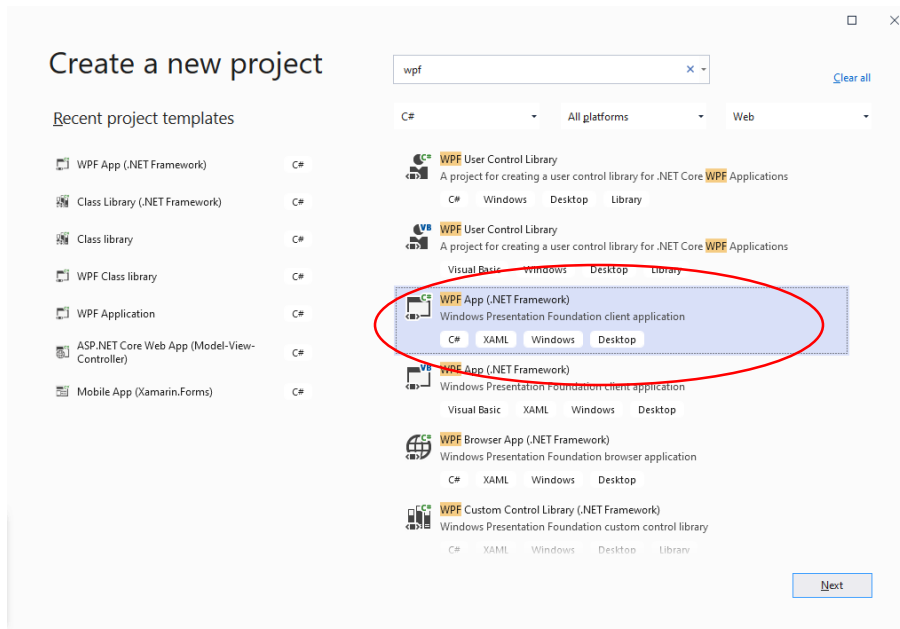


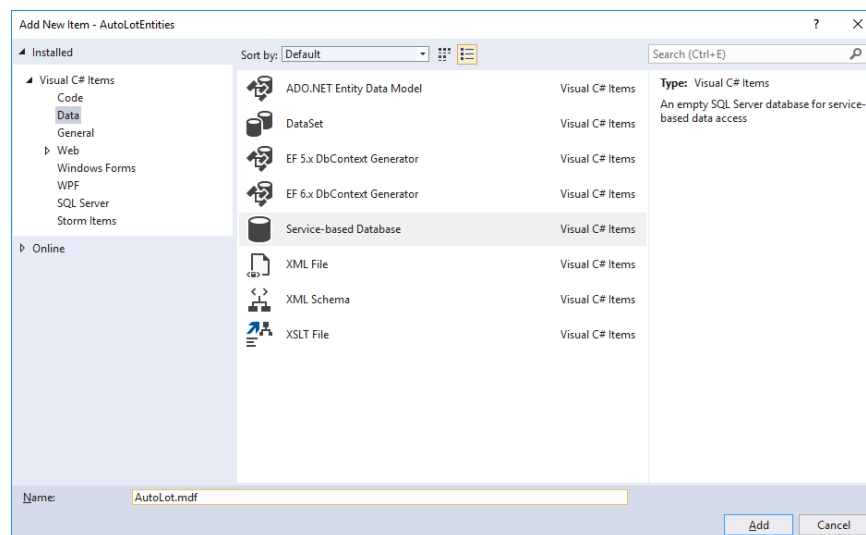
Laborator 5 – Aplicatie WPF folosind Entity Framework - Code First from DataBase

1. Se creează un nou proiect de tipul **WPF App (.NET Framework)** si se denumeste **Nume_Pren_Lab5**

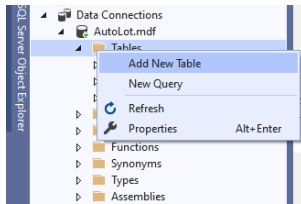


2. Vom crea baza de date. Baza noastră de date va contine 3 tabele.

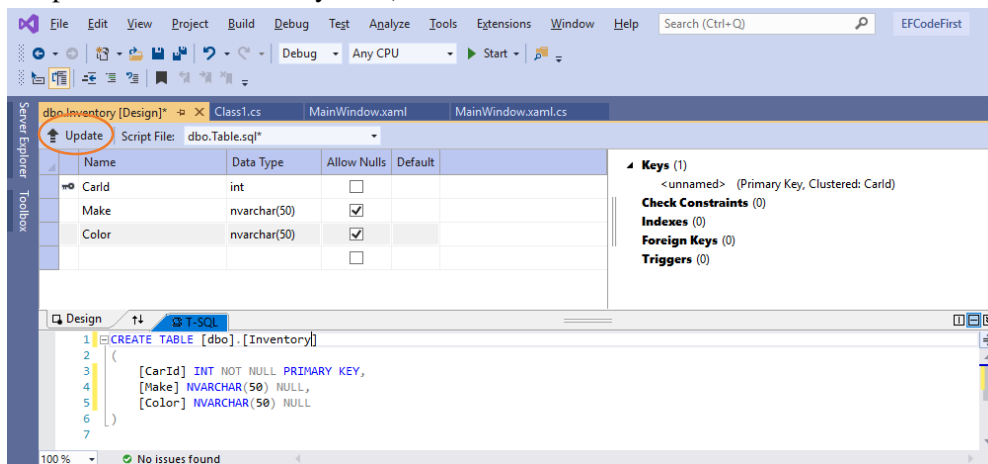
Din meniul **Project** selectăm **Add New Item** care ne va deschide o fereastră de adăugare a unui element nou. În această fereastră, în structura arborescentă de template-uri alegem **Visual C# > Data**, iar din lista alăturată alegem să creem un fișier de tip **Service-based Database**, pe care îl denumim **AutoLot.mdf**, apoi dăm click pe **Add**:



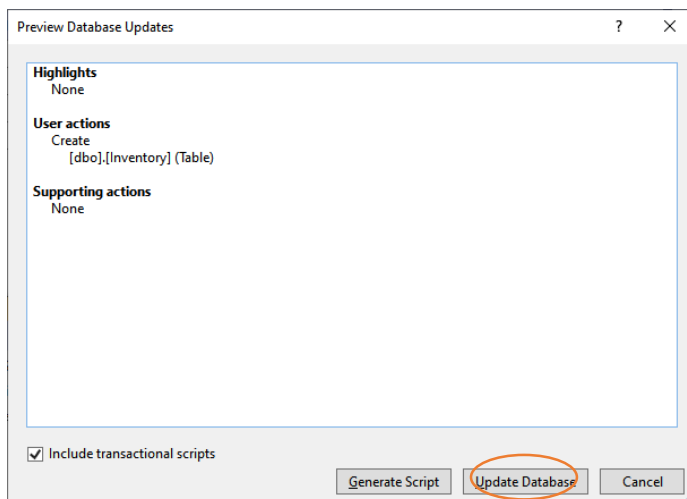
3. Mai departe vom trece la crearea structurii bazei noastre de date. În **Solution Explorer** dăm dublu click pe fișierul **AutoLot.mdf**. Astfel, ni se va deschide fereastra **Server Explorer**, unde observăm folderul **Tables** cu care vom lucra în continuare.



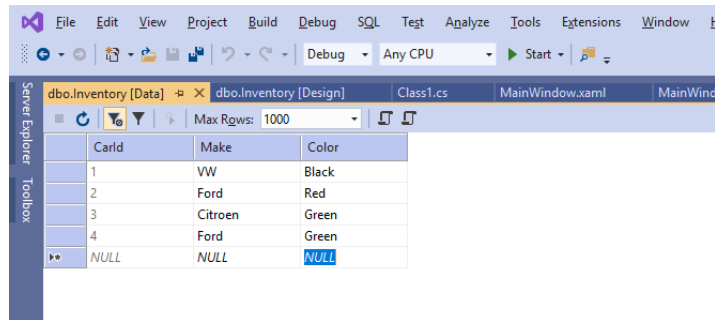
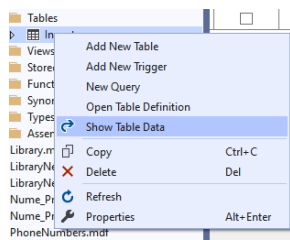
Creem Tabelul Inventory (CarId, Make, Color). Campul CarId va avea tipul int iar celelalte doua nvarchar(50). Ne asiguram ca CarId a fost setat ca si cheie primara (right-click randul CarId si selectam Set Primary Key) si ca Identity (modificam in fereastra Properties pentru CarId la proprietatea Identity Specification->IsIdentity=true).



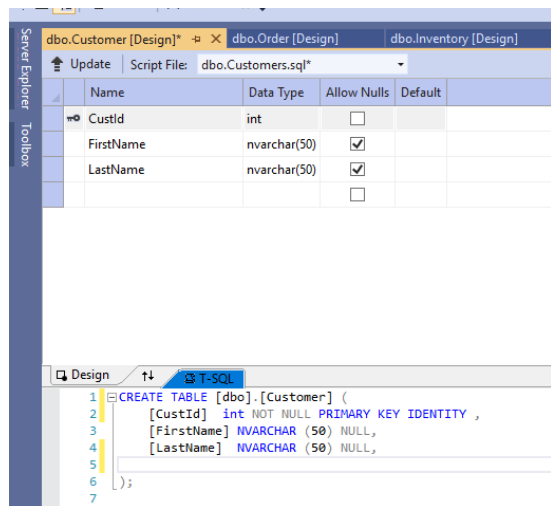
Se apasa apoi optiunea Update iar in fereastra care se deschide se apasa pe butonul Update Database:



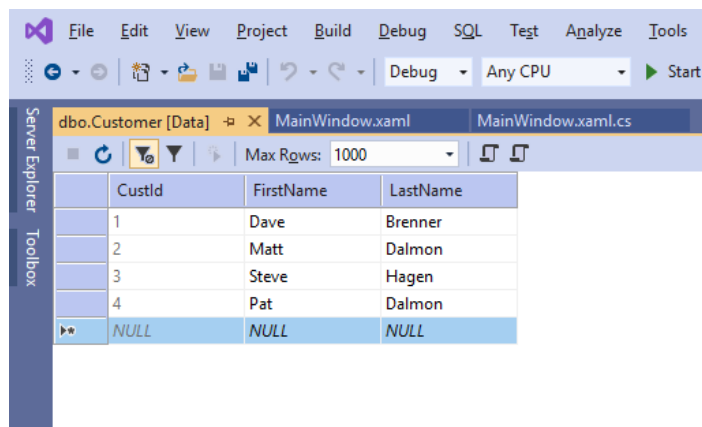
In fereastra **Server Explorer** facem click dreapta pe **Tables** si apoi selectam optiunea **Refresh**. Vom vedea tabelul **Inventory** nou creat si introducem apoi cateva date astfel: Click dreapta pe Inventory-> Show Table Data



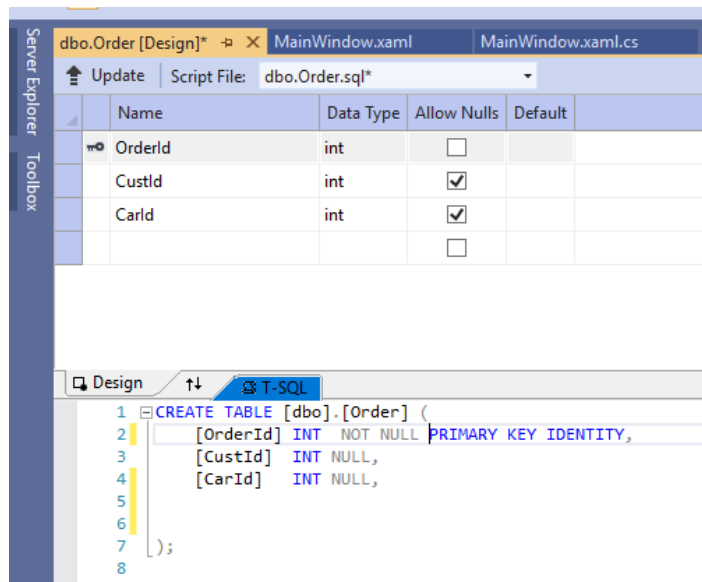
Creem apoi tabelul Customer conform imaginii de mai jos



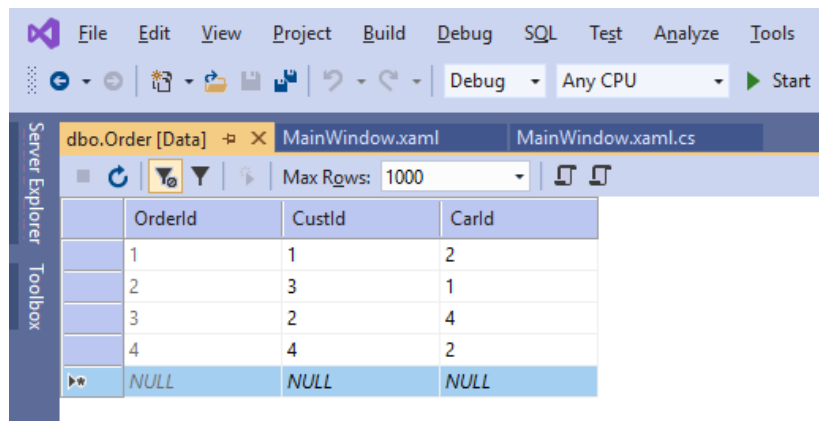
Introducem cateva date in tabelul Customer:



Cream apoi tabelul Order conform imaginii de mai jos:



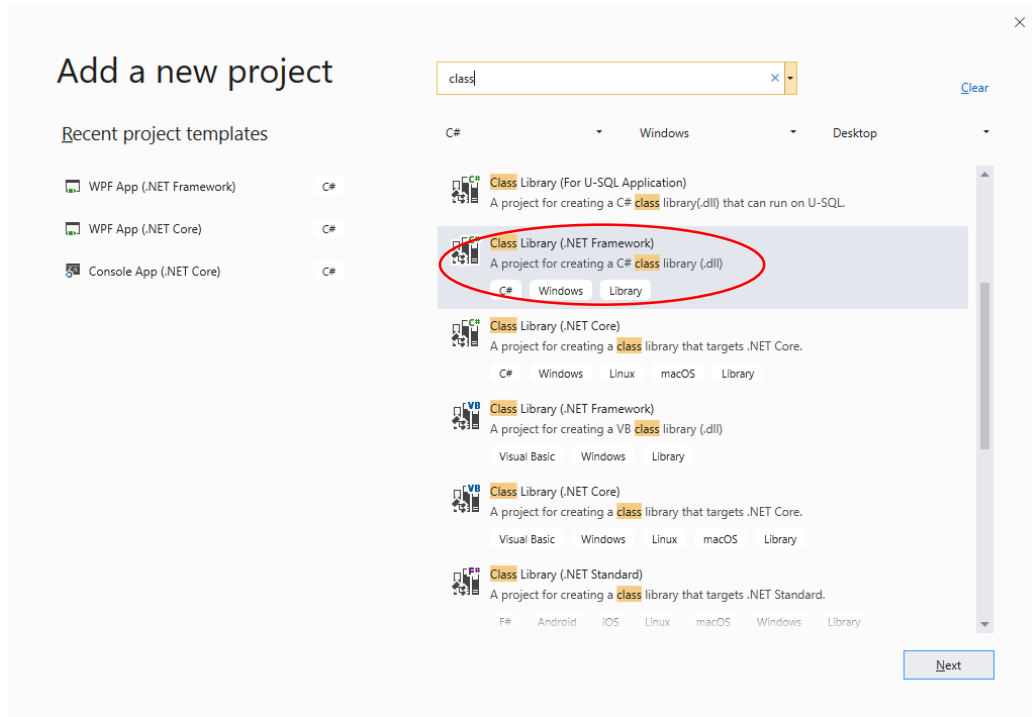
Introducem cateva date,avand grija ca acestea sa fie consistente



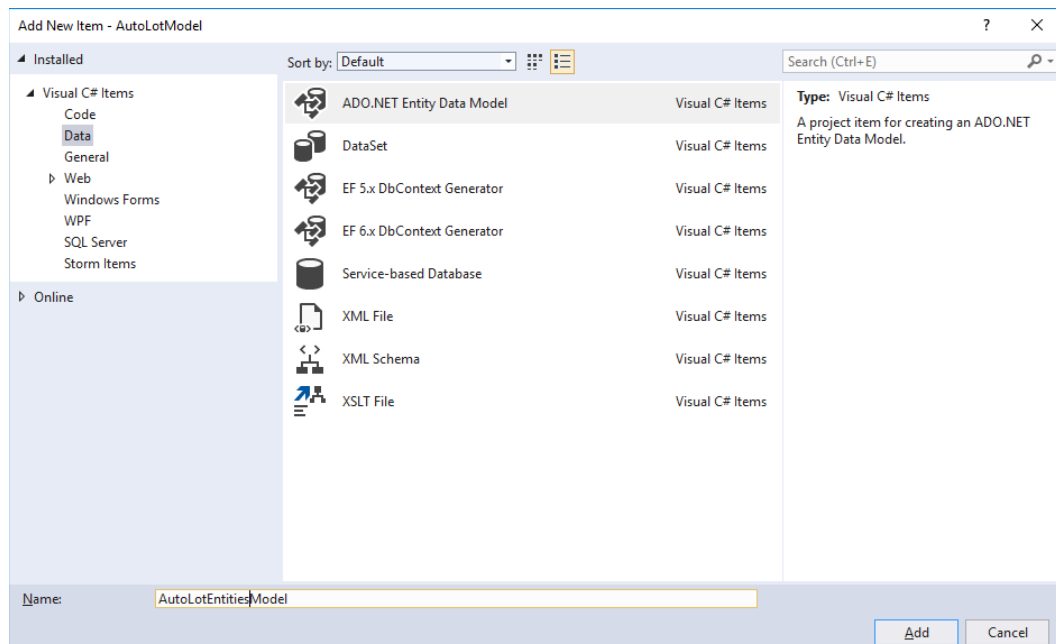
Pentru a asigura consistenta datelor la stergere si acualizate, adaugam următoarele constrangeri la tabelul Order si apoi facem click pe Update Database:

```
CONSTRAINT fk_orders_cust_id  
    FOREIGN KEY (CustId)  
    REFERENCES Customer (CustId)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
CONSTRAINT fk_orders_inv_id  
    FOREIGN KEY (CarId)  
    REFERENCES Inventory (CarId)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE
```


4. La *Solution Nume_Pren_Lab5* se adaugă un *nou proiect* de tipul **Class Library(.Net Framework)** din meniul **File>New Project>Class Library**. Numele proiectului va fi **AutoLotModel**.




5. Din proiectul *AutoLotModel* se va șterge fișierul *Class1.cs*, si se adaugă un **ADO.NET Entity Data Model** din meniul **Project>Add New Item**. Numele modelului va fi **AutoLotEntitiesModel**





Entity Data Model Wizard


 Choose Model Contents

What should the model contain?


EF Designer
from
database


Empty EF
Designer
model


Empty Code
First model


Code First
from
database

Creates a Code First model based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model.


< Previous

Next >

Finish

Cancel

Entity Data Model Wizard

 Choose Your Data Connection

Which data connection should your application use to connect to the database?

AutoLot.mdf

New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

data source=(LocalDB)\MSSQLLocalDB;attachdbfilename="D:\FSEGA\2021-2022\Medii de programare \Laboratoare\5\Nume_Pren_Lab5\Nume_Pren_Lab5\AutoLot.mdf";integrated security=True;MultipleActiveResultSets=True;App=EntityFramework

☒ Save connection settings in App.Config as:

AutoLotEntitiesModel

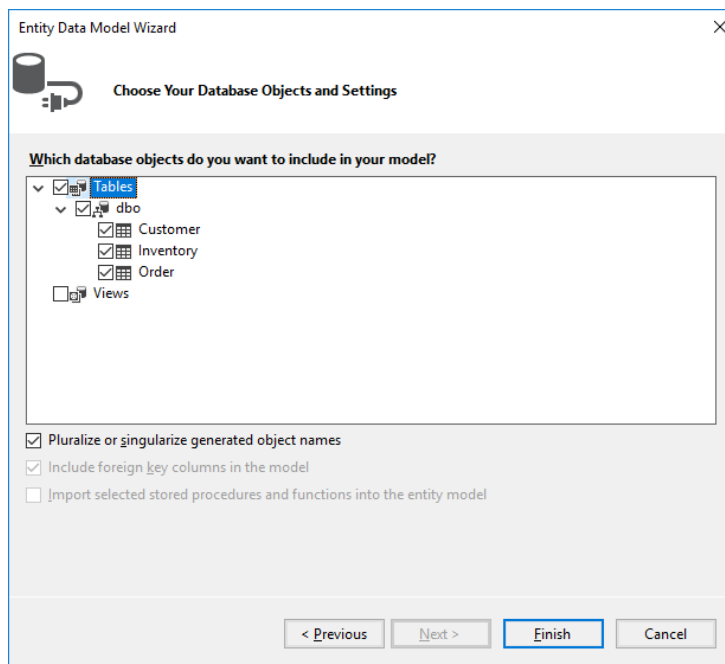
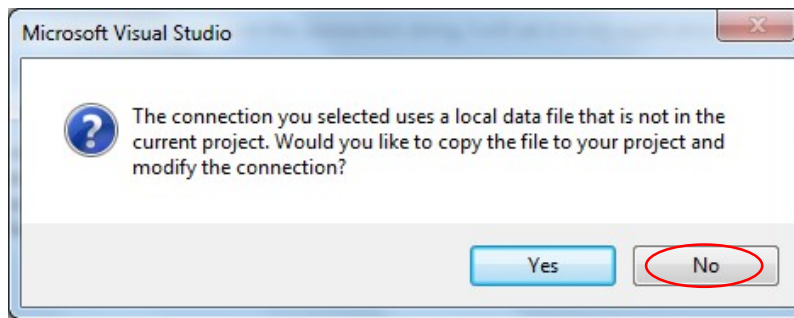
< Previous

Next >

Finish

Cancel

6. Alegem opțiunea **No** pe următorul ecran.

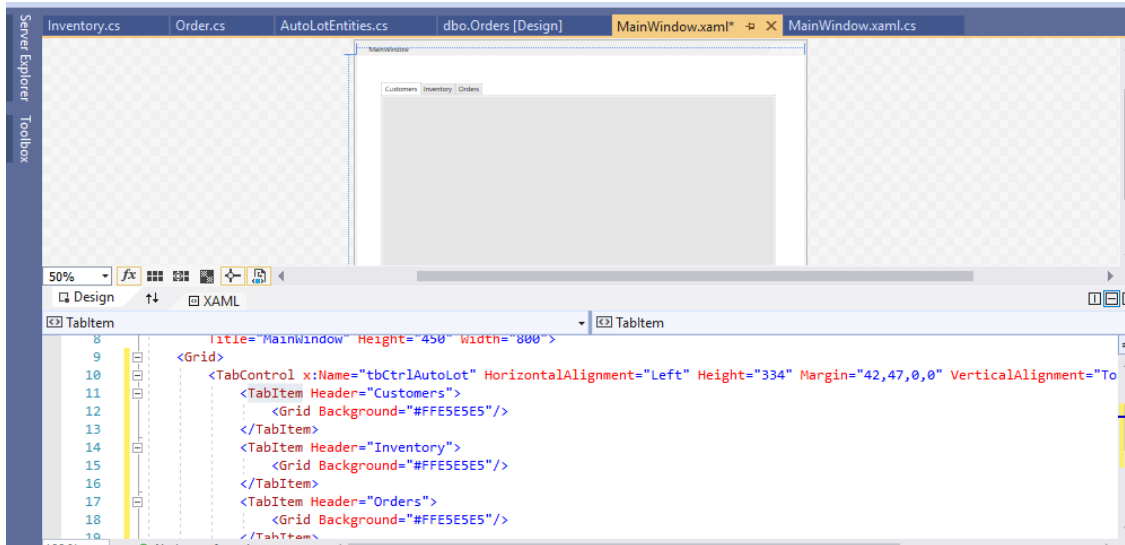


7. Copiem din fișierul App.config al proiectului **AutoLotModel**, EF connectionStrings în fișierul App.config al proiectului **Nume_Pren_Lab5**.

8. În proiectul **Nume_Pren_Lab5** se adaugă o referință spre proiectul **AutoLotModel** (In Solution Explorer - click dreapta pe numele proiectului **Nume_Pren_Lab5**->Add->Reference)

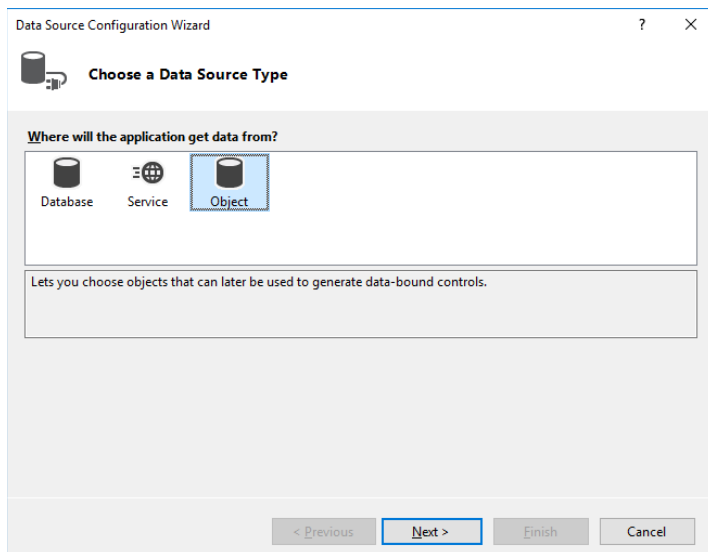
9. În proiectele **Nume_Pren_Lab5** si **AutoLotModel** instalam pachetul EntityFramework. Se lansează de la meniul **Tools -> NuGet Package Manager** prin opțiunea **Manage NuGet Packages for Solution**, de unde alegem să instalăm pachetul **EntityFramework**.

10. Din Toolbox adaugam un TabControl pentru care setam proprietatea Name : „tbCtrlAutoLot”. Mai adaugam un TabItem pe langa cele doua existente si schimbam proprietatea Header la fiecare conform imaginii de mai jos:

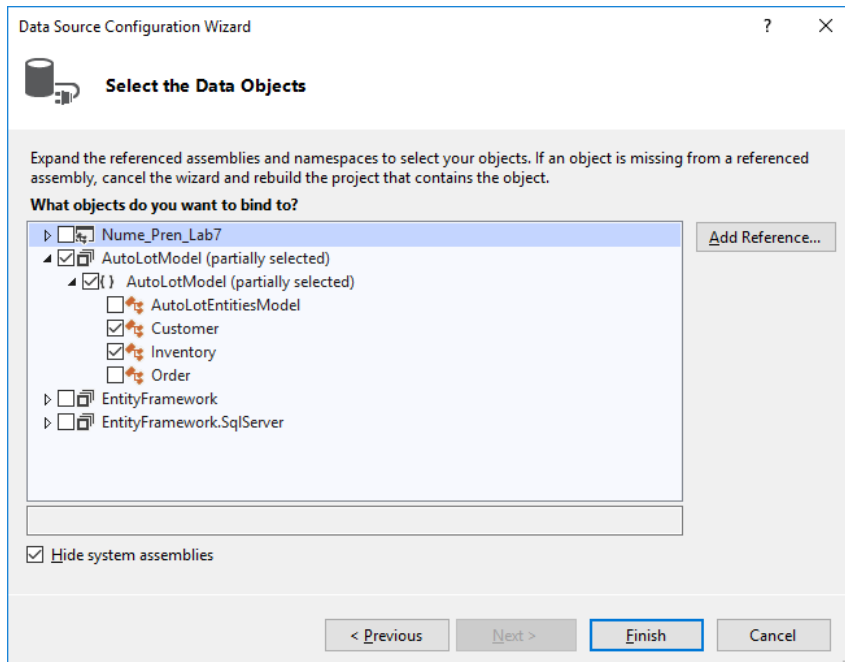


11. Pentru fiecare TabItem vom adauga elementele de interfata si vom realiza operatiile CRUD cu EF. Pentru a realiza databinding-ul este posibil sa scriem cod similar cu laboratorul 5 sau putem sa utilizam facilitatile Visual Studio pentru a realiza acest lucru. Pentru exemplul curent vom utiliza a doua modalitate

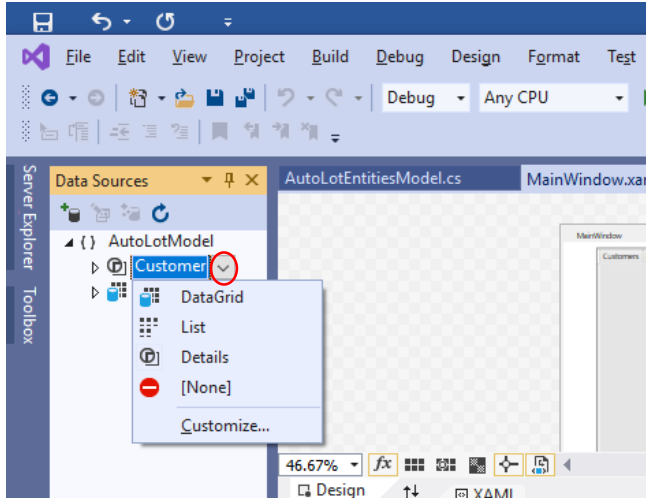
12. Alegem de la meniul **Project** alegem optiunea **Add New DataSource** care ne deschide un wizard. Alegem Object pentru ca facem binding-ul cu clasele din model nu cu baza de date



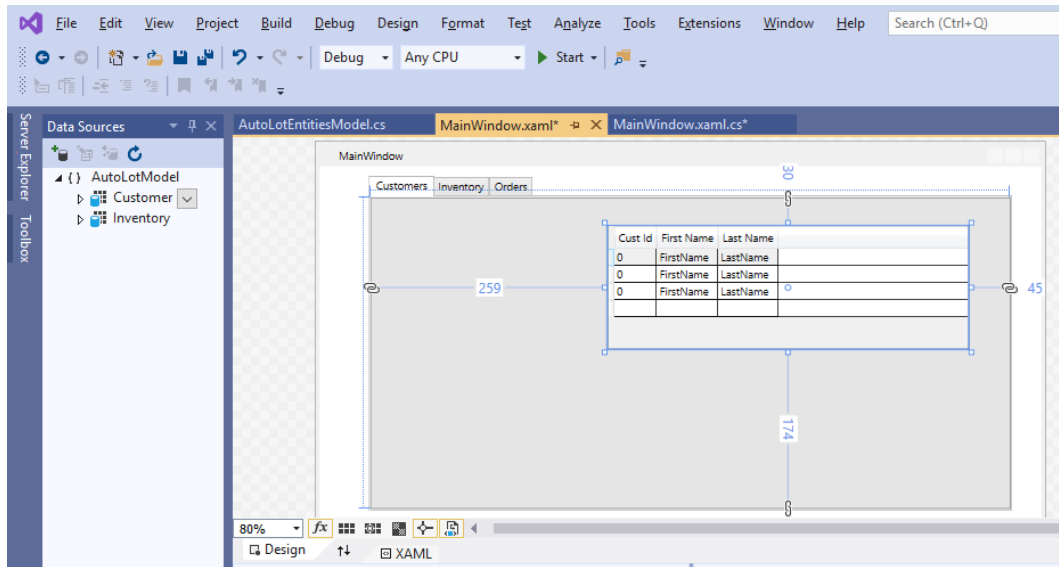
13. Bifam Customers si Inventory (Sursa pentru Orders este automat generata din navigation property Orders de la Customers) apoi dam click pe **Finish**. (In cazul in care AutoLotModel nu este vizibil facem Build la Solutie)



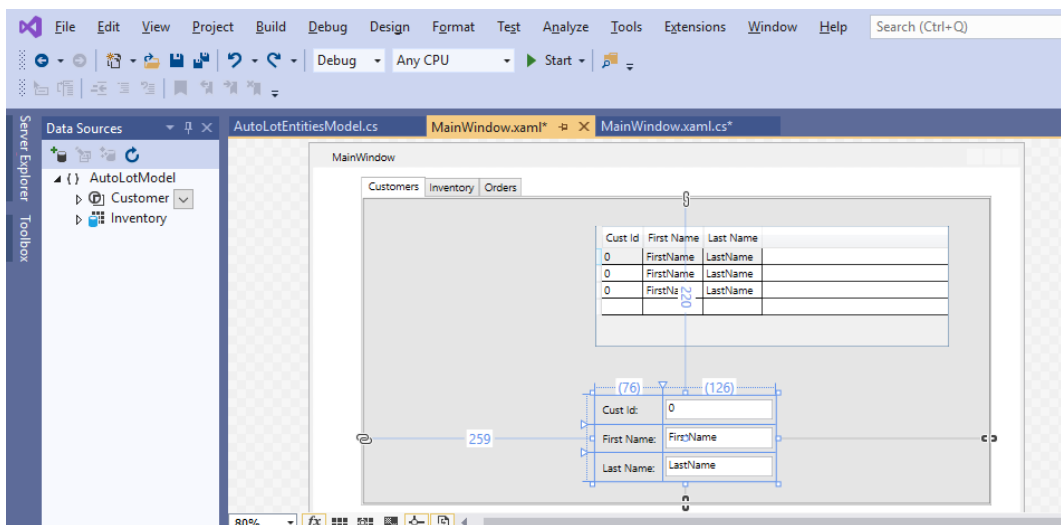
14. Activam fereastra DataSources din meniul **View > Other Windows > Data Sources**
15. Pentru a afisa toate datele despre client vom folosi un datagrid
16. Facem click pe sageata de la Customer, si alegem **DataGrid**



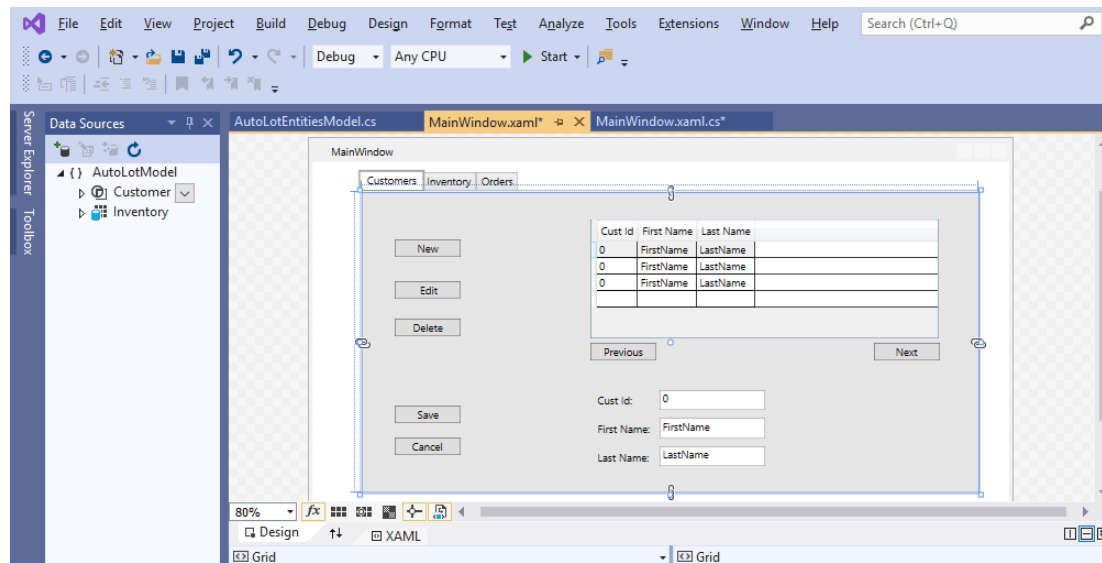
17. Apoi facem drag&drop in MainWindow.xaml



18. Facem click pe sageata de la Customer, si alegem **Details**, iar apoi facem drag&drop in MainWindow.xaml



19. Setam proprietatea `IsEnabled=false` pentru obiectul `custIdTextBox`, deoarece campul `CustId` este de Identity
20. Adaugam butoanele conform imaginii de mai jos si apoi setam proprietatea nume pentru fiecare astfel: `btnNew`, `btnEdit`, `btnDelete`, `btnSave`, `btnCancel`, `btnPrev`, `btnNext`



21. Reluati pasii 16-20 pentru TabItem-ul Inventory si procedati in mod similar cu ceea ca ati facut pentru TabItem-ul Customers