

Desenvolupament d'interfícies

Unitat 5. Accés a base de dades. Dades i controls



Índex

1.	Introducció	2
2.	Origen o font de les dades	2
2.1.	Instal·lació de SQL Server EXPRESS.	3
2.2.	Instal·lació de l'eina d'Administració de SQL Server.	12
2.3.	Configuració i ús de l'eina d'Administració de SQL Server.....	15
2.4.	Administració d'una base de dades des d'un Projecte WPF.....	18
3.	Accés a dades amb WPF.....	25
3.1.	Connectar, consultar i mostrar.....	25
3.2.	Enllaçar dades a controls WPF.....	28
3.3.	Ordenant el codi, utilitzant una classe per al model.	31
4.	Taules: configuració del control DataGridView.....	34
4.1.	Autogeneració de columnes.	34
4.2.	Definició manual.....	35
4.3.	Ordenar i redimensionar.....	37
4.4.	Agrupar.....	37
4.5.	Detalls d'una fila.	38
4.6.	Altres característiques.....	39
5.	Llistes: configuració del control ListBox	40
5.1.	Ordenació	40
5.2.	Agrupar	42
6.	CRUD amb WPF	42
7.	Annex: Exportació d'Esquema i Dades de SQL Server	47

1. Introducció

En aquest punt continuarem estudiant els controls WPF més comuns i començarem a utilitzar altres que ens seran molt útils per a posar un poc d'ordre i trellat a aplicacions amb moltes opcions i més complexitat que s'han de resoldre amb una única finestra.

És a dir, contemplarem l'ús de menús, pestanyes, etc...

I recordeu, que quan menys text haja d'escriure l'usuari, menys errors de dades es produiran.

2. Origen o font de les dades

Per a poder accedir a una bases de dades el primer que s'ha de fer és tindre una, tindre una font de dades a la que accedir.

En aquesta ocasió treballarem amb una versió reduïda i gratuïta de SQL Server anomenada SQL Server EXPRESS.

En la següent imatge podem observar les diferències amb el seu germà major SQL Server.

Characteristics	SQL Server 2019 Standard	SQL Server 2019 Express
Maximum number of Cores	24 Cores	4 Cores
Storage: Maximum buffer pool memory per Instance	128 GB	1410 MB
Storage: Maximum cache segment column storage per instance	32 GB	352 MB
Storage: Maximum Data with memory per Database optimization	32 GB	352 MB
Maximum Database Size	524 GB	10 GB
RAM memory for storing caches of information	Unlimited, Will depend on the Servers' RAM	Maximum 1,5 GB

No obstant per a moltes aplicacions que xicotet tamany i poca demanda de recursos, SQL Server EXPRESS és una opció molt més que acceptable.

2.1. Instal·lació de SQL Server EXPRESS.

Per a començar l'instal·lació el primer que hem de fer és descarregar el producte. Per a fer això obrim un navegador i visitem al url <https://www.microsoft.com/es-es/sql-server/sql-server-downloads>

I baixant cap a baix de pàgina trobem l'enllaç corresponent al SQL Server EXPRESS:

O bien, descarga una edición especializada gratuita



Desarrollador

SQL Server 2019 Developer es una edición gratuita con todas las características que se puede usar como base de datos de desarrollo y pruebas en un entorno que no sea de producción.

Descargar ahora >



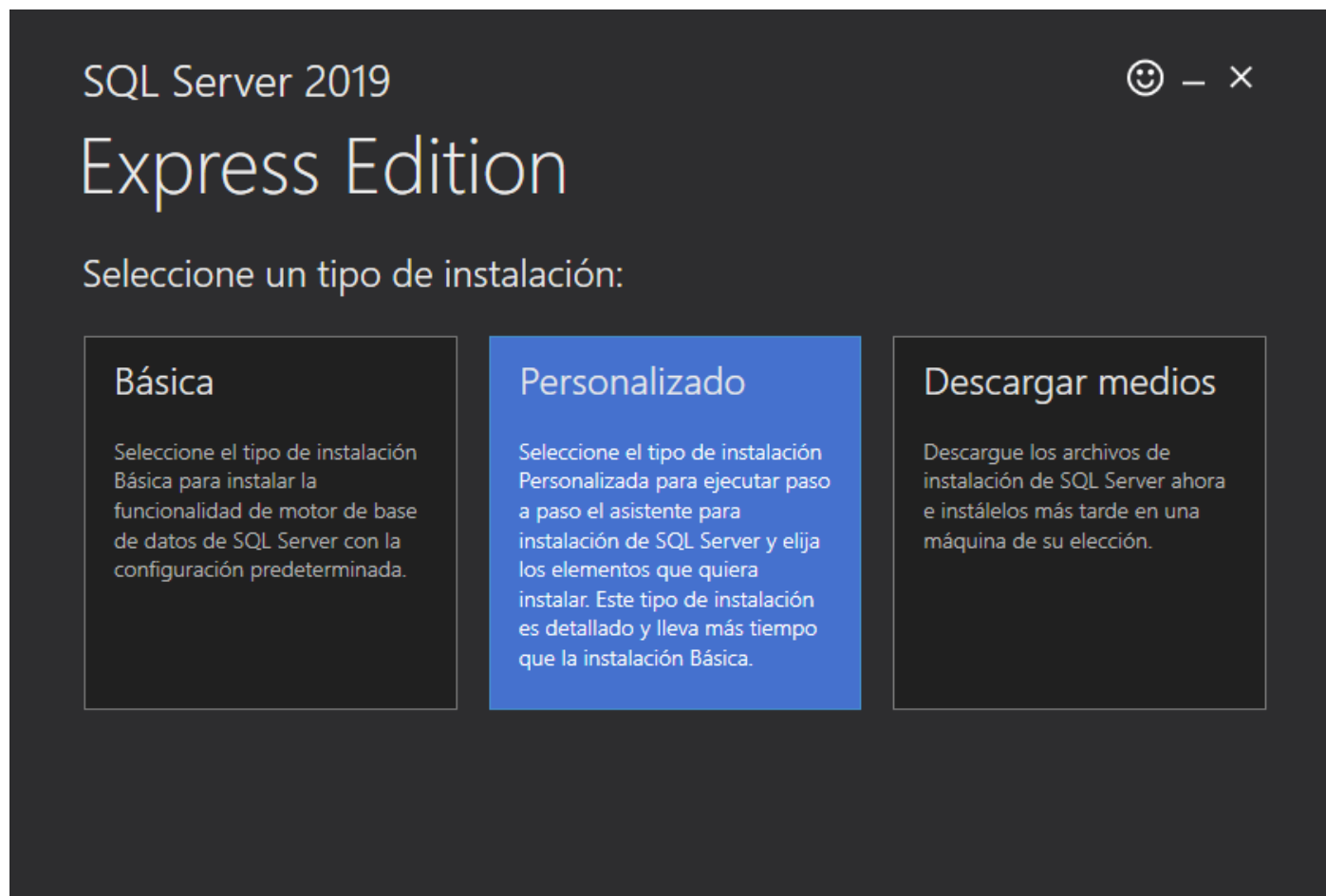
Express

SQL Server 2019 Express es una edición gratuita de SQL Server ideal para el desarrollo y la producción de aplicaciones de escritorio, aplicaciones web y pequeñas aplicaciones de servidor.

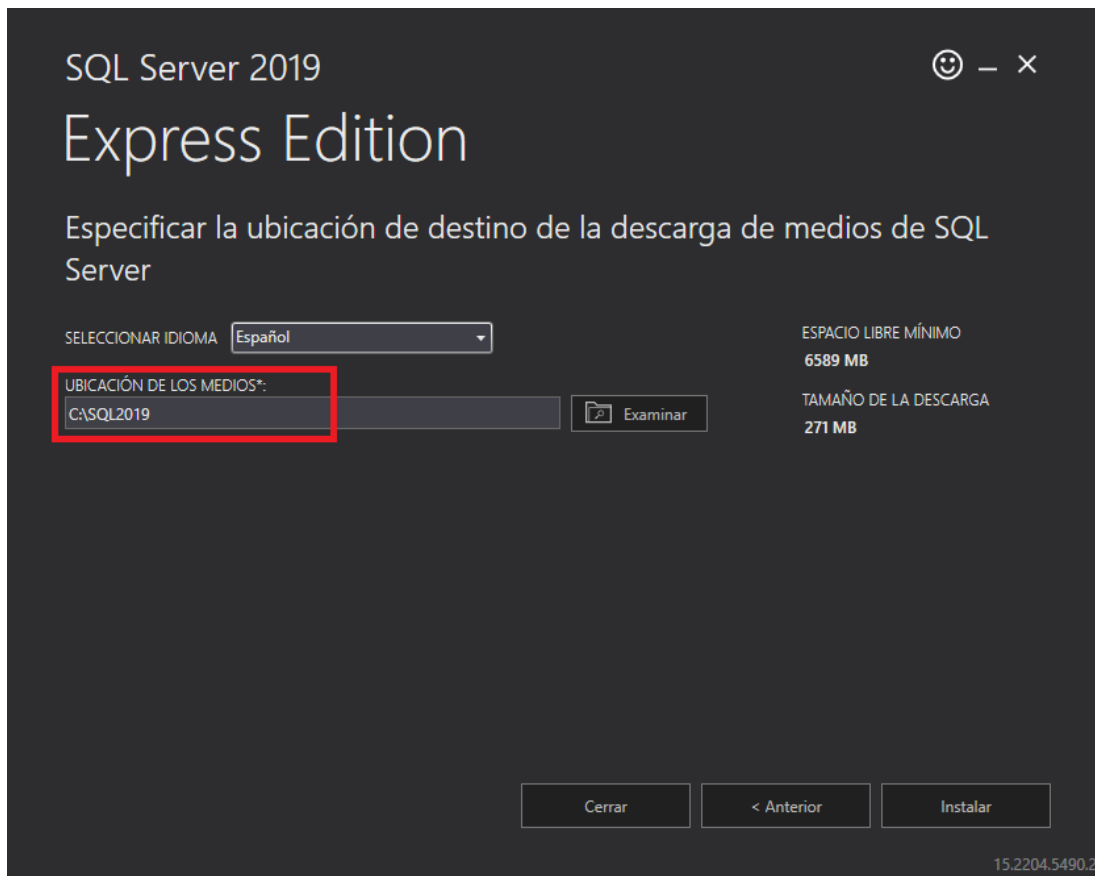
Descargar ahora >

Triem la versió **Express** i la descarreguem.

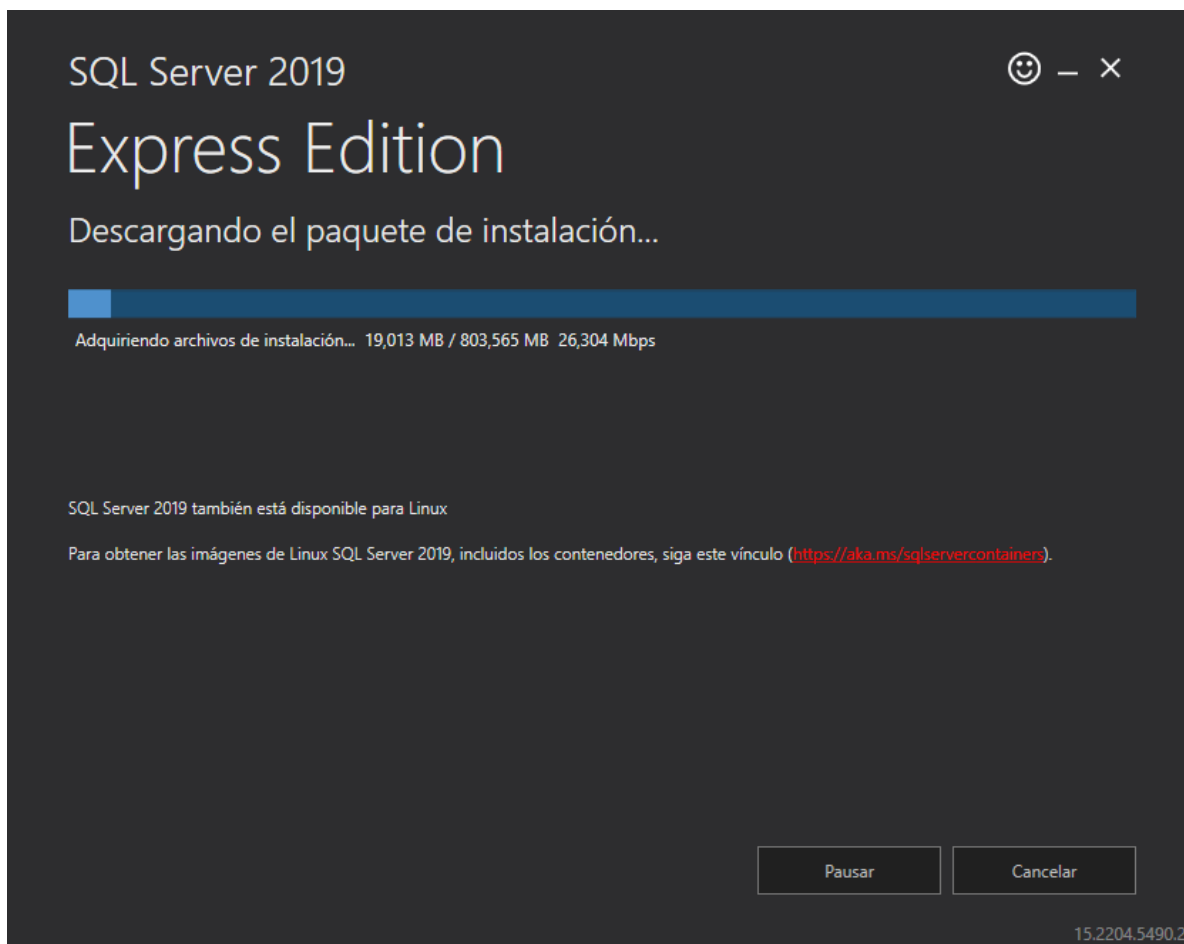
A continuació executem el paquet descarregat i el primer que ens trobem serà aquesta finestra:



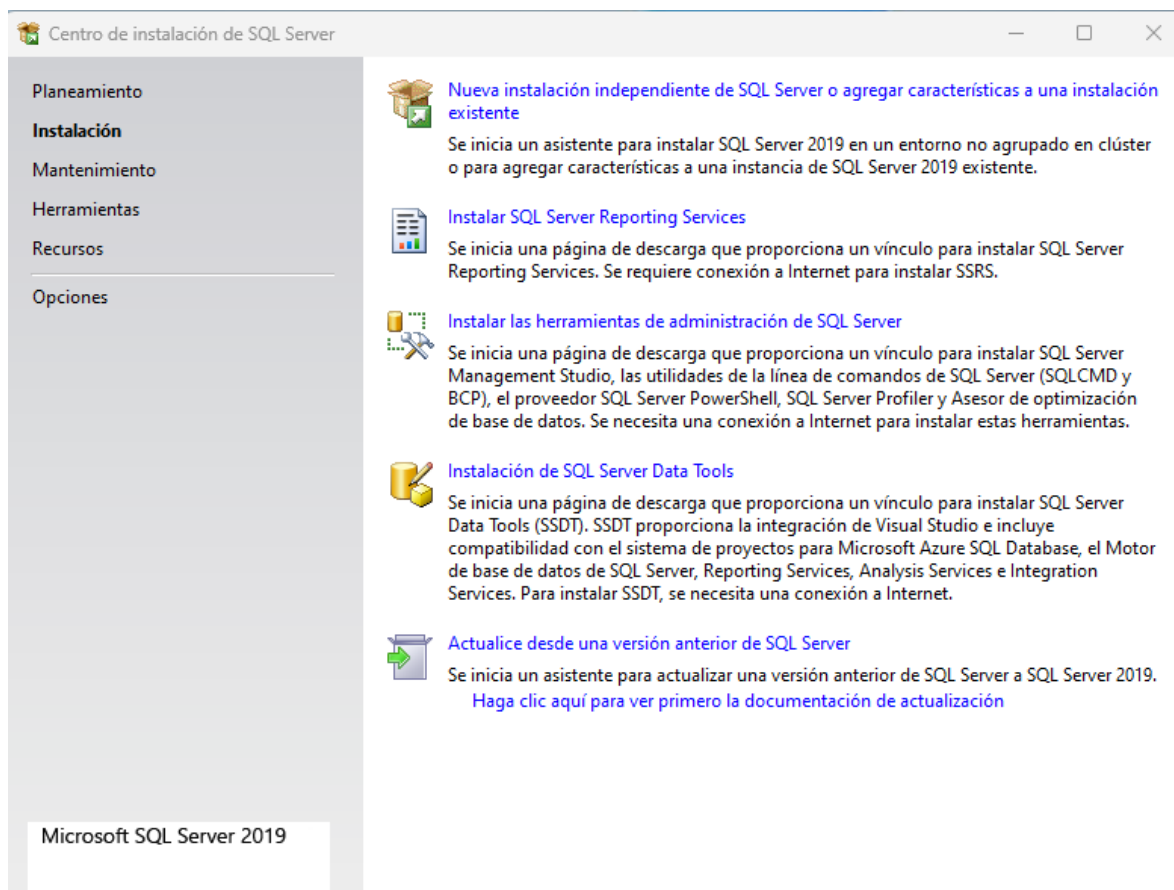
Triem la versió personalitzada.



Deixem la ubicació d'instal·lació com està i polsem en **Instalar**:

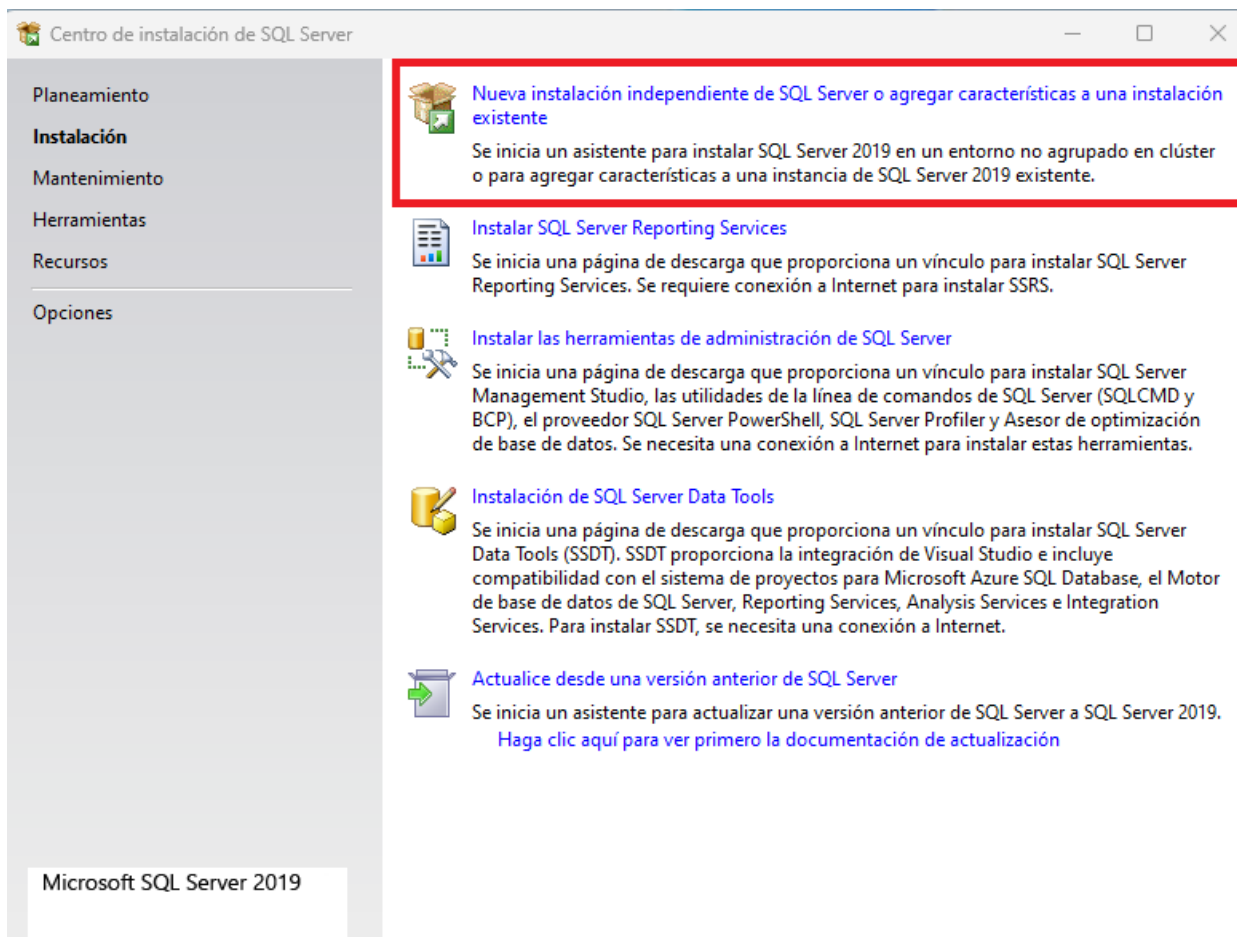


Quan termine l'instal·lació trobarem aquesta finestra:

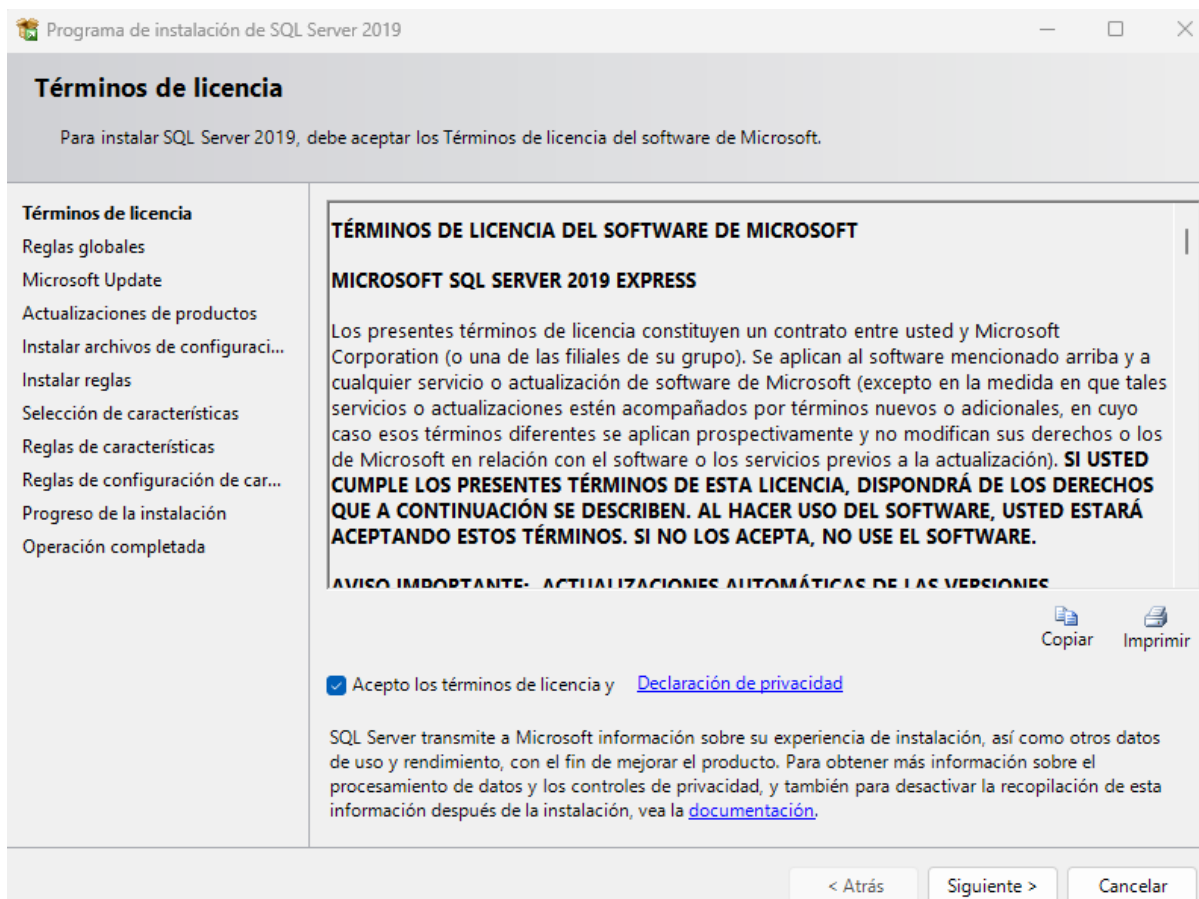


En aquesta finestra podem observar els diferents productes que podem instal·lar. Per a cobrir les nostres necessitats únicament instal·larem el servidor SQL Server (primera opció) i les eines d'administració (tercera opció).

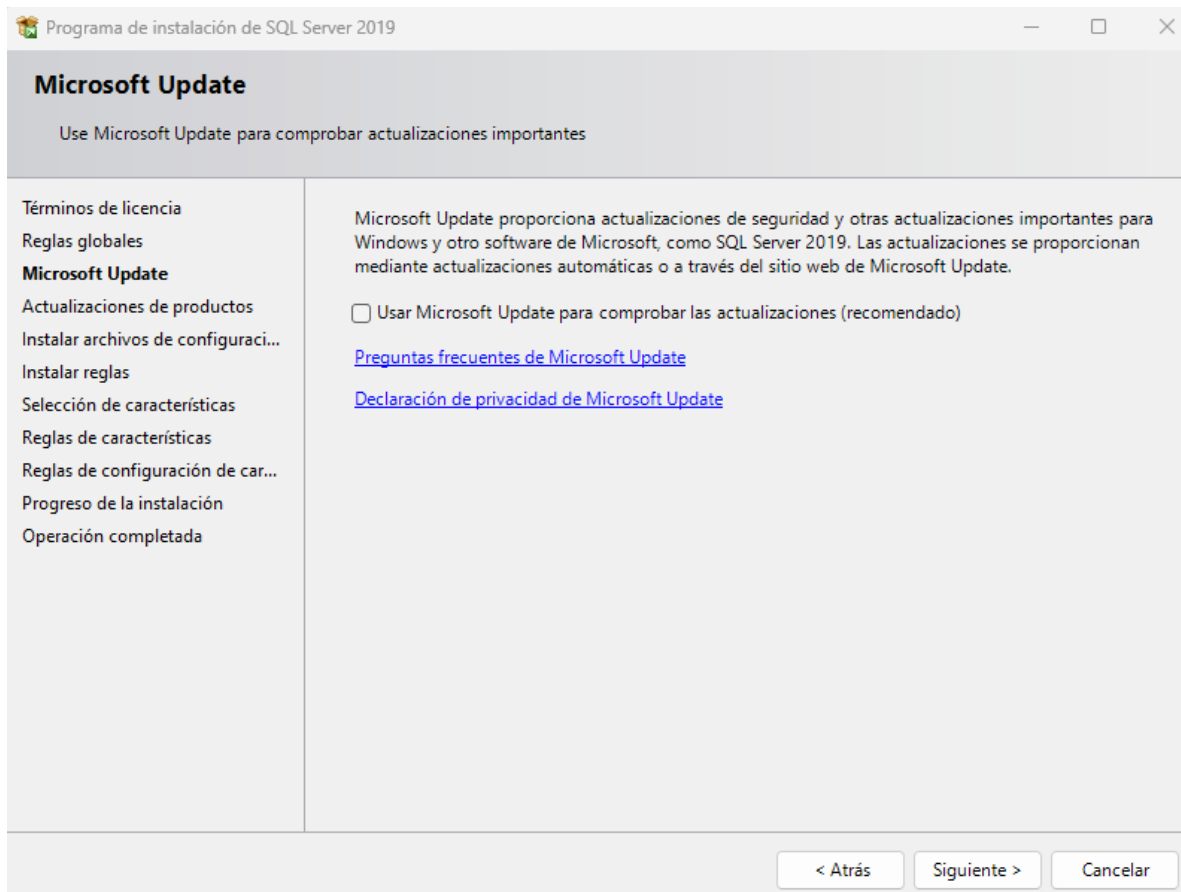
Comencem per el servidor.



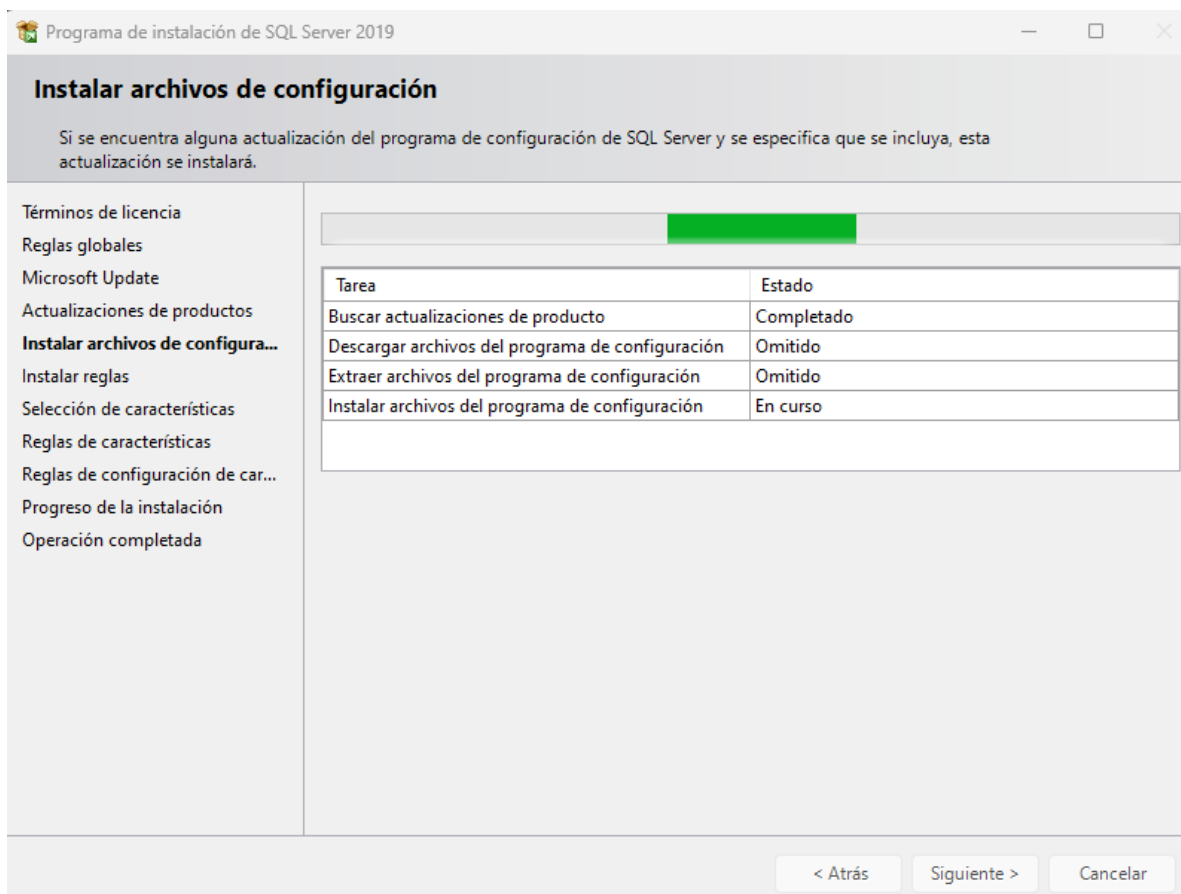
Polsem la primera opció i trobarem aquesta finestra:



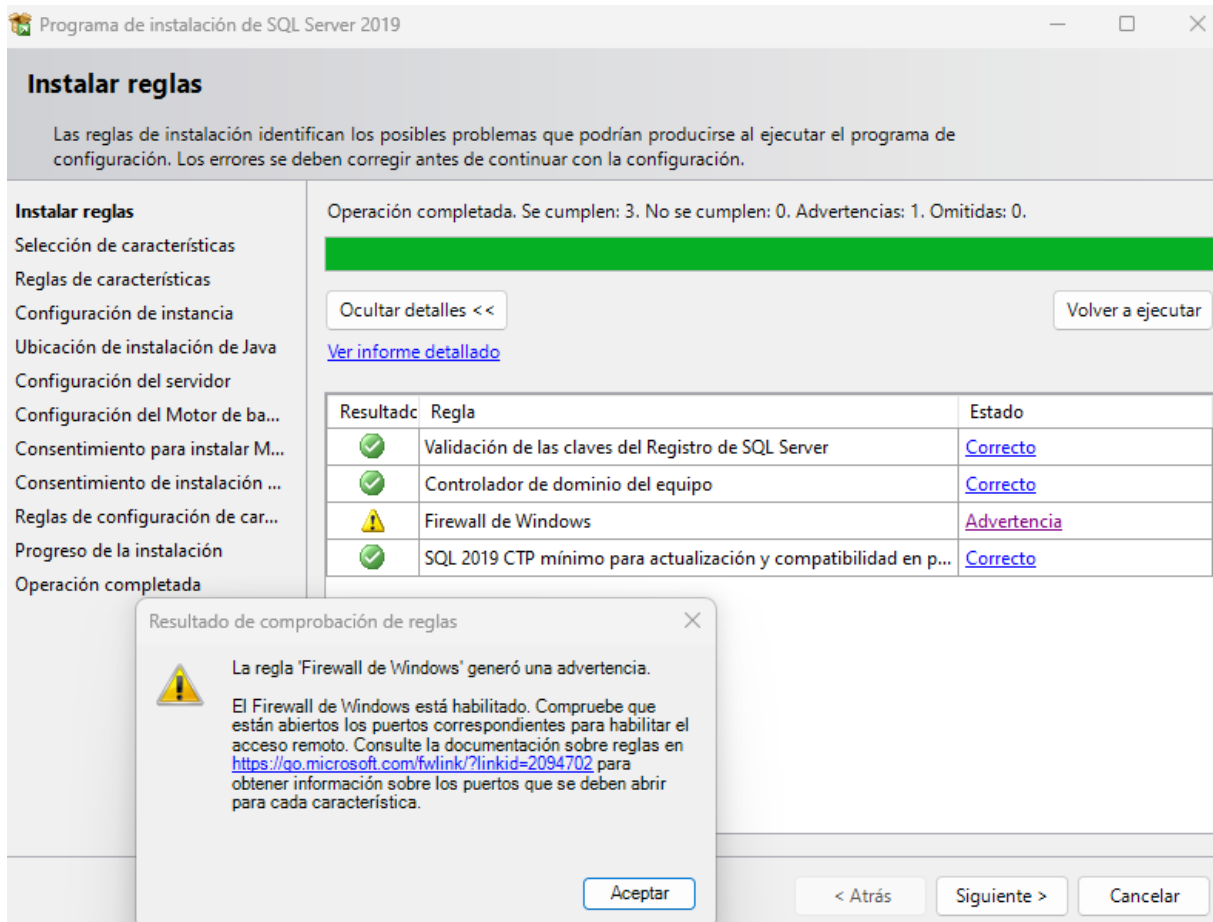
Acceptem els termes i polsem **Següent**:



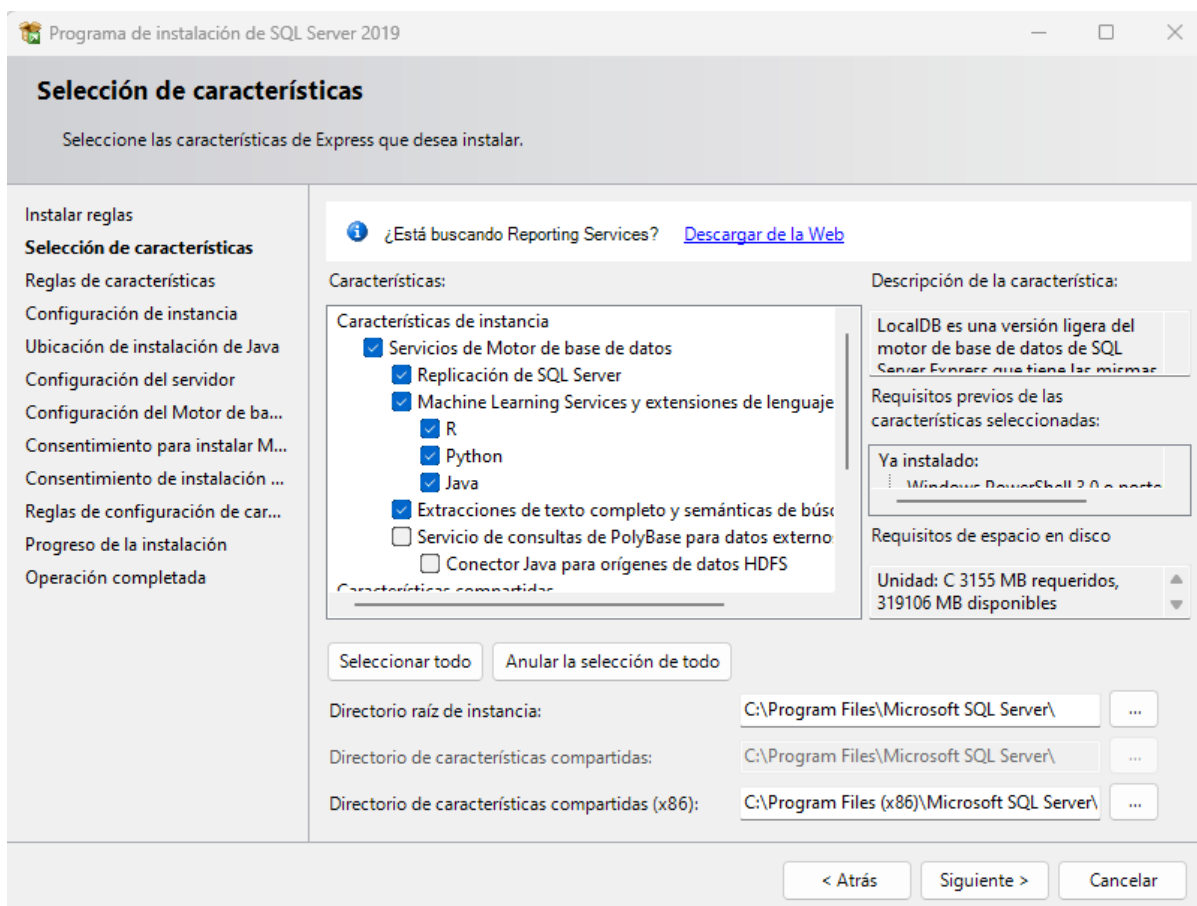
Podriem triar usar Microsoft Update per actualitzar el servidor SQL Server, ho deixem desmarcat i polsem **següent**:



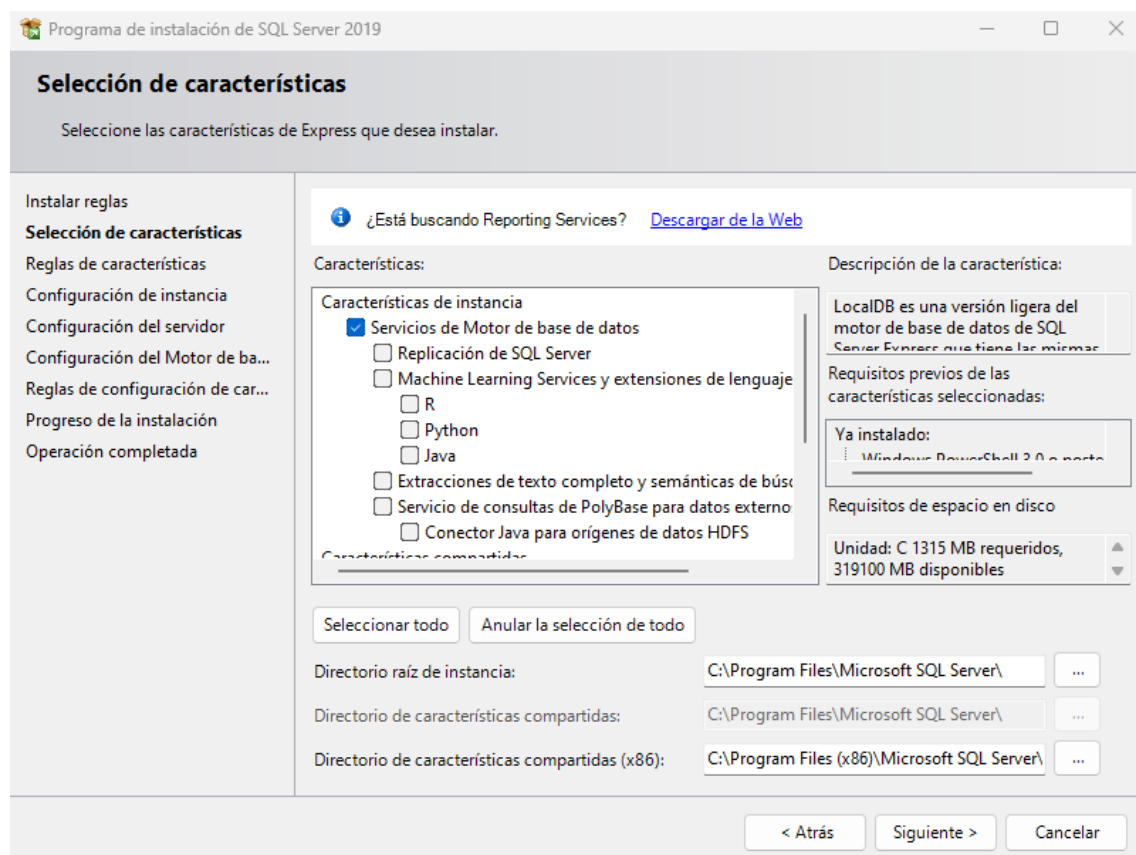
La instal·lació comença i quan termine trobarem aquesta finestra:



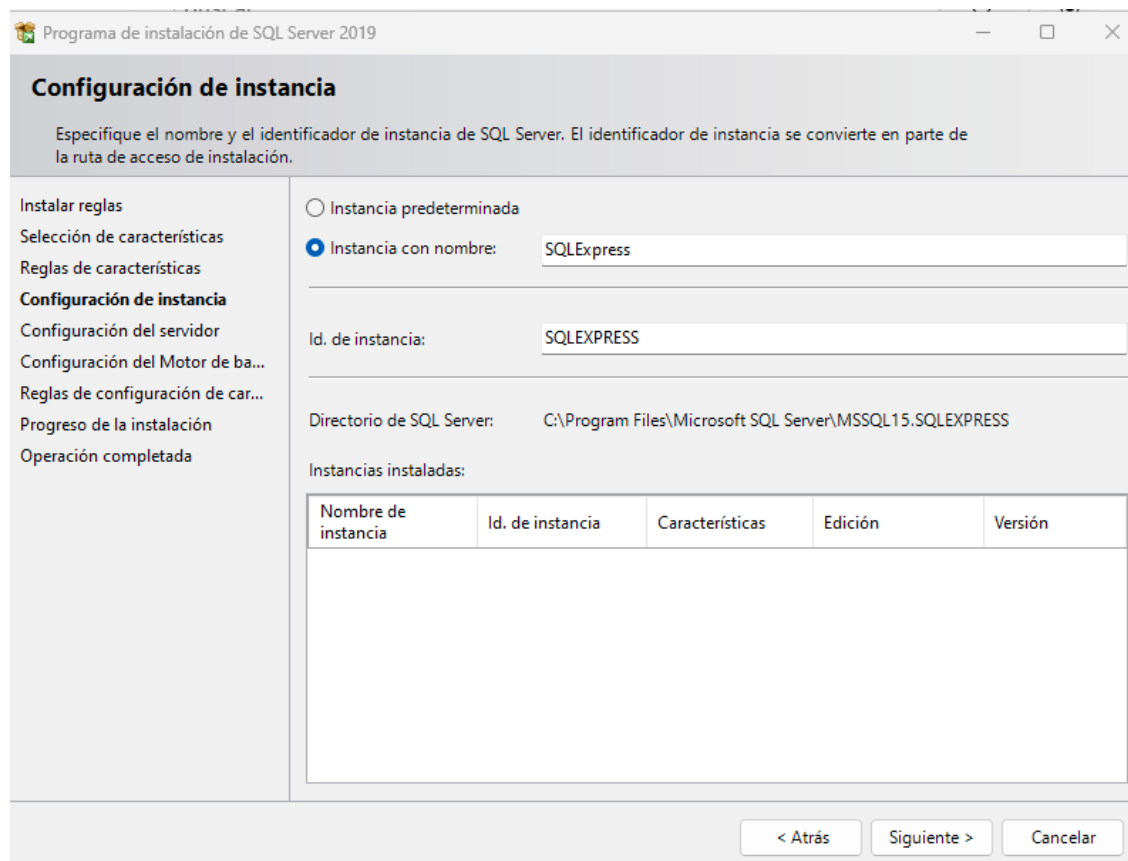
Hauriem de comprovar el firewall per a poder connectar amb el servidor SQL Server des de l'exterior. Per al funcionament local no cal comprovar res al firewall. Polsem **Acceptar** i continuem amb el botó **Següent**:



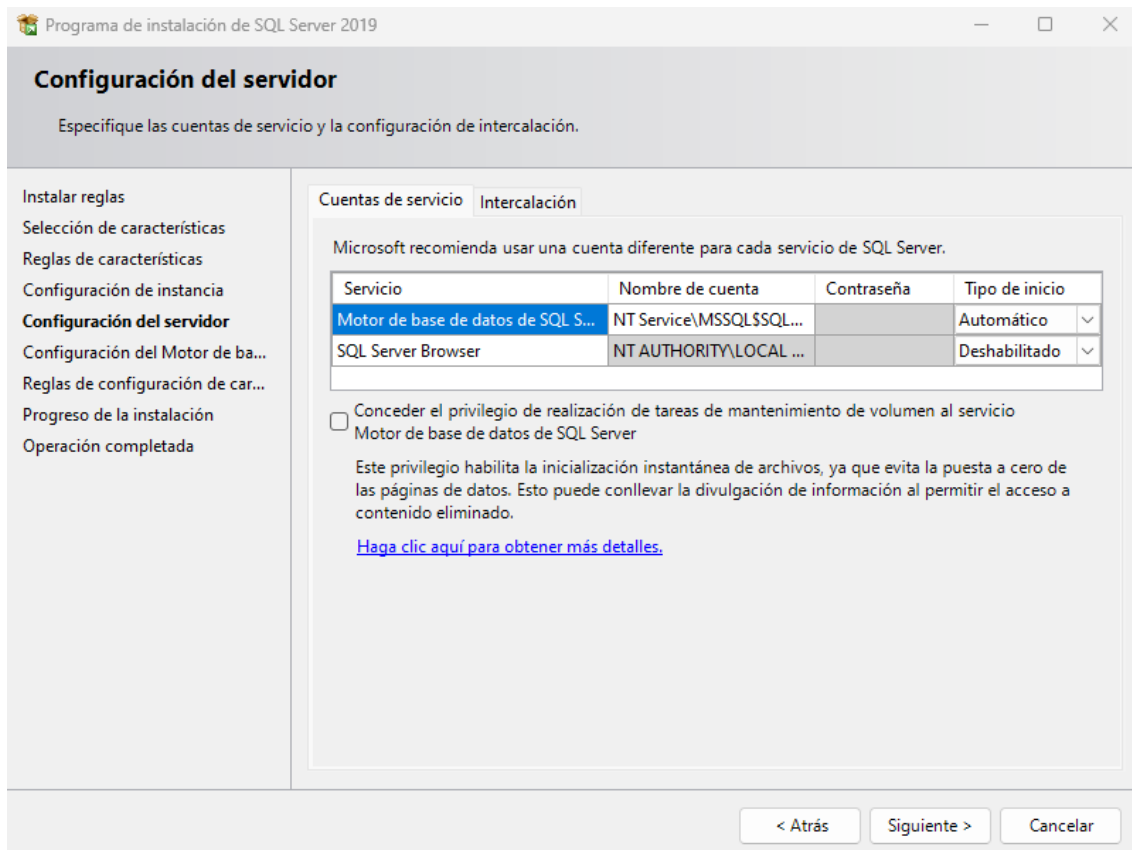
Ara es moment de triar que components del servidor volem instal·lar. Nosaltres únicament necessitem el servidor pur, per tant desmarcarem la resta d'opcions:



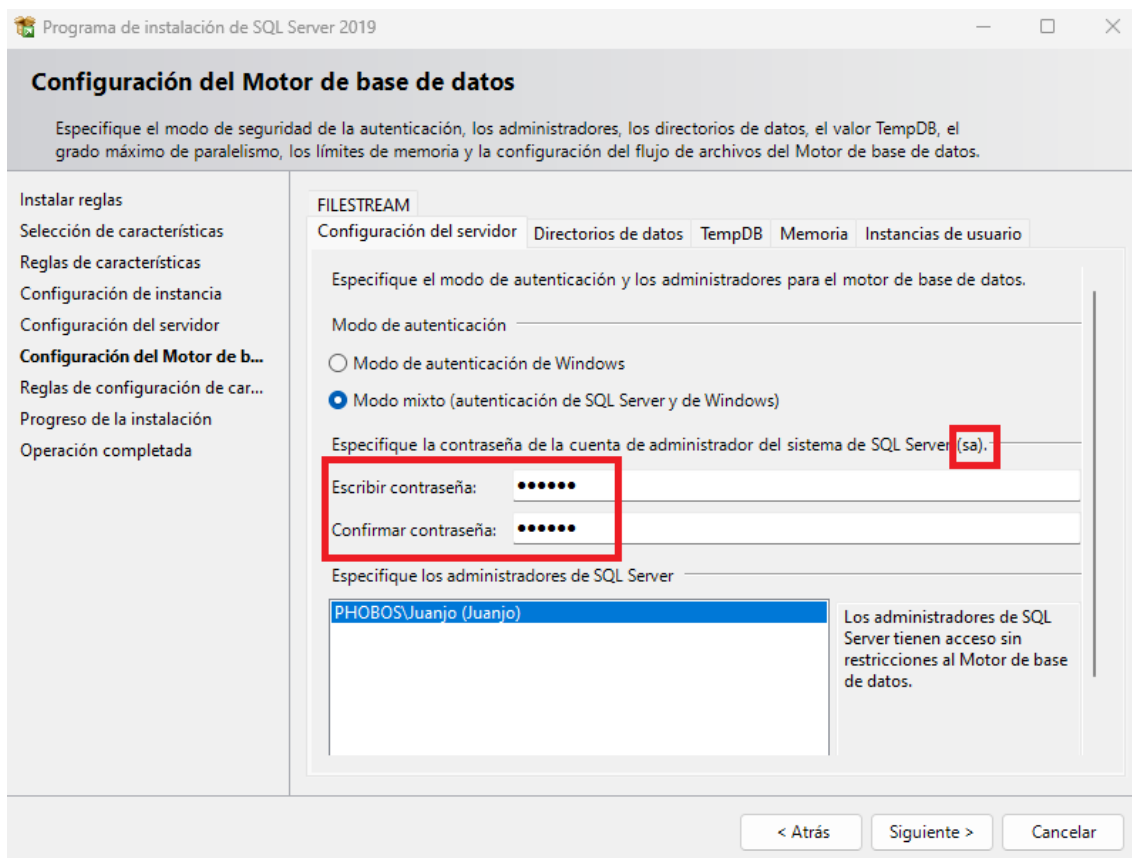
Polsem **següent** i continuem:



En aquesta finestra decidim el nom de la nostra instància de SQL Server, ho deixem com està i polsem **Següent**

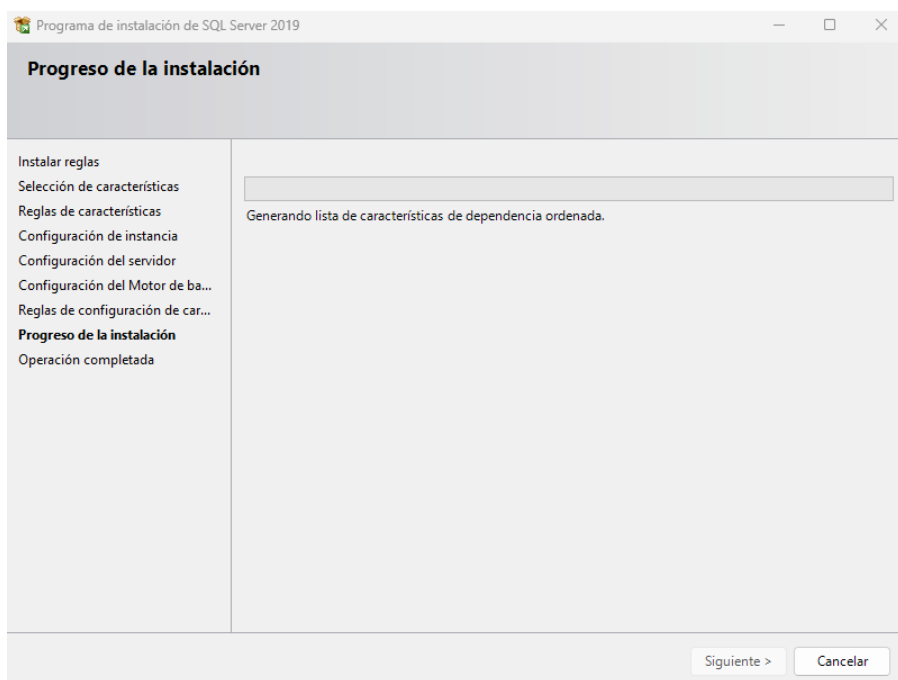


No toquem res i polsem **Següent**



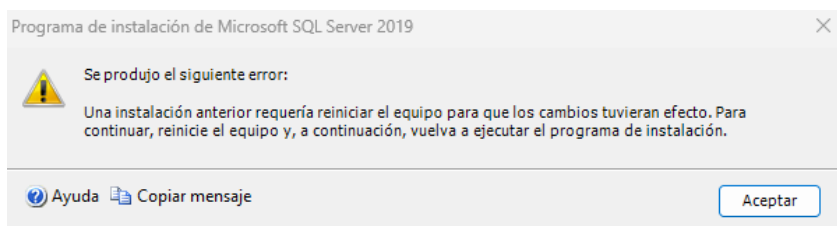
En aquesta finestra hem decidir el tipus d'autenticació del SQL Server. Podem triar entre autenticació Windows (usuaris del S.O.) o **autenticació mixta** (també amb usuaris del propi SQL Server). Triem aquesta última i indiquem una contrasenya per a l'usuari administrador (**sa**) del SQL

Server. Com no esteu treballant en un entorn de producció us recomane posar una contrasenya sencilla, per exemple **saroot** o **altra que siga facil de recordar**. A continuació polsem **següent**:

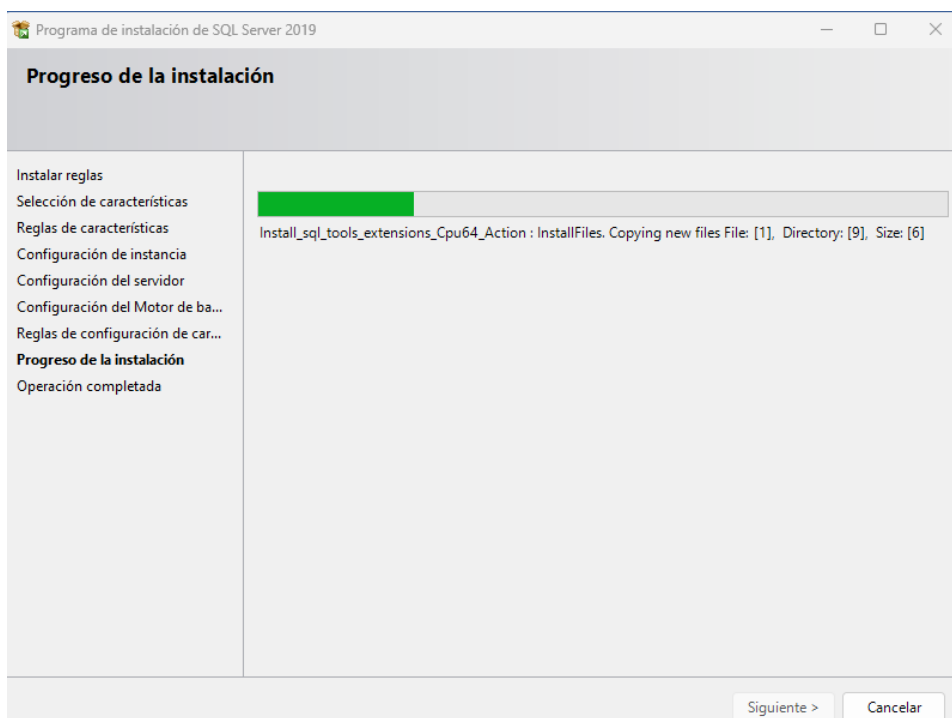


I per fi comença l'instal·lació de SQL Server EXPRESS.

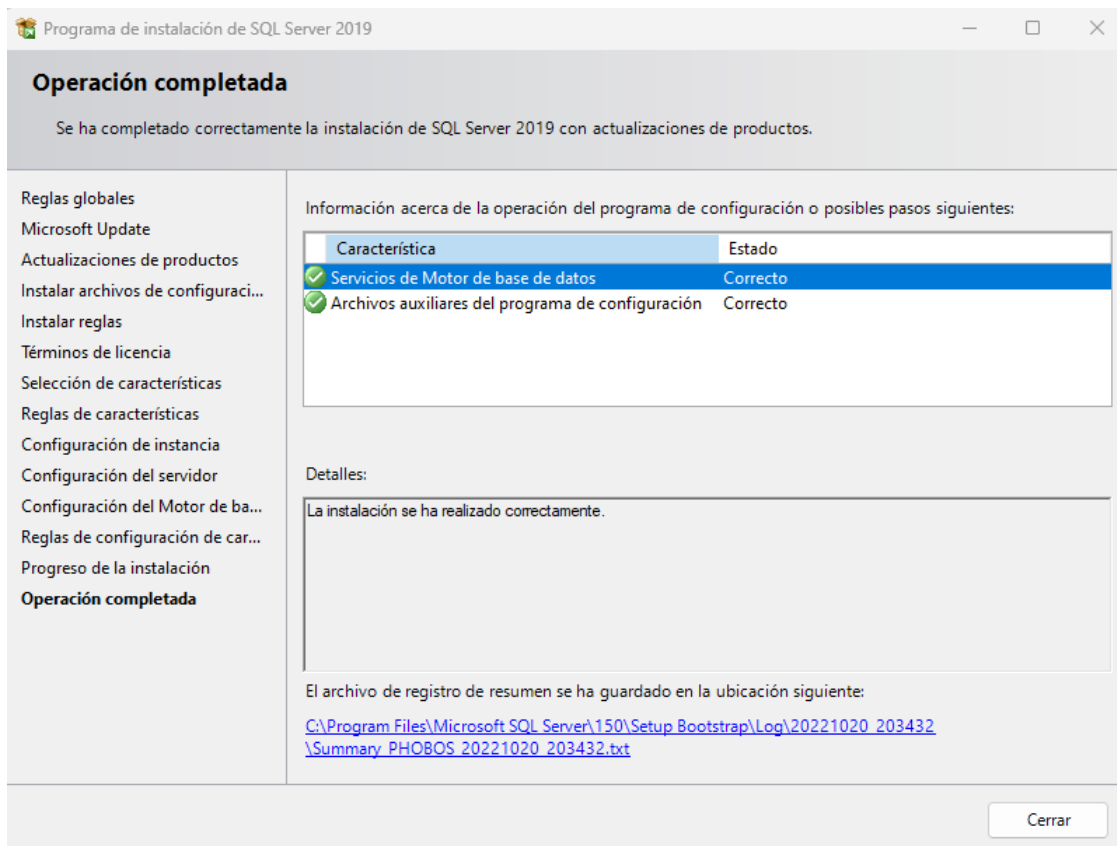
Es possible que durant la mateixa aparegua aquest missatge:



La instal·lació fallarà i tindreu que reiniciar l'equip. Ho feu i comenceu de nou a instal·lar SQL Server EXPRESS, en esta segona ocasió l'instal·lació continuarà sense problemes:



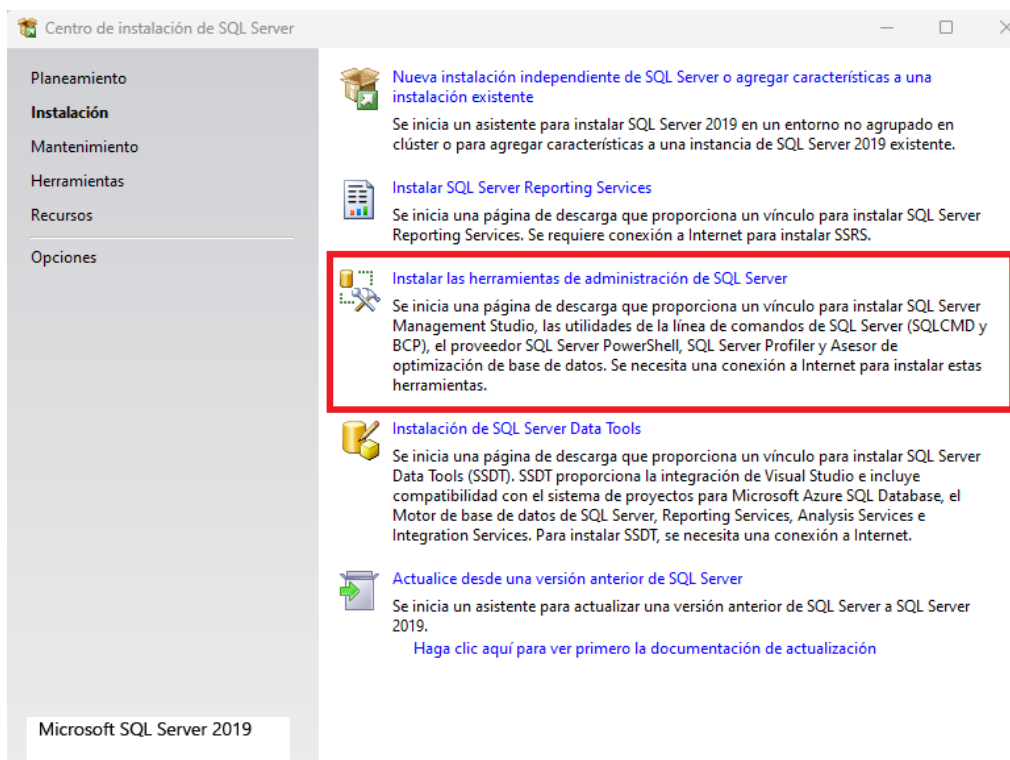
Quan termine es mostrarà una finestra com aquesta:



Polseu **Tancar** i tornarem a la finestra principal d'instal·lació d'elements de SQL Server EXPRESS

2.2. Instal·lació de l'eina d'Administració de SQL Server.

Ja tenim instal·lat el servidor. Ara necessitem un software gràfic per poder administrar-lo (encara que és possible fer-ho a l'interfície de comandaments). Emn la finestra principal d'instal·lació d'elements de SQL Server triarem el tercer element, **Eines d'administració de SQL Server**.



Aquesta opció ens durà a la següent url <https://learn.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver16>

En aquesta pàgina podrem descarregar l'eina SSMS (**SQL Server Managment Studio**)

Podem descarregar-lo en anglés

Descarga de SQL Server Management Studio (SSMS)

Artículo • 26/09/2022 • Tiempo de lectura: 7 minutos • 45 colaboradores

[Comentarios](#)

Se aplica a: SQL Server (todas las versiones admitidas) Azure SQL Database Azure SQL Managed Instance Azure Synapse Analytics

SQL Server Management Studio (SSMS) es un entorno integrado para administrar cualquier infraestructura de SQL, desde SQL Server a Azure SQL Database. SSMS proporciona herramientas para configurar, supervisar y administrar instancias de SQL Server y bases de datos. Use SSMS para implementar, supervisar y actualizar los componentes de nivel de datos que usan las aplicaciones, además de compilar consultas y scripts.

Use SSMS para consultar, diseñar y administrar bases de datos y almacenes de datos, estén donde estén, en el equipo local o en la nube.

Descargar SSMS

Para descargar SSMS 19 Preview 3, visite [Descarga de SSMS 19](#).

[↓ Descarga gratuita de SQL Server Management Studio \(SSMS\) 18.12.1](#)

SSMS 18.12.1 es la última versión de disponibilidad general (GA). Si tiene instalada una versión GA anterior de SSMS 18, esta se actualizará a 18.12.1 con la nueva instalación.

- Número de versión: 18.12.1
- Número de compilación: 15.0.18424.0
- Fecha de publicación: 21 de junio de 2022

O baixar un poc la pàgina i trobarem una versió en castellà:

Idiomas disponibles

Esta versión de SSMS puede instalarse en los idiomas siguientes:

SQL Server Management Studio 18.12.1:

[Chino \(simplificado\)](#) | [Chino \(tradicional\)](#) | [Inglés \(Estados Unidos\)](#) | [Francés](#) | [Alemán](#) | [Italiano](#) | [Japonés](#) | [Coreano](#) | [Portugués \(Brasil\)](#) | [Ruso](#) | [Español](#)

Triem el que desitgem i l'executem per a començar l'instal·lació:



VERSIÓN 18.12.1

Microsoft SQL Server Management Studio con Azure Data Studio

Bienvenido. Haga clic en 'Instalar' para comenzar.

Ubicación:

C:\Program Files (x86)\Microsoft SQL Server Management Studio 18

Cambiar

Al hacer clic en el botón "Instalar", confirmo que acepto la [Declaración de privacidad](#) y los términos de licencia para [SQL Server Management Studio](#) y [Azure Data Studio](#)

SQL Server Management Studio transmite a Microsoft información sobre su experiencia de instalación, así como otros datos de uso y rendimiento, con el fin de mejorar el producto. Para obtener más información sobre el procesamiento de datos y los controles de privacidad, y también para desactivar la recopilación de esta información después de la instalación, vea la [documentación](#)

Instalar

Cerrar

El software ens proposa una ruta d'instal·lació, la deixem com està i polsem **Instalar**:



VERSIÓN 18.12.1

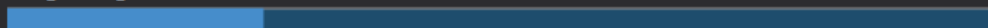
Microsoft SQL Server Management Studio con Azure Data Studio

Progreso del paquete



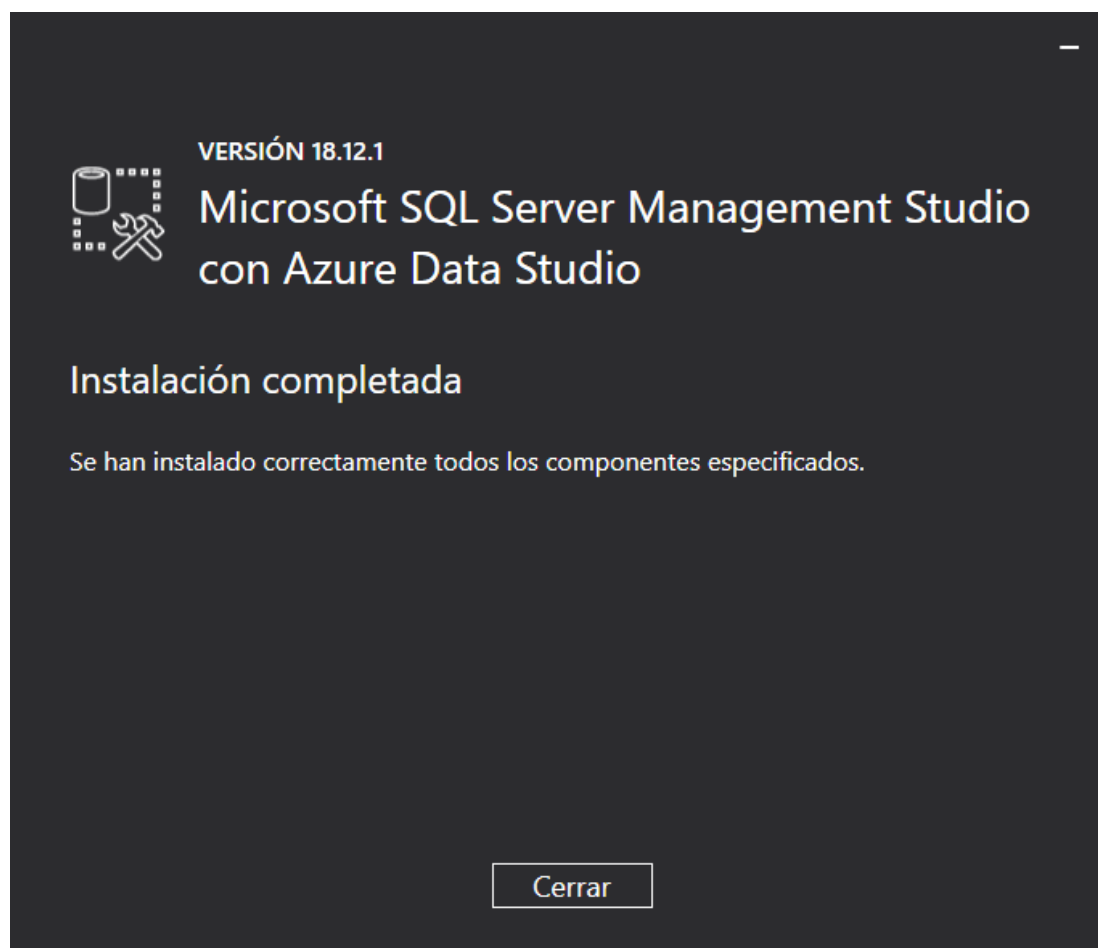
Microsoft OLE DB Driver for SQL Server

Progreso general



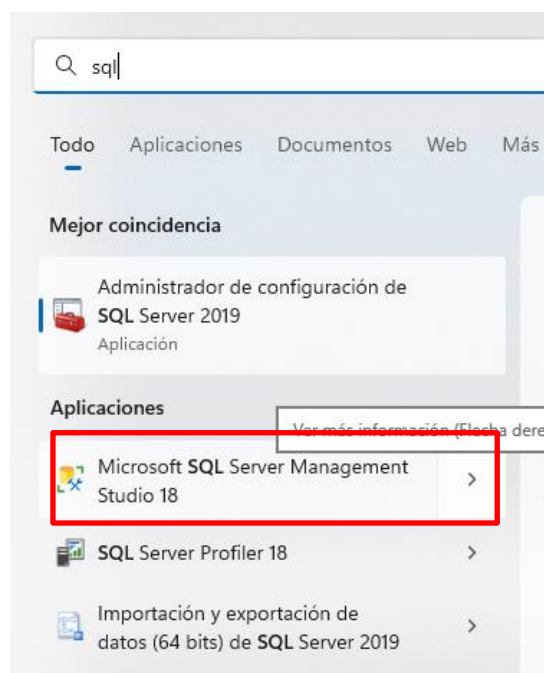
Cancelar

Quan termine arribareu a aquesta finestra

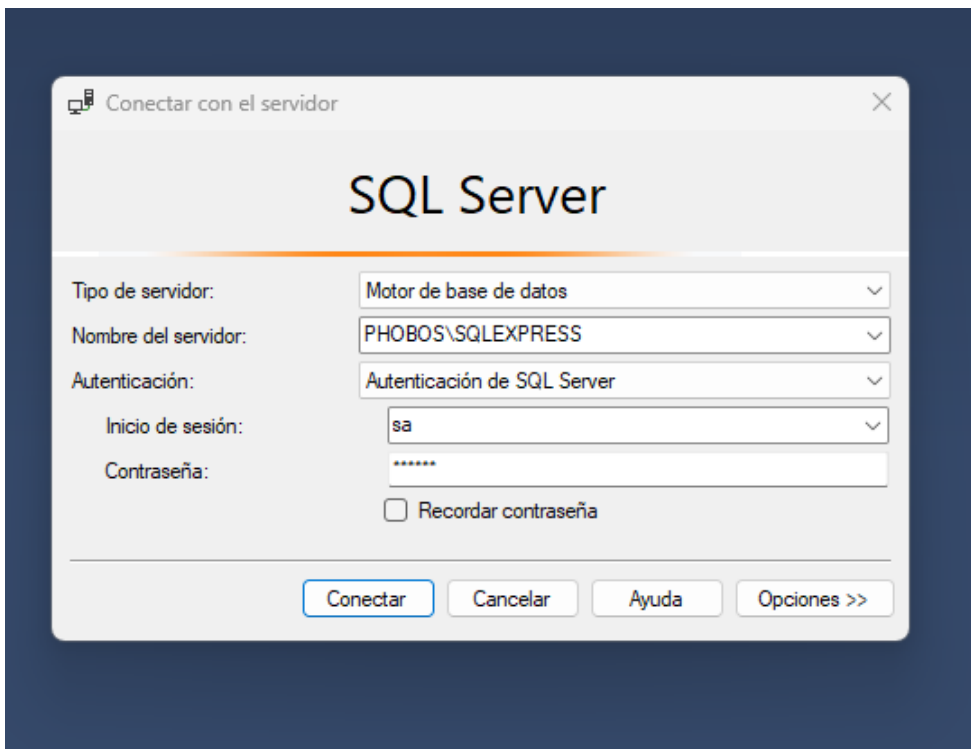


2.3. Configuració i ús de l'eina d'Administració de SQL Server.

Podem trobar l'eina d'administració instal·lada al menú de recerca del S.O.:



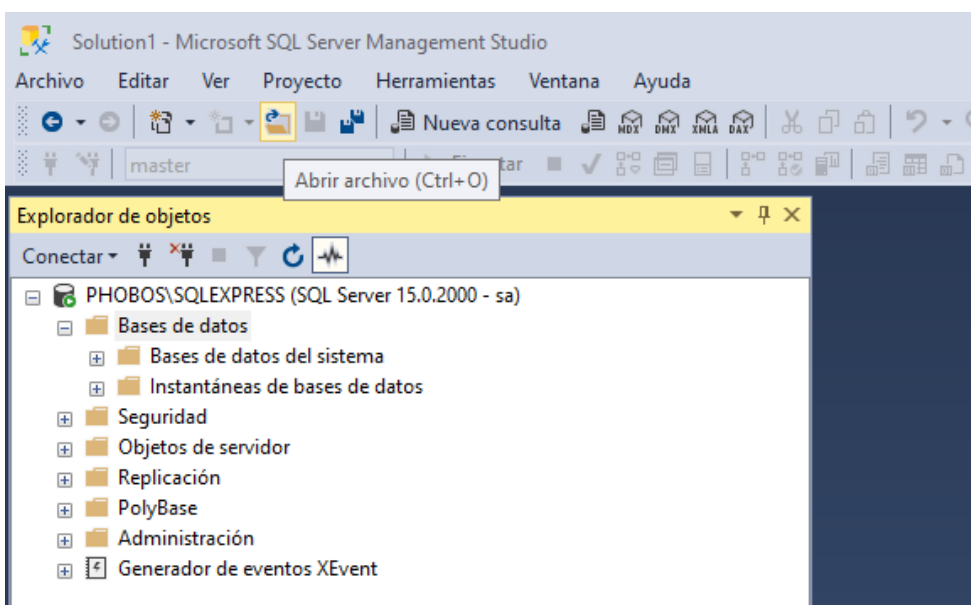
L'executem i la primera finestra que trobem és aquesta:



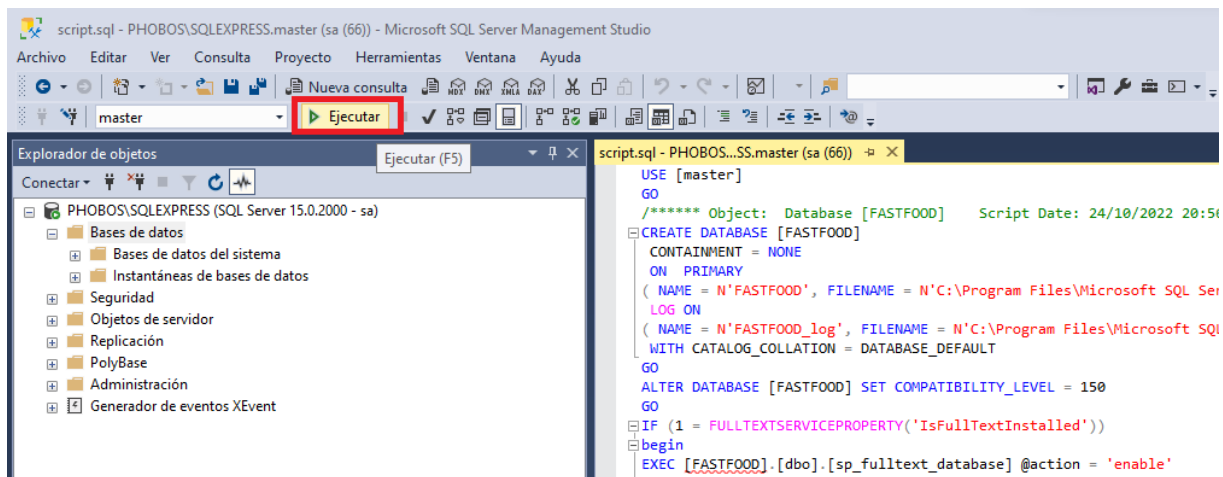
El primer que ens demana és que ens autèntiquem. Indiquem el tipus d'autenticació (Autenticació SQL Server) i l'usuari administrador i contrasenya del SQL Server. A continuació polsem **Connectar**

Ja podem administrar el servidor SQL Server, però com vegeu a l'imatge encara no tenim base de dades. Únicament vegem les bases de dades de sistema (on s'emmagatzemen coses com els usuaris del servidor SQL Server, definicions d'altres bases de dades, etc.) i les instantànies de les bases de dades.

Podem crear una base de dades nova desde 0 o importar una. Al nostre cas importarem una anomenada FASTFOOD mitjançant l'execució d'un script de lleguatge SQL que incorpora l'esquema i les dades de la base de dades. Per fer això polsem sobre l'icona "**Obrir Arxiu**" i triem el fitxer .sql:



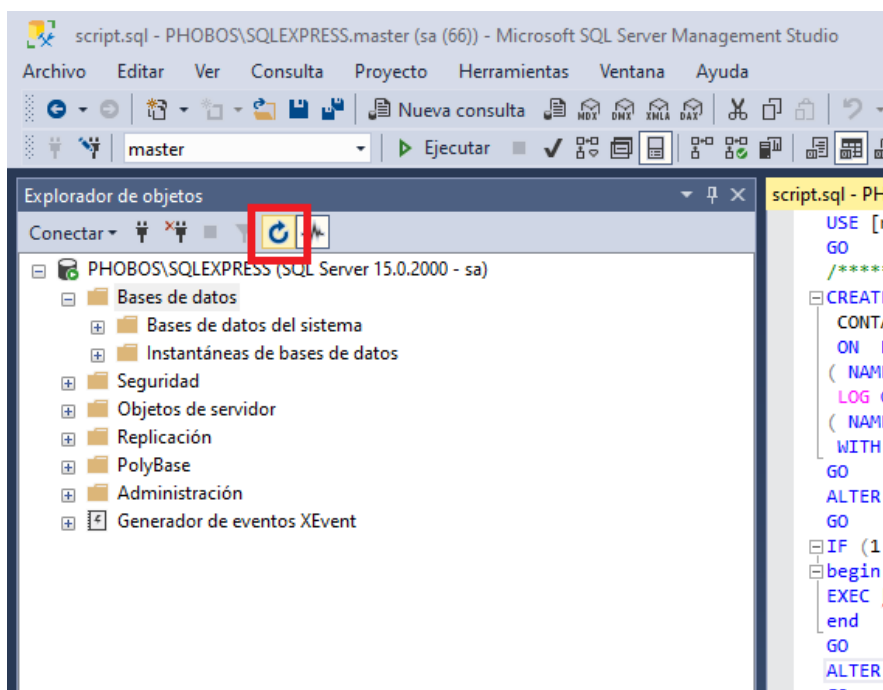
El software ens carregarà el fitxer sql en una finestra a la dreta de l'entorn:



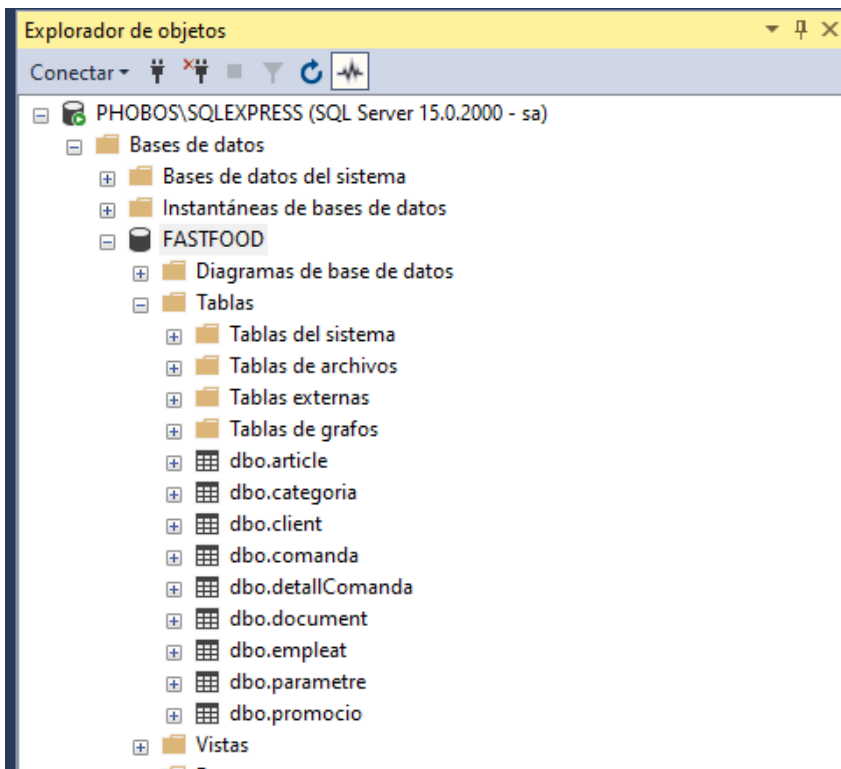
Podem el botó “**Executar**” i si no apareix cap errada ja tindrem la base de dades importada:



Però encara no es reflecteix a la finestra de l'esquerra. Polsem el botó “**Actualitzar**” per a refrescar els objectes d'aquesta finestra:



I ja tindrem disponible la base de dades **FASTFOOD**:

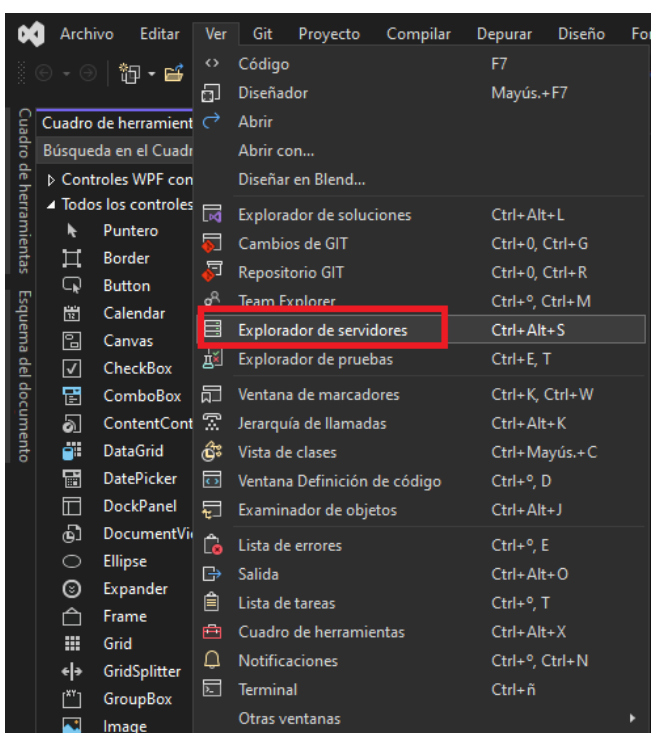


Podem navegar entre les taules i trobarem opcions per a dissenyar l'esquema de les taules, insertar fileres, llançar SQL's, etc ...

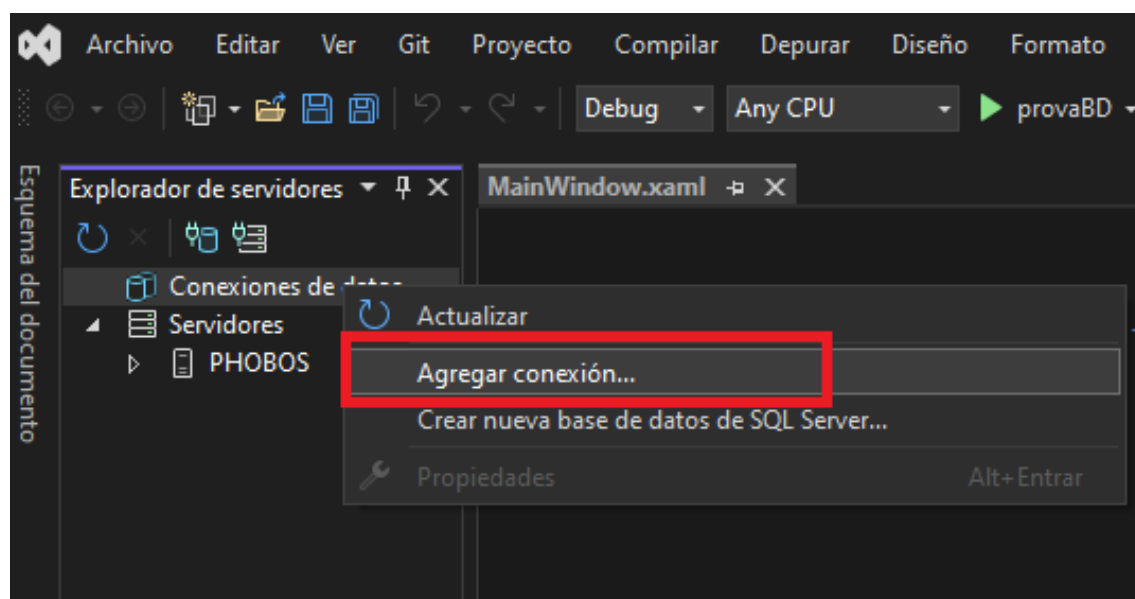
2.4. Administració d'una base de dades des d'un Projecte WPF.

També existeix la possibilitat d'integrar l'administració de SQL Sever dins d'un projecte WPF, de forma que podríem consultar o editar la base de dades des del mateix projecte. Vegem com ho podem fer.

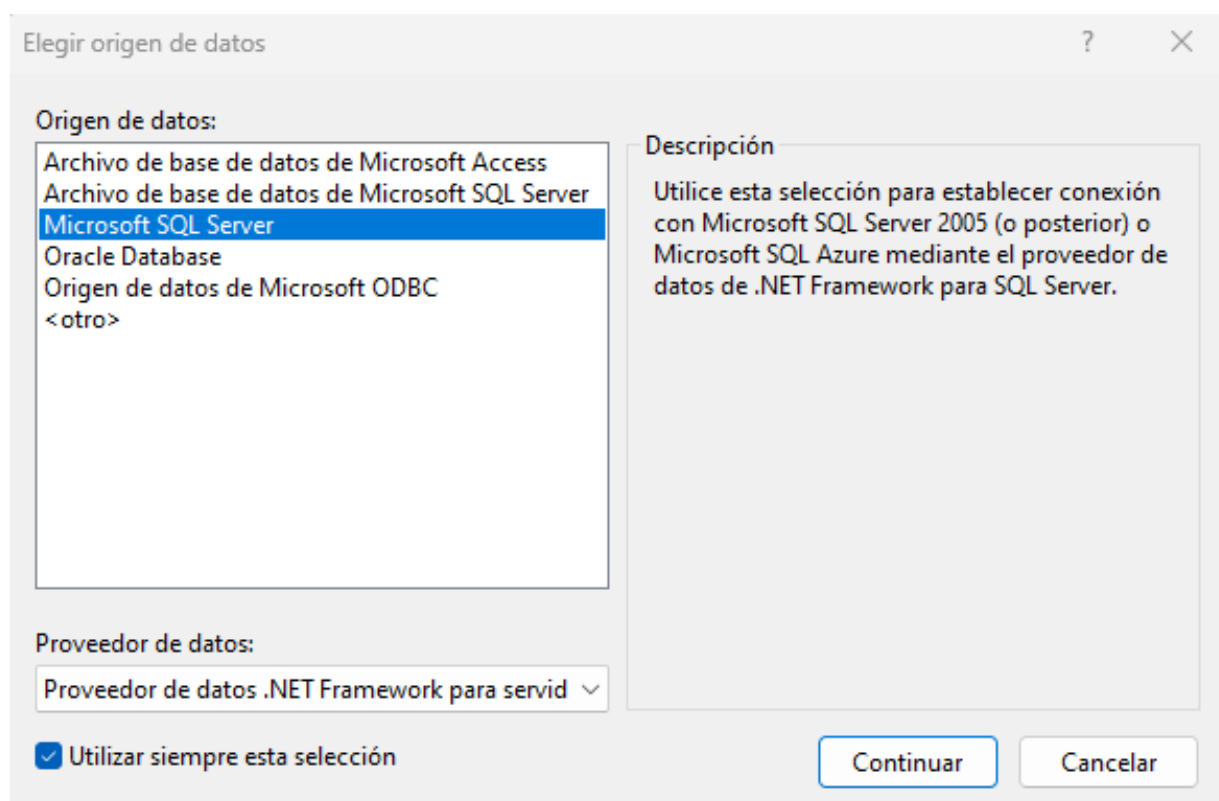
Creem un nou projecte WPF i fem clic sobre el menu "**Ver**" i triem l'opció "**Explorador de Servidores**":



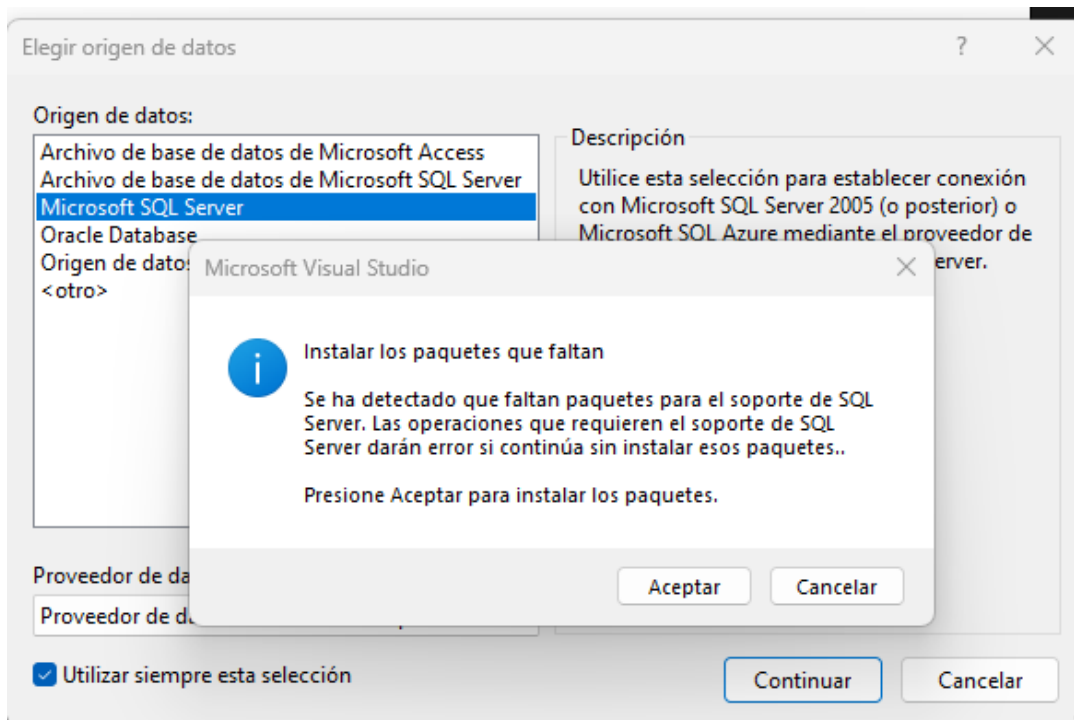
A continuació apareixerà un menú a l'esquerra on apareixerà el nom del nostre equip i dalt un ellaç per a crear una nova connexió. Triem aquesta opció: **Afegir una connexió**



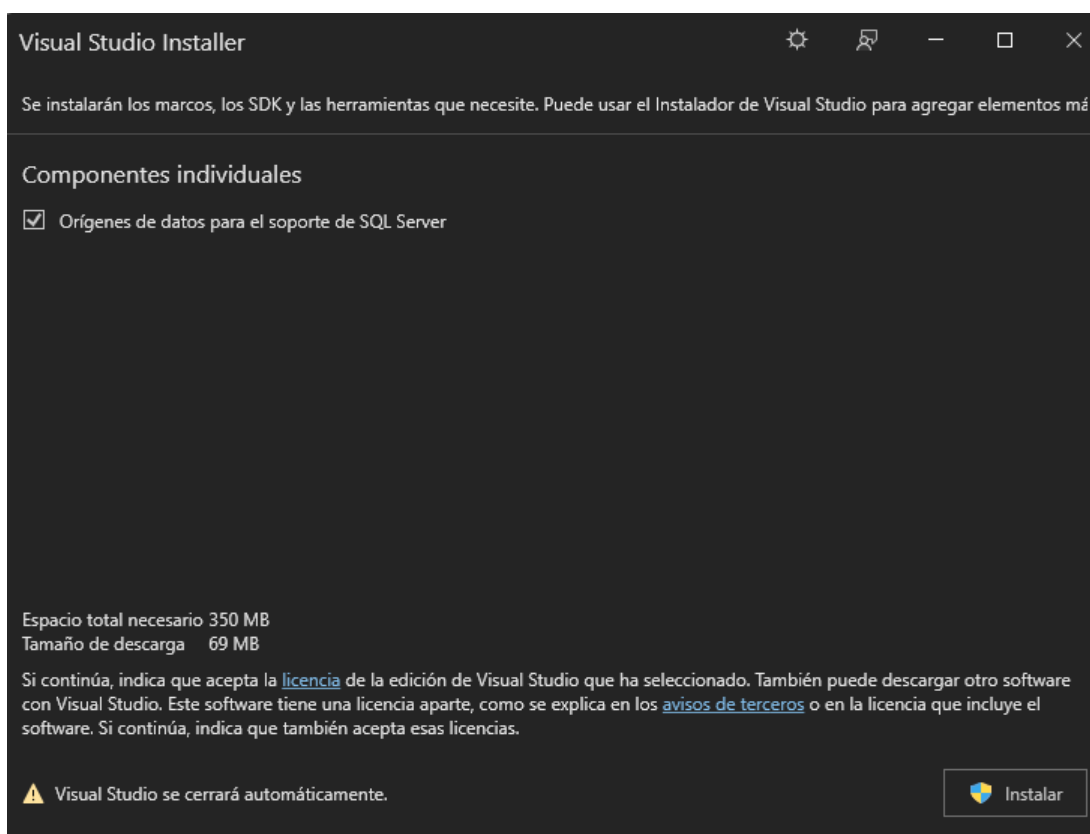
Apareixerà una finestra per a triar el tipus d'origen de dades;



Triem l'opció **Microsoft SQL Server** i polsem **Continuar**

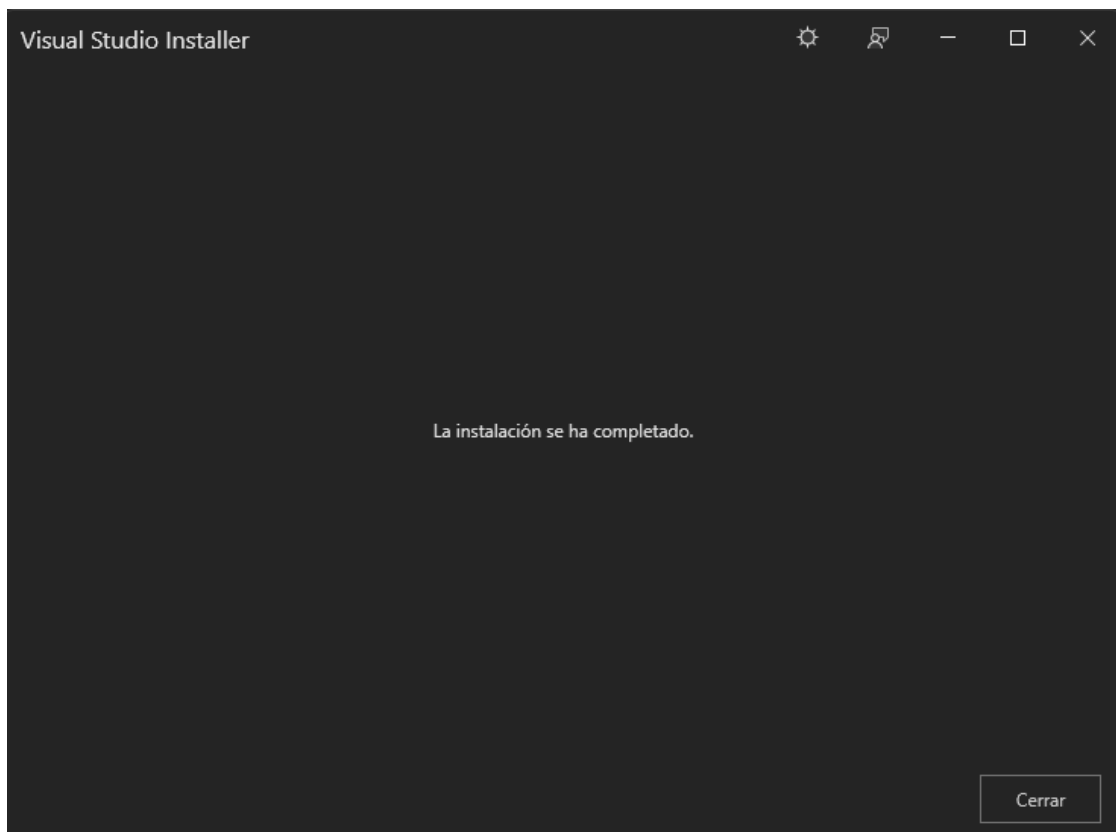
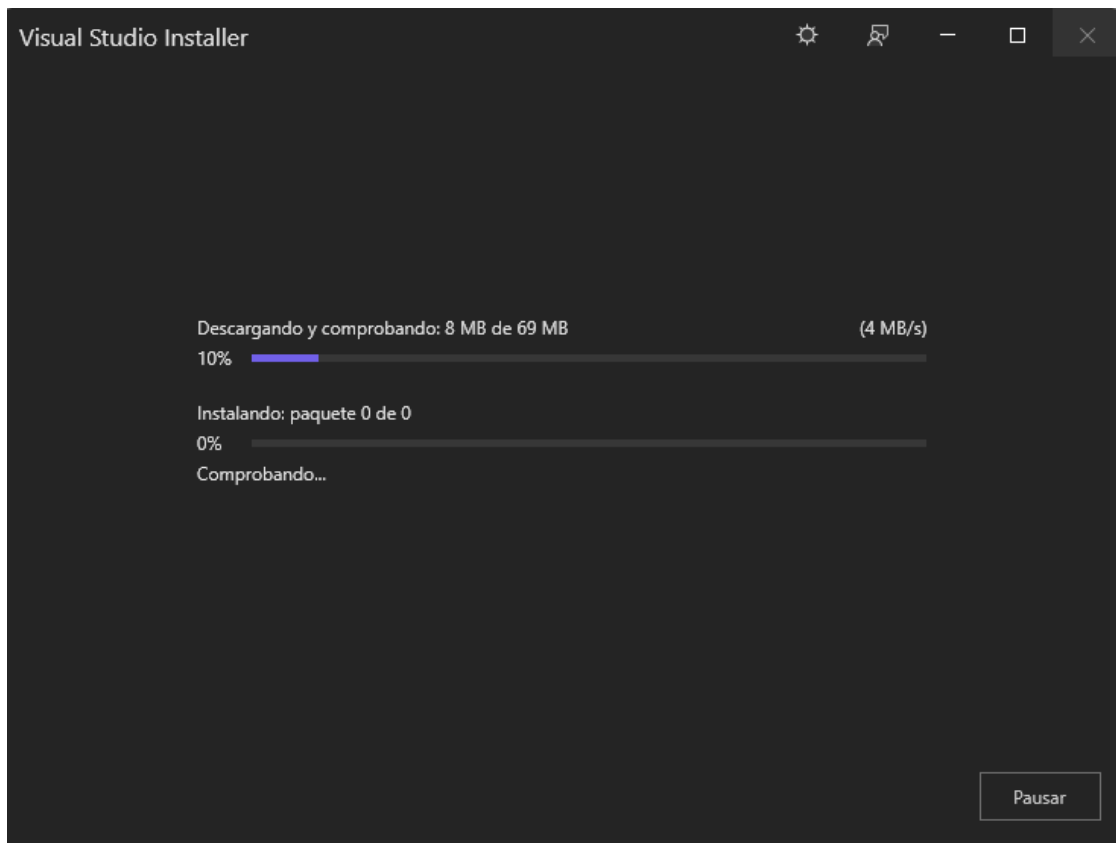


Si ens demana actualitzar uns paquets polsem **Aceptar**



A continuació apareix l'instal·lador dels paquets necessaris. Polsem **Instal·lar**

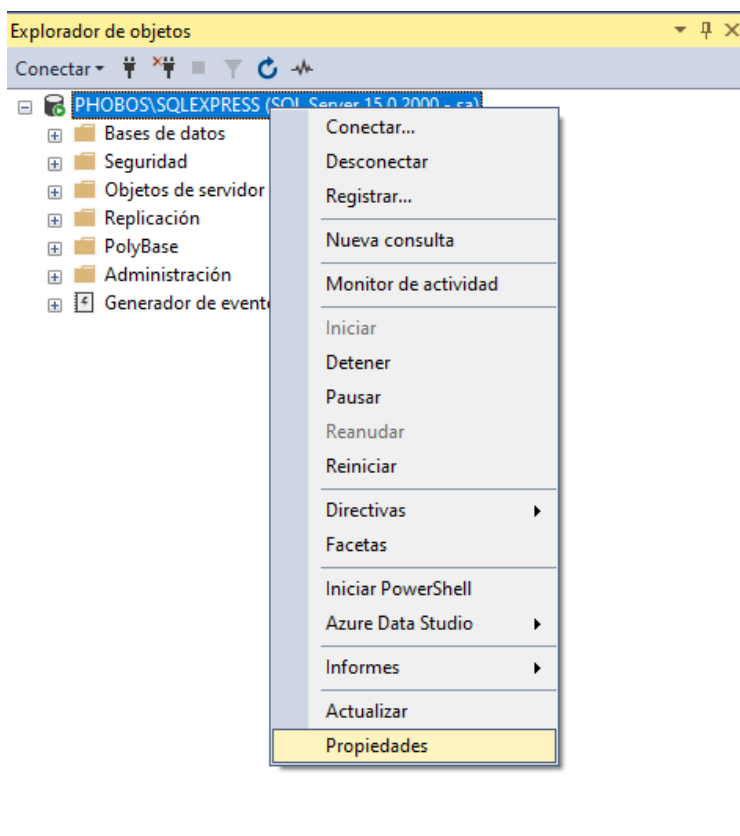
Comença l'instal·lació dels paquets:



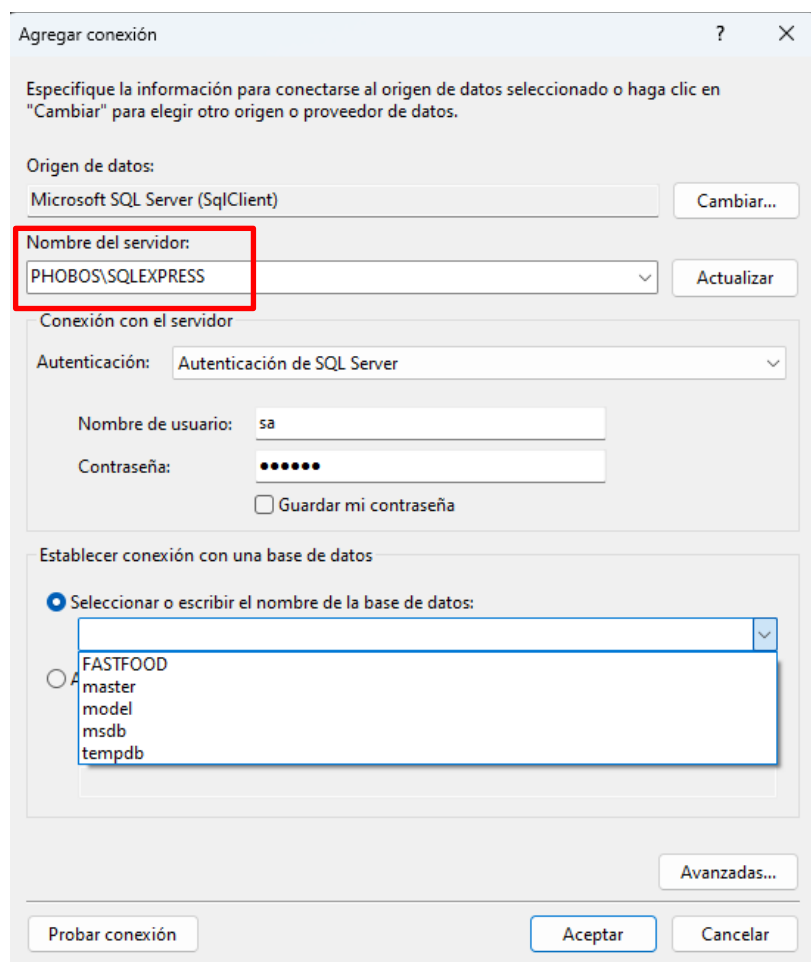
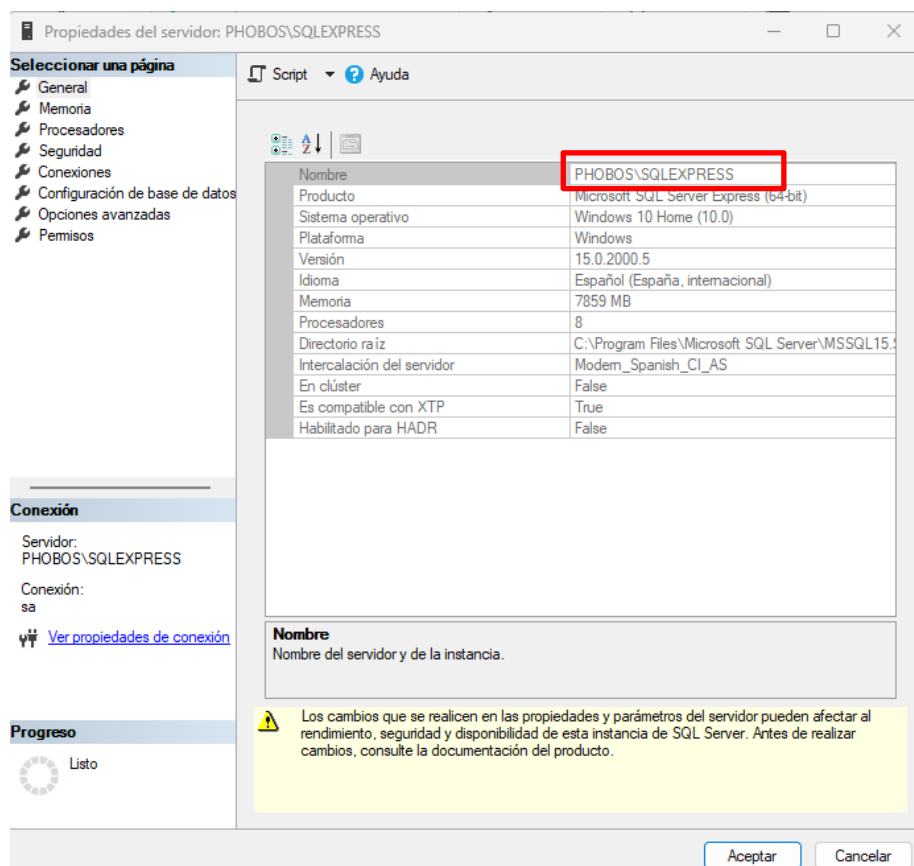
Una vegada instal·lats els paquets polsem l'opció **Tancar** i passarem a configurar la connexió amb la base de dades

Observa bé aquesta finestra. Cal configurar el **nom del servidor SQL Sever**, el **tipus d'autenticació** (Windows o SQL Server) i en el cal de autenticació SQL Server l'**usuari i contrasenya** d'accés.

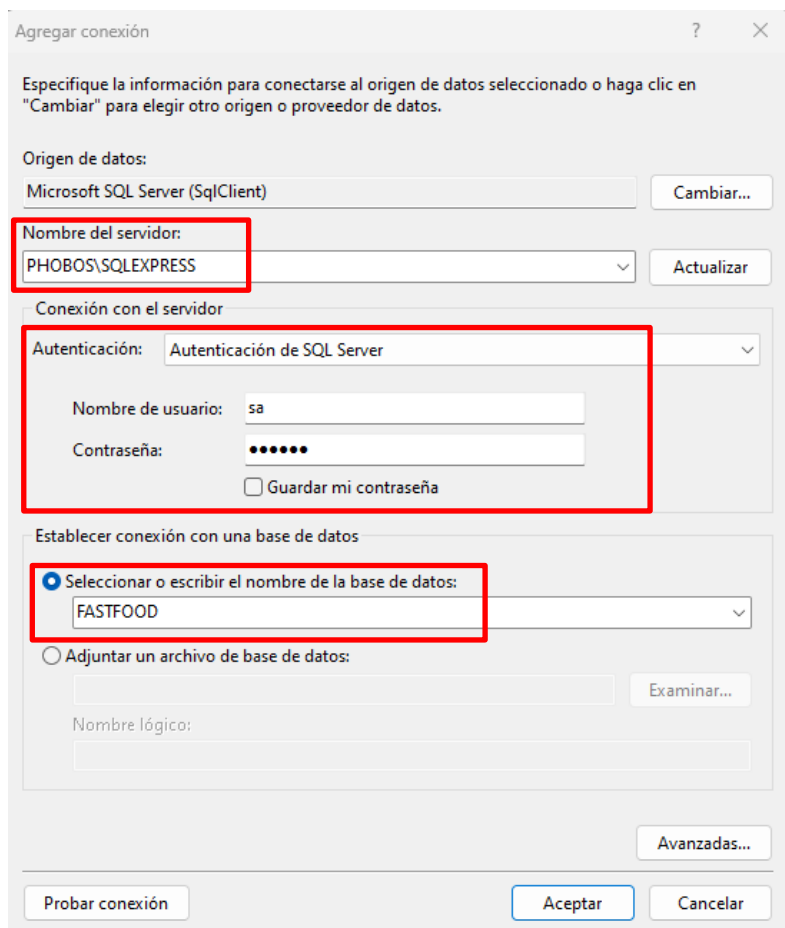
Normalment el nom que és veu a la casella "**nom del servidor**" no és el nom del servidor SQL Server. Per a coneixer el nom del servidor SQL Server podem anar a l'**Administrador de SQL Server** i pulsar sobre el servidor amn el botó dret del ratolí i triar l'opció **propietats**:



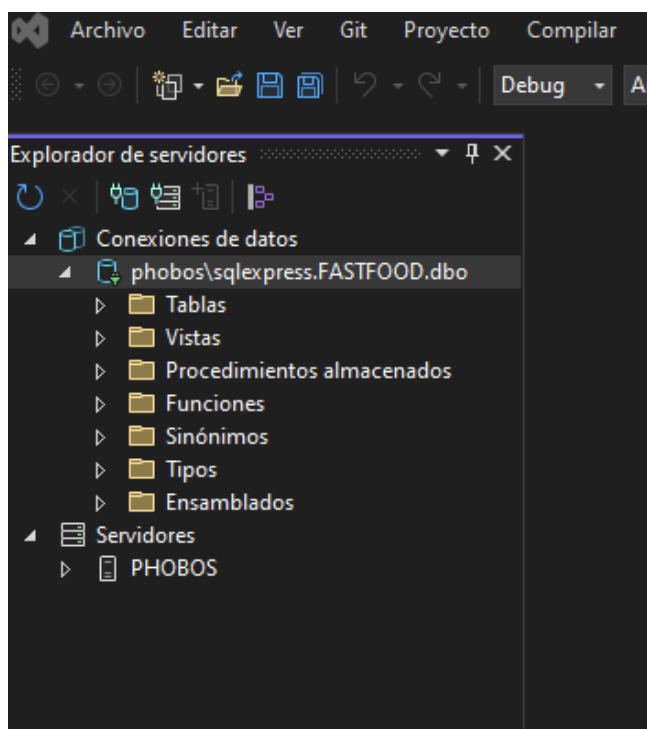
A continuació podem copiar el nom del servidor i pegar-lo en la casella del nom del servidor en la finestra de configuració de la connexió:



En acabar la configuració de la connexió aquesta finestra hauria de quedar així:



Polsem “**Aceptar**” i tindrem la connexió creada:



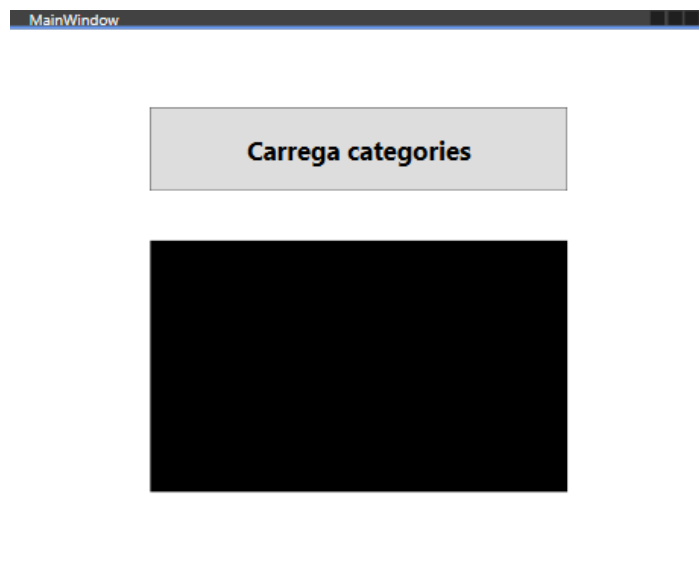
Finalment des de aquesta finestra integrada en el VSC podem vore taules, editarles, insertar dades, etc. sense necessitat d'utilitzar l'Administrador de SQL Server.

3. Accés a dades amb WPF

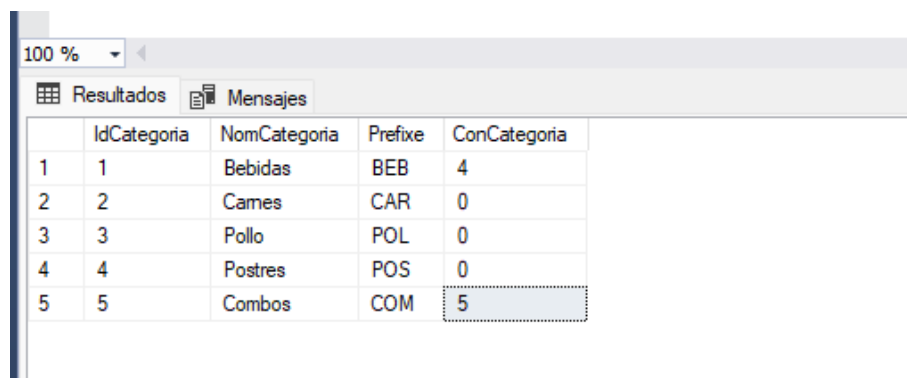
Ja tenim un origen o font de dades, ara vorem com poder accedir a aquestes dades des d'un projecte WPF. Comencem amb un exemple en el que anem a escriure les categories de productes de la base de dades FASTFOOD en un control TextBlock.

3.1. Connectar, consultar i mostrar.

Creem un projecte nou amb un botó i un textblock:



I el contingut de la taula a la que anem accedir és aquest

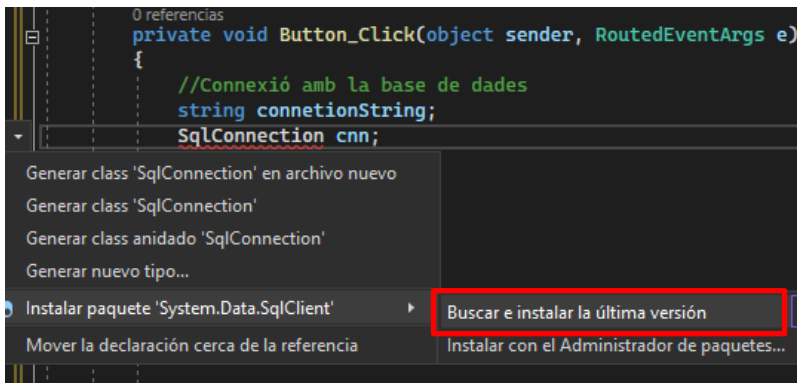


	IdCategoria	NomCategoria	Prefixe	ConCategoria
1	1	Bebidas	BEB	4
2	2	Carnes	CAR	0
3	3	Pollo	POL	0
4	4	Postres	POS	0
5	5	Combos	COM	5

Dins de l'event Click del botó configurarem la connexió, llançarem una SQL (Select * from categorias" i carregarem el Textblock. Començarem amb aquest codi:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    //Connexió amb la base de dades
    string connectionString;
    SqlConnection cnn;
```

El projecte no reconeix l'objecte SqlConnection, per tant hem d'afegir-lo al projecte. Per a fer-ho polsem amb el botó dret sobre l'objecte no reconegut i instal·lem el paquet **System.Data.SqlClient**



Aquesta acció afegirà una referència al paquet **System.Data.SqlClient** a la finestra:

```
using System.Data.SqlClient;
```

A continuació afegim la tira de connexió i obrim la connexió:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    //Connexió amb la base de dades
    string connetionString;
    SqlConnection cnn;

    //Tira de connexió
    connetionString = @"Data Source=PHOBOS\SQLEXPRESS;Initial Catalog=FASTFOOD;User ID=sa;Password=saroot";
    cnn = new SqlConnection(connetionString);

    //Obrim la connexió
    cnn.Open();
}
```

Observa la tira de connexió, en ella s'estableix:

- El nom del servidor SQL Server (Data Source)
- El nom de la base de dades del servidor (Initial Catalog)
- L'usuari i contrasenya.

El següent seria llançar una consulta a la base de dades que torne registres. Creem una variable de tipus string amb la **SQL** i la executem amb un objecte anomenat **SqlCommand**. També necessitarem un objecte **SqlDataReader** per emmagatzemar els resultats de la consulta.

```
//Connexió amb la base de dades
string connetionString;
SqlConnection cnn;

//Tira de connexió
connetionString = @"Data Source=PHOBOS\SQLEXPRESS;Initial Catalog=FASTFOOD;User ID=sa;Password=saroot";
cnn = new SqlConnection(connetionString);

//Obrim la connexió
cnn.Open();

//Preparem la SQL
String sql = "select * from categoria";
//Preparem l'objecte SqlCommand amb la SQL
SqlCommand command = new SqlCommand(sql, cnn);
//Llançem la SQL i o,plim el DataReader
SqlDataReader dataReader = command.ExecuteReader();
```

A continuació podem recórrer els registres del **SQLDataReader** i mostrar-los en el **TextBlock**.

Amb el mètode **Read** del DataReader ens desplaçarem per les files i amb mètode **GetValue(index)** del DataReader obtindrem cada camp de la fila actual.

Observa el codi següent:

```
//Connexió amb la base de dades
string connetionString;
SqlConnection cnn;

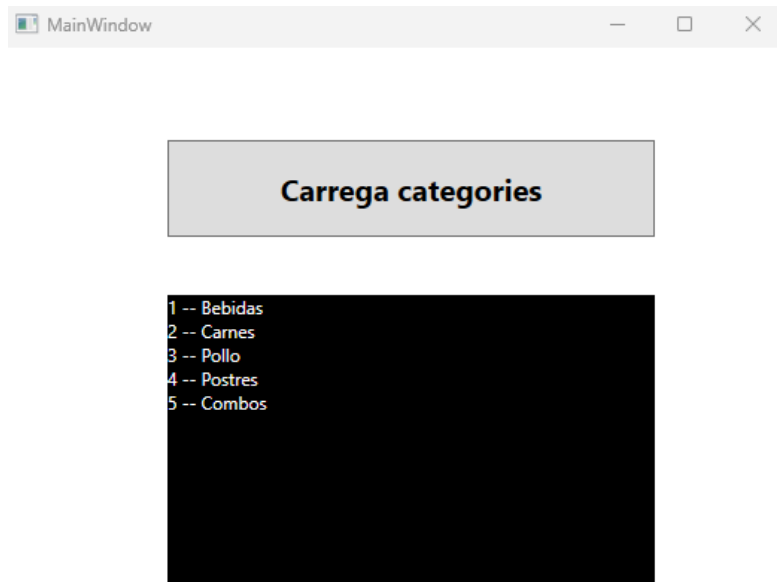
//Tira de connexió
connetionString = @"Data Source=PHOBOS\SQLEXPRESS;Initial Catalog=FASTFOOD;User ID=sa;Password=saroot";
cnn = new SqlConnection(connetionString);

//Obrim la connexió
cnn.Open();

//Preparem la SQL
String sql = "select * from categoria";
//Preparem l'objecte SqlCommand amb la SQL
SqlCommand command = new SqlCommand(sql, cnn);
//Llançem la SQL i o,plim el DataReader
SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())
{
    txtRegistres.Text = txtRegistres.Text + dataReader.GetValue(0) + " -- " + dataReader.GetValue(1) + "\n";
}
```

I el TextBlock quedaria així:

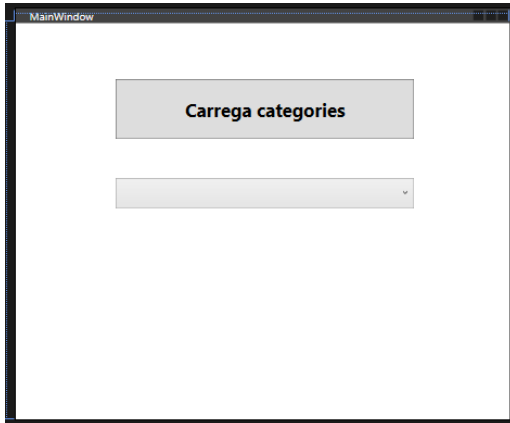


3.2. Enllaçar dades a controls WPF.

A l'exemple anterior carregàvem les dades recorrent les files de l'objecte DataReader. També es possible enllaçar directament els resultats d'una SQL a un control de dades WPF. Vegem com:

Enllaçar dades a un ComboBox

Crearem un projecte amb un botó i un comboBox amb l'objectiu d'omplir el comboBox al pulsar el botó:



El codi XAML d'aquest disseny seria el següent:

```
<Window x:Class="Ud5_ComboBox_BD.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Ud5_ComboBox_BD"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="542">
    <Grid>
        <Button Click="Button_Click" Content="Carrega categories" HorizontalAlignment="Left" Margin="110,62,0,0" VerticalAlignment="Top" Height="30" Width="180"/>
        <ComboBox x:Name="cmbCat" SelectionChanged="cmbCat_SelectionChanged" HorizontalAlignment="Left" Margin="110,170,0,0" VerticalAlignment="Top" Height="30" Width="180"/>
    </Grid>
</Window>
```

Al fer clic al botó s'executaria aquest codi:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    //Connexió amb la base de dades
    string connetionString;
    SqlConnection cnn;

    //Tira de connexió
    connetionString = @"Data Source=PHOBOS\SQLEXPRESS;Initial Catalog=FASTFOOD;User ID=sa;Password=saroot";
    cnn = new SqlConnection(connetionString);

    //Obrim la connexió
    cnn.Open();

    //Preparem la SQL
    String sql = "select * from categoria";
    //Llançem la SQL amb un DataAdapter
    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM CATEGORIA", cnn);
    //Creem i omplim un DataSet amb la info del DataAdapter
    DataSet ds = new DataSet();
    dataAdapter.Fill(ds, "categorias");

    //Enllaçem el comboBoc amb el DataSet
    cmbCat.ItemsSource = ds.Tables["categorias"].DefaultView;
    cmbCat.DisplayMemberPath = "NomCategoria";
    cmbCat.SelectedValuePath = "IdCategoria";

    dataAdapter.Dispose();
    cnn.Close();
}
```

Observa els nous objectes de base de dades que utilitzem:

SqlDataAdapter: És l'objecte que ens permeteix omplir un **DataSet** i reflectirà els canvis en la base de dades cas que el **DataSet** siga modificat. Més informació en <https://learn.microsoft.com/es-es/dotnet/api/system.data.sqlclient.sqldataadapter?view=dotnet-plat-ext-6.0>

DataSet: Es l'objecte que podem associar a un control de dades WPF amb el paràmetre ItemsSource del control. Aquest objecte podem omplir-lo amb un **SQLDataAdapter** o amb altres objectes. Més info en <https://learn.microsoft.com/es-es/dotnet/api/system.data.dataset?view=net-6.0>

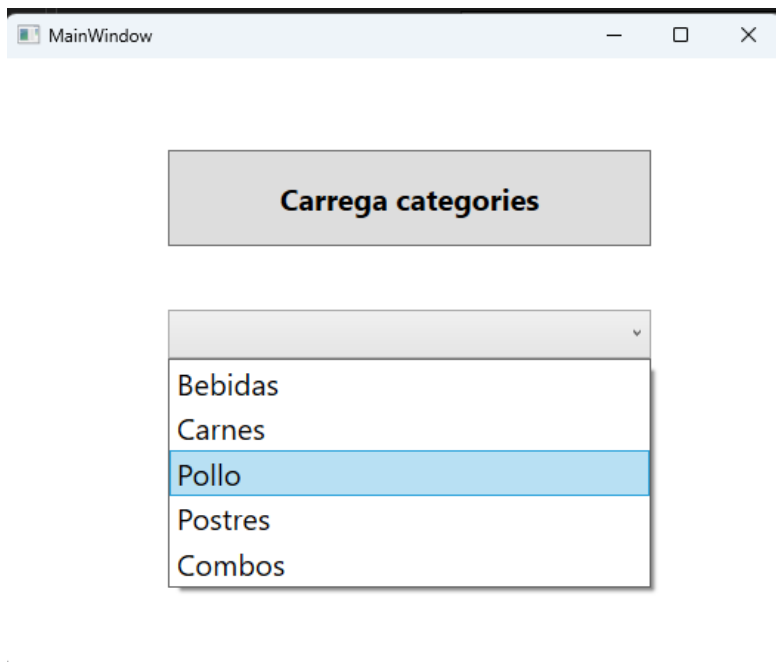
Fixat en el codi, com associem les dades amb el comboBox:

- A la propietat ItemsSource associem el DataSet omplert
`cmbCat.ItemsSource = ds.Tables["categorias"].DefaultView;`
- Establim el camp associat al valor del comboBox i el camp que mostrarà.
`cmbCat.DisplayMemberPath = "NomCategoria";`
`cmbCat.SelectedValuePath = "IdCategoria";`

I per a terminar alliberem la memòria ocupada pel SqlDataAdapter i tanquem la connexió:

```
dataAdapter.Dispose();  
cnn.Close();
```

Finalment, l'aspecte del desplegable després de polsar el botó seria aquest:



Enllaçar dades a un ListBox

El listBox és un control similar al comboBox, l'enllaç de dades és realitza de la mateixa forma que l'exemple anterior.

Enllaçar dades a un DataGrid

Crearem un projecte amb un botó i un datagrid amb l'objectiu d'omplir la cuadrícula al pulsar el botó:



El codi associat al botó seria molt similar al del comboBox:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    //Connexió amb la base de dades
    string connectionString;
    SqlConnection cnn;

    //Tira de connexió
    connectionString = @"Data Source=PHOBOS\SQLEXPRESS;Initial Catalog=FASTFOOD;User ID=sa;Password=saroot";
    cnn = new SqlConnection(connectionString);

    //Obrim la connexió
    cnn.Open();

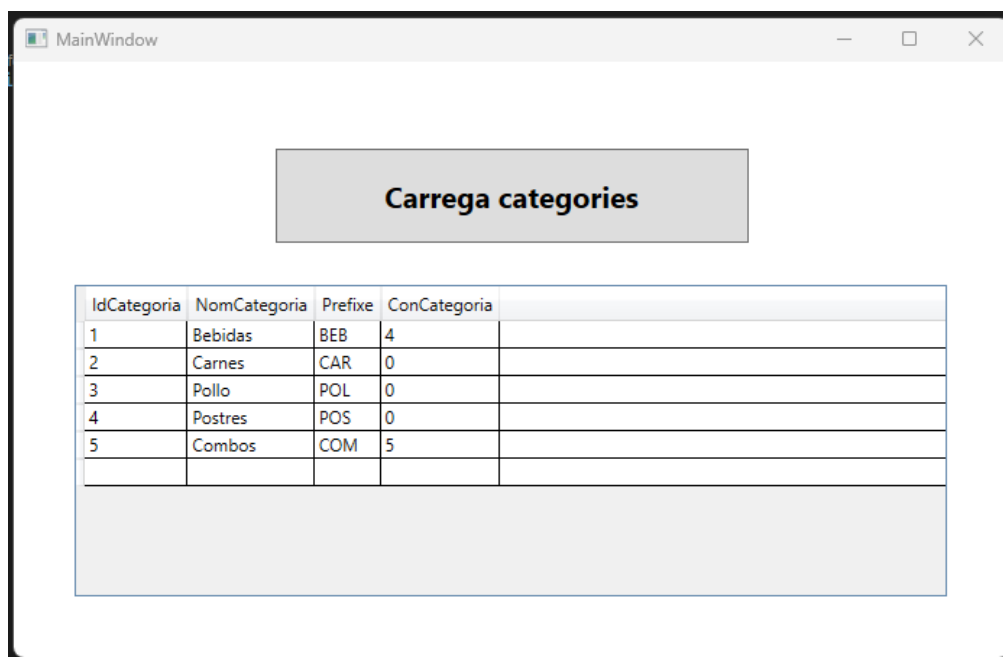
    //Preparem la SQL
    String sql = "select * from categoria";
    //Llançem la SQL amb un DataAdapter
    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM CATEGORIA", cnn);
    //Creem i omplim un DataSet amb la info del DataAdapter
    DataSet ds = new DataSet();
    dataAdapter.Fill(ds, "categorias");

    //Enllaçem el datagrid amb el DataSet
    dgCat.ItemsSource = ds.Tables["categorias"].DefaultView;

    dataAdapter.Dispose();
    cnn.Close();
}
```

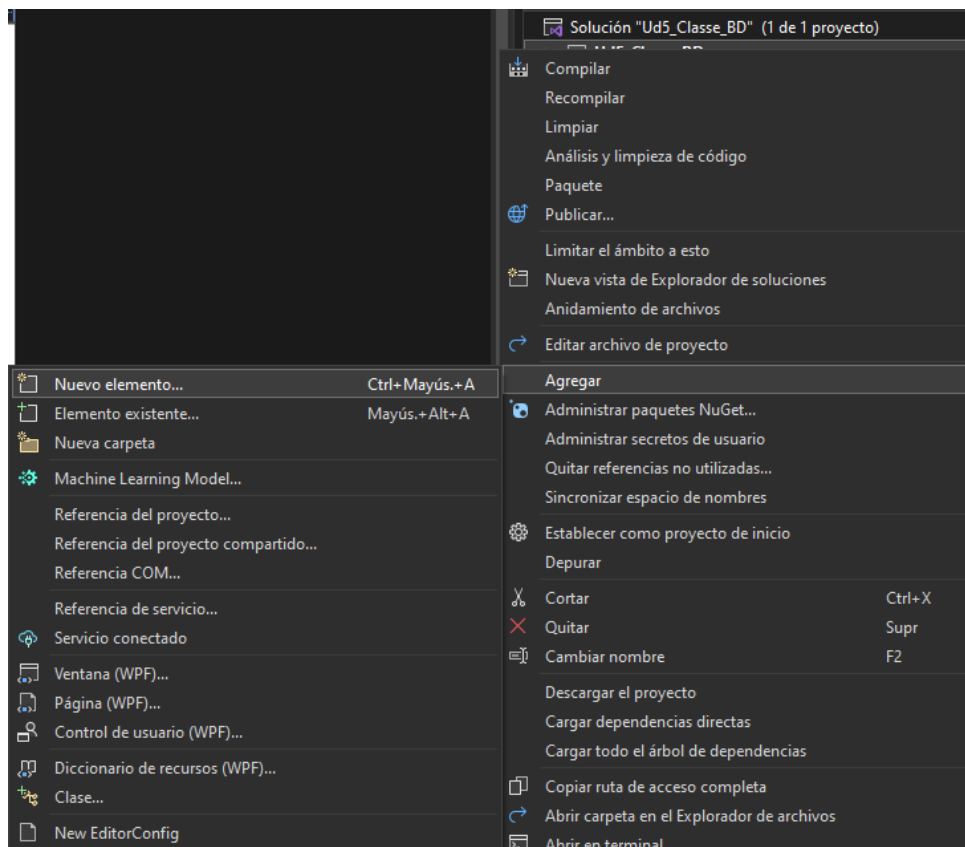
Amb el datagrid únicament és necessari associar el **dataset** a la propietat **ItemsSource**.

Finalment, l'aspecte del datagrid després de polsar el botó seria aquest:

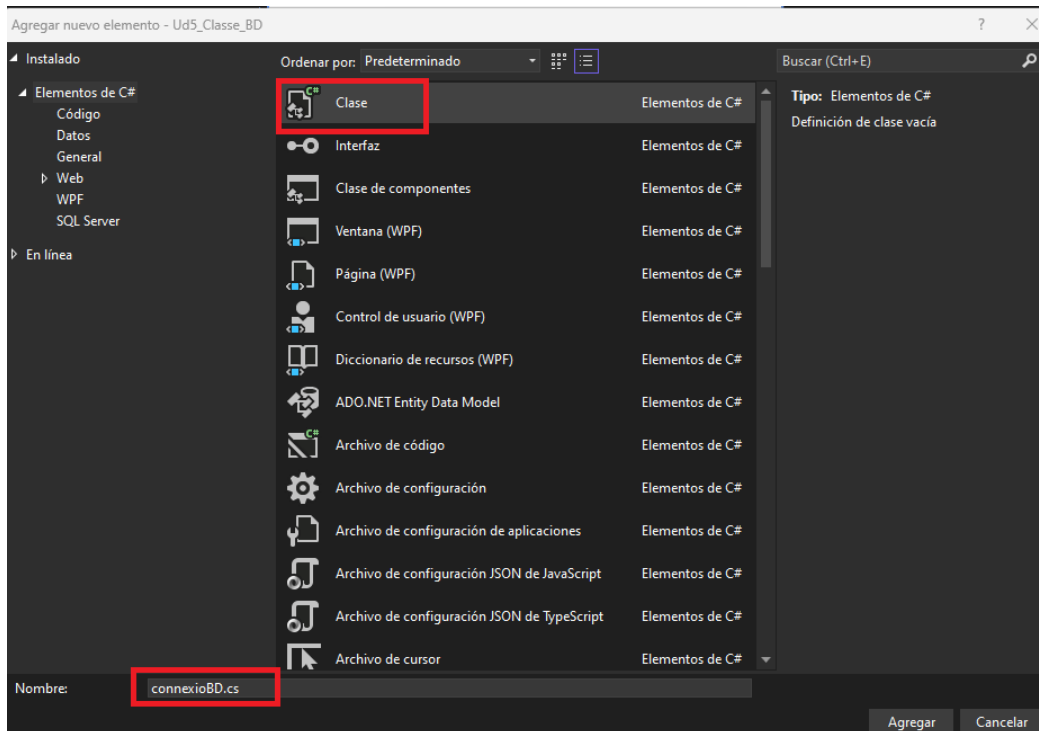


3.3. Ordenant el codi, utilitzant una classe per al model.

Una bona pràctica és separar el codi d'accés a dades de la resta de codi, i així continuarem construint un bon model MVC als nostres projectes. En el cas de accés a dades, una bona idea és crear un classe per connectar-se a la base de dades i accedir a les dades. Per agregar una classe al projecte podem fer-ho polsant amb botó dret del ratolí sobre el projecte i triar les opcions **Agregar → Nou Element**



En la següent finestra triarem l'element **classe** i baix especificarem un nom per a la classe, per exemple **connexioBD.cs**



Polsem **Agregar** i una classe buida amb nom **connexioBD** s'agregarà al nostre projecte:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Ud5_Classe_BD
8 {
9     0 referencias
10     internal class connexioBD
11     {
12     }
13 }
```

I a continuació podem continuar escrivint la nostra classe. Un exemple d'aquesta classe **connexióBD** podria ser el següent (és possible que vosaltres penseu en crear la classe i els mètodes d'altra forma, cadascú pot tindre una idea diferent de la classe):

```
internal class connexioBD
{
    private SqlConnection connexio;
    private string servidor;
    private string bd;
    private string usuari;
    private string contrasenya;

    // Constructor
    1 referencia
    public connexioBD()
    {
        Initialize();
    }

    // Inicialitzem valors
    1 referencia
    private void Initialize()
    {
        servidor = @"PHOBOS\SQLEXPRESS";
        bd = "FASTFOOD";
        usuari = "sa";
        contrasenya = "saroot";
        string connectionString = "Data Source=" + servidor + ";Initial Catalog=" + bd + ";User ID=" + usuari + ";Password=" + contrasenya;
        connexio = new SqlConnection(connectionString);
    }
}
```

```

// Obrim la connexió a la base de dades
0 referencias
private bool ObrirConnexio()
{
    try
    {
        connexio.Open();
        return true;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
        return false;
    }
}

// Tanquem la connexió
0 referencias
private bool TancarConnexio()
{
    try
    {
        connexio.Close();
        return true;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return false;
    }
}
}

```

Observa els mètodes per a Obrir i Tancar la connexió.

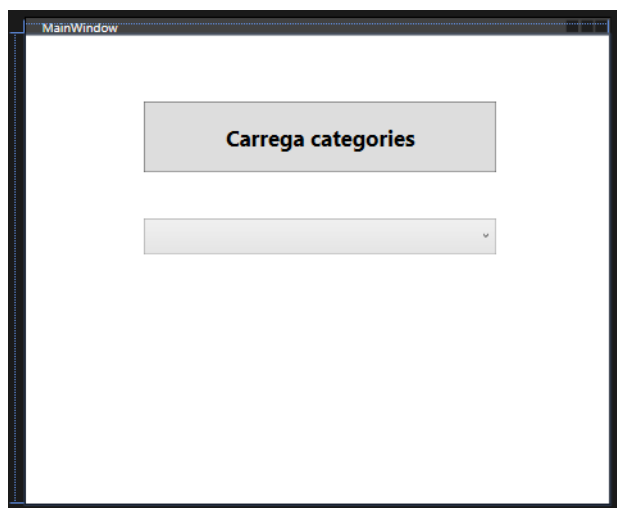
També podríem afegir un mètode que tornara una **DataSet** amb la taula **categories**

```

1 referencia
public DataSet categories()
{
    DataSet ds = new DataSet();
    try
    {
        this.ObrirConnexio();
        SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM CATEGORIA", this.connexio);
        dataAdapter.Fill(ds, "categorias");
        MessageBox.Show("f");
        this.TancarConnexio();
        return ds;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return ds;
    }
}

```

I podríem utilitzar la classe (per exemple) amb l'exemple anterior del comboBox.



El codi (utilitzant la classe) per a carregar el comboBox al pulsar el botó seria el següent:

```
public partial class MainWindow : Window
{
    0 referencias
    public MainWindow()
    {
        InitializeComponent();
    }

    1 referencia
    private void Button_Click(object sender, RoutedEventArgs e)
    {
        connexioBD cnn = new connexioBD();
        DataSet categories = cnn.categories();
        cmbCat.ItemsSource = categories.Tables["categorias"].DefaultView;
        cmbCat.DisplayMemberPath = "NomCategoria";
        cmbCat.SelectedValuePath = "IdCategoria";
    }
}
```

4. Taules: configuració del control DataGrid

En aquest apartat aprofundirem un poc més en un dels components més utilitzats en interfícies com són les taules per a mostrar **dades**, es a dir el **DataGrid**.

El component té com a principals objectes les files i les columnes. Depenent del framework utilitzat per a les interfícies tindrem controls amb més o menys funcionalitats com ordenar, filtrar, agrupar, ...

En els següents apartats anirem veient algunes de les funcionalitats de què disposa el component **DataGrid** per a WPF.

4.1. Autogeneració de columnes.

Per a treballar amb el component DataGrid en WPF resulta bastant senzill si utilitzem l'autogeneració de columnes. Com havíem vist abans, en aquest cas simplement hem d'associar la llista d'objectes que volem representar amb la propietat **ItemsSource** del nostre component DataGrid.

```
<DataGrid ItemsSource="{Binding Customers}" />
```

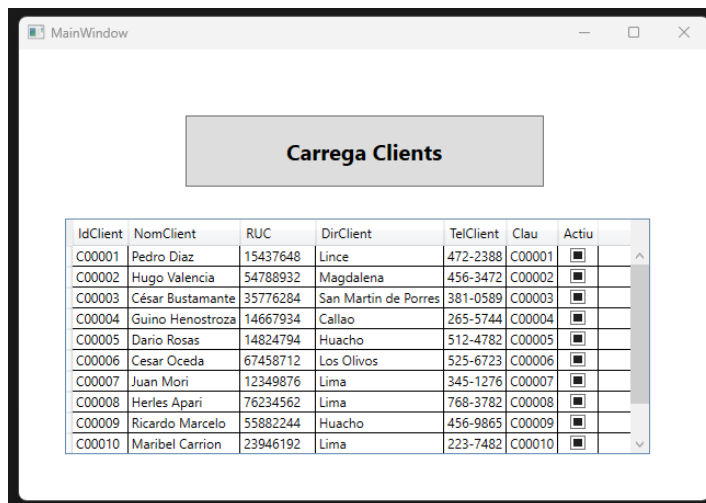
Després d'aquesta assignació, el component llegirà les dades i els representarà de manera automàtica, separant cada propietat dels objectes. A més és capaç de detectar el tipus i adequar el component que se situa en cada cel·la.

Per exemple, si la llista conté un camp de tipus **bool**, aleshores en la cel·la es mostrarà un component **Casella de selecció**.

Els tipus de columnes que hi ha per defecte són les següents:

- **TextBox**: per a valors de tipus string

- **CheckBox:** per a valors de tipus boolean
- **ComboBox:** per a columnes amb valors enumerables, és a dir, que tenen associat una sublista
- **Hyperlink:** per a valors tipus URI



4.2. Definició manual.

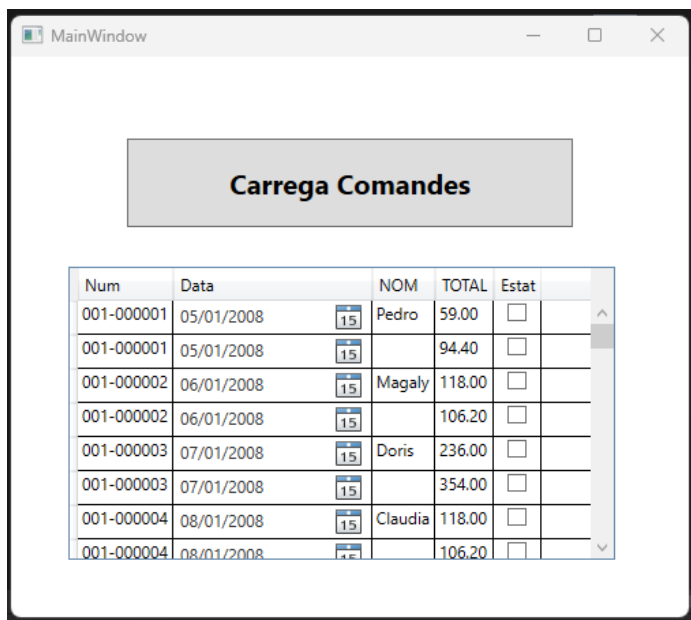
Si no volem mostrar tots els camps dels objectes que conté la llista podem seleccionar-los i mostrar solament les columnes que s'associen a ells. Per a realitzar això hem de deshabilitar l'autogeneració de les columnes (propietat **AutoGenerateColumns = false**).

Això fa que el component no tinga definit cap columna i que hàgem de definir-les nosaltres. Els tipus de columnes que tenim definits per defecte són els següents:

- **DataGridCheckBoxColumn** per a valors booleans
- **DataGridComboBoxColumn** per a valors que són sublistas
- **DataGridHyperlinkColumn** per a valors tipus URI
- **DataGridTextColumn** per a valors de tipus text
- Finalment podem definir tipus personalitzats per a incloure, dates, imatges, etc. Per a això utilitzarem **DataGridTemplateColumn**

Vegem un exemple:

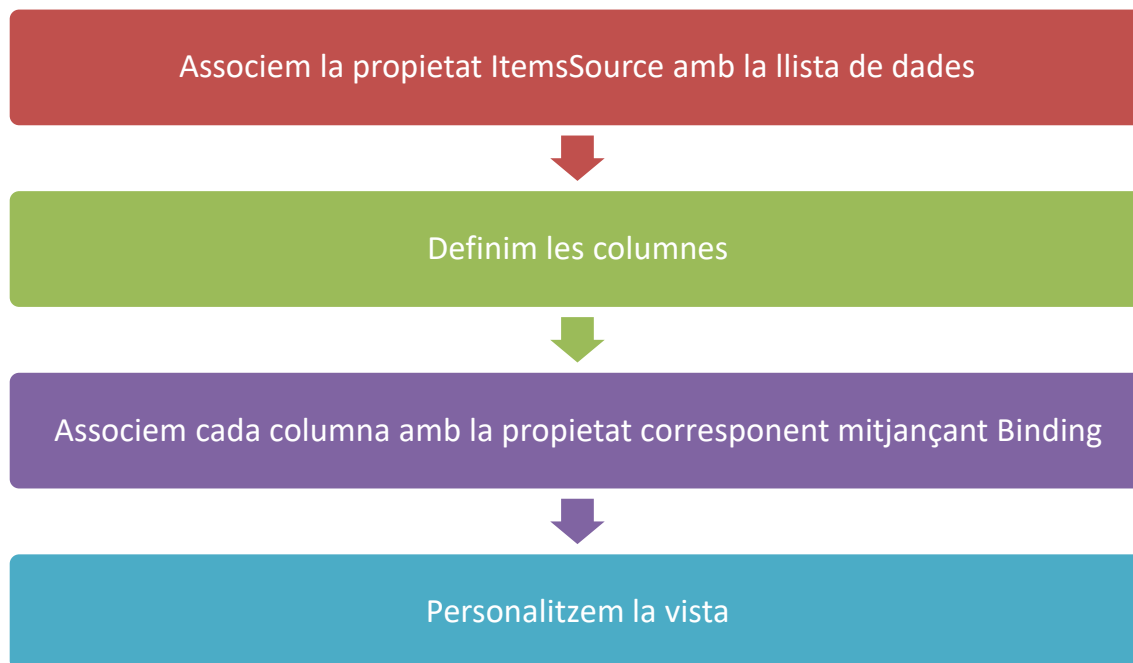
```
<DataGrid x:Name="dgCat" AutoGenerateColumns = "False" Margin="42,154,54,42"
Grid.RowSpan="2">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Num" Binding="{Binding NumDocument}" />
        <DataGridTemplateColumn Header="Data">
            <DataGridTemplateColumn.CellTemplate>
                <DataTemplate>
                    <DatePicker SelectedDate="{Binding Data}" BorderThickness="0" />
                </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
        </DataGridTemplateColumn>
        <DataGridTextColumn Header="NOM" Binding="{Binding NomClient}" />
        <DataGridTextColumn Header="TOTAL" Binding="{Binding Total}" />
        <DataGridCheckBoxColumn Header="Estat" Binding="{Binding Estat}" />
    </DataGrid.Columns>
</DataGrid>
```



En l'exemple podem veure com definim columnes per a **text**, una columna amb un **CheckBox** i una personalitzada amb un **DatePicker** integrat en la celda.

No cal dir que el tipus de columna personalitzada ha de tindre trellat amb el tipus de dades a mostrar (no és pot associar un DatePicker al nom del client).

En el següent dibuix podem resumir els punts per a la utilització de les taules amb WPF



4.3. Ordenar i redimensionar.

El component té la possibilitat d'establir propietats que ens permeten realitzar operacions com ara ordenar, redimensionar o canviar l'ordre de les columnes. Per defecte totes aquestes propietats estan activades. El alguna ocasió potser ens interese canviar el comportament del DataGrid.

Simplement amb activar o desactivar la propietat corresponent podem canviar el comportament per defecte del control.

- **CanUserReorderColumns:** permet canviar les columnes de lloc i reorganitzar-les
- **CanUserResizeColumns:** permet canviar la grandària predefinida de les columnes
- **CanUserResizeRows:** permet canviar la grandària predefinida de les files
- **CanUserSortColumns:** permet ordenar les files en funció del valor d'una o diverses columnes.

```
<DataGrid CanUserReorderColumns="True" CanUserResizeColumns="True"  
          CanUserResizeRows="False" CanUserSortColumns="True"/>
```

4.4. Agrupar.

El component suporta la possibilitat d'agrupar les files per un camp determinat. Per a poder aprofitar aquesta característica hem d'utilitzar com a contenidor de dades per al control DataGrid la classe **ListCollectionView** d'aquesta forma (l'objecte ds és un DataSet), afegint el camp pel qual agruparem els registres o files. :

```
CollectionViewSource miColeccio = new CollectionViewSource();  
miColeccio.Source = ds.Tables["comanda"];  
//Agrupem pel nom de Client  
miColeccio.GroupDescriptions.Add(new PropertyGroupDescription("NomClient"));  
//Enllaçem el datagrid amb la col·lecció  
dgCat.ItemsSource = miColeccio.View;
```

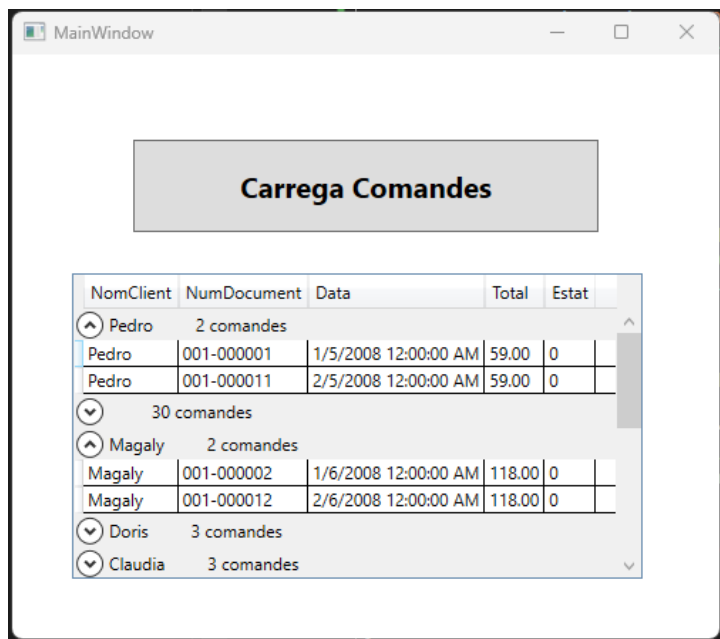
I després la part més complexa, aplicar un estil de group al datagrid:

```

<DataGrid.GroupStyle>
  <GroupStyle>
    <GroupStyle.HeaderTemplate>
      <DataTemplate>
        <StackPanel>
          <TextBlock Text="{Binding Path=Name}" />
        </StackPanel>
      </DataTemplate>
    </GroupStyle.HeaderTemplate>
    <GroupStyle.ContainerStyle>
      <Style TargetType="{x:Type GroupItem}">
        <Setter Property="Template">
          <Setter.Value>
            <ControlTemplate TargetType="{x:Type GroupItem}">
              <Expander>
                <Expander.Header>
                  <StackPanel Orientation="Horizontal">
                    <TextBlock Text="{Binding Path=Name}" />
                    <TextBlock Text="{Binding Path=ItemCount}" Margin="30,0,0,0"/>
                    <TextBlock Text="comandes"/>
                  </StackPanel>
                </Expander.Header>
                <ItemsPresenter />
              </Expander>
            </ControlTemplate>
          </Setter.Value>
        </Setter>
      </Style>
    </GroupStyle.ContainerStyle>
  </GroupStyle>
</DataGrid.GroupStyle>

```


El resultat seria aquest:



4.5. Detalls d'una fila.

Una altra característica interessant és desplegar una fila per a mostrar més dades associades a aqueix objecte. Per exemple, si teniu una llista de tipus de productes, en desplegar podem veure més informació sobre el producte, com mostrar la imatge o mostrar un camp llarg de descripció que nos és mostra a les columnes del DataGrid.

First Name	FirstName	LastName	Gender	Web	ReceiveNewsletter	Image
Christian	Christian	Moser	Male	http/	<input checked="" type="checkbox"/>	Images/christian.jpg
Peter	Peter	Meyer	Male	http/	<input type="checkbox"/>	Images/peter.jpg
Lisa	Lisa	Simpson	Female	http/	<input type="checkbox"/>	Images/lisa.jpg



© wpftutorial.net

Betty	Betty	Bossy	Female	http/	<input type="checkbox"/>	Images/betty.jpg
					<input type="checkbox"/>	

```

<DataGrid AutoGenerateColumns = "False" >
    <DataGrid.Columns>
        <DataGridTextColumn Header="First Name" Binding="{Binding FirstName}" />
        ...
    </DataGrid.Columns>
    <DataGrid.RowDetailsTemplate>
        <DataTemplate>
            <Image Height="100" Source="{Binding Image}" />
        </DataTemplate>
    </DataGrid.RowDetailsTemplate>
</DataGrid>

```

4.6. Altres característiques.

El control DataGrid també aporta altres característiques que poden resultar interessants. Vegem algunes d'elles:

- **Fons altern:** podem canviar el fons de les files de manera alterna.

```

<DataGrid AlternatingRowBackground="Gainsboro" AlternationCount="2"/>

```

FirstName	LastName	Gender	WebSite
Christian	Moser	Male	http://www.wpftutorial.net
Peter	Meyer	Male	http://www.petermeyer.com
Lisa	Simpson	Female	http://www.thesimpsons.com
Betty	Bossy	Female	http://www.bettybossy.ch

- **Columnes fixes:** també podem deixar fixes algunes columnes

```

<DataGrid FrozenColumnCount="2" />

```


FirstName	LastName		Rec	Image
Peter	Meyer	meyer.com	<input type="checkbox"/>	Images/peter.j
Lisa	Simpson	mpsons.com	<input type="checkbox"/>	Images/lisa.jpg
Christian	Moser	torial.net	<input checked="" type="checkbox"/>	Images/christia
Betty	Bossy	bossy.ch	<input type="checkbox"/>	Images/betty.j
			<input type="checkbox"/>	

- **Visibilitat capçalera:** o amagar les capçaleres de les columnes

```
<DataGrid HeadersVisibility="None" />
```

None ▾			
Betty	Bossy	Female	http://www.bettybossy.ch
Christian	Moser	Male	http://www.voftutorial.net
Lisa	Simpson	Female	http://www.thesimpsons.com
Peter	Meyer	Male	http://www.petermeyer.com

5. Llistes: configuració del control ListBox

Es tracta d'un component més senzill que un DataGrid però amb funcionalitats similars, bàsicament es tracta de mostrar una llista d'objectes, dels quals es mostra una propietat o unes algunes d'aquest.

S'utilitza a nivell informatiu en la majoria dels casos, com per exemple el resultat d'una recerca o com una llista de missatges.

5.1. Ordenació

Les llistes tenen un ordre per defecte, el qual podem canviar perquè altre camp el que predomine en la manera d'ordenar-la.

Per a això utilitzarem el següent codi en el mètode on vulguem ordenar-la.

```
miColeccio.Source = ds.Tables["comandes"];
miColeccio.SortDescriptions.Add(new SortDescription("NomClient",
ListSortDirection.Descending));
miColeccio.SortDescriptions.Add(new SortDescription("Data",
ListSortDirection.Ascending));
```

Utilitzem el mètode **SortDescriptions**, al qual afegim tants objectes de la classe **SortDescription** com a capes d'ordenació vulguem afegir.

En l'exemple anterior es realitzarà un ordre pels camps *NomClient* i *Data* en sentit descendent el primer i ascendent el segon.

Vegem l'exemple complet:

Codi C# que carrega el ListBox:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    //Connexió amb la base de dades
    string connetionString;
    SqlConnection cnn;

    //Tira de connexió
    connetionString = @"Data Source=PHOBOS\SQLEXPRESS;Initial Catalog=FASTFOOD;User ID=sa;Password=saroot";
    cnn = new SqlConnection(connetionString);

    //Obrim la connexió
    cnn.Open();

    //Preparem la SQL
    String sql = "select * from comanda";
    //Llançem la SQL amb un DataAdapter
    SqlDataAdapter dataAdapter = new SqlDataAdapter(sql, cnn);
    //Creem i omplim un DataSet amb la info del DataAdapter
    DataSet ds = new DataSet();
    dataAdapter.Fill(ds, "comandes");

    CollectionViewSource miColeccio = new CollectionViewSource();
    miColeccio.Source = ds.Tables["comandes"];
    miColeccio.SortDescriptions.Add(new SortDescription("NomClient", ListSortDirection.Descending));
    miColeccio.SortDescriptions.Add(new SortDescription("Data", ListSortDirection.Ascending));

    listComandes.ItemsSource = miColeccio.View;

    dataAdapter.Dispose();
    cnn.Close();
}
```

Codi XAML que dona format al ListBox:

```
<Grid>
<Button Click="Button_Click" Content="Carrega comandes" HorizontalAlignment="Left" Margin="110,62,0,0" VerticalAlignment="Top" Height="30" Width="150"/>
<ListBox x:Name="listComandes" Margin="39,164,50,46">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="30" SharedSizeGroup="Column1"/>
                    <ColumnDefinition Width="100" SharedSizeGroup="Column2"/>
                    <ColumnDefinition Width="150" SharedSizeGroup="Column3"/>
                </Grid.ColumnDefinitions>
                <!-- Assumes MVVM and you wish to bind to properties and commands -->
                <TextBlock Grid.Column="0" Text="{Binding idComanda}"/>
                <TextBlock Grid.Column="1" Text="{Binding Data, StringFormat=dd-MM-yyyy}"/>
                <TextBlock Grid.Column="2" Text="{Binding NomClient}"/>
            </Grid>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
</Grid>
```

I el resultat:



5.2. Agrupar

També podem realitzar l'agrupació dels ítems en funció d'un camp i que els elements es vegen de manera contigua.

```
PropertyGroupDescription groupDescription = new
PropertyGroupDescription();

groupDescription.PropertyName = "Data";

listingDataView.GroupDescriptions.Add(groupDescription);
```

En l'exemple podem veure la manera d'agrupar els elements per la propietat **Data**.

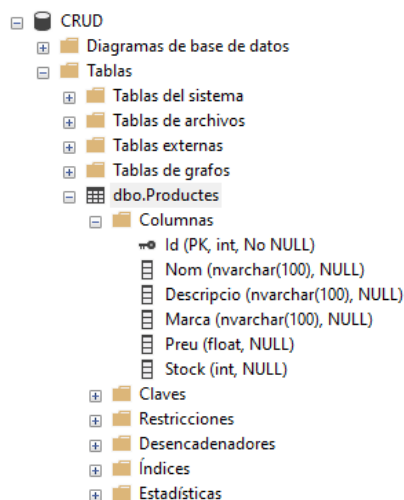
Recorda que sempre hem de fer ús d'un objecte **CollectionViewSource** (al exemple és l'objecte **listingDataView**)

6. CRUD amb WPF

Les sigles CRUD signifiquen les 4 operacions bàsiques que es poden realitzar amb una base de dades: **C**reate, **R**ead, **U**ppdate i **D**eleate

Podem denominar aplicació CRUD a aquella capaç fer les 4 operacions des de la mateixa finestra de dades.

Vegem un exemple de com fer-ho amb una base de dades simple amb una taula, per exemple aquesta (amb una única taula Productes):



Resultados		Mensajes				
	Id	Nom	Descripcio	Marca	Preu	Stock
1	1	Gaseosa	3 litres	marcacola	7	24
2	2	Xocolata	Tauleta 100 grams	iberica	12	36
3	3	Cola	2 litres	marcacola	8	24
4	5	Gin	0,80 litres	schweppes	7	24

El primer que hem de fer és definir una classe per a la connexió a la base de dades i les operacions amb ella (Select, Update, Insert i Delete):

```

internal class ConnexioBD
{
    private SqlConnection connexio;
    private string servidor;
    private string bd;
    private string usuari;
    private string contrasenya;

    // Constructor
    3 referencias
    public ConnexioBD()
    {
        Initialize();
    }

    // Inicialitzem valors
    1 referencia
    private void Initialize()
    {
        servidor = @"PHOBOS\SQLEXPRESS";
        bd = "CRUD";
        usuari = "sa";
        contrasenya = "saroot";
        string connectionString = "Data Source=" + servidor + ";Initial Catalog=" + bd + ";User ID=" + usuari + ";Password=" + contrasenya;
        connexio = new SqlConnection(connectionString);
    }

    // Obrim la connexió a la base de dades
    4 referencias
    private bool ObrirConnexio()
    {
        try
        {
            connexio.Open();
            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
            return false;
        }
    }
}

```

```

// Tanquem la connexió
4 referencias
private bool TancarConnexio()
{
    try
    {
        connexio.Close();
        return true;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return false;
    }
}

1 referencia
public DataSet getProductes()
{
    DataSet ds = new DataSet();
    try
    {
        this.ObrirConnexio();
        SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM PRODUCTES", this.connexio);
        dataAdapter.Fill(ds, "productes");
        this.TancarConnexio();
        return ds;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return ds;
    }
}

```

```
1 referencia
public void Editar(string nom, string desc, string marca, double preu, int stock, int id)
{
    this.ObrirConnexio();
    String SqlUpdate = "update Productes set Nom='" + nom + "', Descripcio='" + desc + "', " +
        "Marca='" + marca + "', Preu=" + preu + ", Stock=" + stock + " where Id=" + id;
    SqlCommand comand = new SqlCommand(SqlUpdate, this.connexio);
    comand.ExecuteNonQuery();
    this.TancarConnexio();
}
```

```
1 referencia
public void Nou(string nom, string desc, string marca, double preu, int stock)
{
    this.ObrirConnexio();
    String SqlInsert = "Insert into Productes (Nom, Descripcio, Marca, Preu, Stock) values " +
        "(" + nom + "', '" + desc + "', '" + marca + "', " + preu + ", " + stock + " )";
    MessageBox.Show(SqlInsert);
    SqlCommand comand = new SqlCommand(SqlInsert, this.connexio);
    comand.ExecuteNonQuery();
    this.TancarConnexio();
}
```

```
1 referencia
public void Eliminar( int id)
{
    this.ObrirConnexio();
    String SqlDelete = "delete from Productes where Id=" + id;
    SqlCommand comand = new SqlCommand(SqlDelete, this.connexio);
    comand.ExecuteNonQuery();
    this.TancarConnexio();
}
```

Observa els mètodes per a modificar la base de dades (Insert, Update, Delete). Utilitzen un objecte anomenat **SqlCommand**. Aquest objecte ens serveix per a llançar SQL's que tornen resultats.

El següent pas seria definir la finestra de treball amb un datagrid, botons i caixes de text per a editar o inserta nous productes. El seu aspecte seria aquest:

El codi XAML de la finestra seria aquest:

```
Window x:Class="Ud5_CRUD.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Ud5_CRUD"
    mc:Ignorable="d"
    Title="Productes" Height="450" Width="856">
    <Grid Margin="10,0,0,0">
        <Grid.ColumnDefinitions>
            <ColumnDefinition x:Name="gridCol0" Width="500" SharedSizeGroup="Column1"/>
            <ColumnDefinition x:Name="gridCol1" Width="325" SharedSizeGroup="Column2"/>
        </Grid.ColumnDefinitions>
        <DataGrid x:Name="dgProductes" Margin="0,10,0,84" FontSize="20"/>
        <Button x:Name="btnEditar" Click="btnEditar_Click" Content="Editar" Background="Black" Foreground="White" HorizontalAlignment="Left" Margin="0,0,0,0"/>
        <Button x:Name="btnEliminar" Click="btnEliminar_Click" Content="Eliminar" Background="Black" Foreground="White" HorizontalAlignment="Left" Margin="0,0,0,0"/>
        <Button x:Name="btnNou" Click="btnNou_Click" Content="Nou" Background="Black" Foreground="White" HorizontalAlignment="Left" Margin="0,0,0,0"/>
        <Button x:Name="btnGuardar" Click="btnGuardar_Click" Content="Guardar" Background="Black" Foreground="White" HorizontalAlignment="Left" Margin="0,0,0,0"/>
        <TextBox x:Name="txtId" Grid.Column="1" HorizontalAlignment="Left" Margin="99,19,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="226"/>
        <TextBox x:Name="txtNom" Grid.Column="1" HorizontalAlignment="Left" Margin="99,69,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="226"/>
        <TextBox x:Name="txtDesc" Grid.Column="1" HorizontalAlignment="Left" Margin="99,115,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="226"/>
        <TextBox x:Name="txtMarca" Grid.Column="1" HorizontalAlignment="Left" Margin="99,161,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="226"/>
        <TextBox x:Name="txtPreu" Grid.Column="1" HorizontalAlignment="Left" Margin="99,207,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="226"/>
        <TextBox x:Name="txtStock" Grid.Column="1" HorizontalAlignment="Left" Margin="99,253,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="226"/>
        <Label Grid.Column="1" Content="ID" HorizontalAlignment="Left" Margin="20,19,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.34,0.21"/>
        <Label Grid.Column="1" Content="NOM" HorizontalAlignment="Left" Margin="20,69,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.34,0.21"/>
        <Label Grid.Column="1" Content="DESC." HorizontalAlignment="Left" Margin="20,115,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.34,0.21"/>
        <Label Grid.Column="1" Content="MARCA" HorizontalAlignment="Left" Margin="20,161,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.34,0.21"/>
        <Label Grid.Column="1" Content="PREU" HorizontalAlignment="Left" Margin="20,207,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.34,0.21"/>
        <Label Grid.Column="1" Content="STOCK" HorizontalAlignment="Left" Margin="20,253,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.34,0.21"/>
    </Grid>
</Window>
```

Observem amb deteniment el codi:

- S'ha dividit el contenidor (grid) en 2 columnes, una para el datagrid i l'altra per a l'edició o adició de producte.
- Tenim 4 botons: Editar, Eliminar, Nou i Guardar. Els 2 primers únicament seran efectius si s'ha seleccionat una filera del DataGrid.

A continuació observem els mètodes que cal definir al codi C# associat a la finestra principal:

```
public String estat = "";
0 referencias
public MainWindow()
{
    InitializeComponent();
    MostrarProductes();
}

3 referencias
private void MostrarProductes()
{
    ConnexioBD cnn = new ConnexioBD();
    DataSet ds = cnn.getProductes();
    dgProductes.ItemsSource = ds.Tables["productes"].DefaultView;
    gridCol1.Width = new GridLength(0, GridUnitType.Star);
}
```

Al iniciar l'aplicació i cada vegada que fem un canvi en la base de dades (Insert, Update o Delete) cridarem al mètode **MostrarProductes()** per a actualitzar el contingut del DataGrid. També ocultarà la columna de la dreta col·locant el seu ample a 0.

També definim la propietat estat per a saber si estem **Editant** o **Insertant** quan polsem el botó **Guardar**.

El mètode controlador del botó Editar (**btnEditar_Click**) comprovarà que s'ha seleccionat una filera i carregarà en els quadres de text la informació corresponent a eixa filera (el quadre de text amb el Id de l Producte sempre estarà deshabilitat). També mostrarà la columna de la dreta col·locant el seu ample a 325.

```
private void btnEditar_Click(object sender, RoutedEventArgs e)
{
    if (this.dgProductes.SelectedItem != null)
    {
        var rowview = dgProductes.SelectedItem as DataRowView;
        if (rowview != null)
        {
            DataRow row = rowview.Row;
            txtId.Text = row.ItemArray[0].ToString();
            txtNom.Text = row.ItemArray[1].ToString();
            txtDesc.Text = row.ItemArray[2].ToString();
            txtMarca.Text = row.ItemArray[3].ToString();
            txtPreu.Text = row.ItemArray[4].ToString();
            txtStock.Text = row.ItemArray[5].ToString();
            gridColl.Width = new GridLength(325, GridUnitType.Star);
            estat = "Editant";
        }
    }
    else MessageBox.Show("Has de seleccionar una fila", "Informació", MessageBoxButton.OK, MessageBoxImage.Information);
}
```

El mètode controlador del botó Nou (**btnNou_Click**) buidarà els quadres de text i mostrarà la columna de la dreta col·locant el seu ample a 325.

```
1 referencia
private void Nou_Click(object sender, RoutedEventArgs e)
{
    txtId.Text = "";
    txtNom.Text = "";
    txtDesc.Text = "";
    txtMarca.Text = "";
    txtPreu.Text = "";
    txtStock.Text = "";
    gridColl.Width = new GridLength(325, GridUnitType.Star);
    estat = "Nou";
}
```

El mètode controlador del botó Guardar (**btnGuardar_Click**) cridarà al mètode adequat de la classe ConnexioBD en funció de si la propietat **estat** té un valor o altre.

```
private void btnGuardar_Click(object sender, RoutedEventArgs e)
{
    ConnexioBD cnn = new ConnexioBD();
    if (estat == "Editant")
    {
        cnn.Editar(txtNom.Text, txtDesc.Text, txtMarca.Text, double.Parse(txtPreu.Text), int.Parse(txtStock.Text), int.Parse(txtId.Text));
    }
    if (estat == "Nou")
    {
        cnn.Nou(txtNom.Text, txtDesc.Text, txtMarca.Text, double.Parse(txtPreu.Text), int.Parse(txtStock.Text));
    }
    MostrarProductes();
}
```

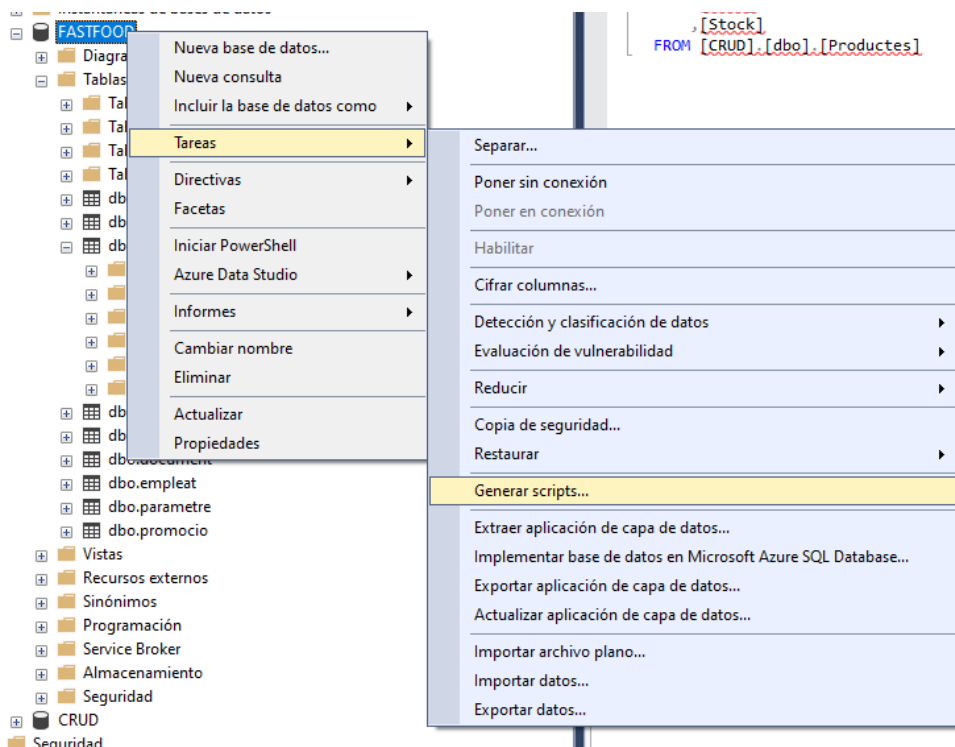

Per últim tindrem el mètode controlador del botó Eliminar (**btnEliminar_Click**), que comprovarà que l'usuari vol eliminar el registre i en cas afirmatiu cridarà al mètode **Eliminar** de la classe ConnexioBD.

```
private void btnEliminar_Click(object sender, RoutedEventArgs e)
{
    if (this.dgProductes.SelectedItem != null)
    {
        var rowview = dgProductes.SelectedItem as DataRowView;
        if (rowview != null)
        {
            DataRow row = rowview.Row;
            String id = row.ItemArray[0].ToString();
            MessageBoxResult result = MessageBox.Show("Vols eliminar el registre " + id + "?", "Confirmació", MessageBoxButton.YesNo, Mes
            if (result==MessageBoxResult.Yes)
            {
                ConnexioBD cnn = new ConnexioBD();
                cnn.Eliminar(int.Parse(id));
                MostrarProductes();
            }
        }
    }
    else MessageBox.Show("Has de seleccionar una fila", "Informació", MessageBoxButton.OK, MessageBoxImage.Information);
}
```

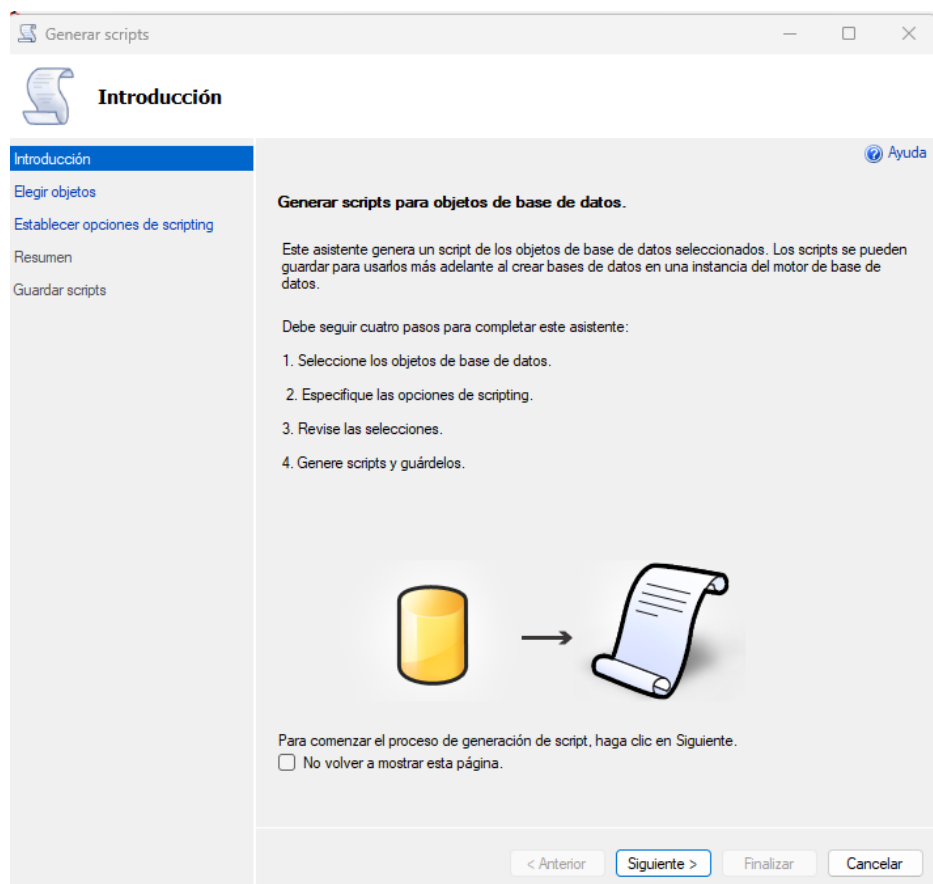
7. Annex: Exportació d'Esquema i Dades de SQL Server

Des de l'eina d'Administració de SQL Server podem exportar en format SQL l'esquema de una base de dades i els seu contingut. Vegem com:

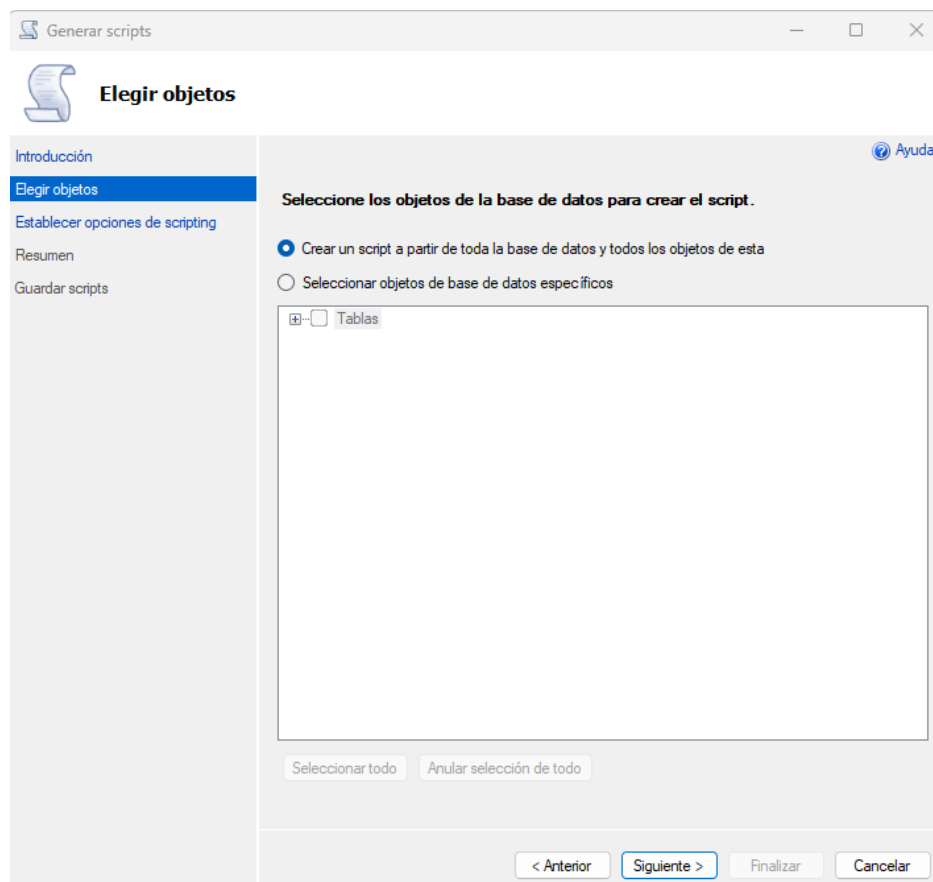
Primer hem de fer click amb el botó dret sobre la base de dades que volem exportar, triant l'opció Tasques → : Generar Scripts



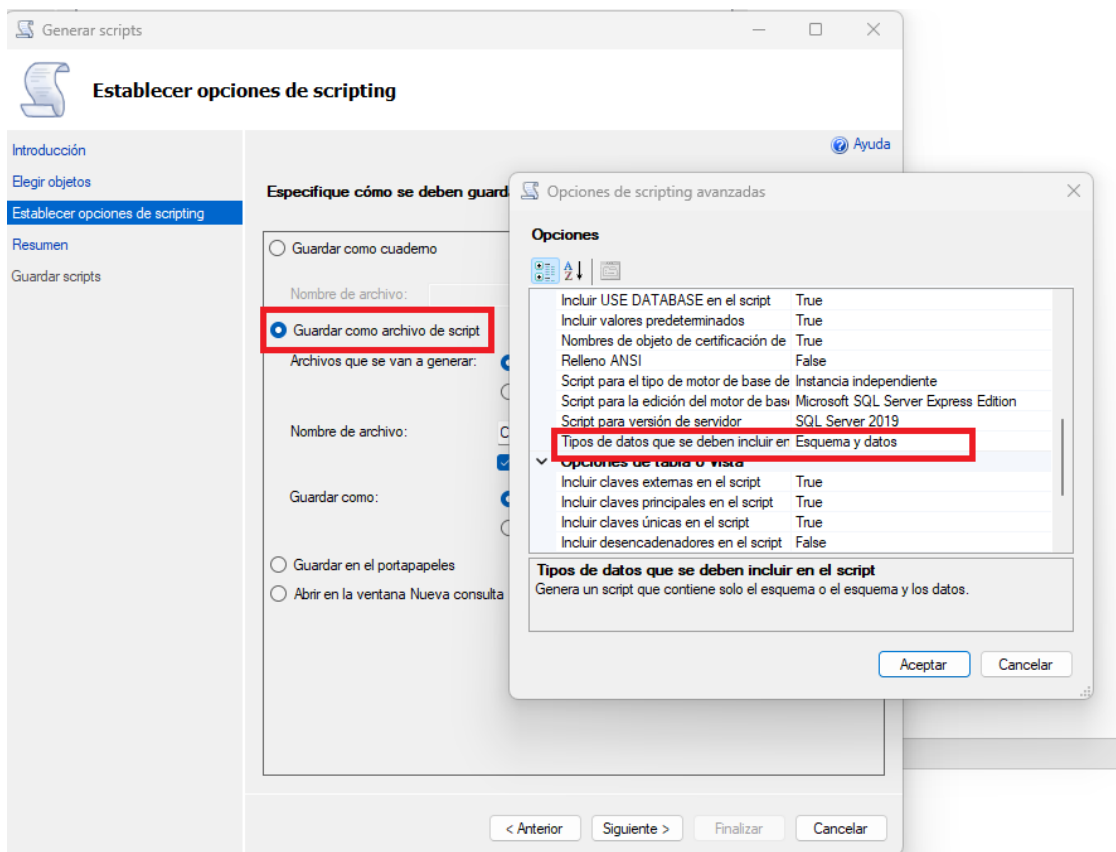
Apareixerà la següent finestra:



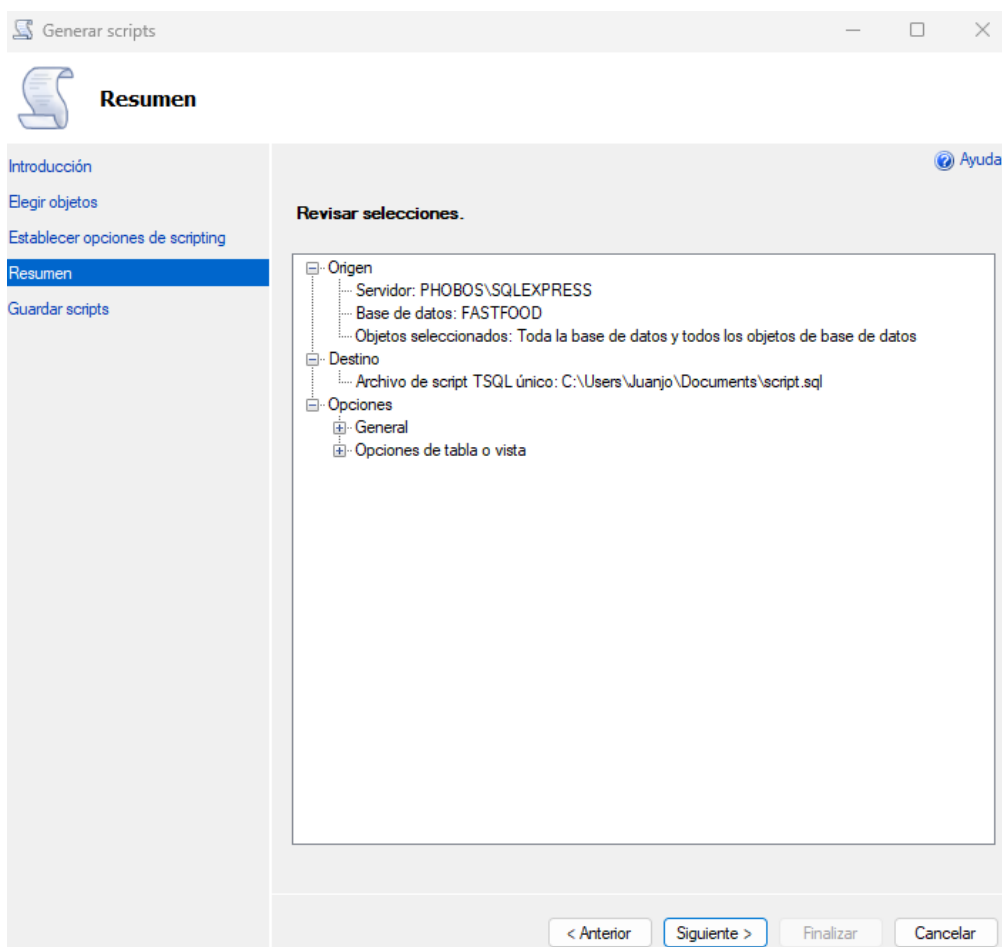
Polsem Següent:



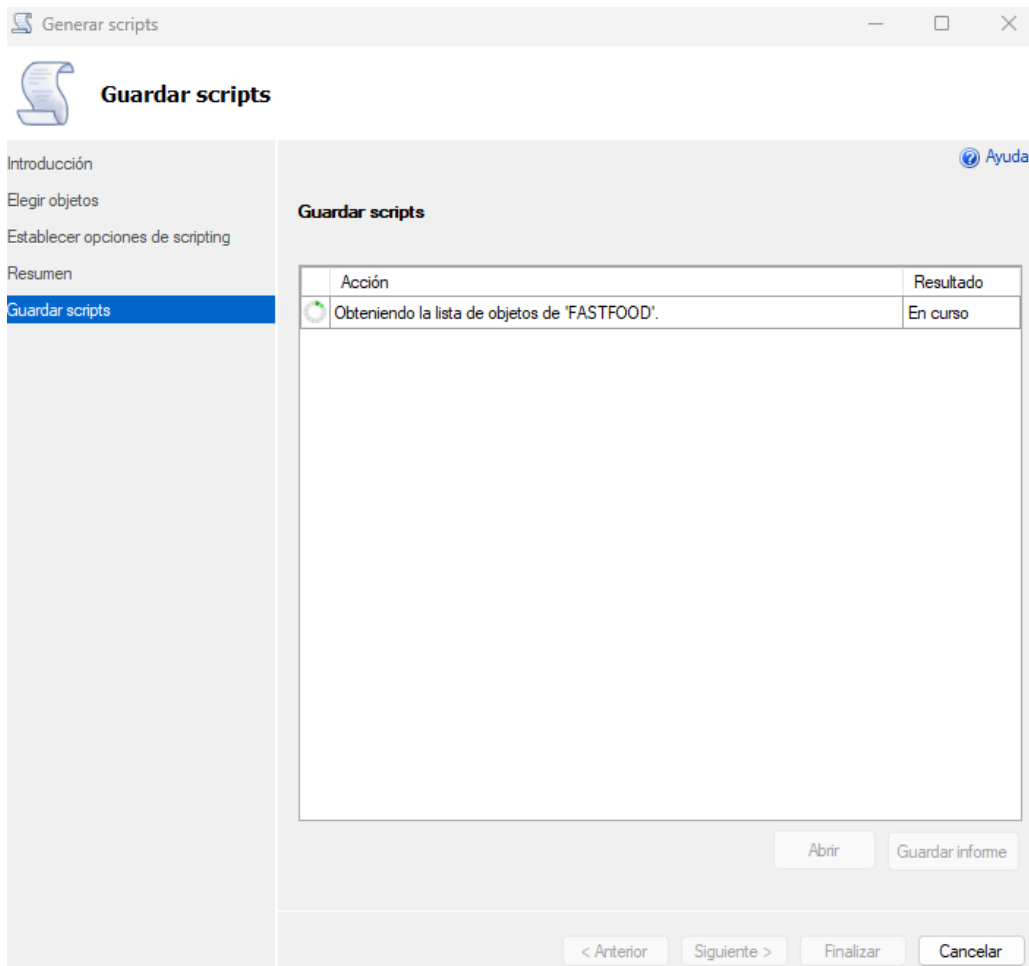
Exportarem tota la base de dades, polsem **Següent**:



En aquesta finestra triarem l'opció **“Guardar com arxiu de script”** i polsarem el botó **“Avançades”** i triarem exportat l'**esquema i les dades**. Polsem **Següent**:



Ara ens mostra un resum de les opcions triades i al polsar Següent començarà l'exportació:



Quan finalitze tindrem aquesta finestra i l'arxiu SQL en la carpeta proposada.

