



PROGRAMAÇÃO O.O.




(C#)

{ DataGridView, DateTimePicker e PictureBox
Professor: João Luiz Lagôas

}




Controles úteis!



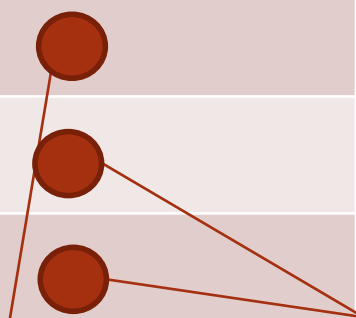
Controle	Nome	Atributo/Propriedade que armazena a informação	Tipo da informação
	TextBox		
	NumericUpDown		
	ComboBox		

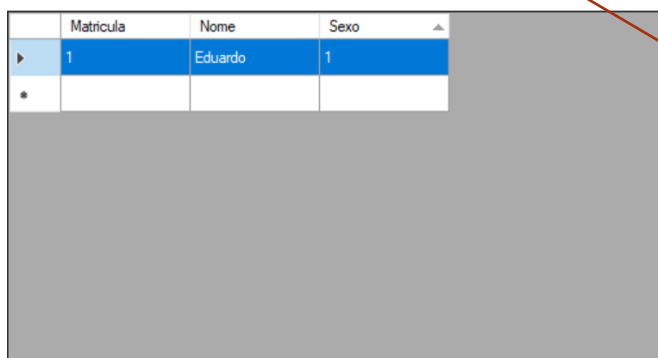
Controles úteis!



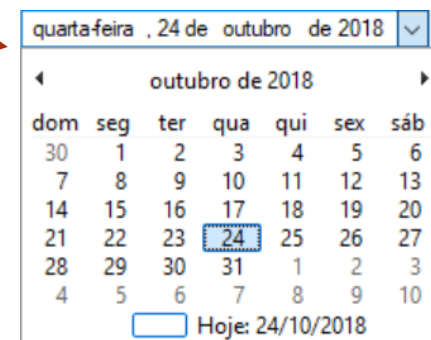
Controle	Nome	Atributo/Propriedade que armazena a informação	Tipo da informação
	TextBox	Text	string
	NumericUpDown	Value	Decimal
	ComboBox	Items	List<object>

Controles úteis!

Controle	Nome	Atributo/Propriedade que armazena a informação	Tipo da informação
	DataGridView		
	DateTimePicker		
	PictureBox		



Matricula	Nome	Sexo
1	Eduardo	1
*		


quarta-feira, 24 de outubro de 2018

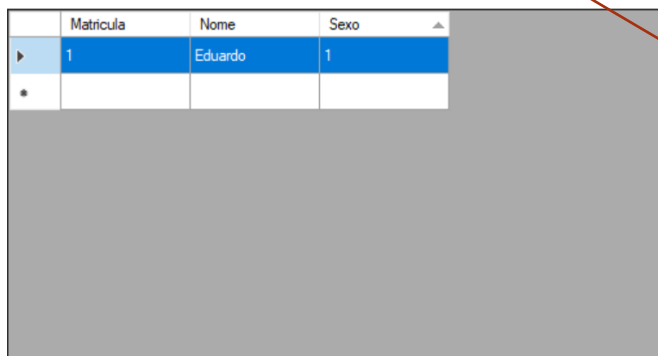
outubro de 2018

dom	seg	ter	qua	qui	sex	sáb
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

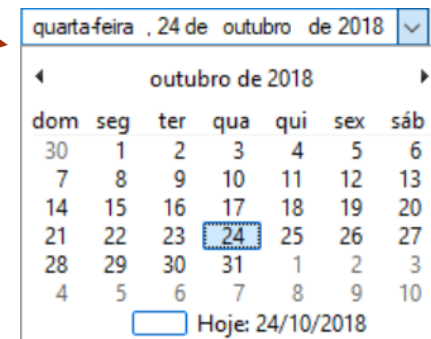
Hoje: 24/10/2018

Controles úteis!

Controle	Nome	Atributo/Propriedade que armazena a informação	Tipo da informação
	DataGridView	DataSource	DataTable
	DateTimePicker	Value	DateTime
	PictureBox	Image	Image



Matricula	Nome	Sexo
1	Eduardo	1
*		



DataGridView Control



- O **DataGridView** é um dos mais úteis controles para armazenar dados em formato de tabelas. Por conta disso, esse controle se torna muito útil para armazenar informações cuja origem é um banco de dados.
- Um **DataGridView** armazena um objeto do tipo **DataTable** no atributo **DataSource**.

```
DataTable tabela = new DataTable();  
dataGridView1.DataSource = tabela;
```

DataGridView Control

Lembrando da classe DataTable



- Um **DataTable** é uma classe presente na plataforma .NET usada para armazenar informações em forma de tabela. Funciona como uma espécie de matriz mas é repleta de recursos (métodos e atributos) para tornar a manipulação dos dados simples e otimizada.
- Podemos criar um **DataTable** para armazenar dados em memória e realizar operações para **incluir, alterar e excluir** linhas.
- A classe **DataTable** pode ser encontrada no *namespace System.Data*.



DataGridView Control

Lembrando da classe DataTable



- Um objeto **DataTable** apresenta um nome e um conjunto de Colunas e Linhas. A primeira informação é representada pelo atributo **Name** enquanto que as outras duas são representadas pelos atributos **Columns** e **Rows**.
- A propriedade **Rows** e **Columns** armazenam objetos do tipo **DataRow** e **DataColumn**, respectivamente. Ambos podem ser interpretados como uma **List**.

DataGridView Control

Lembrando da classe DataTable



- Até este momento, viemos utilizando a classe DataTable através do seu carregamento/preenchimento automático. Isso era realizado ao chamar o método de assinatura:

```
public DataTable Load(SqlDataReader reader)
```

```
conexao.Open();  
comando = new SqlCommand(comandoDQL, conexao);  
  
leitor = comando.ExecuteReader();  
  
DataTable tabela = new DataTable();  
  
tabela.Load(leitor);  
  
return tabela;
```

Note que estamos preenchendo a instância tabela chamando o método Load. Ele recebe um objeto do tipo SqlDataReader e carrega todos os dados lidos no comando DQL.

Trecho do método ConsultarBanco que implementamos na classe DataBaseManager.cs

DataGridView Control

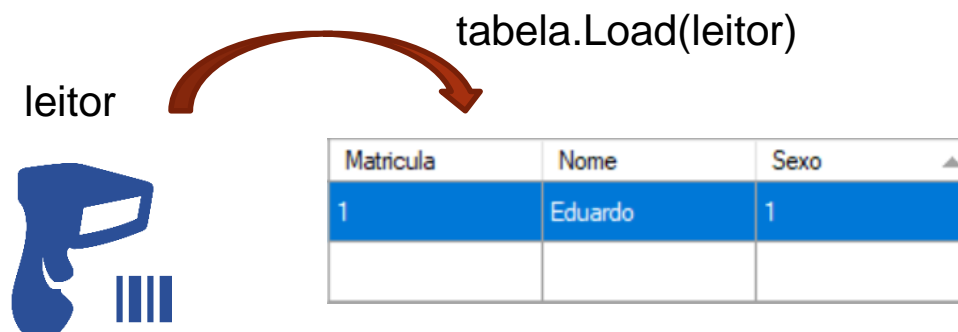
DataTable



Uma das principais características de um DataTable é a capacidade de armazenar dados advindos de qualquer fonte de dados (arquivos texto, XML, banco de dados, etc).



- O método **Load()** da classe **DataTable** recebe como argumento um objeto do tipo **SqlDataReader**.
- Ao ser executado o método **Load()** carrega o objeto **DataTable** de acordo com a consulta/tabela que um leitor esteja fazendo.



```
dataGridView.DataSource = tabela;
```

DataGridView Control

DataTable



```
string consultaSQL = "SELECT Nome, Sexo FROM Aluno";

SqlConnection conexao = new SqlConnection(connectionString);
conexao.Open();

SqlCommand comando = new SqlCommand(consultaSQL, conexao);

SqlDataReader leitor = comando.ExecuteReader();

DataTable tabela = new DataTable();
tabela.Load(leitor);
dataGridView1.DataSource = tabela;
```

Carregamento do objeto tabela a partir do leitor!

Depois da chamada ao método Load(leitor), a DataTable já está preenchida.

DataGridView Control

*DataTable: construindo e adicionando uma linha
SEM usar o método Load*



```
DataTable tabela = new DataTable("Alunos");

tabela.Columns.Add("Matricula");
tabela.Columns.Add("Nome");
tabela.Columns.Add("Sexo");

DataRow novaLinha = tabela.NewRow();

novaLinha["Matricula"] = 1;
novaLinha["Nome"] = "Eduardo";
novaLinha["Sexo"] = '1';

tabela.Rows.Add(novaLinha);
```

objeto do
tipo **DataTable**



Name: Aluno

Rows { **DataRow**

Matricula	Nome	Sexo
1	Eduardo	1

DataColumn

Columns

DataGridView Control

DataTable: *iterando com **foreach** ou com **for***

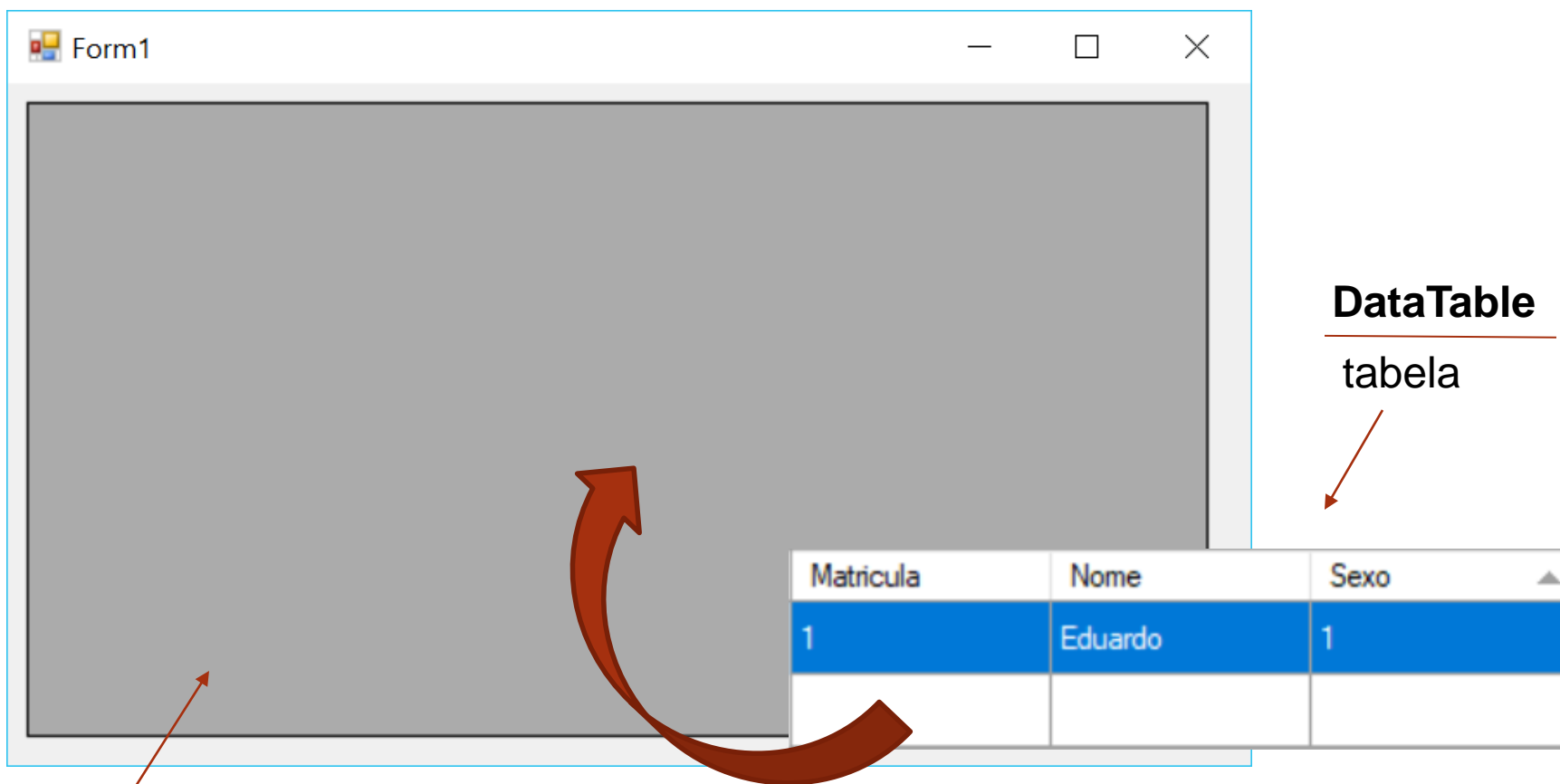


```
foreach(DataRow linha in tabela.Rows)
{
    string nome = linha["Nome"].ToString();
    listBox1.Items.Add(nome);
}
```

```
for(int i = 0; i < tabela.Rows.Count; i++)
{
    string nome= tabela.Rows[i]["Nome"].ToString();
    listBox1.Items.Add(nome);
}
```

DataGridView Control

DataTable: carregando um DataTable no Grid



DataGridView
dataGridView1

```
dataGridView1.DataSource = tabela;
```

DataGridView Control

DataTable: carregando um DataTable no Grid



```
private void Form1_Load(object sender, EventArgs e)
```

```
{
```

```
    DataTable tabela = new DataTable("Alunos");
```

```
    tabela.Columns.Add("Matricula");
```

```
    tabela.Columns.Add("Nome");
```

```
    tabela.Columns.Add("Sexo");
```

```
    DataRow novaLinha = tabela.NewRow();
```

```
    novaLinha["Matricula"] = 1;
```

```
    novaLinha["Nome"] = "Eduardo";
```

```
    novaLinha["Sexo"] = '1';
```

```
    tabela.Rows.Add(novaLinha);
```

```
    dataGridView1.DataSource = tabela;
```

```
}
```

Construção e
preparação
do objeto
DataTable

Atribuição do
DataTable ao
grid

DataGridView Control

DataTable



- Como um DataGridView apresenta como propriedade armazenadora de suas informações principais um objeto do tipo **DataTable**, então é comum utilizarmos o método ConsultarBanco(string comandoSQL) que retorna esse valor!

```
private void button1_Click(object sender, EventArgs e)
{
    string comandoSql = "SELECT * FROM Aluno" ;

    DataTable tabela = gerenciador.ConsultarBanco(comandoSql);
    dataGridView1.DataSource = tabela;
}
```

gerenciador.Consultar
Banco("SELECT *
FROM Aluno")



Matricula	Nome	Sexo
1	Eduardo	1

dataGridView.DataSource = tabela;



DateTimePicker Control



- Um objeto da classe **DateTimePicker** permite que o usuário selecione uma data específica.
- Um **DateTimePicker** armazena um objeto do tipo **DateTime** no atributo **Value**.

quarta-feira , 24 de outubro de 2018 ▼

outubro de 2018						
dom	seg	ter	qua	qui	sex	sáb
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Hoje: 24/10/2018

Atributos:

- Value: recupera a data que está selecionada. Retorna um objeto do tipo **DateTime**.

Eventos:

- ValueChanged: acontece quando o atributo **Value** do objeto muda.

DateTimePicker Control

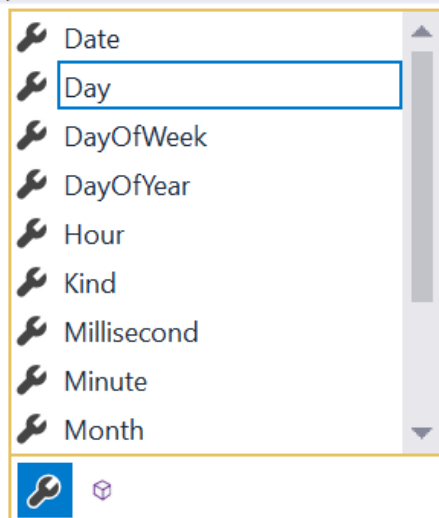
DateTime



- Um objeto da classe **DateTime** armazena dados relacionados a uma data e horário específicos. Seus atributos são bem intuitivos.

```
DateTime data = dateTimePicker1.Value;
```

data.

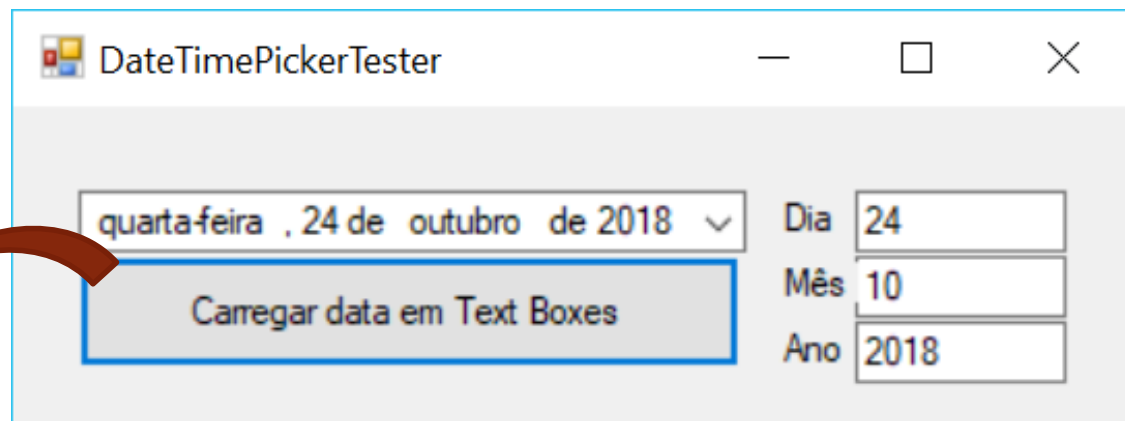


Atributos:

- Day: retorna o dia do objeto DateTime.
- Month: retorna o mês do objeto DateTime.
- Year: retorna o ano do objeto DateTime.
- Second: retorna os segundos do objeto DateTime.
- Minute: retorna os minutos do objeto DateTime.
- Hour: retorna as horas do objeto DateTime.

DateTimePicker Control

DateTime: *ler dados do DateTimePicker*



```
DateTime data = dateTimePicker1.Value;
```

```
textBox1.Text = Convert.ToString(data.Day);  
textBox2.Text = Convert.ToString(data.Month);  
textBox3.Text = Convert.ToString(data.Year);
```

DateTimePicker Control

DateTime

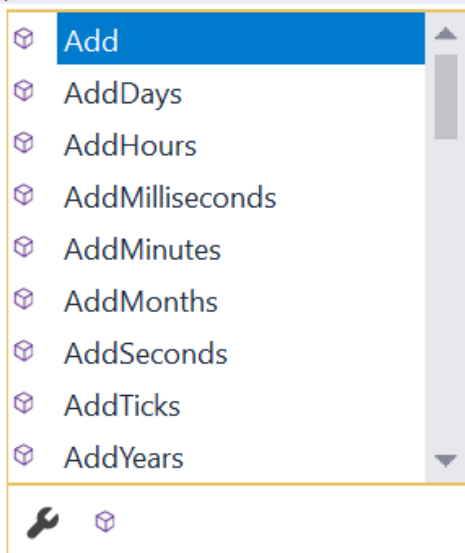


- Além de armazenar essas informações, há métodos que podem ser úteis para trabalhar com esses objetos que representam uma data e uma hora.

```
DateTime data = dateTimePicker1.Value;
```

Métodos:

data.



- AddDays: adiciona os dias deste objeto DateTime a outro que seja passado como parâmetro. Retorna um novo DateTime com o resultado especificado.
- AddMonth: adiciona os dias deste objeto DateTime a outro que seja passado como parâmetro. Retorna um novo DateTime com o resultado especificado.

DateTimePicker Control

DateTime



- Vale mencionar alguns métodos estáticos da classe DateTime:

```
bool bissexto = DateTime.IsLeapYear(1997);  
DateTime agora = DateTime.Now();  
DateTime hoje = DateTime.Today();
```

Retorna true se o ano é bissexto.

Retorna um DateTime contendo o horário atual do sistema.

Retorna um DateTime contendo a data e horário atual do sistema.

DateTimePicker Control

DateTime



- Exemplo: descobrindo a sua idade.

Form1

Data de Nascimento quarta-feira, 1 de agosto de 1990

Idade 28

```
private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{

}

}
```

DateTimePicker Control

DateTime



- Exemplo: descobrindo a sua idade.

Form1

Data de Nascimento quarta-feira, 1 de agosto de 1990

Idade 28

```
private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
    DateTime nascimento = dateTimePicker1.Value;
    DateTime hoje = DateTime.Today;

    int idade = hoje.Year - nascimento.Year;

    textBox1.Text = idade.ToString();
}
```

PictureBox Control



- O **PictureBox** é um controle utilizado para armazenar e exibir uma imagem no formulário.
- Para se armazenar uma imagem numa **PictureBox** basta passar um objeto do tipo **Image** para o seu atributo **Image**.

```
Image foto = Image.FromFile(caminhoDaFoto);  
pictureBox1.Image = foto;
```


PictureBox Control

Image



- **Image** é uma classe que fornece funcionalidades para se instanciar na memória imagens.
- Há classes que são filhas de **Image**, como Bitmap e Metafile. Todas elas são capazes de tratar imagens e manipular seus pixels, por exemplo.
- A classe **Image** pode ser encontrada no *namespace System.Drawing*.



PictureBox Control

Image



- Para se criar um objeto de imagem, basta chamar o método estático `FromFile()` da classe `Image` e passar como parâmetro o caminho da imagem.

```
Image foto = Image.FromFile(caminhoDaFoto);  
pictureBox1.Image = foto;
```

- Para associar esse objeto `Image` à `PictureBox` e, assim, exibí-la no formulário, basta realizar a atribuição do objeto ao atributo `Image`.

PictureBox Control

Image



- O caminho de onde a imagem está localizada pode ser encontrado da mesma maneira que buscávamos caminhos quando trabalhávamos com arquivos!

```
OpenFileDialog janela = new OpenFileDialog();
janela.Filter = "Imagens(.jpg, .png)|*.png;*.jpg";

DialogResult botaoClicado = janela.ShowDialog();
if (botaoClicado == DialogResult.OK)
{
    string caminhoDaImagem = janela.FileName;

    Image imagem = Image.FromFile(caminhoDaImagem);
    pictureBox1.Image = imagem;
}
```



O código ao lado irá criar uma OpenFileDialog para que possamos procurar o caminho de alguma imagem.

Armazenamento de Informações no banco de dados






- Como dados como **DataTable**, **DateTime** e **Image** são armazenados no banco de dados?!



Armazenamento de Informações no banco de dados






Controle	Nome	Atributo/Propriedade que armazena a informação	Tipo da informação
	TextBox	Text	string
	NumericUpDown	Value	Decimal
	ComboBox	Items	List<object>

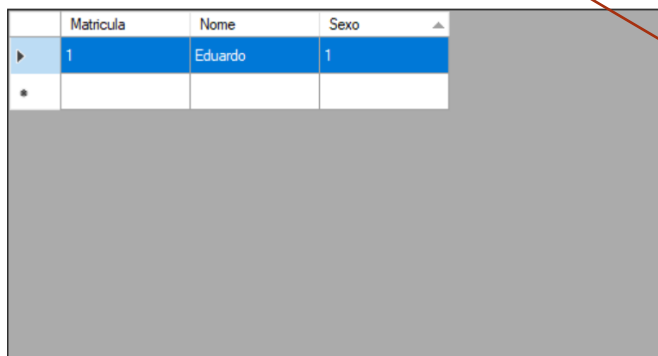
varchar

float

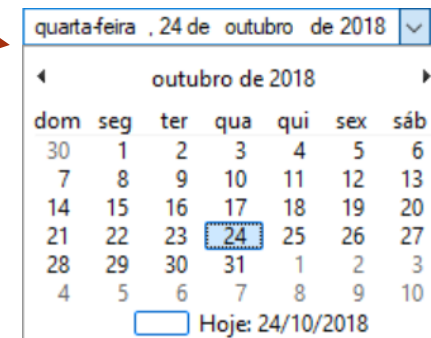
Armazenamento de Informações no banco de dados

varbinary(MAX) ← **datetime**

Controle	Nome	Atributo/Propriedade que armazena a informação	Tipo da informação
	DataGridView	DataSource	DataTable
	DateTimePicker	Value	DateTime
	PictureBox	Image	Image



Matricula	Nome	Sexo
1	Eduardo	1



Pergunta!



- Como **inserir** dados de **datetime** e **varbinary(MAX)** no banco de dados?
- Como **recuperar** dados de **datetime** e **varbinary(MAX)** do banco e usá-los na minha aplicação C#?



Tipo **datetime** no **SQL Server**

Comando SQL: INSERINDO



- Os comandos abaixo mostram a sintaxe que deve ser seguida para se inserir valores do tipo **datetime** no SQL Server.

```
INSERT INTO
```

```
Aluno(Matricula, Nome, Sexo, Turma, Foto, Data_de_Nascimento)
```

```
VALUES
```

```
(1234, 'Pedro', 'M', 7, NULL, CONVERT(DATETIME, '31/07/1985', 103));
```

```
UPDATE Aluno
```

```
SET Data_de_Nascimento = CONVERT(DATETIME, '31/07/1985', 103)
```

```
WHERE Matricula = 5;
```


Tipo **datetime** no **SQL Server**

Comando SQL: INSERINDO



- Por conta disso, devemos ter cuidado no momento de concatenarmos nossa instrução SQL na aplicação C#.

Exemplo

```
INSERT INTO
Aluno(Matricula, Nome, Sexo, Turma, Foto, Data_de_Nascimento)
VALUES
(1234, 'Pedro', 'M', 7, NULL, CONVERT(DATETIME, '31/07/1985', 103));
```



```
string data = dateTimePicker.Value.ToString();
string instrução = "INSERT INTO
Aluno(Matricula, Nome, Sexo, Turma, Foto, Data_de_Nascimento)
VALUES
(1234, 'Pedro', 'M', 7, NULL, CONVERT(DATETIME, '"+ data +"', 103))";
```

Tipo **datetime** no **SQL Server**

Comando SQL: INSERINDO



- Por conta disso, devemos ter cuidado no momento de concatenarmos nossa instrução SQL na aplicação C#.

Exemplo

```
UPDATE Aluno  
SET Data_de_Nascimento = CONVERT(DATETIME, '31/07/1985', 103)  
WHERE Matricula = 5;
```



```
string data = dateTimePicker.Value.ToString();  
string instrução =  
"UPDATE Aluno  
SET Data_de_Nascimento = CONVERT(DATETIME, '"+ data +"', 103)  
WHERE Matricula = 5";
```

Tipo **varbinary** no **SQL Server**

Comando SQL: INSERINDO



- Os comandos abaixo mostram a sintaxe que deve ser seguida para se inserir valores do tipo **varbinary** no SQL Server.

```
INSERT INTO Aluno (Nome, Foto) VALUES  
( 'Lucas', (SELECT * FROM OPENROWSET(BULK  
N'C:/caminho/da/minha/foto.png', SINGLE_BLOB) AS PhotoImage))
```



Isso é necessário pois sua imagem é um arquivo binário! Portanto ela precisa ser armazenada no banco de uma maneira especial, ao contrário de strings ou números.

Uma informação armazenada no tipo **varbinary** – o que significa quantidade variável de binários – é, essencialmente, um vetor de bytes (byte[]).

Tipo varbinary no **SQL Server**

Comando SQL: INSERINDO



- Por conta disso, devemos ter cuidado no momento de concatenarmos nossa instrução SQL na aplicação C#.

Exemplo

```
INSERT INTO Aluno (Nome, Foto) VALUES
('Lucas', (SELECT * FROM OPENROWSET(BULK
N'C:/caminho/da/minha/foto.png', SINGLE_BLOB) AS PhotoImage))
```



```
string auxSql = @"(SELECT * FROM OPENROWSET(BULK N'" +
caminhoImagem + "', SINGLE_BLOB) AS PhotoImage)";

string nome = txtNome.Text;

string sqlStatement = "INSERT INTO Aluno (Nome, Foto) VALUES ('" +
nome + "'," + auxSql + ")";
```

Tipo varbinary no **SQL Server**

Comando SQL: INSERINDO



- Note que o caminho de onde a imagem se encontra pode ser obtido através de uma **OpenFileDialog**.

```
OpenFileDialog janela = new OpenFileDialog();  
janela.Filter = "Imagens(.jpg, .png)|*.png;*.jpg";
```

```
DialogResult botaoClicado = janela.ShowDialog();
```

```
if (botaoClicado == DialogResult.OK)  
    caminhoImagem = janela.FileName;
```

```
string auxSql = @"(SELECT * FROM OPENROWSET(BULK N'" + caminhoImagem + "',  
SINGLE_BLOB) AS PhotoImage)";
```

```
string nome = txtNome.Text;
```

```
string sqlStatement = "INSERT INTO Aluno (Nome, Foto, Foto) VALUES ('" +  
nome + "'," + auxSql + ")";
```

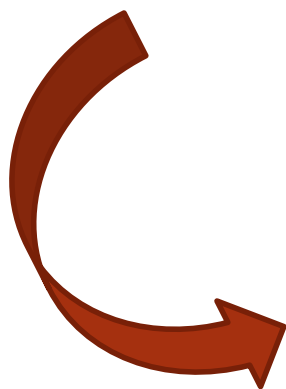
Tipo **datetime** no **SQL Server**

Comando SQL: RECUPERANDO



- Para se recuperar informação de um banco de dados já sabemos como fazer. Basta escrever a consulta **SELECT** e pedir as colunas.

```
SELECT Nome, Data_de_Nascimento FROM Aluno
```



Nome	Data_de_Nascimento
Maria	09/10/2000
Pedro	12/04/1997
Lucas	15/06/2002
Leonardo	11/01/2001

Tipo **datetime** no **SQL Server**

Comando SQL: RECUPERANDO



- Fizemos a consulta, recuperamos a DataTable associada ao resultado do SELECT e agora iremos converter o valor armazenado em linha["Data_de_Nascimento"] para um objeto DateTime.

```
DataTable tabela = gerenciador.ConsultarBanco("SELECT Nome,  
Data_de_Nascimento FROM Aluno");  
  
int idadeTotal = 0;  
  
foreach (DataRow linha in tabela.Rows)  
{  
    DateTime nascimento =(DateTime) linha["Data_de_Nascimento"];  
    int idade = DateTime.Today.Year - nascimento.Year;  
  
    //Fazer alguma coisa com a idade...  
}
```

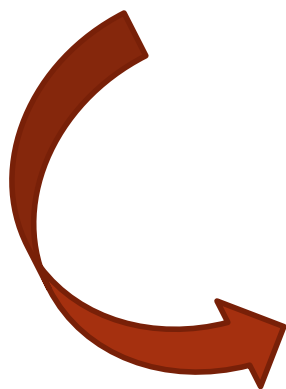
Tipo varbinary no **SQL Server**

Comando SQL: RECUPERANDO



- Para se recuperar informação de um banco de dados já sabemos como fazer. Basta escrever a consulta SELECT e pedir as colunas.

```
SELECT Nome, Foto FROM Aluno
```



Nome	Foto
Maria	0xFFA004829FFB10...
Pedro	0xFFA004829FFB10...
Lucas	0xFFA004829FFB10...
Leonardo	0xFFA004829FFB10...

Tipo varbinary no **SQL Server**

Comando SQL: RECUPERANDO



- Fizemos a consulta, recuperamos a DataTable associada ao resultado do SELECT e agora iremos converter o valor armazenado em linha["Foto"] para um objeto do tipo Image.

```
DataTable tabela = gerenciador.ConsultarBanco("SELECT
Nome, Foto FROM Aluno WHERE Matricula = " + matricula);

if (tabela != null)
{
    DataRow linha = tabela.Rows[0];

    string Nome = linha["Nome"].ToString();

    byte[] imagemBinaria = (byte[]) linha["Foto"];
    MemoryStream ms = new MemoryStream(imagemBinaria);
    Image imagemReal = Image.FromStream(ms);

    pctPersonagem.Image = imagemReal;
}
```



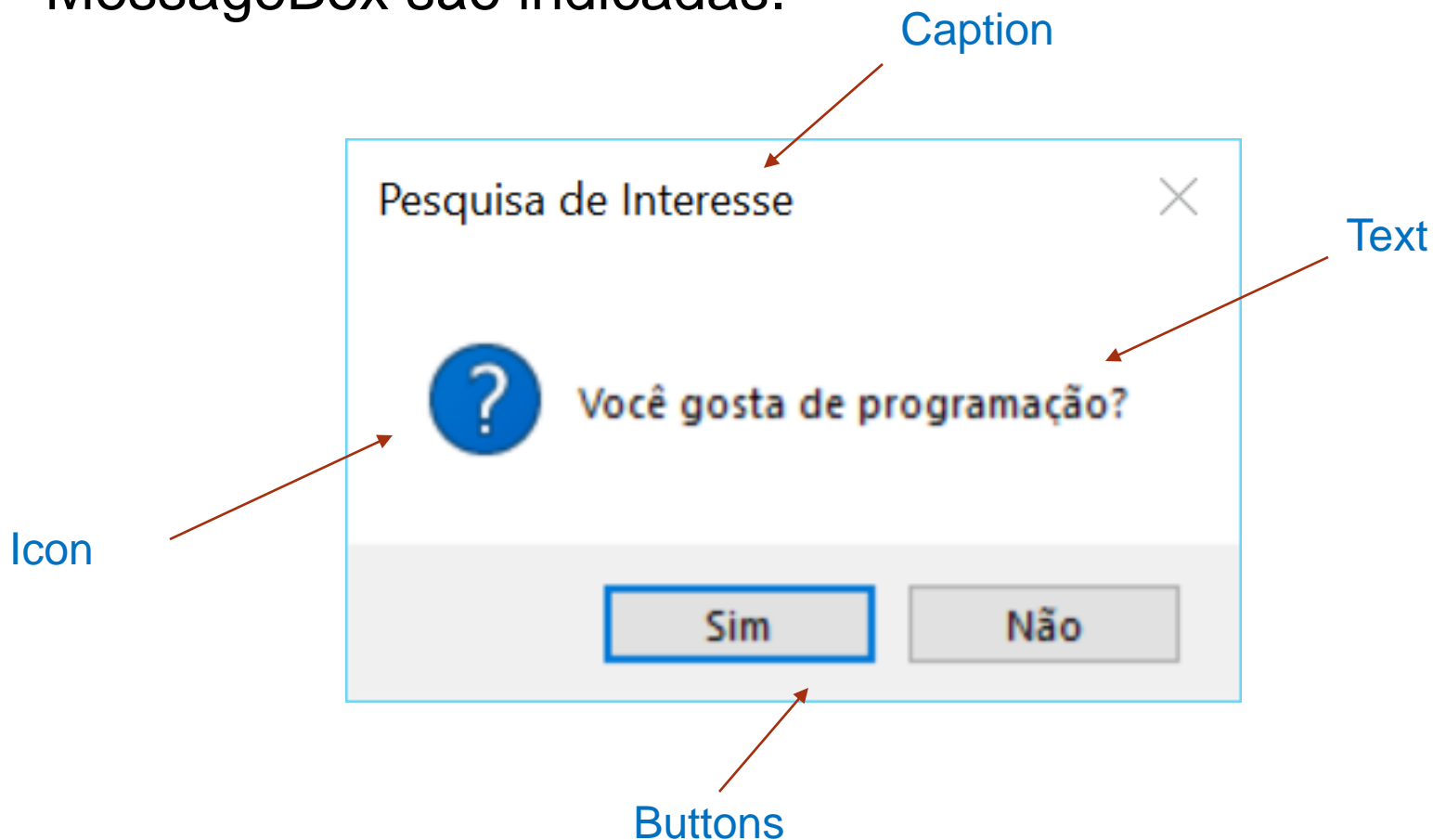
MessageBox



- Muitas vezes, numa aplicação, você irá querer transmitir alguma informação para o seu usuário.
- Essa informação pode ser uma mensagem de cortesia como “**Novo registro inserido!**” ou uma pergunta requerendo um feedback “**Você realmente quer deletar esse registro?**”.
- Visual C# provê um excelente mediador para se realizar esse tipo de tarefa através da classe **MessageBox**.

MessageBox

- No exemplo abaixo, as diferentes partes de uma MessageBox são indicadas.



MessageBox

- Para usar uma MessageBox você tem que chamar o método estático Show e passar como parâmetro para eles as informações de:

- Text
 - Caption
 - Icon
 - Buttons
- string**
- MessageBoxIcon**
- MessageBoxButtons**
- São enumerações pré-definidas para MessageBox

```
MessageBox.Show(Text, Caption, Buttons, Icon);
```

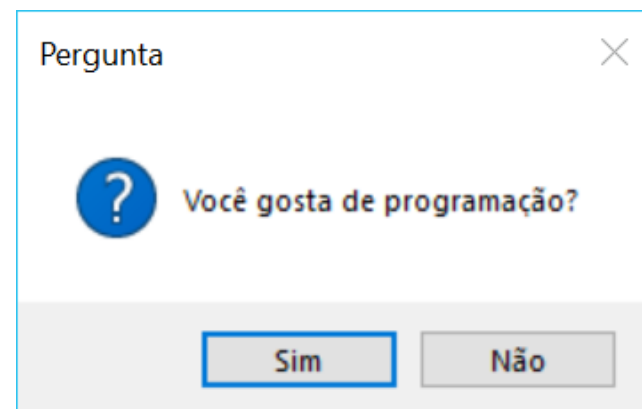
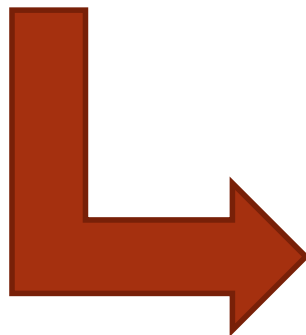
MessageBox



- Exemplos:

```
MessageBox.Show(Text, Caption, Buttons, Icon);
```

```
MessageBox.Show("Você gosta de programação?", "Pergunta",  
                MessageBoxButtons.YesNo, MessageBoxIcon.Question);
```



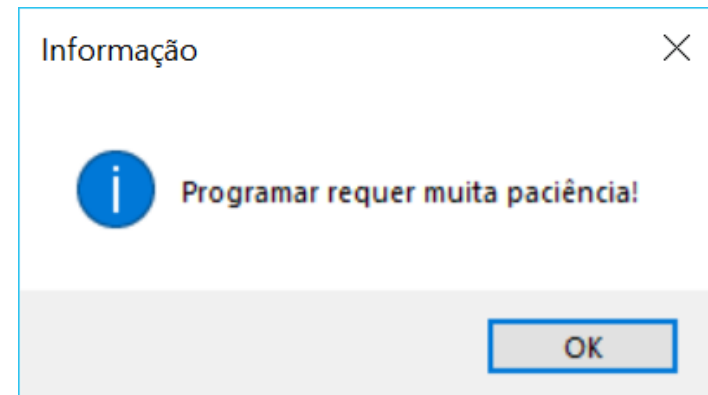
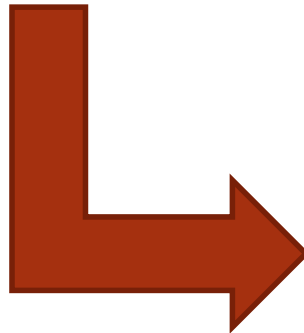
MessageBox



- Exemplos:

```
MessageBox.Show(Text, Caption, Buttons, Icon);
```

```
MessageBox.Show("Programar requer muita paciência!", "Informação",  
                MessageBoxButtons.OK, MessageBoxIcon.Information);
```



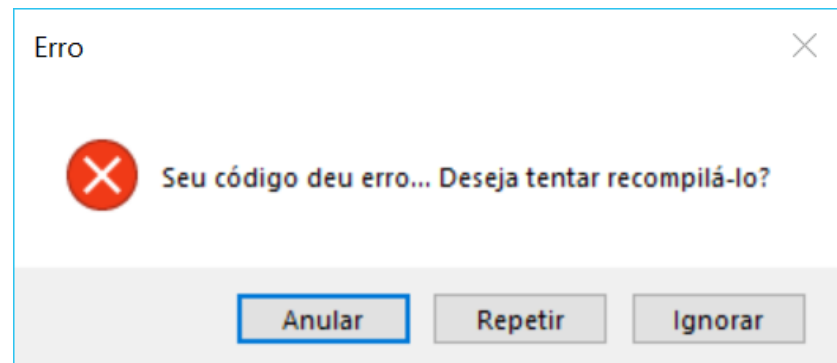
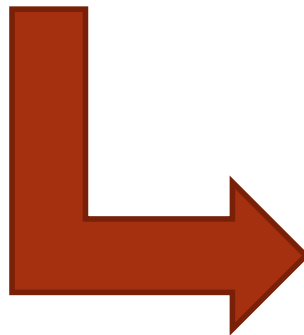
MessageBox



- Exemplos:

```
MessageBox.Show(Text, Caption, Buttons, Icon);
```

```
MessageBox.Show("Seu código deu erro... Deseja tentar recompilá-lo?", "Erro",  
                MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);
```



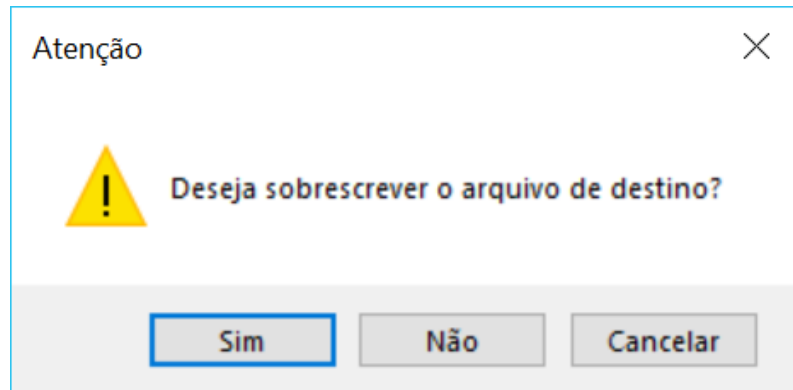
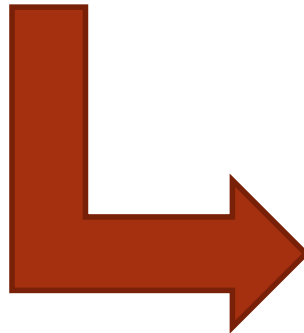
MessageBox



- Exemplos:

```
MessageBox.Show(Text, Caption, Buttons, Icon);
```

```
MessageBox.Show("Deseja sobrescrever o arquivo de destino?", "Atenção",  
    MessageBoxButtons.YesNoCancel, MessageBoxIcon.Exclamation);
```



MessageBox



- A classe MessageBox é sobrecarregada com vários métodos Show. Os mais comuns são:

- `MessageBox.Show(Text);`
- `MessageBox.Show(Text, Caption);`
- `MessageBox.Show(Text, Caption, Buttons);`
- `MessageBox.Show(Text, Caption, Buttons, Icon);`

No exemplo usamos esta sobrecarga!

- Os dois primeiros argumentos do método são definem qual texto será utilizado e qual legenda a MessageBox apresentará. São eles: **Text** e **Caption** (*strings*). Os próximos argumento do método são definidos por enumerações no .NET.
- **Buttons** pode assumir os seguintes valores de enumeração:
 - `AbortRetryIgnore`, `OK`, `OKCancel`, `RetryCancel`, `YesNo`, `YesNoCancel`.
- **Icons** pode assumir os seguintes valores de enumeração:
 - `IconAsterisk`, `IconInformation`, `IconError`, `IconHand`, `IconNone`, `IconStop`, `IconExclamation`, `IconWarning`, `IconQuestion`

MessageBox



- Quando o método Show é chamado a MessageBox é exibida na tela e o usuário então pode interagir com seus botões.
- O método retorna uma variável do tipo DialogResult dependendo de onde o usuário clicou. A variável DialogResult pode assumir os valores:
 - **Abort, Cancel, Ignore, No, OK, Retry, Yes**

```
DialogResult resultado = MessageBox.Show("Você gosta de programação?",  
"Pergunta", MessageBoxButtons.YesNo, MessageBoxIcon.Question);  
  
if (resultado == DialogResult.Yes)  
    MessageBox.Show("Boa! Programar sempre aprimora nosso  
                    raciocínio lógico!");  
else  
    MessageBox.Show("Que pena! :/");
```