

UNIFEV
ENGENHARIA DE COMPUTAÇÃO

GUILHERME AVELLAR VIÇOSO
RA: 105033

PROVA PRÁTICA E SISTEMA DESKTOP (.NET) C#
TÓPICOS EM LINGUAGEM DE PROGRAMAÇÃO I

VOTUPORANGA – SP
2023

RESUMO

Nesse projeto vimos a estrutura de um projeto desktop .Net em C#.

Criamos telas e alguns menus de funções, como, menu principal, tela de cadastro entre outros. Esse foi basicamente o desenvolvimento do projeto dentre esse semestre, finalizado e garantindo que tudo está rodando.

Projetado para atender a empresas onde sofre com problema de estoque e gerenciamento, um simples software que pode melhorar a empresa em grande escala.

Sumário

1 – SISTEMA EMPRESARIAL DESKTOP .NET C#.....	4
2 – TELAS E CÓDIGOS	5
MENU PRINCIPAL.....	6
TELA DE LOGIN	10
TELA ESTOQUE.....	13
TELA DE CADASTRO	18
TELA DE RELATÓRIO.....	21
TELA DE AJUDA	24
LINK ACESSO AO DRIVE DO PROJETO	25
LINK ACESSO AO GIT	25
CONCLUSÃO	26

1 – SISTEMA EMPRESARIAL DESKTOP .NET C#

Sistema empresarial é um software para gestão do seu negócio, serve para ter controle do que entra e sai da sua empresa e gestão do seu estoque, que foi o caso desse software, voltado para a parte de gestão de estoque.

Foi utilizado a linguagem C# para ser feito, e feito em modo desktop(.Net), utilizando como plataforma de programação o Visual Studio 2022, um recurso atual e avançado. Com ele eu criei as telas e os códigos que geram funções para o programa.

Utilizei também o Sql Sever 2022 para criar o banco de dados, onde armazenei os dados e gerei as tabelas para controle dentro do software, linkando tudo isso dentro do VS, usando método de Dapper, estudado e feito no decorrer das aulas.

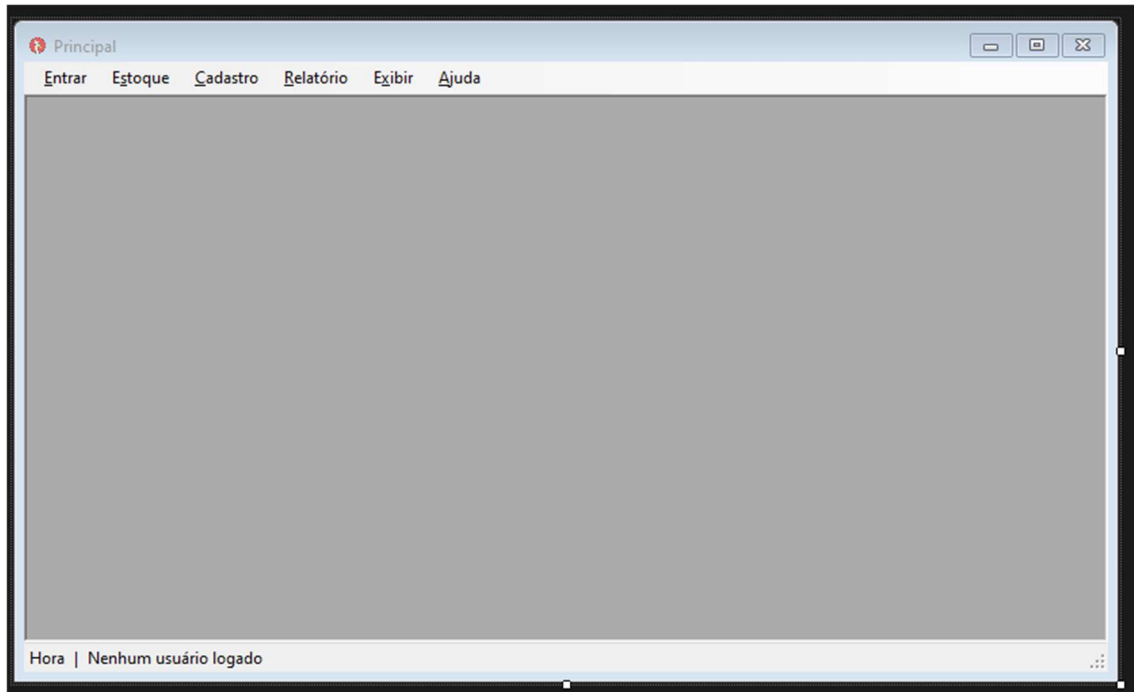
2 – TELAS E CÓDIGOS

Vou deixar aqui as telas junto com seus códigos feito no projeto, explicando o que é cada parte.

Foram criadas 6 telas no programa, sendo elas Menu Principal, Estoque, Cadastro, Login, Ajuda e Relatório. Foi feito também as Classes, desenvolvendo códigos de conexões, como por exemplo EstoqueDAO dentre outros.

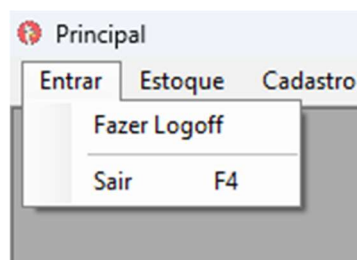
Todos os Forms e Clases foram separadas em pastas e também foi desenvolvido em um layout com imagens que fica mais fácil de compreender a sua funcionalidade, Como por exemplo Fechar e Excluir.

MENU PRINCIPAL



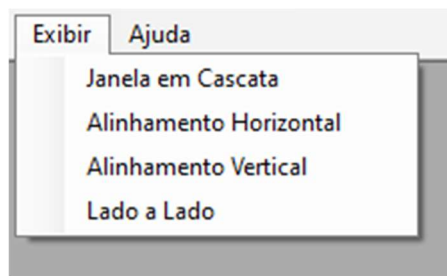
Essa é a aparência do menu principal, tendo um Menu Strip com as principais funções, Entrar, Estoque, Cadastro, Relatório, Exibir e Ajuda.

No Entrar temos duas opções:



O sair, para fechar o programa, ou o Fazer Logoff para deslogar o usuário conectado, ou Fazer Login quando não estiver logado ainda.

Temos também o Exibir, que mostra formatos de exibição da tela:



Esses modos são modos que sua tela de Estoque ou Cadastro podem ficar dentro do seu layout principal.

As outras funções jogam para telas diferentes.

O código do Menu Principal (Form), ficou extenso, por se tratar do form principal, temos muitas funções nele.

```
namespace SistemaProva
{
    4 references
    public partial class Principal : Form
    {
        1 reference
        public Principal()
        {
            InitializeComponent();
        }

        1 reference
        private void horaTimer_Tick(object sender, EventArgs e)
        {
            horaToolStripStatusLabel.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
        }

        1 reference
        private void Principal_Load(object sender, EventArgs e)
        {
            horaTimer.Start();
            VerificarLogin();
        }

        1 reference
        private void janelaEmCascataToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.LayoutMdi(MdiLayout.Cascade);
        }

        1 reference
        private void alinhamentoHorizontalToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.LayoutMdi(MdiLayout.TileHorizontal);
        }
    }
}
```

```

1 reference
private void alinhamentoVerticalToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileVertical);
}

1 reference
private void ladoALadoToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.ArrangeIcons);
}

```

Esse é o código apenas do exibir, onde cada um define a maneira que será exibido.

```

1 reference
private void sobreToolStripMenuItem_Click(object sender, EventArgs e)
{
    AjudaForm Ajuda = new AjudaForm();
    Ajuda.MdiParent = this;
    Ajuda.Show();
    Ajuda.WindowState = FormWindowState.Normal;
}

1 reference
private void sairToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}

1 reference
private void estoqueToolStripMenuItem_Click(object sender, EventArgs e)
{
    EstoqueForm Estoque = new EstoqueForm();
    Estoque.MdiParent = this;
    Estoque.Show();
    Estoque.WindowState = FormWindowState.Maximized;
}

1 reference
private void cadastroToolStripMenuItem_Click(object sender, EventArgs e)
{
    CadastroForm Cadastro = new CadastroForm();
    Cadastro.MdiParent = this;
    Cadastro.Show();
    Cadastro.WindowState = FormWindowState.Maximized;
}

```

```

1 reference
private void relatórioToolStripMenuItem_Click(object sender, EventArgs e)
{
    VisualizadorRelatorio visualizadorRelatorioForm = new
    VisualizadorRelatorio();
    visualizadorRelatorioForm.GerarRelatorioCidades();
    visualizadorRelatorioForm.ShowDialog();
}

```


Aqui está sendo declarado algumas funções como chamar a tela de Ajuda, Estoque e Cadastro e da opção de fechar.

```
2 references
void VerificarLogin()
{
    if (Global.UsuarioLogado == null)
    {
        nomeUsuarioToolStripStatusLabel.Text = "Nenhum usuário logado";
        //fecha os forms abertos
        foreach (Form f in this.MdiChildren)
            f.Close();
        fazerOLoginToolStripMenuItem.Text = "Entrar";
        //chama a tela de login
        new LoginForm().ShowDialog();
    }
    if (Global.UsuarioLogado != null)
    {
        //só habilita o menu de usuários se for admin (nível 1)
        fazerOLoginToolStripMenuItem.Text = "Fazer Logoff";
        nomeUsuarioToolStripStatusLabel.Text = Global.UsuarioLogado.Nome;
    }
}

1 reference
private void fazerOLoginToolStripMenuItem_Click(object sender, EventArgs e)
{
    Global.UsuarioLogado = null;
    VerificarLogin();
}
```

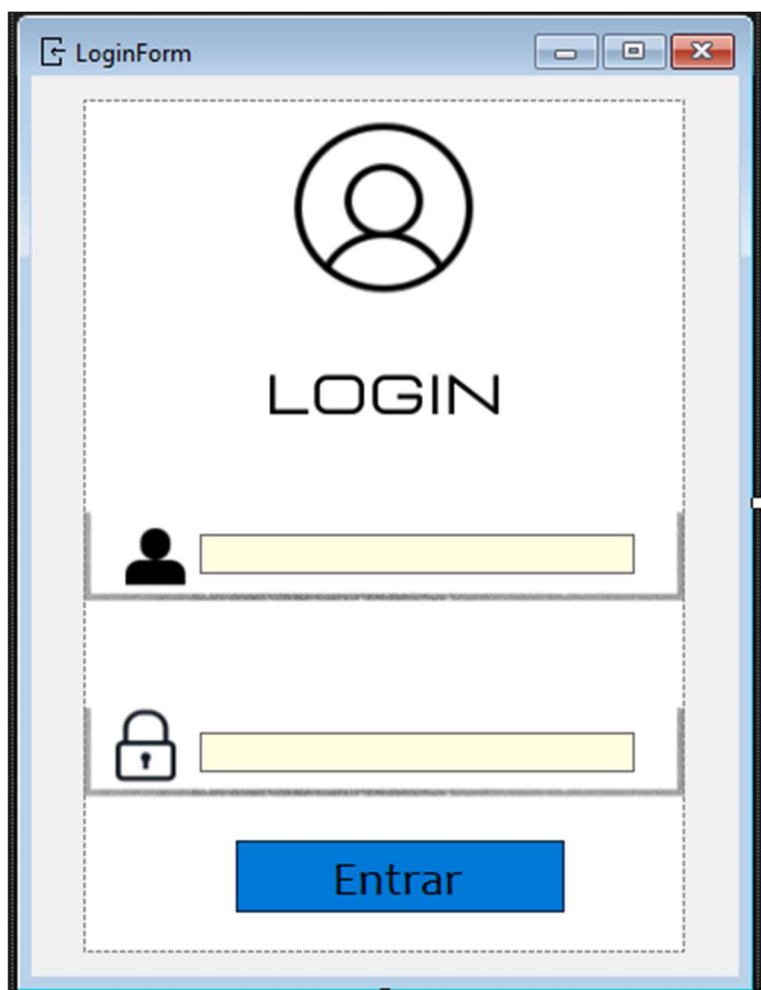
Em seguida fizemos a função para verificar o login e puxar a tela para logar, verificando o usuário e se for correto ele entra no sistema.

TELA DE LOGIN

Quando vamos iniciar o sistema, logo de início ele pede para logar, usando os usuários criados no banco de dados, sendo eles:

- Mondenez, senha:8354
- Avellar, senha:4632

Esses são os usuários de login criados, e o layout da tela é essa:



The image shows a screenshot of a login form window titled "LoginForm". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is enclosed in a dashed border and contains the following elements:

- A large circular icon representing a user profile at the top center.
- The word "LOGIN" in a large, bold, sans-serif font below the icon.
- A horizontal input field for the username, preceded by a small black silhouette icon of a person.
- A horizontal input field for the password, preceded by a small icon of a padlock.
- A blue rectangular button labeled "Entrar" (Enter) at the bottom center.

Layout bem simples e intuitivo, para o usuário não ter erro na hora de usar.

```

public partial class LoginForm : Form
{
    1 reference
    public LoginForm()
    {
        InitializeComponent();
    }

    1 reference
    private void entrarButton_Click(object sender, EventArgs e)
    {
        Global.UsuarioLogado = new UsuariosDAO().Login(loginTextBox.Text, senhaTextBox.Text);
        //não encontrou
        if (Global.UsuarioLogado == null)
        {
            MessageBox.Show("Usuário e senha não encontrado!", ProductName,
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            senhaTextBox.Clear();
            loginTextBox.Focus();
        }
        else
        {
            //encontrou e testamos se está ativo
            if (Global.UsuarioLogado.Ativo == false)
            {
                MessageBox.Show("Usuário desabilitado!", ProductName,
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                senhaTextBox.Clear();
                loginTextBox.Focus();
            }
            else
            {
                //está tudo certo, fechamos a tela
                Close();
            }
        }
    }
}

```

Esse é o código por trás dessa tela, fazendo a verificação e validação do usuário, usando também algumas classes de conexões, para puxar as informações.

```

6 references
public class Usuario
{
    1 reference
    public int ID { get; set; }
    2 references
    public string Nome { get; set; }
    1 reference
    public string Login { get; set; }
    1 reference
    public string Senha { get; set; }
    1 reference
    public int Nivel { get; set; }
    2 references
    public bool Ativo { get; set; }
}

```

A classe Usuario, que puxa as informações do banco de dados para dentro do projeto.

```
7 references
public class Global
{
    7 references
    public static Usuario UsuarioLogado { get; set; }
}
```


E a classe global, que verifica o usuário, assim validando ele se esta correto ou não.


TELA ESTOQUE


Quando clicamos em Estoque no Menu Principal, ele nos joga para outra tela, dentro do próprio form principal.


	ID	Nome do Produto	Sigla
▶	10	Canivete Volks	VW
	11	Miolo F250	Ford
	12	Excentrico Strada Curto	Fiat
	13	Centrinho 410	GM
	14	Canivete	Jeep
	15	Canivete	Citroen
	16	Canivete	Peugeot
	17	Excentrico Renault Master	Renault
	18	Carcaca	Toyota
	19	Canivete	Mercedes
	20	Canivete	BMW
	21	Canivete HB20	Hyundai
	22	Canivete	Audi
	23	682	Yale
	24	Chave Ford KA	Fiat

Registros encontrados: 17

 Excluir

 Alterar

 Novo

 Fechar

Essa é a tela de Estoque, onde temos o botão para buscar seus produtos e mostra sua quantidade de produtos cadastrados lá embaixo. Temos também 4 botões de ações, o de Excluir, que você seleciona um item e deleta ele, o de alterar e novo te jogam para uma nova tela, onde você faz tanto a alteração quanto adiciona um novo produto. O botão fechar ele sai dessa tela de Estoque.

```

namespace SistemaProva
{
    5 references
    public partial class EstoqueForm : Form
    {
        1 reference
        public EstoqueForm()
        {
            InitializeComponent();
            Application.DoEvents();
            //desabilita os botões
            excluirButton.Enabled = false;
            alterarButton.Enabled = false;
        }

        1 reference
        private void fecharButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        1 reference
        private void buscarButton_Click(object sender, EventArgs e)
        {
            var dados = new EstoqueDAO().ListarTodas();
            quantidadeLabel.Text = $"Registros encontrados: {dados.Count()}";
            listaDataGridView.DataSource = dados;
            excluirButton.Enabled = dados.Count() > 0;
            alterarButton.Enabled = excluirButton.Enabled;
        }
    }
}

```

Aqui iniciamos nosso código fazendo que os botões de excluir e alterar sejam desativados até o momento que se puxa o estoque todo. Logo temos o botão de buscar, onde encontra os dados do banco e mostra todos eles, também executa a função de mostrar a quantidade de produtos.

```

1 reference
private void excluirButton_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Confirma a exclusão do Produto?", ProductName,
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {
        int id =
            Convert.ToInt32(listaDataGridView.SelectedRows[0].Cells["idColumn"].Value);
        new EstoqueDAO().Excluir(id);
        buscarButton.PerformClick();
        MessageBox.Show("Produto excluído com sucesso!", ProductName,
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

1 reference
private void alterarButton_Click(object sender, EventArgs e)
{
    int id = Convert.ToInt32(listaDataGridView.SelectedRows[0].Cells["idColumn"].Value);
    new CadastroForm(id).ShowDialog();
    buscarButton.PerformClick();
}

1 reference
private void novoButton_Click(object sender, EventArgs e)
{
    int id = Convert.ToInt32(listaDataGridView.SelectedRows[0].Cells["idColumn"].Value);
    new CadastroForm().ShowDialog();
    buscarButton.PerformClick();
}

```

Aqui está sendo feito os comandos para altera, novo e excluir, puxando-os de uma classe onde foram feitas as funções principais.

```

5 references
public class EstoqueDAO : Conexao
{
    //Função para Adicionar Nova Cidade
    1 reference
    public void Adicionar(string nome, string sigla)
    {
        conexao.Execute("INSERT INTO Estoque (Nome, Sigla) VALUES (@Nome, @Sigla)",
            new { nome, sigla });
    }

    // Função para Atualizar uma Cidade
    1 reference
    public void Atualizar(int id, string nome, string sigla)
    {
        conexao.Execute("UPDATE Estoque SET Nome=@Nome, Sigla=@Sigla WHERE ID=@ID",
            new { nome, sigla, id });
    }

    // Função para Excluir uma Cidade
    1 reference
    public void Excluir(int id)
    {
        conexao.Execute("DELETE FROM Estoque WHERE ID=@ID",
            new { id });
    }

    // Função para Buscar uma Cidade pelo ID
    1 reference
    public Estoque Buscar(int id)
    {
        return conexao.Query<Estoque>("SELECT * FROM Estoque WHERE ID=@ID",
            new { id }).SingleOrDefault();
    }

    // Função para Listar Todas as Cidades (foi modificado para ficar no padrão
    1 reference
    public List<Estoque> ListarTodas()
    {
        return conexao.Query<Estoque>("SELECT * FROM Estoque order by ID").ToList();
    }
}

```

Essa é a classe EstoqueDAO, onde gerei as funções para os comandos da tela de Estoque. Se reparar temos duas classes sendo puxadas, a Estoque e a Conexão.

```

3 references
public class Conexao
{
    public SqlConnection conexao = null;

    0 references
    public Conexao()
    {
        conexao = new SqlConnection(
            ConfigurationManager.ConnectionStrings["banco"].ConnectionString);
    }

    0 references
    internal static object Query<T>(string v, object value)
    {
        throw new NotImplementedException();
    }
}

```

Essa é a classe Conexão, onde fiz conexão com o bando de dados através do app.config


```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="banco"
      connectionString="Server=(local)\SQLEXPRESS;Database=BancoProva;Trusted_Connection=True;" />
    </connectionStrings>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>

```

Também fizemos a classe do Estoque.

```

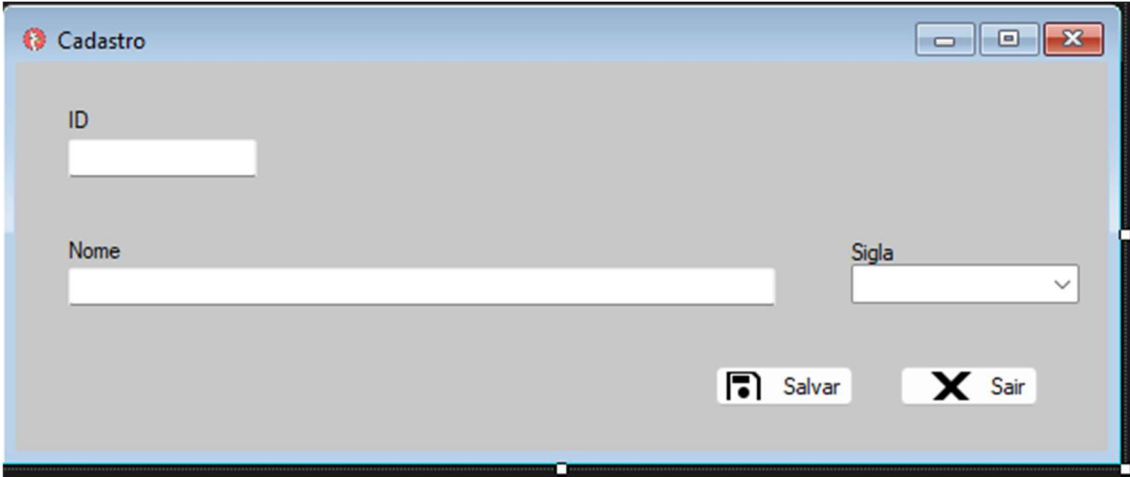
4 references
public class Estoque
{
    1 reference
    public int Id { get; set; }
    1 reference
    public string Nome { get; set; }
    1 reference
    public string Sigla { get; set; }
}

```

Onde referencia os valores e conteúdos do meu banco de dados.

TELA DE CADASTRO

Essa é a tela que pode ser puxada tanto no Menu Principal, quanto na tela de Estoque por meio dos botões de Novo e Alterar.



A imagem mostra a interface de uma janela de cadastro. No topo, há uma barra azul com o título "Cadastro" e botões de controle de janela. Abaixo, o formulário contém:

- Um campo de texto rotulado "ID".
- Um campo de texto rotulado "Nome".
- Um campo de texto rotulado "Sigla" com uma seta para baixo no final.
- Dois botões na base à direita: "Salvar" (com ícone de disquete) e "Sair" (com ícone de X).

Quando é feito uma alteração ou inserir um novo produto, essa tela é puxada, o ID é automático, só tem que ser colocado o Nome e a Sigla.

```

2 references
public CadastroForm()
{
    InitializeComponent();
    Text = "Novo Produto";
    idTextBox.Text = "Automático";
    idTextBox.ReadOnly = true;
    nomeTextBox.Focus();

    ListarSiglas();
}

1 reference
public CadastroForm(int id)
{
    InitializeComponent();
    var cadastro = new EstoqueDAO().Buscar(id);

    Text = "Alteração de Produto";
    idTextBox.Text = cadastro.Id.ToString();
    idTextBox.ReadOnly = true;

    nomeTextBox.Text = cadastro.Nome;
    nomeTextBox.Focus();

    ListarSiglas();
    Application.DoEvents();
    siglaComboBox.SelectedValue = cadastro.Sigla;
}

2 references
void ListarSiglas()
{
    var lista = new MarcasDAO().Listar();
    siglaComboBox.DisplayMember = "Nome";
    siglaComboBox.ValueMember = "Sigla";
    siglaComboBox.DataSource = lista;
}

```

Aqui já está sendo feito os comandos quando se puxa do botão de alterar ou de Adicionar um Novo produto.

```

1 reference
private void sairButton_Click(object sender, EventArgs e)
{
    Close();
}

1 reference
private void salvarButton_Click(object sender, EventArgs e)
{
    if (Text == "Novo Produto")
    {
        //faz a inclusão
        new EstoqueDAO().Adicionar(nomeTextBox.Text, siglaComboBox.SelectedValue.ToString());
        MessageBox.Show("Cidade adicionada com sucesso!", ProductName,
            MessageBoxButtons.OK, MessageBoxIcon.Information);

        Close();
    }
    else
    {
        //faz a alteração
        new EstoqueDAO().Atualizar(Convert.ToInt32(idTextBox.Text),
            nomeTextBox.Text, siglaComboBox.SelectedValue.ToString());
        MessageBox.Show("Cidade alterada com sucesso!", ProductName,
            MessageBoxButtons.OK, MessageBoxIcon.Information);

        Close();
    }
}

```

Aqui está fazendo a função e salvar ou altera os produtos, junto com o botão de fechar a tela.

TELA DE RELATÓRIO

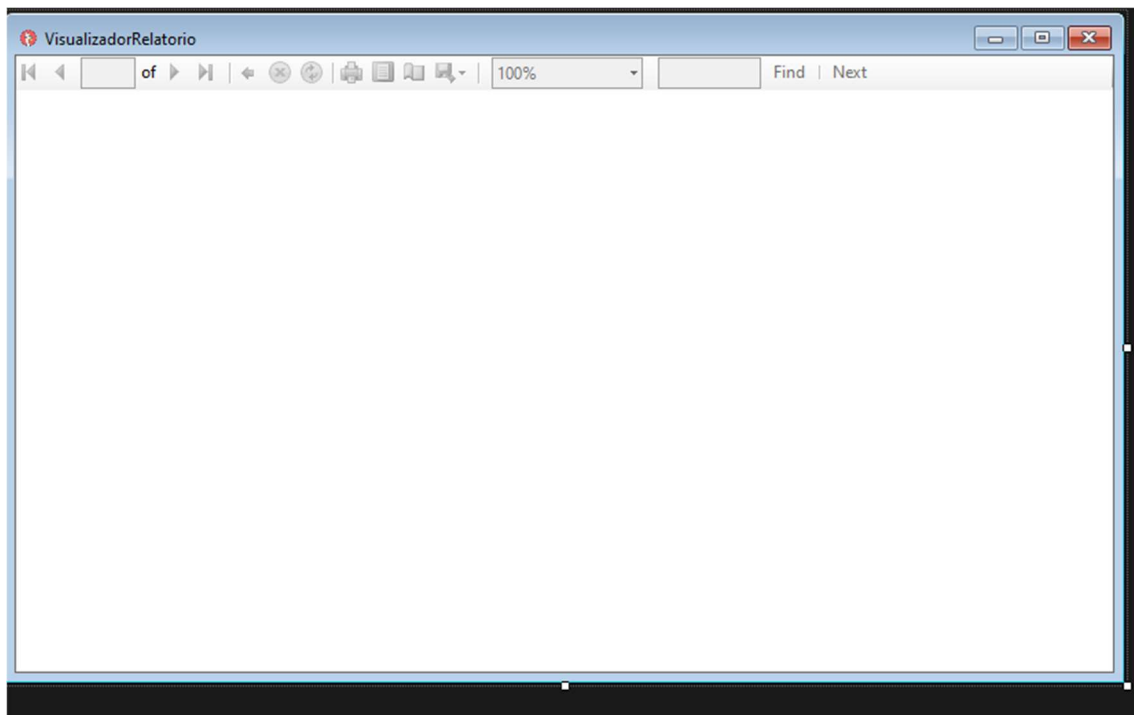
Nessa tela é uma das partes mais chatas para ser feita, pois usamos o método de ReportView e usamos mais outra tela para a exibição, tirando o código de conexão.



The mockup shows a report interface. At the top left is a circular logo with a red background and a white duck head. To its right is a title box containing the text "RELATÓRIO DE LISTA DE ESTOQUE". Below the title is a table with three columns: "ID", "Nome", and "Siglas". The first row of the table has placeholder text "[ID]", "[Nome]", and "[Siglas]" respectively. At the bottom of the interface, there are two buttons, each labeled "«Expr»".

ID	Nome	Siglas
[ID]	[Nome]	[Siglas]

Esse é o layout do Relatório, onde será passado o ID, Nome e Sigla do produto.



Essa tela será puxada dentro desse formulário, trazendo suas informações.

```
5 references
public partial class VisualizadorRelatorio : Form
{
    1 reference
    public VisualizadorRelatorio()
    {
        InitializeComponent();
    }

    1 reference
    private void VisualizadorRelatorio_Load(object sender, EventArgs e)
    {
        this.reportViewer1.RefreshReport();
    }

    1 reference
    public void GerarRelatorioCidades()
    {
        using (var connection = new SqlConnection("Server=(local)\\SQLEXPRESS;Database=BancoProva;Trusted_Connection=True;"))
        {
            var listaEstoque = connection.Query<EstoqueGris>("SELECT ID, Nome, Sigla FROM Estoque").ToList();

            reportViewer1.LocalReport.ReportPath = Path.Combine(Application.StartupPath, "EstoqueListaReport.rdlc");
            reportViewer1.LocalReport.DataSources.Clear();
            reportViewer1.LocalReport.DataSources.Add(
                new ReportDataSource(
                    "DataSet1",
                    listaEstoque)
            );
            reportViewer1.SetDisplayMode(DisplayMode.PrintLayout);
            reportViewer1.ZoomMode = ZoomMode.PageWidth;
            reportViewer1.RefreshReport();
        }
    }
}
```

Aqui está todo o desenvolvimento por trás, onde está sendo puxado os valores das tabelas do banco de dados e jogados dentro do relatório.

```

1 reference
public class EstoqueGris
{
    0 references
    public int ID { get; set; }
    0 references
    public string Nome { get; set; }
    0 references
    public string Sigla { get; set; }
}

```

E usamos essa conexão, onde é puxado os valores.

VisualizadorRelatorio

1 de 2

Largura da Página

Localizar | Avançar

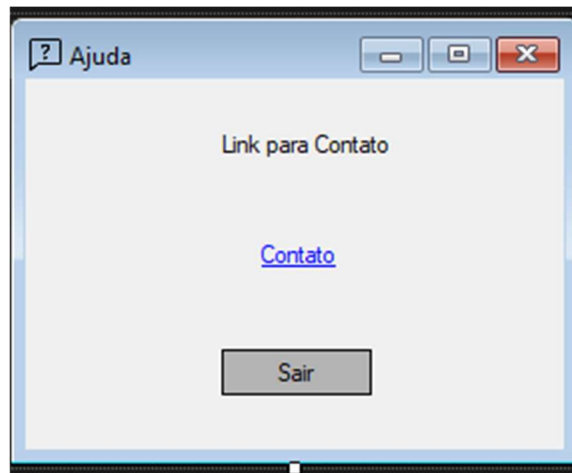
RELATÓRIO DE LISTA DE ESTOQUE

ID	Nome	Siglas
10	Canivete Volks	
11	Miolo F250	
12	Excentrico Strada Curto	
13	Centrinho 410	
14	Canivete	
15	Canivete	
16	Canivete	
17	Excentrico Renault Master	
18	Carcça	
19	Canivete	
20	Canivete	
21	Canivete HB20	
22	Canivete	
23	682	
24	Chave Ford KA	
25	Chave Kadet	
27	Excentrico Prisma	

E esse é o layout final, apenas não está puxando as siglas.

TELA DE AJUDA

Essa é a última tela, a tela de ajuda, caso sinta dificuldade ou algum problema é nela que você entra e assim chama contato via WhatsApp.



Uma tela bem simples, mais que resolve todos seus problemas a respeito do sistema.

```
5 references
public partial class AjudaForm : Form
{
    1 reference
    public AjudaForm()
    {
        InitializeComponent();
    }

    1 reference
    private void sairButton_Click(object sender, EventArgs e)
    {
        Close();
    }

    1 reference
    private void contatoLinkLabel_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        try
        {
            VisitLink();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Clique AQUI para entrar em Contato!");
        }
    }

    1 reference
    private void VisitLink()
    {
        contatoLinkLabel.LinkVisited = true;
        System.Diagnostics.Process.Start
        ("https://api.whatsapp.com/send?phone=5517996626833&text=01%C3%A1,%20queria%20saber%20a%20respeito%20do%20Sistema.%20Es");
    }
}
```

Aqui é o código desenvolvido nele, puxando um comando VisitaLink, onde coloquei um link do meu WhatsApp.

LINK ACESSO AO DRIVE DO PROJETO

https://drive.google.com/drive/folders/1Duolqk9apF-qOaJOKZt7Giym5TpksGHI?usp=drive_link

LINK ACESSO AO GIT

<https://github.com/GuiVellar/SistemaProva>

CONCLUSÃO

Chegamos ao fim do desenvolvimento do sistema. Esse foi meu software apresentado, voltado para área de empresas, focado na melhora da gestão e controle do estoque.