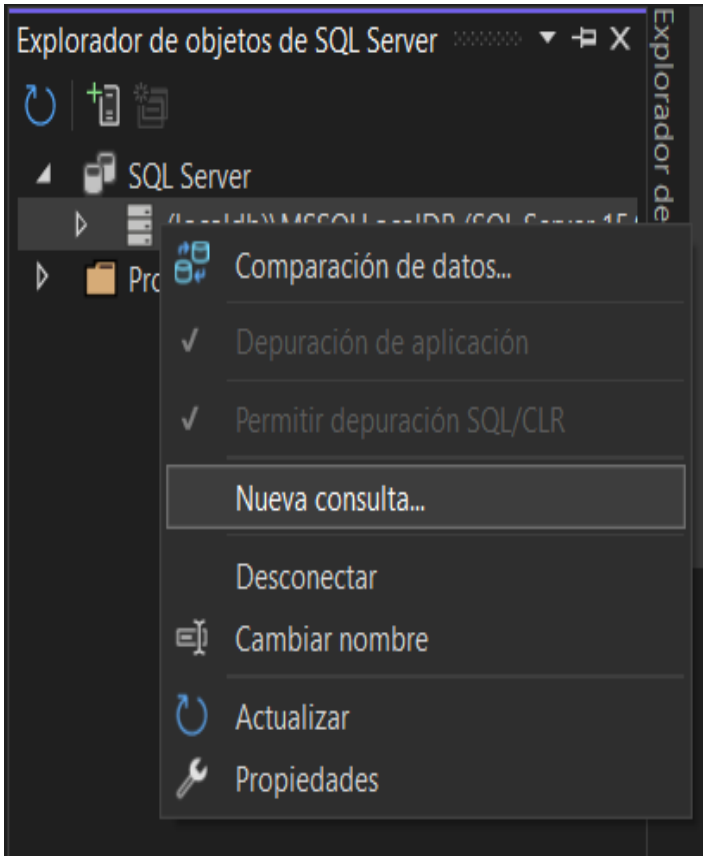
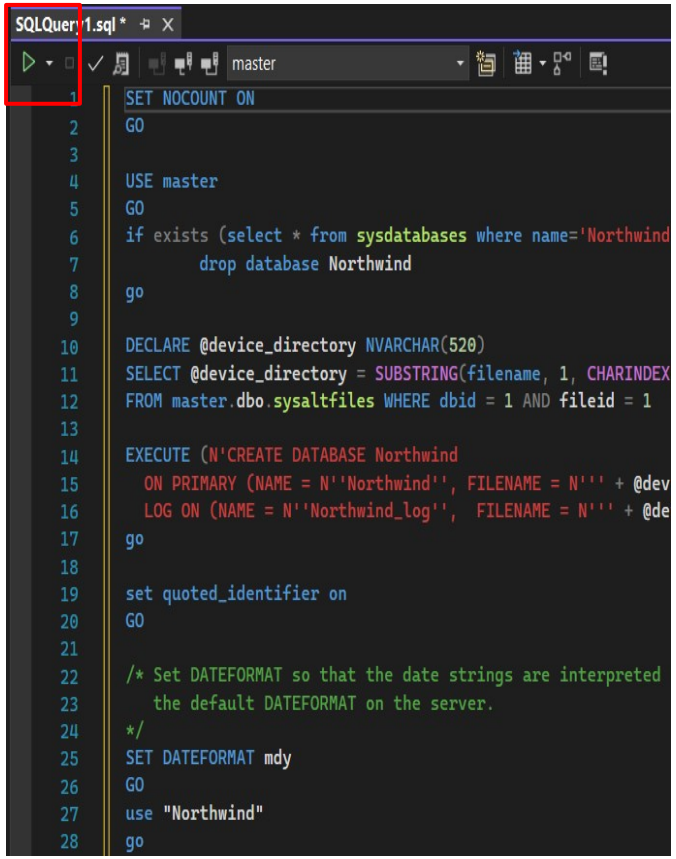
EF Core - DataBase First

(localdb)\MSSQLLocalDB

Importar una BD





EF Core - DataBase First

Generar Modelos de la BD

Para realizar esta acción tendremos que hacer "ingeniería inversa" usando el comando Scaffold-DBContext

Esta acción generará las clases de los modelos y la clase de contexto (derivada de DBContext) basándose en el esquema de la base de datos existente.

NOTA: Es necesario tener instalados los paquetes NuGet:

- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools

EF Core - DataBase First

Generar Modelos de la BD

El comando a usar tiene la siguiente Sintaxis:

```
Scaffold-DbContext [-Connection] [-Provider] [-OutputDir] [-Context] [-Schemas>] [-Tables>]
                   [-DataAnnotations] [-Force] [-Project] [-StartupProject] [<CommonParameters>]
```

Usaremos la “Consola de paquetes NuGet” para escribir el comando, que tendrá el siguiente aspecto:

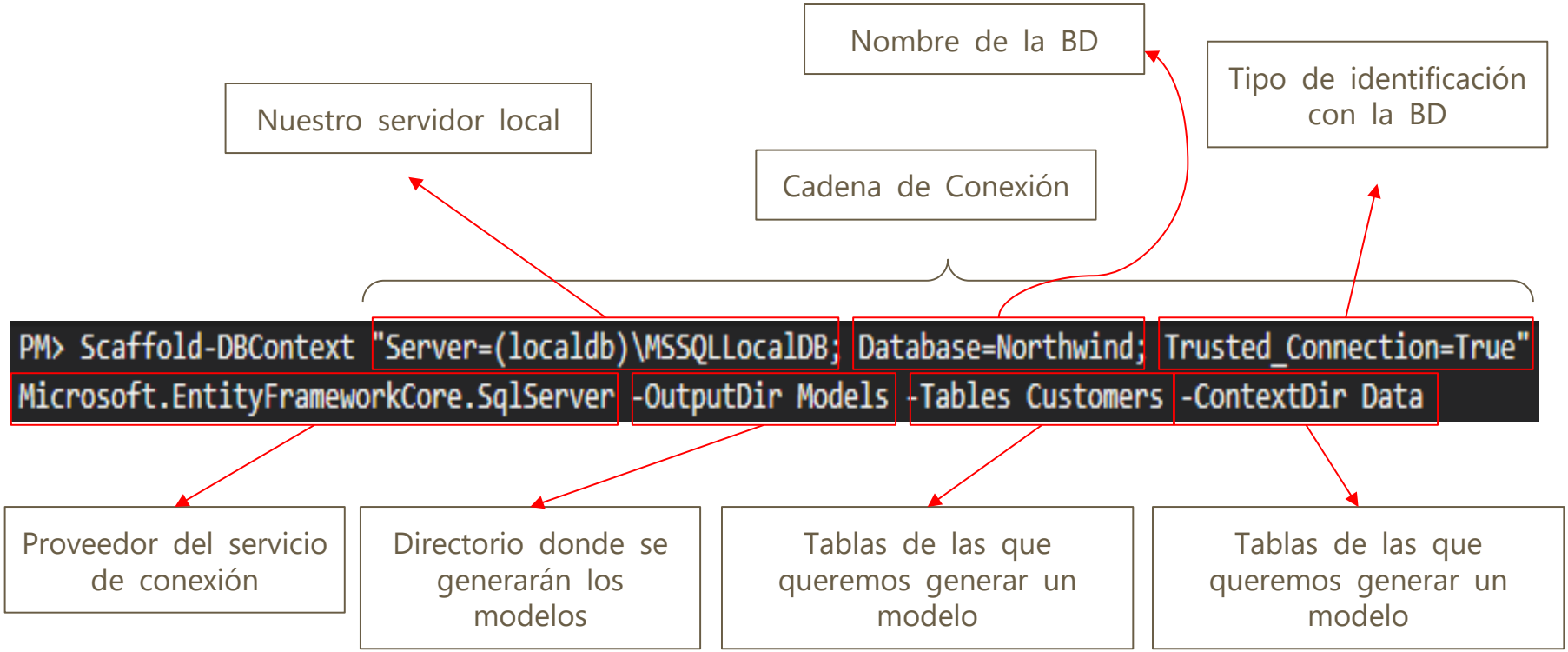
EF Core - DataBase First

Generar Modelos de la BD

Scaffold-DBContext "Server=(localdb)\MSSQLLocalDB; Database=NombreDB; Trusted_Connection=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Tables Tabla1,Tabla2,Tabla3 -ContextDir Data

El comando a usar tiene la siguiente Sintaxis:

-f para sobrescribir las tablas



EF Core - DataBase First

Generar Controlador y Vistas del Modelo

En este punto ya podemos realizar el *scafoldding* en base al modelo, de la misma forma que lo hicimos en el enfoque *Code First* pero esta vez con el modelo generado por ingeniería inversa.

HACER SCAFFOLDING IGUAL QUE EN CODE FIRST, OSEA LOS CONTROLADORES Y VISTAS, NO HACE FALTA EL SEDDER Y MIGRACIONES

NOTA: Esta vez no tendremos que generar una nueva Clase de contexto de datos, ya que la hemos generado con el comando *Scaffold-DbContext.*

Por ello, tendremos que seleccionar la clase generada, y guardada en el directorio Data, en el desplegable correspondiente al hacer el *Scaffold*

EF Core - DataBase First

```
"ConnectionStrings": {
  "NameContext": ""
}
```

Añadir Clase de Contexto al Builder de la Aplicación

Finalmente, para que nuestro proyecto pueda realizar consultas con la base de datos importada, tenemos que añadir nuestra clase de contexto (DBBuilder) como servicio en el builder de nuestra WebApplication

Esto lo realizaremos en el archivo "Program.cs"

```
builder.Services.AddDbContext<ChinookContext>(options =>
{
    options.UseSqlServer
(builder.Configuration.GetConnectionString("DbContext"));
});
```

```
builder.Services.AddDbContext<NorthwindContext>();
```

Si miramos en "NorthwindContext.cs" vemos que la cadena de conexión está definida dentro, junto a un mensaje *Warning*.

Lo recomendable es incluir esa cadena en el archivo appsettings.json