

Licenciatura em Engenharia Informática

Bases de Dados

Livraria

Gabriel Hall - 102851
Leonardo Flório - 103360

Bases de Dados	1
Introdução	2
Análise de requisitos	3
Diagramas	4
Diagrama Entidade-Relacionamento	4
Diagrama Relacional	5
Diagrama auto-gerado em SQL	6
Inserção de dados	7
Triggers	8
Adicionar tipo de utilizador	8
Adicionar tipo de produto	8
Update stock	9
Update linha	10
Procedures	10
UDF	11
VIEW	12
C#	14
Inserir Livros	14
Inserir Fornecedor/Cliente	15

Introdução

No projeto final da Unidade Curricular de Base de Dados, com vista a reunir todas as competências lecionadas nas aulas práticas e teórico-práticas da disciplina, foi pedido aos alunos que escolhessem um tema.

Desta forma, o tema escolhido e apresentado neste relatório tem como objetivo a gestão de uma Livraria. Vai permitir tratar de entidades como:

- Editora
- Autor
- Produto
- Livro
- Pacote
- Documento
- Fatura de Compra
- Fatura de Venda
- Fornecedor
- Cliente

É, também, importante salientar que a base de dados foi realizada em SQL Server, e a interface gráfica foi construída usando Windows Forms c#.

Análise de requisitos

O sistema de gestão de stock terá de permitir:

- Inserir
 - Autores
 - Editoras
 - Livros
 - Fornecedores
 - Clientes
 - Documentos (venda e compra)
 - Linhas nos documentos

- Listar
 - Inventário (stocks)
 - Histórico de vendas
 - Faturação

É de notar que é possível apagar qualquer editora, autor, livro, fornecedor ou cliente.

É importante que o stock seja atualizado automaticamente quando se insere uma fatura nova. Também é pertinente atualizar a faturação total de cada vez que ocorre uma venda.

O foco deste projeto é permitir uma fácil gestão de stock e também uma visão sobre a faturação da livraria.

Diagramas

Como surgiram necessidades de implementar tabelas novas e relações novas, os diagramas foram alterados várias vezes com o decorrer do projeto.

Diagrama Entidade-Relacionamento

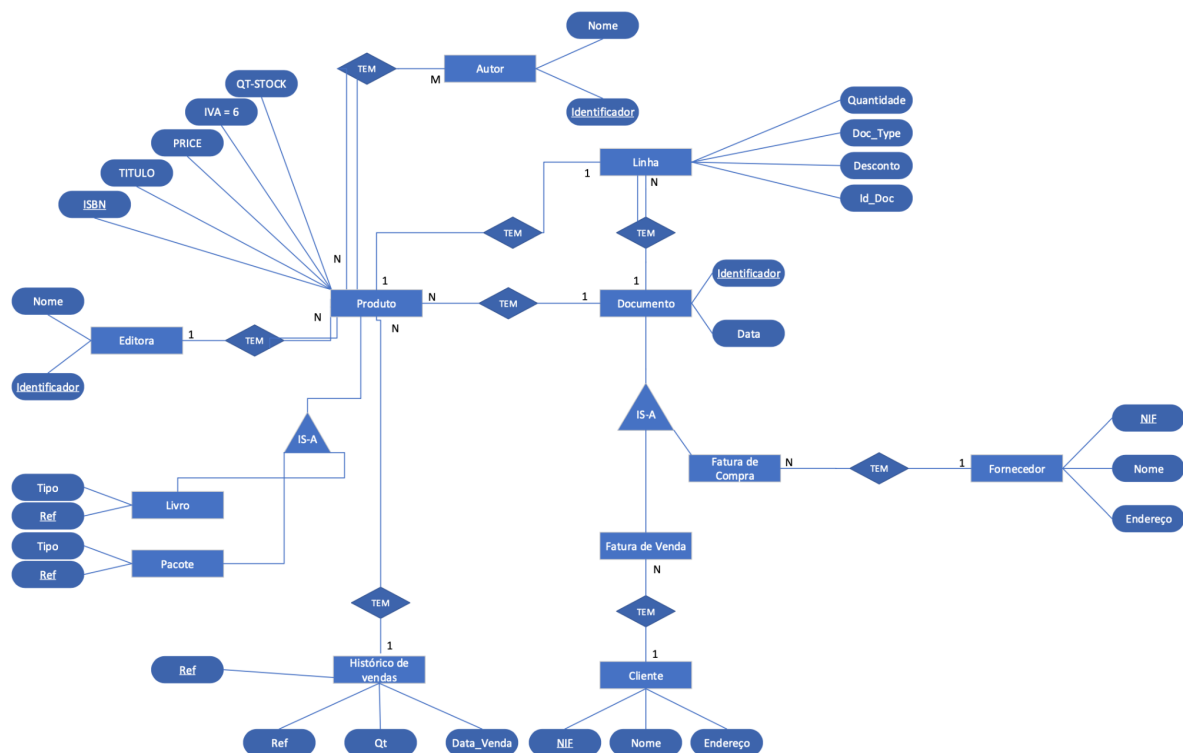


Diagrama Relacional

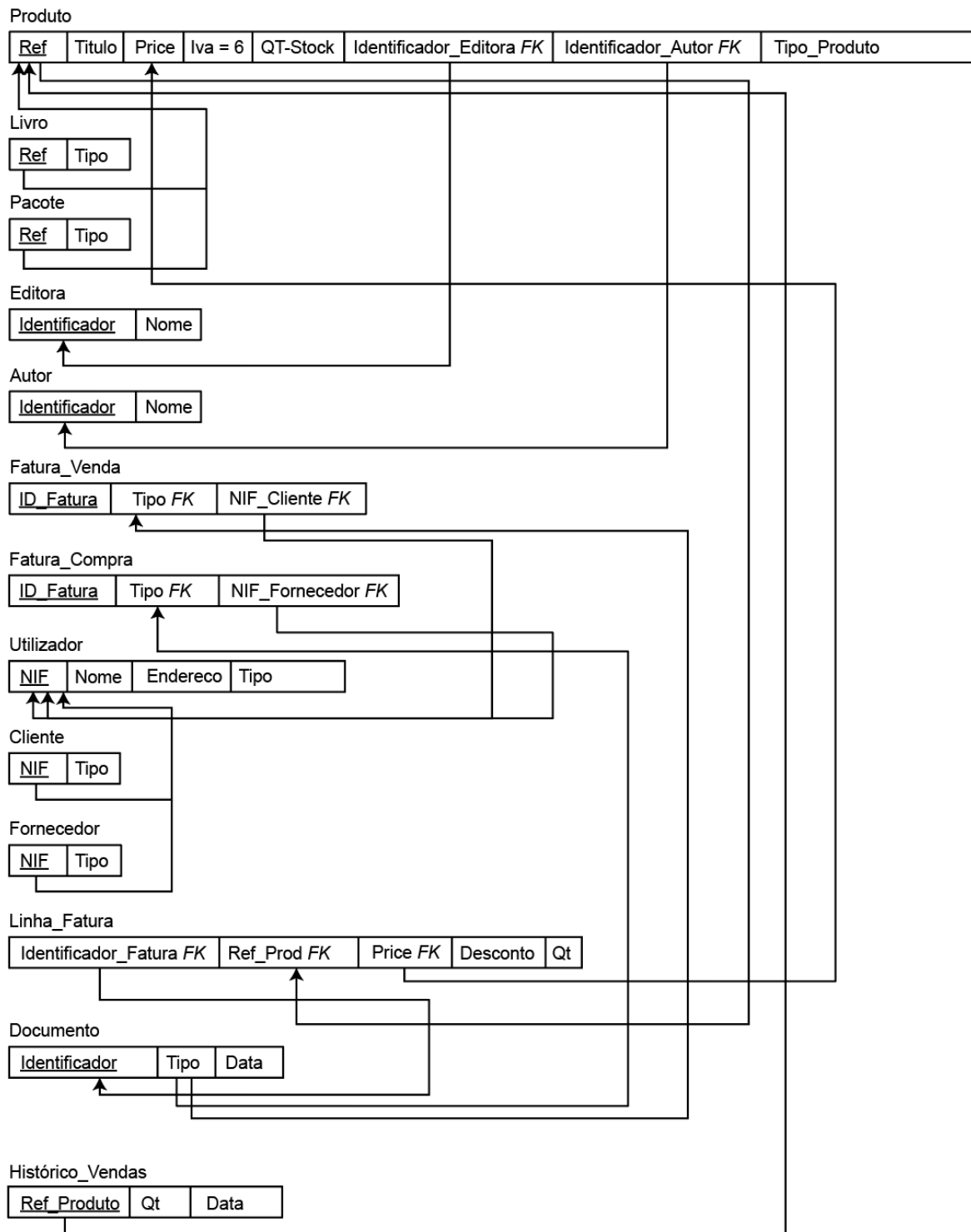
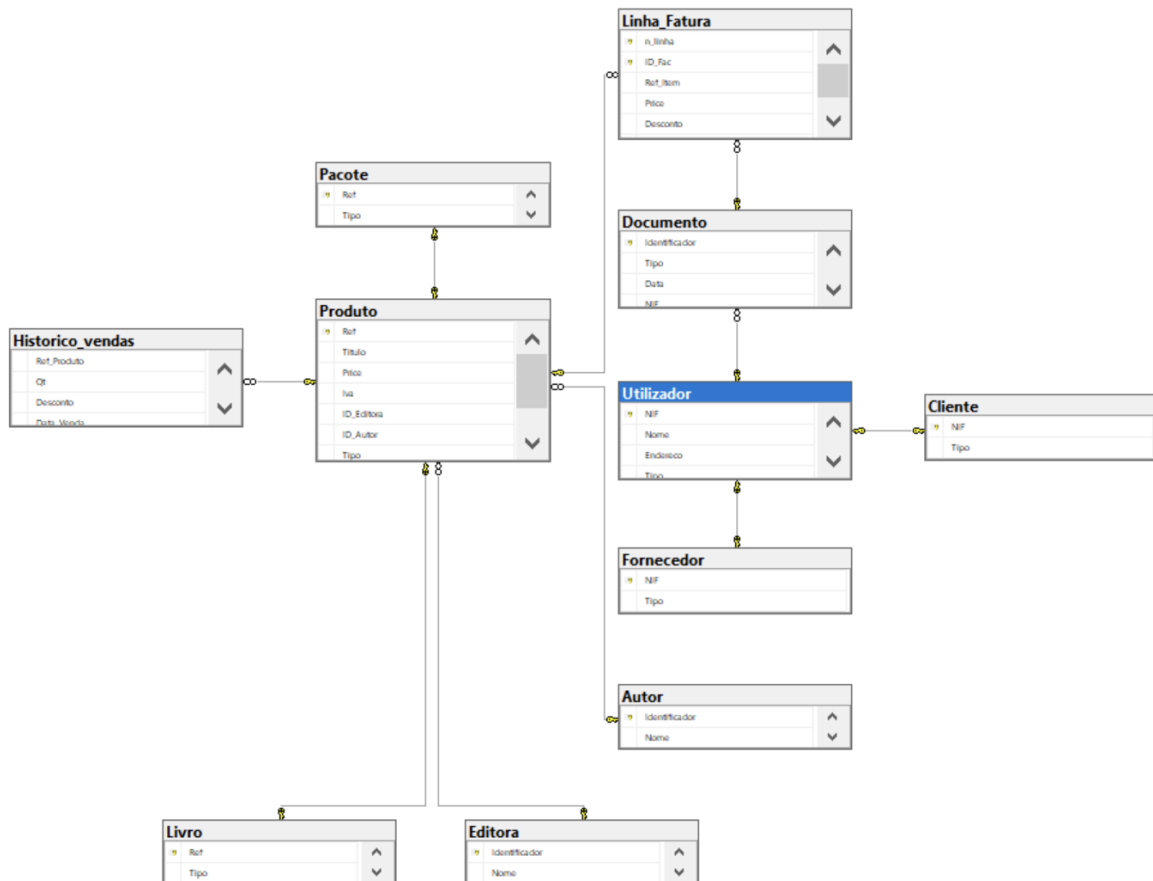


Diagrama auto-gerado em SQL



Inserção de dados

Alguns dados foram inseridos diretamente na base de dados (ficheiro insert.sql) com operações INSERT INTO, nomeadamente:

- INSERT INTO EDITORA
- INSERT INTO AUTOR
- INSERT INTO PRODUTO
- INSERT INTO UTILIZADOR
- INSERT INTO DOCUMENTO
- INSERT INTO LINHA_FATURA

Os restantes dados foram inseridos através da aplicação desenvolvida. Os inserts foram também utilizados para testar os triggers.

Triggers

- Adicionar tipo de utilizador

```
CREATE TRIGGER addUserType
ON Utilizador
FOR INSERT
AS
BEGIN
    DECLARE @NIF BIGINT, @Tipo INT
    Select @NIF = NIF, @Tipo = Tipo from inserted
    IF (@Tipo = 1)
    Begin
        INSERT INTO Cliente (NIF, Tipo)
            VALUES (@NIF, 'Cliente')
    END

    IF (@Tipo = 2)
    Begin
        INSERT INTO Fornecedor (NIF, Tipo)
            VALUES (@NIF, 'Fornecedor')
    END
END
GO
```

Aquando da inserção de um novo Utilizador o trigger vai buscar a primary key (NIF), e o parâmetro Tipo do Utilizador e cria um novo Fornecedor ou Cliente conforme o Tipo de Utilizador inserido.

- Adicionar tipo de produto

```
CREATE TRIGGER addProdType
ON Produto
FOR INSERT
AS
BEGIN
    DECLARE @Ref BIGINT, @Tipo INT
    Select @Ref = Ref, @Tipo = Tipo from inserted
    IF (@Tipo = 1)
    Begin
        INSERT INTO Livro (Ref, Tipo)
        VALUES (@Ref, 'Livro')
    End
    IF (@Tipo = 2)
    Begin
        INSERT INTO Pacote (Ref, Tipo)
        VALUES (@Ref, 'Pacote')
    End
END
GO
```

Semelhante ao trigger anterior, este trigger, consoante o tipo de Produto inserido criará também um novo Livro ou Pacote.

- Update stock

```
create trigger trg_updateStocks
ON Linha_Fatura
FOR INSERT
AS
BEGIN
    DECLARE @ID_Fac INT, @Qt INT, @Ref BIGINT, @Doc_type INT, @Prod_type INT, @data DATETIME, @Desc INT

    SELECT @ID_Fac = ID_Fac, @Qt = Quantity, @Ref = Ref_Item, @Desc = Desconto from inserted

    SELECT @data = Documento.Data, @Doc_type = AVG(Documento.Tipo) from Linha_Fatura JOIN Documento ON ID_Fac = Identificador
    where Identificador = @ID_Fac
    GROUP BY Documento.Data

    SELECT @Prod_type = Produto.Tipo from Linha_Fatura join Produto on Ref_Item = Ref WHERE Ref = @Ref

    IF (@Doc_type = 1)
    BEGIN
        IF (@Prod_type = 1)
        BEGIN
            Update Produto
            SET Stock = Stock - @Qt
            where ref = @Ref

            Insert into Historico_vendas (Ref_Produto, Qt, Desconto, Data_Venda)
            VALUES (@Ref, @Qt, @Desc, @data)
        END
    END

    IF (@Doc_type = 2)
    BEGIN
        IF (@Prod_type = 1)
        BEGIN
            Update Produto
            SET Stock = Stock + @Qt
            where ref = @Ref
        END
    END
END
GO
```

Este trigger quando se insere uma nova linha numa fatura vai buscar o tipo de documento (fatura de venda/fatura de compra) e, consoante o tipo vai aumentar ou decrementar o stock do produto identificado pela linha.

- Update linha

```
create trigger dbo.update_Line
ON dbo.Linha_Fatura
FOR INSERT
AS
BEGIN

    declare @ID_Fac INT

    select @ID_Fac = ID_Fac from inserted

    --check if doc exists
    --if exists(select Identificador from Documento where Identificador = @ID_Fac)
    --Begin
        --get lines from doc

        declare @num INT
        select @num = count(n_linha) from dbo.Linha_Fatura where ID_Fac = @ID_Fac
        update dbo.Linha_Fatura set n_linha = @num, ID_Fac = @ID_Fac where n_linha = 999
    --end

END
go
```

Este trigger serve para organizar linhas por cada fatura. Por default cada linha é inserida com o valor 999. Com o objetivo de fazer com que a linha seja identificada pela primary key (n_linha, ID_Fac), o número da linha terá de ser incrementado a cada linha inserida. Por isso, o trigger trata de dar update ao valor n_linha e incrementá-lo sempre que se insere uma nova linha.

Procedures

Os procedures criados (procedures.sql), nomeadamente:

- INSERT_EDITORA
- DELETE_EDITORA
- INSERT_AUTOR
- DELETE_AUTOR

São executados nestas 2 operações (insert, delete) em duas entidades (autor, editora).

```
SqlCommand cmd = new SqlCommand("exec delete_autor @Name = @Nome", con);
```

```
SqlCommand cmd = new SqlCommand("exec insert_autor @Name = @Nome", con);
```

Os 4 procedures são simples com apenas comandos de select, delete e insert.

Para inserir um Autor ou uma Editora basta inserir o nome. O identificador é incrementado automaticamente pois está definido como identity(1,1)
Uma vez que a entidade Editora é muito semelhante à entidade Autor, o código e a interface são também muito semelhantes.

UDF

UDF que permite calcular a faturação total.

```
Create function get_Faturacao()
Returns INT
as
begin
return
(
SELECT SUM(Price*Qt*(100-Desconto)/100) AS Total FROM Historico_vendas JOIN Produto on Historico_vendas.Ref_Produto = Produto.Ref
)
end
go

select dbo.get_Faturacao() as Total
```

Form1

Histórico de Vendas

Inserir Livro

Inserir Editora

Inserir Autor

Inventário

Inserir Fornecedor

Inserir Cliente

Inserir Documento

Histórico Vendas

	Ref_Produto	Título	Price	Qt	Data_Venda	Desconto
▶	978483987	Mundo no Chao	7	1	6/7/2022	0
	978930031	A Vingança do G...	7	2	6/7/2022	0
	978483987	Mundo no Chao	7	3	6/7/2022	0
	978930031	A Vingança do G...	7	2	6/7/2022	0
	978930031	A Vingança do G...	7	4	6/7/2022	0
	978923658	Urso Castanho	11	1	6/24/2022	0
*						

Faturação total: 95€

[refresh](#)

Esta UDF permite registar a faturação (montante das vendas ocorridas)

VIEW

Esta view permite visualizar o inventário de produtos da livraria.

```
create view inventario
as
SELECT Ref, Titulo, Price, Editora.Nome AS Editora, Autor.Nome AS Autor, Stock
FROM Produto JOIN Editora ON ID_Editora = Editora.Identificador JOIN Autor ON ID_Autor = Autor.Identificador
```

Com o objetivo de filtrar a visualização desenvolvemos também uma função que permite fazer uma query multi-parametrizada.

Os filtros permitidos são todas as combinações possíveis dos 4 parâmetros:

- Editora (nome)
- Autor (nome)
- Título
- ISBN

```

private String CheckIfEmpty(TextBox textBox)
{
    if (!String.IsNullOrEmpty(textBox.Text))
    {
        if(textBox.Name == "Editora" || textBox.Name=="Autor")
            return textBox.Name + ".Nome = '" + textBox.Text + "'";
        if(textBox.Name == "Titulo")
            return "Produto." + textBox.Name + " = '" + textBox.Text + "'";
        if(textBox.Name == "ISBN")
            return "Produto.Ref = '" + textBox.Text + "'";
    }
    return "";
}

private void button1_Click(object sender, EventArgs e)
{
    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-JLR4AMV;Initial Catalog=Livraria;Integrated Security=True"))
    {
        con.Open();
        String total = "";

        List<TextBox> lst = new List<TextBox> { Editora, Autor, Titulo, ISBN };
        foreach (TextBox tb in lst)
        {
            if(total.Length == 0)
            {
                if(CheckIfEmpty(tb) != "")
                    total += "WHERE " + CheckIfEmpty(tb);
            }
            else
            {
                if (CheckIfEmpty(tb) != "")
                    total += " AND " + CheckIfEmpty(tb);
            }
        }

        SqlDataAdapter sqlDa = new SqlDataAdapter("SELECT Ref, Titulo, Price, Editora.Nome AS Editora, Autor.Nome AS Autor, Stock FROM " +
        "Produto JOIN Editora ON ID_Editora = Editora.Identificador JOIN Autor ON ID_Autor = Autor.Identificador " +
        total, con);
        DataTable dt = new DataTable();
        sqlDa.Fill(dt);
        dataGridView1.DataSource = dt;
    }
}

```

Form1

Inventário

Inserir Livro

Inserir Editora

Inserir Autor

Inventário

Inserir Fornecedor

Inserir Cliente

Inserir Documento

Histórico Vendas

Filtrar por:

Editora

Autor

Título

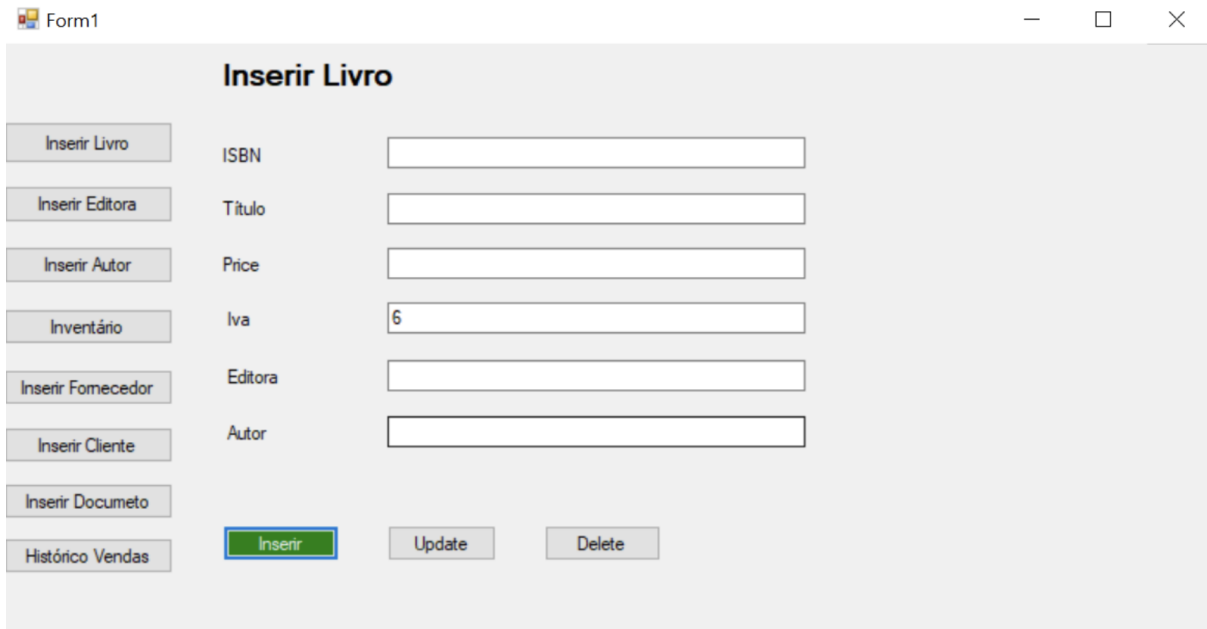
ISBN

Filtrar

	Ref	Título	Price	Editora	Autor	Stock
	935718148	Joaninha Resmu...	15	Planeta Tangerina	Ines Machado	5
	973258628	P de Pai	13	Leya	Manuel Afonso	5
	978483761	A Toupeira	12	Orfeu Negro	Ana Pessoa	0
	978483987	Mundo no Chao	7	Planeta Tangerina	Madalena Matoso	1
	978907285	Homem da Lua	7	Planeta Tangerina	Joaquim Agil	5
	978923658	Urso Castanho	11	Bruaa	Joao Fazedra	5

C#

Inserir Livros



Aqui foi usada uma insert query que permite ao user inserir livros novos na base de dados.

Para atribuir uma Editora e um Autor ao livro foi também feita uma função que permite saber se estes parâmetros já estão registados na base de dados. Como faz mais sentido colocar o nome do autor/editora em vez do seu identificador foi também feita uma função que com base no nome vai buscar o seu identificador previamente à inserção do novo livro.

```
SqlCommand cmd = new SqlCommand("INSERT INTO Produto (Ref, Titulo, Price, Iva, ID_Editora, ID_Autor, Tipo) " +  
    "Values (@Ref, @Titulo, @Price, @Iva, @ID_Editora, @ID_Autor, @Tipo)", con);
```

```
cmd.Parameters.AddWithValue("@ID_Editora", getIdFromNomeEditora(textBox5.Text));  
cmd.Parameters.AddWithValue("@ID_Autor", getIdFromNomeAutor(textBox6.Text));
```

```
private int getIdFromNomeEditora(string Nome)  
{  
    SqlConnection con = new SqlConnection("Data Source=DESKTOP-JLR4AMV;Initial Catalog=Livraria;Integrated Security=True");  
  
    con.Open();  
  
    SqlCommand cmd = new SqlCommand("SELECT Identificador From Editora WHERE Nome = '" + Nome + "'", con);  
    var res = cmd.ExecuteReader();  
    res.Read();  
    int id = (int)res["Identificador"];  
  
    return id;  
}
```

```
private int getIdFromNomeAutor(string Nome)
{
    SqlConnection con = new SqlConnection("Data Source=DESKTOP-JLR4AMV;Initial Catalog=Livraria;Integrated Security=True");

    con.Open();

    SqlCommand cmd = new SqlCommand("SELECT Identificador From Autor WHERE Nome = '" + Nome + "'", con);
    var res = cmd.ExecuteReader();
    res.Read();
    int id = (int)res["Identificador"];
    res.Close();

    return id;
}
```

Inserir Fornecedor/Cliente

The screenshot shows a Windows application window titled "Form1". On the left is a vertical sidebar with buttons: "Inserir Livro", "Inserir Editora", "Inserir Autor", "Inventário", "Inserir Fornecedor", "Inserir Cliente" (which is highlighted with a blue border), "Inserir Documento", and "Histórico Vendas". The main content area is titled "Inserir Cliente". It contains three text input fields labeled "NIF", "Nome", and "Endereço". Below these fields are three buttons: "Inserir" (green), "Update", and "Delete".

À semelhança das entidades Autor e Editora, as entidades Cliente e Fornecedor também são muito parecidas e partilham de todos os mesmos atributos menos o tipo.

Para inserir clientes e fornecedores fizemos uma insert query que insere utilizadores do tipo cliente/fornecedor. A distinção é feita pelo formulário em que se encontra. Tanto os clientes como os fornecedores são criados quando o trigger `addUserType` é acionado.

```

private void button1_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection("Data Source=DESKTOP-JLR4AMV;Initial Catalog=Livraria;Integrated Security=True");

    con.Open();

    SqlCommand cmd = new SqlCommand("INSERT INTO Utilizador VALUES(@NIF,@Nome,@Address,@Type)", con);
    var myCommand = new SqlCommand("SELECT NIF From Utilizador WHERE NIF = @NIF", con);

    myCommand.Parameters.AddWithValue("@NIF", int.Parse(textBox1.Text));
    var rd = myCommand.ExecuteReader();

    bool exists = rd.HasRows;
    if (!exists)
    {
        if (!String.IsNullOrEmpty(textBox2.Text))
        {
            cmd.Parameters.AddWithValue("@NIF", int.Parse(textBox1.Text));
            cmd.Parameters.AddWithValue("@Nome", textBox2.Text);
            cmd.Parameters.AddWithValue("@Address", textBox3.Text);
            cmd.Parameters.AddWithValue("@Type", 1);
            rd.Close();
            cmd.ExecuteNonQuery();
            MessageBox.Show("Inserted with success");
        }
        else
        {
            MessageBox.Show("Nome é obrigatório");
        }
    }
    else
    {
        {
            MessageBox.Show("NIF já está atribuido a uma Cliente");
        }
    }

    con.Close();
}

```