Técnicas de Conexão em Banco de Dados ActiveX Data Objects (**ADO**)

Técnicas de Programação / Linguagem de Programação / Banco de Dados

Professor Antonio Marcos Alvarez

Vamos contextualizar:

O banco de dados sempre foi um ponto forte na aplicação, uma possível falha na modelagem (preparação) pode prejudicar o desempenho da própria aplicação e do processo como um todo, portanto, é fundamental o conhecimento para que possa preparar da melhor forma possível, tal como dimensionar a solução a ser adotada.

Logo no início, para se ter acesso às informações do banco de dados era necessário ter vasto conhecimento das API (**A**pplication **P**rogramming **I**nterface) de comunicação, e o programador acabava implementando seu código exclusivamente para cada versão de driver como a DBLIB (SQL Server), dedicando com isso muito tempo a este tipo de implementação.

Com o crescimento do mercado e a persistência desse problema foi criado em 1990 com apoio da Microsoft e um consórcio de empresas, o padrão ODBC (**O**pen **D**ata**B**ase **C**onnectivy), sendo uma camada intermediária encarregada de cuidar da comunicação com a API deixando para o programador uma interface única e padrão para todo acesso ao SGDB.

Origem do OLEDB (**O**bject **L**inking and **E**mbedding **D**ata**B**ase)

A partir do sucesso do ODCB e necessidade de evolução foram surgindo outras propostas como o DAO (Data Access Objects), sendo substituído logo depois pelo RDO (Remote Data Objects).

O grande avanço da época deu-se em torno do OLEDB que se assemelhou muito com a arquitetura do ODCB porém trouxe a implementação de interfaces COM (Component Object Model) e a estratégia da Microsoft UDA (Universal Data Access) com objetivo de armazenamento distribuído, semelhante ao ODBC, o OLEDB também foi sucesso sendo aderido até por banco de dados de padrão aberto.

Para facilitar sua utilização foi criado o ADO (Activex Data Objects) com objetivo de consumir os recursos oferecidos pelos OLEDB.

Processo de Conexão (ADO)

Etapas:

1) Identificar SGBD;
2) Localizar o “Provider” adequado para esta conexão;
3) Verificar a existência do drive adequado para a realização da conexão em sua plataforma de desenvolvimento e aplicação;
4) Identificar o path (caminho) correto para acesso ao Banco de Dados;
5) Descrever as necessidades (processos), relações e ações relativas as operações com as tabelas e seus registros.

Processo básico para conexão ADO

1) Crie um objeto de **conexão** para se **conectar** ao banco de dados;
2) Crie um objeto de conjunto de registros para receber dados;
3) Abra a **conexão**.
4) Após, efetue as operações necessárias, conforme descrito na documentação dos processos. Exemplos: Instruções e Comandos SQL.

C# (C Sharp) / .NET Framework

Principais **“Namespaces”** (inserir com “**using**”):

✓ Acesso a dados para o Microsoft SQL Server: “System.Data.SqlClient”;

✓ Fontes de dados expostas usando o OLE DB: “System.Data.OleDb”;

✓ Fontes de dados expostas usando ODBC: “System.Data.Odbc”;

✓ Fontes de dados Oracle, cliente Oracle(v 8.1.7, e superiores): “System.Data.OracleClient”;

✓ Acesso a dados para aplicativos de Modelo de Dados de Entidade (EDM): “System.Data.EntityClient”;

✓ Fornece acesso a dados para o Microsoft SQL Server Compact 4,0: “System. Data. SqlServerCe”.

Principais **“Objetos”** de provedores de Dados e suas Classes:

Connection: Estabelece uma conexão com uma fonte de dados específica. A classe base para todos os objetos de “Connection” é a classe “**DbConnection**”;

Command: Executa um comando em uma fonte de dados. Expõe “Parameters” e pode ser executado no escopo de uma “Transaction” de “Connection”. A classe base para todos os objetos de “Command” é a classe “**DbCommand**”;

DataReader: Ler um fluxo de dados apenas de encaminhamento e somente leitura de uma fonte de dados. A classe base para todos os objetos de “DataReader” é a classe “**DbDataReader**”;

DataAdapter: Preenche um “DataSet” e resolve atualizações com a fonte de dados. A classe base para todos os objetos de “DataAdapter” é a classe “**DbDataAdapter**”.

C# (C Sharp) / .NET Framework

Outras Classes:

Transaction: Insere comandos nas transações na fonte de dados. A classe base para todos os objetos de “Transaction” é a classe “**DbTransaction**”. O ADO.NET também fornece suporte para transações usando classes no namespace “**System.Transactions**”;

CommandBuilder: Um objeto auxiliar que gera automaticamente propriedades de comando de um “DataAdapter” ou deriva informações de parâmetro de um procedimento armazenado e preenche a coleção “Parameters” de um objeto “Command”. A classe base para todos os objetos de “CommandBuilder” é a classe “**DbCommandBuilder**”;

ConnectionStringBuilder: Um objeto auxiliar que fornece uma maneira simples de criar e gerenciar o conteúdo de cadeias de conexão usadas por objetos de “Connection”. A classe base para todos os objetos de “ConnectionStringBuilder” é a classe “**DbConnectionStringBuilder**”;

Parameter: Define os parâmetros de valores de entrada, saída e retorno para comandos e procedimentos armazenados. A classe base para todos os objetos de “Parameter” é a classe “**DbParameter**”;

Exception: Retornado quando um erro é encontrado na fonte de dados. Para um erro encontrado no cliente, .NET Framework provedores de dados geram uma exceção .NET Framework. A classe base para todos os objetos de “Exception” é a classe “**DbException**”;

Error: Expõe as informações de um aviso ou erro retornado por uma fonte de dados.

ClientPermission: Fornecido para .NET Framework atributos de segurança de acesso ao código do provedor de dados. A classe base para todos os objetos de “ClientPermission” é a classe “**DBDataPermission**”;

Exemplo de Conexão (Preparando o Ambiente)

Seguindo Tópicos (Pag. 5) – Conexão com Banco de Dados (ACCESS)

```
//Montar uma String de Conexão, Usamos o @ para que a variável (string) aceite os "\"
string strCon = @"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\Users\User\Desktop\DataBase\Banco\base.mdb";
```

Observar o provedor registrado em seu ambiente de desenvolvimento e aplicação, exemplos:

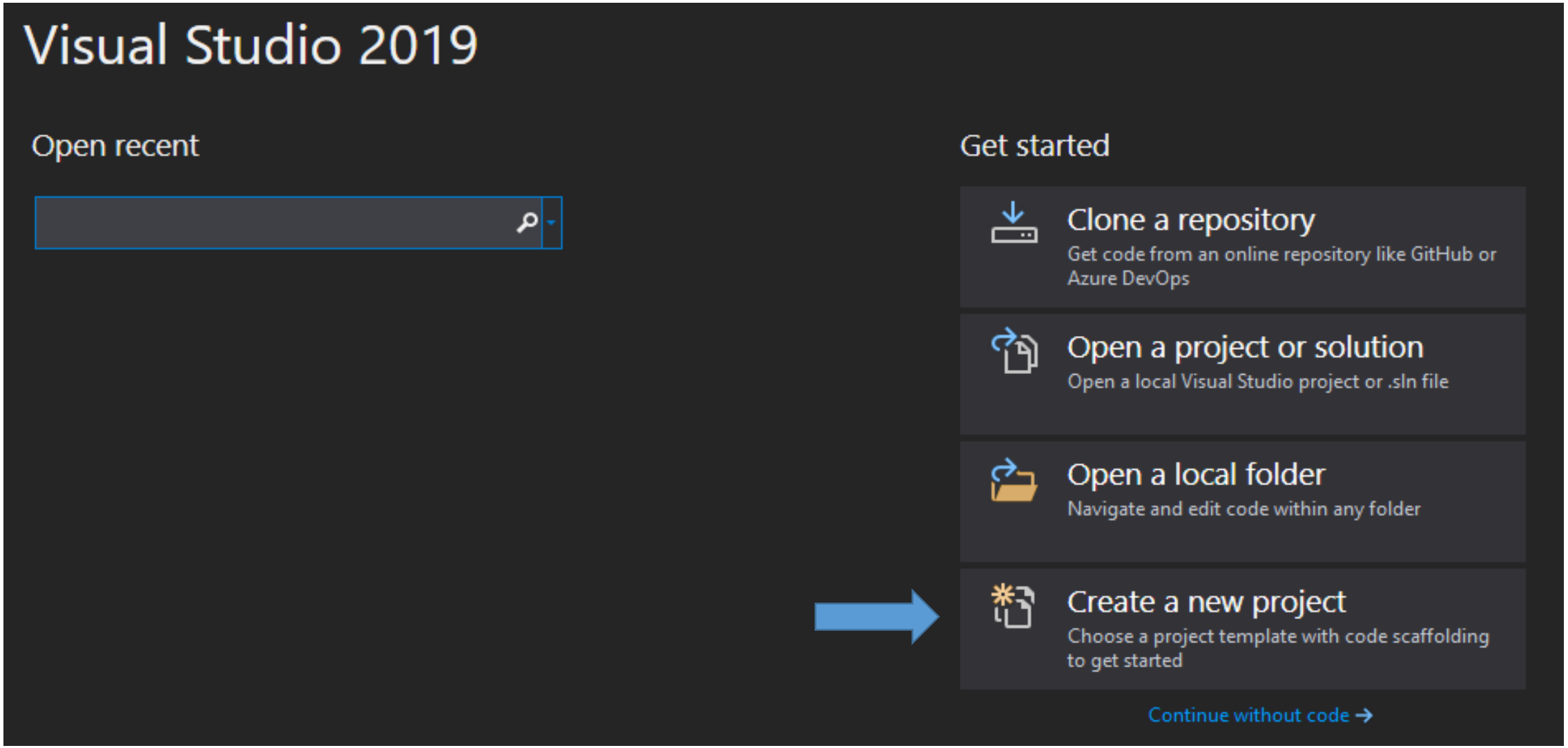
- ✓ Microsoft.ACE.OLEDB.12.0 >> Para acesso de Banco de Dados ambiente 64bits
- ✓ Microsoft.Jet.OLEDB.4.0 >> Para acesso de Banco de Dados ambiente 32Bits

```
//Criar e Estanciar Objeto da Classe de Conexão com o Banco de Dados
OleDbConnection conectar = new OleDbConnection(strCon);

//Criar o Objeto da Classe de Comando para armazenar a (Query) comando SQL
OleDbCommand comandoSQL = new OleDbCommand("select * from Contatos", conectar);
```

Exemplo de Conexão (Abrindo DB e Trabalhando com os Registros)

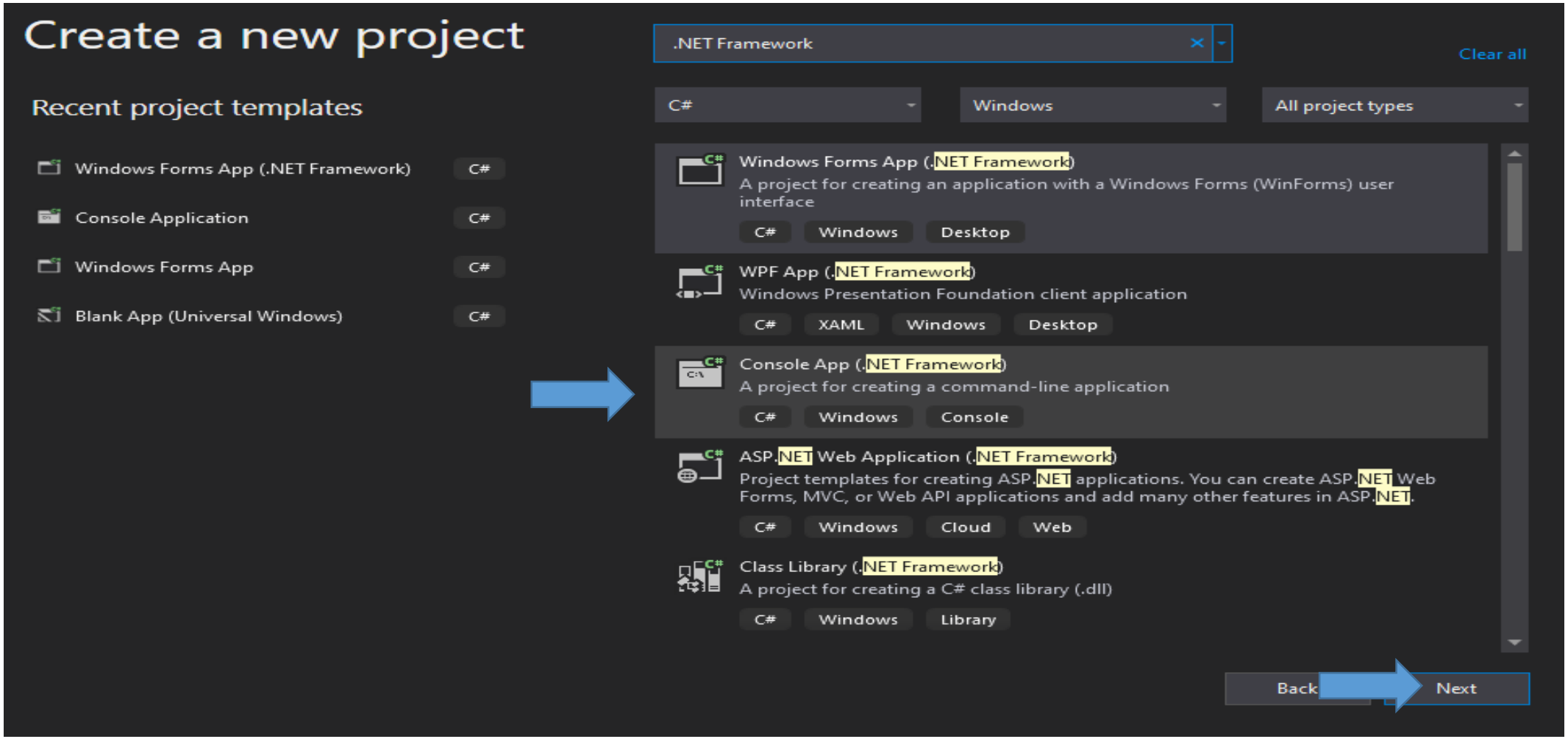
```
//Abrir Objeto da Classe de Conexão (OleDbConnection)
conectar.Open(); //Open, método da classe

//Cria o Objeto da Classe de "DataReader", acesso aos Registros "tabela(s)"
DbDataReader lerDados = comandoSQL.ExecuteReader();

//Faz a interação (Leitura) com o banco de dados lendo os dados da tabela
lerDados.Read() //Read, método da classe
```

Vamos iniciar nosso Projeto (Ex_ConectarBanco01)



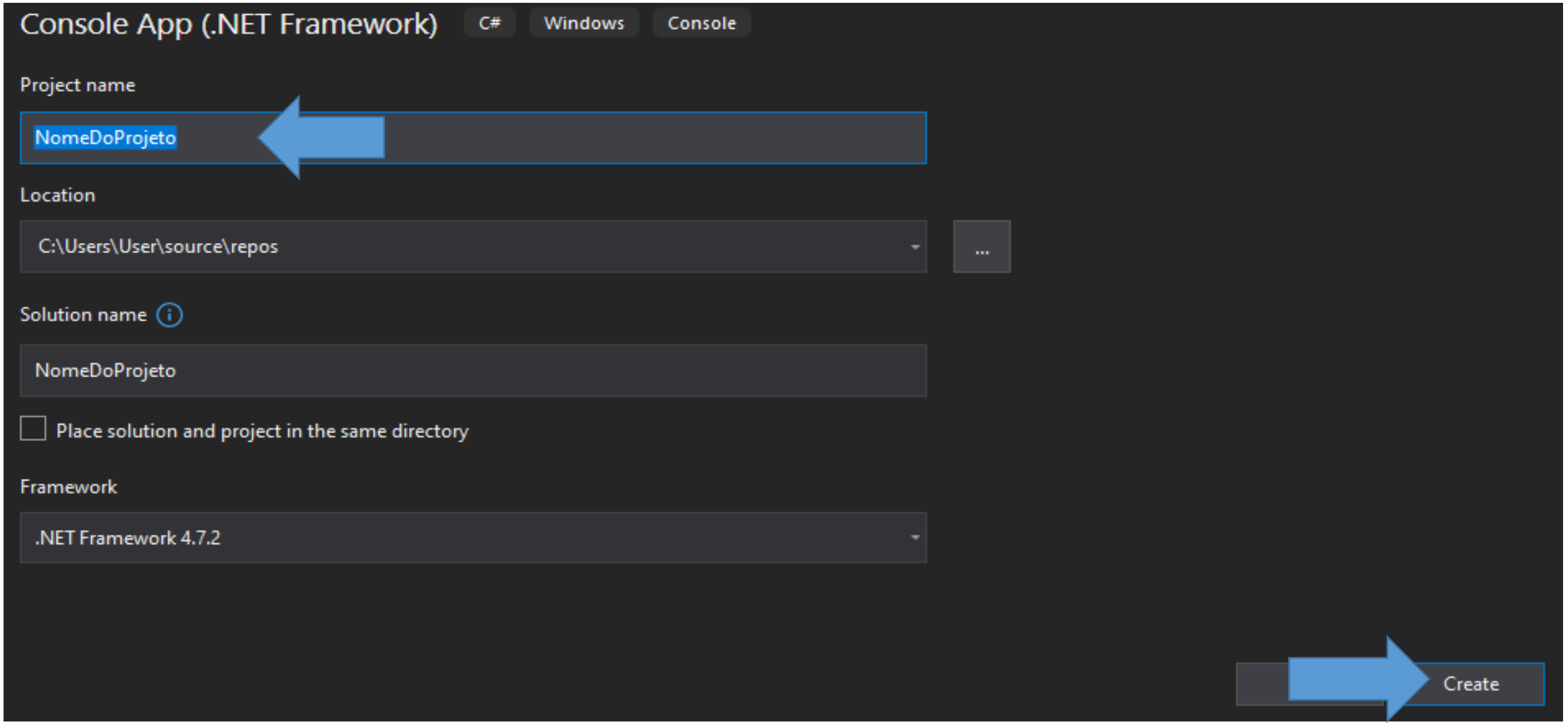
Para Iniciar nosso novo projeto, usamos a opção “**Create a new Project**”

Escolhendo os .NET Framework e ambiente para o Desenvolvimento



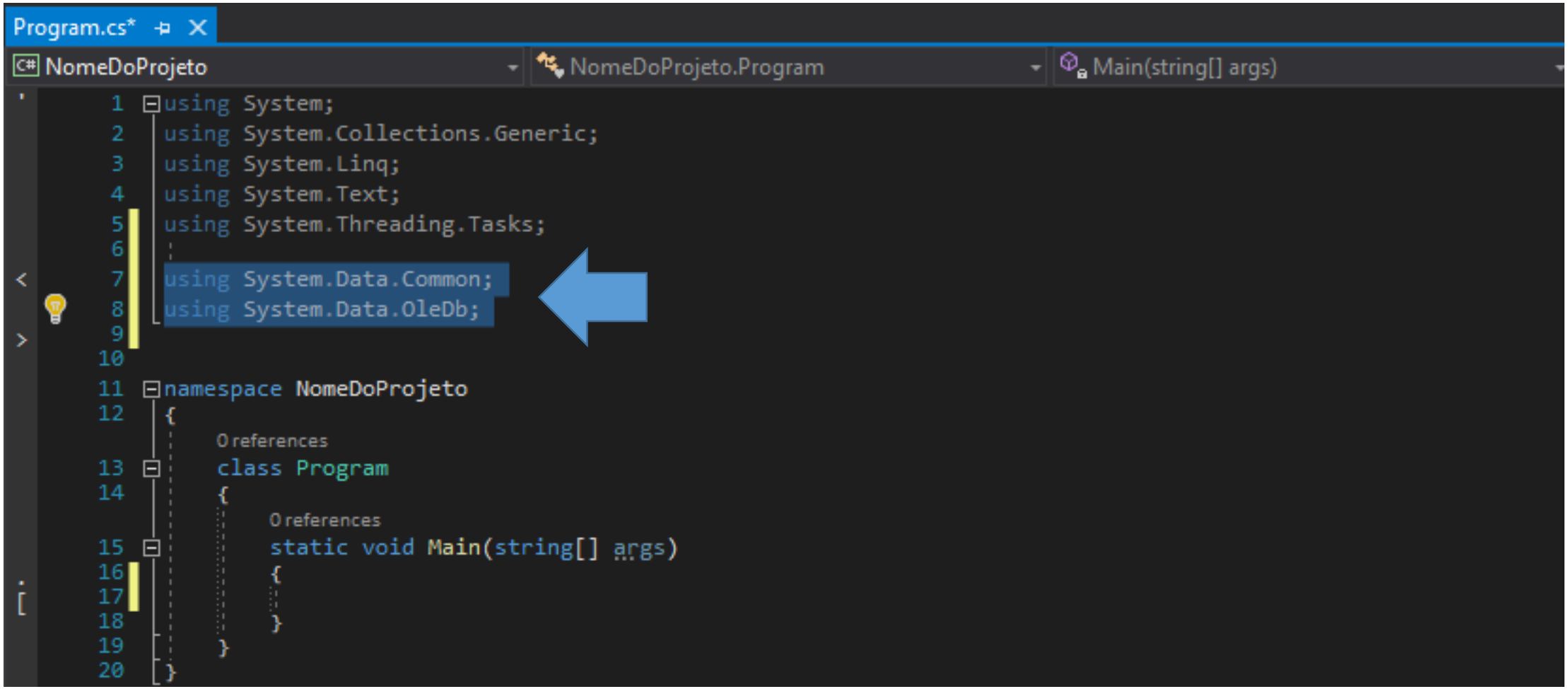
- Buscar o “Project Templates” >> “Console App (.NET Framework)”

“Namespace” do Projeto



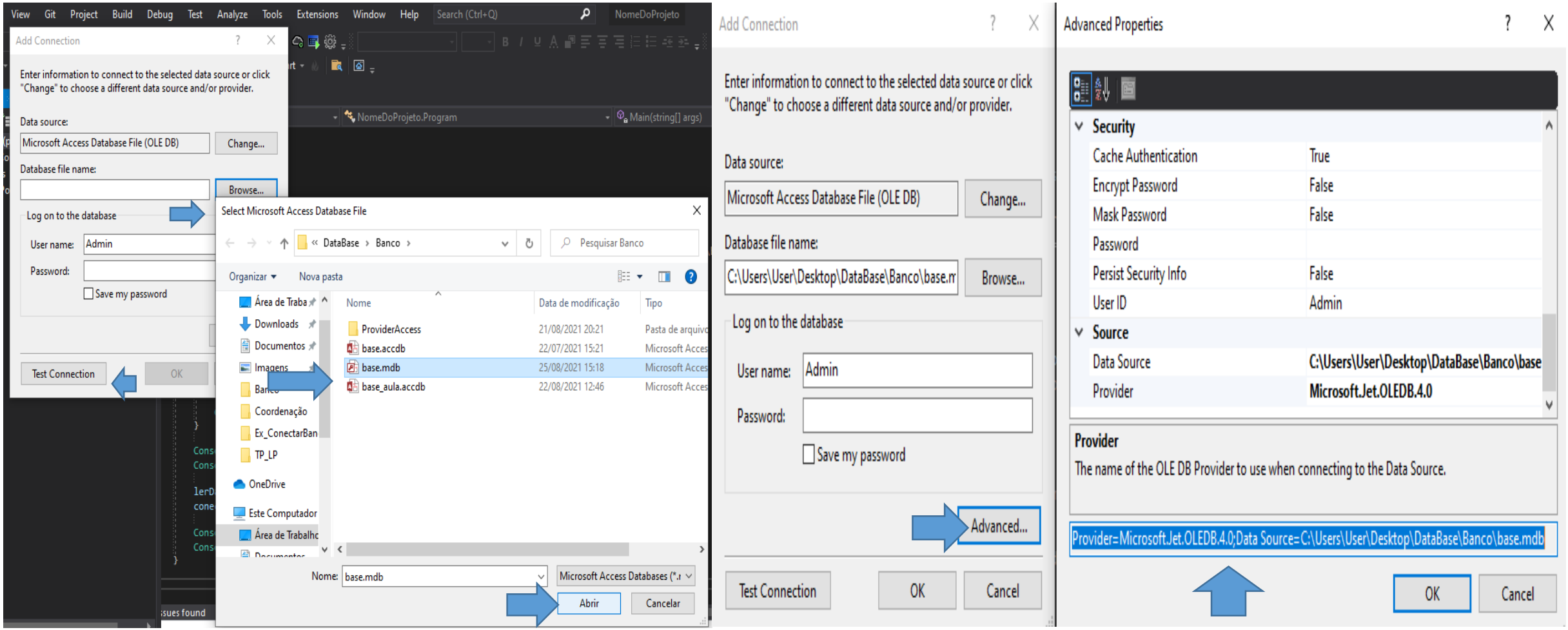
Inserir o Nome de seu Projeto (Nome do Aplicativo – Namespace), depois clicar em “Create”

Iniciando Projeto (Codificação)



using System.Data.Common;
using System.Data.OleDb;

Para Facilitar a montagem da “String” de Conexão, podemos usar o recurso do “Visual Studio Community 2019”, Opções: Tools / Connect to Database



Com este método, podemos selecionar e copiar a “string” de conexão (pronta) para inserirmos em nosso código.

Inserindo nosso código em “static void Main((Main)” (Função Principal do Nosso Projeto)

```
static void Main(string[] args)
{
    string strCon = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Users\User\Desktop\DataBase\Banco\base.mdb";
    OleDbConnection conectar = new OleDbConnection(strCon);
    OleDbCommand comandoSQL = new OleDbCommand("select * from Contatos", conectar);

    conectar.Open();

    DbDataReader lerDados = comandoSQL.ExecuteReader();
    Console.WriteLine("Os valores retornados da tabela são : ");

    while (lerDados.Read())
    {

        Console.WriteLine("Nome:" + lerDados.GetString(1) + " | Telefone:" + lerDados.GetString(2));
    }

    Console.WriteLine("Final dos Registros! Pressionar Qualquer Tecla...");
    Console.ReadLine();

    lerDados.Close();
    conectar.Close();

    Console.WriteLine("Tabela e Conexão Fechados...");
    Console.ReadLine();
}
```

Observe que neste exemplo usamos o provedor (Provider):

✓ Microsoft.Jet.OLEDB.4.0 >> Para acesso de Banco de Dados ambiente 32Bits

Nosso Banco foi gravado com formato para versões inferiores a 2007, em 32Bits e extensão (MDB).

Código: Ex_ConectarBanco01

Segue anexo a pasta com o projeto inteiro, analisar e testar seu funcionamento, observe que o provedor (Provider) está com:

Microsoft.ACE.OLEDB.12.0 >> Para acesso de Banco de Dados ambiente 64bits

Se em seu equipamento não estiver registrado este provedor, você deverá trocar pelo que estiver registrado em seu ambiente de trabalho (desenvolvimento).

Observe também o local correto onde está armazenado seu banco de dados (BASE.MDB), inserindo o local (path) correto na "String" de conexão, na propriedade de "Data Source=".

Se nenhum provedor estiver registrado, poderá baixa-los e instala-los através do link abaixo:

Access 2010: https://www.microsoft.com/en-us/download/details.aspx?id=13255

Access 2016: https://www.microsoft.com/en-us/download/details.aspx?id=54920