**Sistema de Prestação de Contas - Documentação Completa**

**Data:** 28 de Fevereiro de 2025  
**Desenvolvido com:** Grok 3 da xAI

**1. Introdução**

O objetivo desta interação foi desenvolver a funcionalidade de **Prestação de Contas** para um sistema em C# que gerencia vendedores, produtos e entregas. O usuário forneceu modelos iniciais e solicitou ajuda para criar a tela de prestação de contas, incluindo cálculos automáticos e integração com um banco de dados SQLite.

**2. Modelos Iniciais Fornecidos**

**VendedorMODEL**

public class VendedorMODEL

{

public int VendedorID { get; set; }

public string Nome { get; set; }

public string CPF { get; set; }

public string Telefone { get; set; }

}

**ProdutoMODEL**

public class ProdutoMODEL

{

public int ProdutoID { get; set; }

public string NomeProduto { get; set; }

public double Preco { get; set; }

public string Tipo { get; set; }

public int QuantidadePorBloco { get; set; }

}

**EntregasModel**

public class EntregasModel

{

public int EntregaID { get; set; }

public long VendedorID { get; set; }

public long ProdutoID { get; set; }

public long QuantidadeEntregue { get; set; }

public DateTime? DataEntrega { get; set; } = DateTime.Now;

public bool PrestacaoRealizada { get; set; } = false; *// Adicionado posteriormente*

}

**PrestacaoContasModel**

public class PrestacaoContasModel

{

public int PrestacaoID { get; set; }

public int EntregaID { get; set; }

public int QuantidadeVendida { get; set; }

public int QuantidadeDevolvida { get; set; }

public double ValorRecebido { get; set; }

public double Comissao { get; set; }

public DateTime DataPrestacao { get; set; } = DateTime.Now;

}

**3. Estrutura das Tabelas no Banco SQLite**

**Tabela Vendedores**

CREATE TABLE [Vendedores] (

[VendedorID] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

[Nome] TEXT NOT NULL,

[CPF] TEXT NOT NULL,

[Telefone] TEXT  
);

**Tabela Produtos**

CREATE TABLE [Produtos] (

[ProdutoID] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

[NomeProduto] TEXT NOT NULL,

[Preco] REAL NOT NULL,

[Tipo] TEXT,

[QuantidadePorBloco] INTEGER

);

**Tabela Entregas**

CREATE TABLE [Entregas] (

[EntregaID] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

[VendedorID] BIGINT NOT NULL,

[ProdutoID] BIGINT NOT NULL,

[QuantidadeEntregue] BIGINT NOT NULL,

[DataEntrega] DATETIME DEFAULT (CURRENT\_TIMESTAMP) NULL,

[PrestacaoRealizada] INTEGER DEFAULT 0,

CONSTRAINT [FK\_Entregas\_0\_0] FOREIGN KEY ([ProdutoID]) REFERENCES [Produtos] ([ProdutoID]),

CONSTRAINT [FK\_Entregas\_1\_0] FOREIGN KEY ([VendedorID]) REFERENCES [Vendedores] ([VendedorID])

);

**Atualização:**

ALTER TABLE Entregas ADD COLUMN PrestacaoRealizada INTEGER DEFAULT 0;

**Tabela PrestacaoContas**

CREATE TABLE [PrestacaoContas] (

[PrestacaoID] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

[EntregaID] INTEGER NOT NULL,

[QuantidadeVendida] INTEGER NOT NULL,

[QuantidadeDevolvida] INTEGER NOT NULL,

[ValorRecebido] REAL NOT NULL,

[Comissao] REAL NOT NULL,

[DataPrestacao] DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (EntregaID) REFERENCES Entregas(EntregaID)

);

**4. Requisitos da Prestação de Contas**

* **Tela**: Interface para registrar devoluções e valores arrecadados.
* **Cálculos**:
  + QuantidadeVendida = QuantidadeEntregue - QuantidadeDevolvida
  + ValorRecebido = QuantidadeVendida \* Preco
  + Comissao = ValorRecebido \* 0.10 (10%, configurável)
* **Fluxo**:
  + Cadastro de produtos e vendedores.
  + Registro de entregas.
  + Prestação de contas com cálculos automáticos.
  + Atualização do status da entrega.

**5. Desenvolvimento da Tela de Prestação de Contas**

**Controles Sugeridos**

* **ComboBox**: cmbEntregasPendentes (lista entregas com PrestacaoRealizada = 0).
* **TextBox (somente leitura)**: txtVendedor, txtProduto, txtQuantidadeEntregue.
* **TextBox (editável)**: txtQtdDevolvida.
* **TextBox (somente leitura)**: txtQuantidadeVendida, txtValorRecebido, txtComissao.
* **DateTimePicker**: dateTimePicker (data da prestação).
* **Button**: btnSalvar, btnCancelar.

**Código da Tela**

public partial class FormPrestacaoContas : Form

{

private EntregasModel entregaSelecionada;

private string connectionString = "Data Source=seu\_banco.db;Version=3;";

public FormPrestacaoContas()

{  
 InitializeComponent();  
 }

private void FormPrestacaoContas\_Load(object sender, EventArgs e)

{  
 CarregarComboEntregas();  
 }

private void CarregarComboEntregas()

{

var entregas = CarregarEntregasNaoPrestadas();

cmbEntregasPendentes.DataSource = entregas;

cmbEntregasPendentes.DisplayMember = "ToString";

cmbEntregasPendentes.ValueMember = "EntregaID";

}

private List<EntregaComboItem> CarregarEntregasNaoPrestadas()

{

List<EntregaComboItem> entregas = new List<EntregaComboItem>();

string query = @"SELECT EntregaID, VendedorID, ProdutoID, QuantidadeEntregue, DataEntrega

FROM Entregas

WHERE PrestacaoRealizada = 0";

using (SQLiteConnection conn = new SQLiteConnection(connectionString))

{

conn.Open();

using (SQLiteCommand cmd = new SQLiteCommand(query, conn))

{

using (SQLiteDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

entregas.Add(new EntregaComboItem

{

EntregaID = reader.GetInt32(0),

VendedorID = reader.GetInt64(1),

ProdutoID = reader.GetInt64(2),

QuantidadeEntregue = reader.GetInt64(3),

DataEntrega = reader.GetDateTime(4)

});  
 }  
 }  
 }  
 }

return entregas;

}

private void cmbEntregasPendentes\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cmbEntregasPendentes.SelectedItem != null)

{  
 entregaSelecionada = (EntregasModel)cmbEntregasPendentes.SelectedItem;

txtVendedor.Text = BuscarNomeVendedor(entregaSelecionada.VendedorID);

txtProduto.Text = BuscarNomeProduto(entregaSelecionada.ProdutoID);

txtQuantidadeEntregue.Text = entregaSelecionada.QuantidadeEntregue.ToString();  
 }  
 }

private void txtQtdDevolvida\_TextChanged(object sender, EventArgs e)

{

if (entregaSelecionada == null)

{

MessageBox.Show("Selecione uma entrega primeiro!");

return;

}

if (int.TryParse(txtQtdDevolvida.Text, out int devolvida) &&

int.TryParse(txtQuantidadeEntregue.Text, out int entregue))

{

if (devolvida <= entregue)

{

int vendida = entregue - devolvida;

txtQuantidadeVendida.Text = vendida.ToString();

double precoProduto = BuscarPrecoProduto(entregaSelecionada.ProdutoID);

double valorRecebido = vendida \* precoProduto;

txtValorRecebido.Text = valorRecebido.ToString("C");

double comissao = valorRecebido \* 0.10;

txtComissao.Text = comissao.ToString("C");

}

else

{

MessageBox.Show("Quantidade devolvida não pode ser maior que a entregue!");

txtQtdDevolvida.Text = "0";

}  
 }  
 }

private void btnSalvar\_Click(object sender, EventArgs e)

{

var prestacao = new PrestacaoContasModel

{  
 EntregaID = entregaSelecionada.EntregaID,

QuantidadeVendida = int.Parse(txtQuantidadeVendida.Text),

QuantidadeDevolvida = int.Parse(txtQtdDevolvida.Text),

ValorRecebido = double.Parse(txtValorRecebido.Text, NumberStyles.Currency),

Comissao = double.Parse(txtComissao.Text, NumberStyles.Currency),

DataPrestacao = dateTimePicker.Value

};

var entregasDAL = new EntregasDAL();

entregasDAL.SalvarPrestacaoDeContas(prestacao);  
 CarregarComboEntregas();  
 }

private string BuscarNomeVendedor(long vendedorID)

{

using (SQLiteConnection conn = new SQLiteConnection(connectionString))

{  
 conn.Open();

string query = "SELECT Nome FROM Vendedor WHERE VendedorID = @VendedorID";

using (SQLiteCommand cmd = new SQLiteCommand(query, conn))

{

cmd.Parameters.AddWithValue("@VendedorID", vendedorID);

return cmd.ExecuteScalar()?.ToString() ?? "Vendedor não encontrado";

}  
 }  
 }

private string BuscarNomeProduto(long produtoID)

{

using (SQLiteConnection conn = new SQLiteConnection(connectionString))

{  
 conn.Open();

string query = "SELECT NomeProduto FROM Produto WHERE ProdutoID = @ProdutoID";

using (SQLiteCommand cmd = new SQLiteCommand(query, conn))

{

cmd.Parameters.AddWithValue("@ProdutoID", produtoID);

return cmd.ExecuteScalar()?.ToString() ?? "Produto não encontrado";

}  
 }  
 }

private double BuscarPrecoProduto(long produtoID)

{

using (SQLiteConnection conn = new SQLiteConnection(connectionString))

{  
 conn.Open();

string query = "SELECT Preco FROM Produto WHERE ProdutoID = @ProdutoID";

using (SQLiteCommand cmd = new SQLiteCommand(query, conn))

{

cmd.Parameters.AddWithValue("@ProdutoID", produtoID);

object resultado = cmd.ExecuteScalar();

return resultado != null ? Convert.ToDouble(resultado) : 0.0;

}  
 }  
 }  
}

public class EntregaComboItem

{

public int EntregaID { get; set; }

public long VendedorID { get; set; }

public long ProdutoID { get; set; }

public long QuantidadeEntregue { get; set; }

public DateTime DataEntrega { get; set; }

public override string ToString()

{

return $"Entrega {EntregaID} - {QuantidadeEntregue} unidades ({DataEntrega:dd/MM/yyyy})";

}  
}

**6. Camada DAL**

public class EntregasDAL

{

public void SalvarEntregas(EntregasModel entrega)

{

using (var conn = Conexao.Conex())

{

conn.Open();

string query = @"INSERT INTO Entregas (VendedorID, ProdutoID, QuantidadeEntregue, DataEntrega, PrestacaoRealizada)

VALUES (@VendedorID, @ProdutoID, @QuantidadeEntregue, @DataEntrega, @PrestacaoRealizada)";

using (var cmd = new SQLiteCommand(query, conn))

{

cmd.Parameters.AddWithValue("@VendedorID", entrega.VendedorID);

cmd.Parameters.AddWithValue("@ProdutoID", entrega.ProdutoID);

cmd.Parameters.AddWithValue("@QuantidadeEntregue", entrega.QuantidadeEntregue);

cmd.Parameters.AddWithValue("@DataEntrega", entrega.DataEntrega);

cmd.Parameters.AddWithValue("@PrestacaoRealizada", entrega.PrestacaoRealizada ? 1 : 0);

cmd.ExecuteNonQuery();

}  
 }  
 }

public void SalvarPrestacaoDeContas(PrestacaoContasModel prestacao)

{

using (var conn = Conexao.Conex())

{

conn.Open();

string insertQuery = @"INSERT INTO PrestacaoContas (EntregaID, QuantidadeVendida, QuantidadeDevolvida, ValorRecebido, Comissao, DataPrestacao)

VALUES (@EntregaID, @QuantidadeVendida, @QuantidadeDevolvida, @ValorRecebido, @Comissao, @DataPrestacao)";

using (var cmd = new SQLiteCommand(insertQuery, conn))

{

cmd.Parameters.AddWithValue("@EntregaID", prestacao.EntregaID);

cmd.Parameters.AddWithValue("@QuantidadeVendida", prestacao.QuantidadeVendida);

cmd.Parameters.AddWithValue("@QuantidadeDevolvida", prestacao.QuantidadeDevolvida);

cmd.Parameters.AddWithValue("@ValorRecebido", prestacao.ValorRecebido);

cmd.Parameters.AddWithValue("@Comissao", prestacao.Comissao);

cmd.Parameters.AddWithValue("@DataPrestacao", prestacao.DataPrestacao);

cmd.ExecuteNonQuery();  
 }

string updateQuery = @"UPDATE Entregas SET PrestacaoRealizada = 1 WHERE EntregaID = @EntregaID";

using (var cmd = new SQLiteCommand(updateQuery, conn))

{

cmd.Parameters.AddWithValue("@EntregaID", prestacao.EntregaID);

cmd.ExecuteNonQuery();

}  
 }  
 }  
}

**7. Camada BLL**

public class EntregasBLL

{

private EntregasDAL entregasDAL = new EntregasDAL();

public void Salvar(EntregasModel entrega)

{  
 entregasDAL.SalvarEntregas(entrega);  
 }  
}

**8. Salvamento de Entregas no Formulário**

public void SalvarRegistro()

{

try

{

Model.EntregasModel objetoModel = new Model.EntregasModel();

objetoModel.VendedorID = VendedorID;

objetoModel.ProdutoID = ProdutoID;

objetoModel.QuantidadeEntregue = int.Parse(txtQuantidade.Text);

objetoModel.DataEntrega = dtpDataEntregaBilhete.Value;

EntregasBLL objetoBll = new EntregasBLL();

objetoBll.Salvar(objetoModel);

MessageBox.Show("Registro gravado com sucesso!", "Informação!!!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

Utilitario.LimpaCampo(this);

((FrmManutençãodeEntregaBilhetes)Application.OpenForms["FrmManutençãodeEntregaBilhetes"]).HabilitarTimer(true);

}  
 catch (OverflowException ov)  
 {

MessageBox.Show("Overflow Exception deu erro! " + ov);

}  
 catch (Win32Exception erro)

{

MessageBox.Show("Win32 Exception!!! \n" + erro);

}  
}

**9. Principais Correções Realizadas**

1. **Erro em CarregarComboEntregas:**
   * **Problema**: "Não é possível usar a variável local 'entregas' antes de declará-la".
   * **Solução**: Substituí o comentário por chamada a CarregarEntregasNaoPrestadas().
2. **Erro em txtQtdDevolvida\_TextChanged:**
   * **Problema**: "Não é possível converter implicitamente tipo 'string' em 'double'".
   * **Solução**: Substituí BuscarNomeProduto por BuscarPrecoProduto.
3. **Escopo de entregaSelecionada:**
   * **Problema**: Variável fora de escopo.
   * **Solução**: Declarada como variável de instância.
4. **Adição de PrestacaoRealizada:**
   * **Problema**: Tabela Entregas sem controle explícito de prestação.
   * **Solução**: Adicionado campo PrestacaoRealizada com padrão 0.

**10. Testes Recomendados**

* **Salvar Entrega**: Verificar se PrestacaoRealizada = 0 no banco.
* **Carregar ComboBox**: Confirmar que apenas entregas com PrestacaoRealizada = 0 aparecem.
* **Prestar Contas**: Após salvar, verificar se PrestacaoRealizada muda para 1 e o ComboBox é atualizado.

**Fim do Documento**