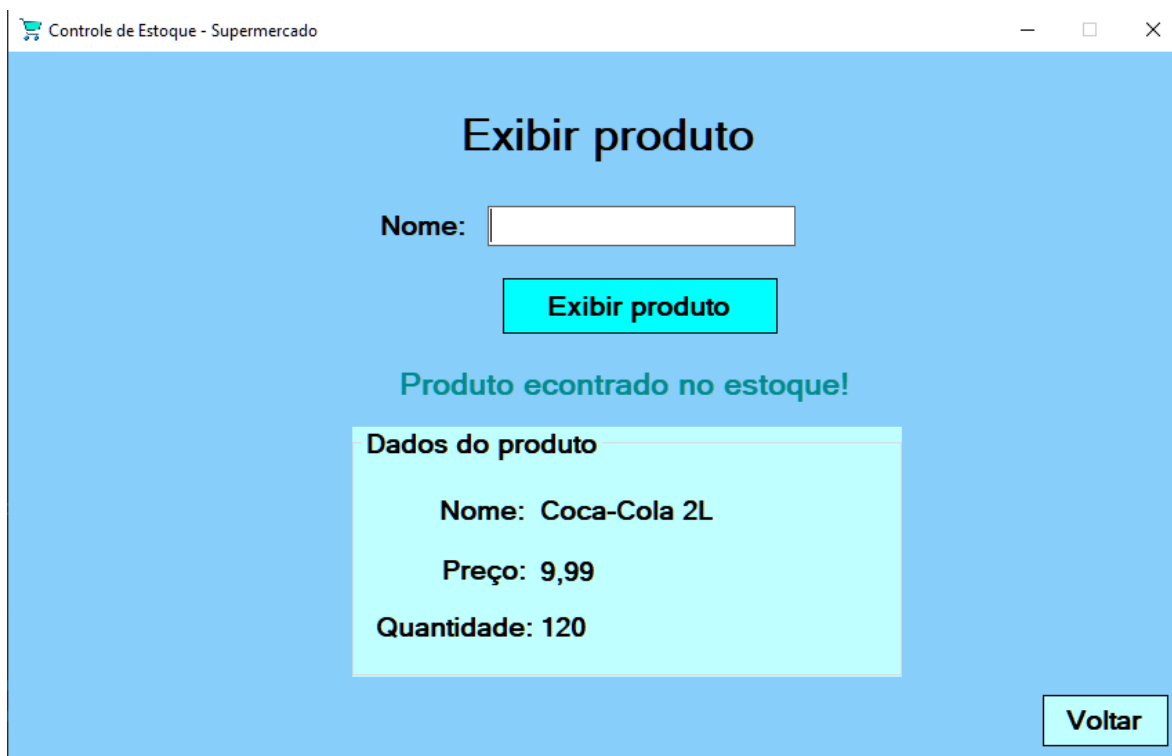


Imagem 4: Tela de exibição de informações sobre um produto

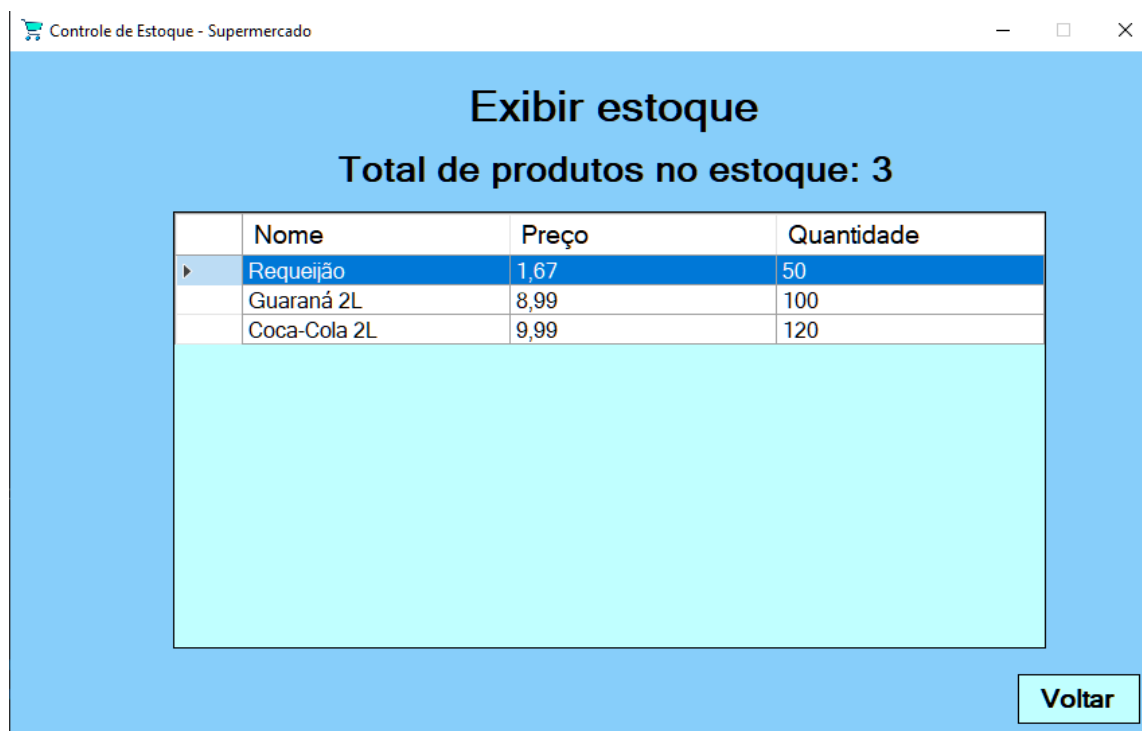


Imagem 5: Tela “Exibir estoque”



Imagem 6: Mensagem exibida ao tentar remover um produto inexistente.

5 Teaser

Teaser da aplicação com 54s: <https://youtu.be/0RbxkJtI94>.

6 Explicação sobre como desenvolver um projeto como este

No desenvolvimento do projeto foi utilizado conhecimentos de C++ e a estrutura de interface de usuário chamada *Windows Form*, integrado com o Visual Studio 2022 da Microsoft. Para o estoque foi utilizado um TAD (Tipo Abstrado de Dado) Lista Cadastral, implementado de forma encadeada, dinâmica e sem repetição de elementos. Essa lista foi escolhida pois permite um melhor gerenciamento de itens. Nesse gerenciamento, foram utilizadas as operações típicas de uma Lista Cadastral, entre elas: adição e remoção de itens em qualquer lugar da lista e busca por nome (chave-valor) .

Para desenvolver um projeto como esse é necessário conhecer as características de uma Lista Cadastral e também possuir um bom entendimento de como manipular ponteiros em C++. Além disso, para criar a interface gráfica, é preciso ter conhecimentos básicos sobre Orientação a Objetos em C++ e aprender a usar a IDE (Integrated Development Environment ou Ambiente) Visual Studio 2022 para conseguir manipular os componentes gráficos da interface de forma correta.

7 Link para executável

O download do executável pode ser encontrado no repositório do projeto, disponível no [link](#).

8 Conclusões

Por fim, podemos concluir e entender a aplicação de uma Estrutura de Dados em casos reais. No nosso programa, a utilização de um modelo de Lista Cadastral nos proporcionou facilidade para manipular uma lista que não depende de ordem, não permite itens repetidos e que é necessário acessar itens através de valores relacionados.

9 Autorizações

Autorizamos a divulgação dos seguintes componentes do projeto, para fins acadêmicos (competições, portfólio, materiais didáticos ou similares), dado que citada a autoria:

	Autoriza a divulgação?	
	Sim	Não
Nomes dos membros da equipe e título do projeto	X	
Imagem do projeto	X	
Teaser (vídeo curto mostrando a execução do projeto)	X	
Documentação (indicar no documento a licença de uso aplicável a seu projeto, se desejado)	X	
Código	X	
Executável	X	
Contatos e Fotos dos membros da equipe		X

10 Código Fonte

O código fonte completo do projeto está disponível no GitHub através do link: <https://github.com/matheusranzani/ProjetoAED1>.

Os itens a seguir contém trechos do código fonte do projeto.

10.1 Estoque.hpp

Segue o código fonte da implementação da Lista Cadastral com suas operações primitivas:

```
#pragma once

#include <iostream>

typedef struct Produto {
    std::string nome;
    double preco;
    int quantidade;
    struct Produto* proximo;
} Produto;

typedef Produto* ProdutoPtr;

typedef struct Estoque {
    ProdutoPtr primeiro;
    ProdutoPtr ultimo;
} Estoque;

void cria(Estoque* e) {
    e->primeiro = NULL;
    e->ultimo = NULL;
}

bool vazio(Estoque* e) {
    return e->primeiro == NULL;
}

bool esta_no_estoque(Estoque* e, std::string nome) {
    ProdutoPtr p_aux = e->primeiro;

    while (p_aux != NULL) {
        if (p_aux->nome == nome) {
            return true;
        }

        p_aux = p_aux->proximo;
    }

    return false;
}
```



```

ProdutoPtr retorna_produto(Estoque* e, std::string nome, bool* ok) {
    ProdutoPtr p_aux = e->primeiro;

    while (p_aux != NULL) {
        if (p_aux->nome == nome) {
            *ok = true;
            return p_aux;
        }

        p_aux = p_aux->proximo;
    }

    *ok = false;
    return NULL;
}

void insere(Estoque* e, Produto p, bool* ok) {
    if (p.preco < .0 || p.quantidade <= 0) {
        *ok = false;
        return;
    }

    ProdutoPtr p_novo = new Produto;

    p_novo->nome = p.nome;
    p_novo->preco = p.preco;
    p_novo->quantidade = p.quantidade;
    p_novo->proximo = NULL;

    if (vazio(e)) {
        e->primeiro = p_novo;
    } else {
        if (esta_no_estoque(e, p_novo->nome)) {
            *ok = false;
            return;
        }

        e->ultimo->proximo = p_novo;
    }

    e->ultimo = p_novo;
    *ok = true;
}

```

```

void retina(Estoque* e, std::string nome, bool* ok) {
    if (!esta_no_estoque(e, nome)) {
        *ok = false;
        return;
    }

    ProdutoPtr p = retorna_produto(e, nome, ok);
    ProdutoPtr p_aux = e->primeiro;
    ProdutoPtr anterior = NULL;

    while (p_aux != NULL) {
        if (p_aux->nome == p->nome) {
            if (anterior == NULL) { // Começo da lista
                e->primeiro = e->primeiro->proximo;
                delete p_aux;
            } else if (p_aux->proximo == NULL) { // Fim da lista
                e->ultimo = anterior;
                anterior->proximo = NULL;
                delete p_aux;
            } else { // Meio da lista
                anterior->proximo = p_aux->proximo;
                delete p_aux;
            }

            *ok = 1;
            return;
        } else {
            anterior = p_aux;
            p_aux = p_aux->proximo;
        }
    }

    *ok = false;
}

void altera_quantidade(Estoque* e, std::string nome, int quantidade, bool* ok) {
    if (!esta_no_estoque(e, nome) || quantidade < 0) {
        *ok = false;
        return;
    }

    ProdutoPtr p_aux = e->primeiro;

```

```

while (p_aux != NULL && p_aux->nome != nome) {
    p_aux = p_aux->proximo;
}

if (p_aux != NULL) {
    p_aux->quantidade = quantidade;
}

*ok = true;
}

#pragma once

```

10.2 frmMenu.h

Segue trecho da implementação das funcionalidades da Lista Cadastral na interface gráfica desenvolvida no Visual Studio 2022:

```

#pragma once

#include "Estoque.hpp"

#include <msclr\marshal_cppstd.h>

namespace ProjetoAED1 {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    Estoque estoque;

    public ref class frmMenu : public System::Windows::Forms::Form {
    public:
        frmMenu(void) {
            InitializeComponent();

            pnlAdicionar->Hide();
            pnlAlterarQuantidade->Hide();
            pnlRemover->Hide();
        }
    };
}

```

```

        pnlExibirProduto->Hide();
        gpbExibirProduto->Visible = false;
        pnlExibirEstoque->Hide();
        dgvEstoque->Visible = false;

        cria(&estoque);
    }

protected:
    ~frmMenu() {
        if (components) {
            delete components;
        }
    }

private:
    System::ComponentModel::Container ^components;

    #pragma region Componentes...
    #pragma region Windows Form Designer generated code...

private: System::Void frmMenu_Load(System::Object^ sender, System::EventArgs^ e) {
    this->ActiveControl = lblEstoque;
}

private: void habilitaBotoesMenu(bool habilita) {
    if (habilita) {
        btnAdicionar->Enabled = true;
        btnAlterarQuantidade->Enabled = true;
        btnRemover->Enabled = true;
        btnExibirProduto->Enabled = true;
        btnExibirEstoque->Enabled = true;
        btnSair->Enabled = true;
    } else {
        btnAdicionar->Enabled = false;
        btnAlterarQuantidade->Enabled = false;
        btnRemover->Enabled = false;
        btnExibirProduto->Enabled = false;
        btnExibirEstoque->Enabled = false;
        btnSair->Enabled = false;
    }
}
}

```

```

        private: System::Void btnAdicionar_Click(System::Object^ sender,
System::EventArgs^ e) {
            habilitaBotoesMenu(false);

            pnlAdicionar->Dock = DockStyle::Fill;
            pnlAdicionar->Show();

            txtNome->Focus();
        }

        private: void limpaAdicionar() {
            txtNome->Text = "";
            txtPreco->Text = "";
            txtQuantidade->Text = "";
        }

        private: System::Void btnAdicionarPanel_Click(System::Object^ sender,
System::EventArgs^ e) {
            if (String::IsNullOrEmpty(txtNome->Text)) {
                MessageBox::Show("O nome do produto precisa ser informado!",
"Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
                txtNome->Focus();
                return;
            }

            if (String::IsNullOrEmpty(txtPreco->Text)) {
                MessageBox::Show("O preço do produto precisa ser informado!",
"Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
                txtPreco->Focus();
                return;
            }

            if (String::IsNullOrEmpty(txtQuantidade->Text)) {
                MessageBox::Show("A quantidade do produto precisa ser informada!",
"Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
                txtQuantidade->Focus();
                return;
            }

            Produto produto;

```

```

        double preco;
        int quantidade;
        bool ok;

        if (Double::TryParse(txtPreco->Text, preco)) {
            produto.preco = preco;
        } else {
            MessageBox::Show("Preço de produto inválido!", "Controle de
Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
            txtPreco->Focus();
            return;
        }

        if (INT::TryParse(txtQuantidade->Text, quantidade)) {
            produto.quantidade = quantidade;
        } else {
            MessageBox::Show("Quantidade de produto inválida!", "Controle de
Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
            txtQuantidade->Focus();
            return;
        }

        msclr::interop::marshal_context context;
        produto.nome = context.marshal_as<std::string>(txtNome->Text);

        insere(&estoque, produto, &ok);

        lblResultado->Visible = true;

        if (ok) {
            limpaAdicionar();
            txtNome->Focus();
            lblResultado->ForeColor = System::Drawing::Color::DarkCyan;
            lblResultado->Text = "Produto adicionado com sucesso!";
        } else {
            limpaAdicionar();
            txtNome->Focus();
            lblResultado->ForeColor = System::Drawing::Color::Magenta;
            lblResultado->Text = "Falha ao adicionar o produto";
        }
    }
}

```

```

        private: System::Void btnAdicionarSair_Click(System::Object^ sender,
System::EventArgs^ e) {
            habilitaBotoesMenu(true);
            pnlAdicionar->Hide();
            limpaAdicionar();
            lblResultado->Visible = false;

            this->Show();
        }

        private: System::Void btnAlterarQuantidade_Click(System::Object^ sender,
System::EventArgs^ e) {
            habilitaBotoesMenu(false);

            pnlAlterarQuantidade->Dock = DockStyle::Fill;
            pnlAlterarQuantidade->Show();

            txtNomeAlterar->Focus();
        }

        private: System::Void btnAlterarPanel_Click(System::Object^ sender,
System::EventArgs^ e) {
            if (String::IsNullOrWhiteSpace(txtNomeAlterar->Text)) {
                MessageBox::Show("O nome do produto precisa ser informado!",
"Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
                txtNomeAlterar->Focus();
                return;
            }

            if (String::IsNullOrWhiteSpace(txtQuantidadeAlterar->Text)) {
                MessageBox::Show("A quantidade a ser alterada precisa ser
informada!", "Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
                txtQuantidadeAlterar->Focus();
                return;
            }

            int quantidade;

            if (!INT::TryParse(txtQuantidadeAlterar->Text, quantidade)) {

```

```

        MessageBox::Show("Quantidade de produto inválida!", "Controle de
Estoque - Supermercado",
        MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
        txtQuantidadeAlterar->Focus();
        return;
    }

    mscclr::interop::marshal_context context;
    std::string nome =
context.marshal_as<std::string>(txtNomeAlterar->Text);
    bool ok;

    altera_quantidade(&estoque, nome, quantidade, &ok);

    lblResultadoQuantidade->Visible = true;

    if (ok) {
        txtNomeAlterar->Text = "";
        txtQuantidadeAlterar->Text = "";
        txtNomeAlterar->Focus();
        lblResultadoQuantidade->ForeColor =
System::Drawing::Color::DarkCyan;
        lblResultadoQuantidade->Text = "Quantidade alterada com sucesso!";
    } else {
        txtNomeAlterar->Text = "";
        txtQuantidadeAlterar->Text = "";
        txtNomeAlterar->Focus();
        lblResultadoQuantidade->ForeColor =
System::Drawing::Color::Magenta;
        lblResultadoQuantidade->Text = "Falha ao alterar a quantidade";
    }
}

private: System::Void btnAlterarQuantidadeSair_Click(System::Object^
sender, System::EventArgs^ e) {
    habilitaBotoesMenu(true);
    pnlAlterarQuantidade->Hide();

    txtNomeAlterar->Text = "";
    txtQuantidadeAlterar->Text = "";
    lblResultadoQuantidade->Visible = false;

    this->Show();
}

```



```

    }

    private: System::Void btnRemover_Click(System::Object^ sender,
System::EventArgs^ e) {
        habilitaBotoesMenu(false);

        pnlRemover->Dock = DockStyle::Fill;
        pnlRemover->Show();

        txtNomeRemover->Focus();
    }

    private: System::Void btnRemoverPanel_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (String::IsNullOrWhiteSpace(txtNomeRemover->Text)) {
            MessageBox::Show("O nome do produto precisa ser informado!",
"Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
            txtNomeRemover->Focus();
            return;
        }

        msclr::interop::marshal_context context;
        std::string nome =
context.marshal_as<std::string>(txtNomeRemover->Text);
        bool ok;

        retira(&estoque, nome, &ok);

        lblResultadoRemover->Visible = true;

        if (ok) {
            txtNomeRemover->Text = "";
            txtNomeRemover->Focus();
            lblResultadoRemover->ForeColor = System::Drawing::Color::DarkCyan;
            lblResultadoRemover->Text = "Produto removido com sucesso!";
        } else {
            txtNomeRemover->Text = "";
            txtNomeRemover->Focus();
            lblResultadoRemover->ForeColor = System::Drawing::Color::Magenta;
            lblResultadoRemover->Text = "Falha ao remover o produto";
        }
    }
}

```

```

        private: System::Void btnRemoverSair_Click(System::Object^ sender,
System::EventArgs^ e) {
            habilitaBotoesMenu(true);
            pnlRemover->Hide();

            txtNomeRemover->Text = "";
            lblResultadoRemover->Visible = false;

            this->Show();
        }

        private: System::Void btnExibirProduto_Click(System::Object^ sender,
System::EventArgs^ e) {
            habilitaBotoesMenu(false);

            pnlExibirProduto->Dock = DockStyle::Fill;
            pnlExibirProduto->Show();

            txtNomeExibirProduto->Focus();
        }

        private: System::Void btnExibirProdutoPanel_Click(System::Object^ sender,
System::EventArgs^ e) {
            if (String::IsNullOrWhiteSpace(txtNomeExibirProduto->Text)) {
                MessageBox::Show("O nome do produto precisa ser informado!",
"Controle de Estoque - Supermercado",
                MessageBoxButtons::OK, MessageBoxIcon::Information);
                txtNomeExibirProduto->Focus();
                return;
            }

            msclr::interop::marshal_context context;
            std::string nome =
context.marshal_as<std::string>(txtNomeExibirProduto->Text);
            bool ok;

            ProdutoPtr produto = retorna_produto(&estoque, nome, &ok);

            lblResultadoProduto->Visible = true;

            if (ok) {
                txtNomeExibirProduto->Text = "";
            }
        }

```

```

        txtNomeExibirProduto->Focus();

        gpbExibirProduto->Visible = true;
        lblResultadoProduto->ForeColor = System::Drawing::Color::DarkCyan;
        lblResultadoProduto->Text = "Produto encontrado no estoque!";

        lblExibeNome->Text = gcnew String(produto->nome.c_str());
        lblExibePreco->Text = Convert::ToString(produto->preco);
        lblExibeQuantidade->Text = Convert::ToString(produto->quantidade);
    } else {
        txtNomeExibirProduto->Text = "";
        txtNomeExibirProduto->Focus();

        gpbExibirProduto->Visible = false;
        lblResultadoProduto->ForeColor = System::Drawing::Color::Magenta;
        lblResultadoProduto->Text = "Produto não encontrado no estoque";
    }
}

private: System::Void btnExibirProdutoSair_Click(System::Object^ sender,
System::EventArgs^ e) {
    habilitaBotoesMenu(true);
    pnlExibirProduto->Hide();

    txtNomeExibirProduto->Text = "";
    lblExibeNome->Text = "";
    lblExibePreco->Text = "";
    lblExibeQuantidade->Text = "";
    lblResultadoProduto->Visible = false;
    gpbExibirProduto->Visible = false;

    this->Show();
}

private: System::Void btnExibirEstoque_Click(System::Object^ sender,
System::EventArgs^ e) {
    habilitaBotoesMenu(false);
    pnlExibirEstoque->Dock = DockStyle::Fill;
    pnlExibirEstoque->Show();
    lblExibirEstoque->Focus();

    if (vazio(&estoque)) {
        lblTotal->Text = "Total de produtos no estoque: 0";
    }
}

```

```

        dgvEstoque->Visible = false;
        lblEstoqueVazio->Visible = true;
    } else {
        dgvEstoque->Visible = true;

        DataTable^ tabela = gcnew DataTable();
        tabela->Columns->Add("Nome");
        tabela->Columns->Add("Preço");
        tabela->Columns->Add("Quantidade");

        ProdutoPtr p_aux = estoque.primeiro;
        int total_produtos = 0;

        do {
            String^ nome = gcnew String(p_aux->nome.c_str());
            double preco = p_aux->preco;
            int quantidade = p_aux->quantidade;

            tabela->Rows->Add(nome, preco, quantidade);
            total_produtos++;

            p_aux = p_aux->proximo;
        } while (p_aux != NULL);

        lblTotal->Text = "Total de produtos no estoque: " +
Convert::ToString(total_produtos);
        dgvEstoque->DataSource = tabela;
    }
}

private: System::Void btnExibeEstoqueSair_Click(System::Object^ sender,
System::EventArgs^ e) {
    habilitaBotoesMenu(true);

    pnlExibirEstoque->Hide();
    this->Show();
}

private: System::Void btnSair_Click(System::Object^ sender,
System::EventArgs^ e) {
    if (MessageBox::Show("Tem certeza que deseja finalizar o programa?",
"Sair", MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
System::Windows::Forms::DialogResult::Yes) {

```

```
        this->Close();  
        Application::Exit();  
    }  
};  
}
```