Práctica Arquitectura por capas

Normativa para la práctica:

- ☐ **El nombre de la solución y del archivo comprimido ha de ser: Apellido1_Nombre_VENTASDAM**
- ☐ Documentar cada uno de los diferentes ficheros de nuestro proyecto de la siguiente forma: ///<author> Antonio Rodríguez</author>
- ☐ La nomenclatura de las variables tiene que ser camelCase y el nombre de las mismas debe ser significativo en cuanto a su contenido.
- ☐ La nomenclatura de los atributos xxxxx siendo el nombre camelCase y el nombre de las mismas debe ser significativo en cuanto a su contenido.
- ☐ La nomenclatura de las propiedades PascalCase y nombre igual que el atributo al que expone, pero en mayúscula.
- ☐ Las soluciones deben basarse en P.O.O.

Vamos a realizar durante este curso, una aplicación que gestione una base de datos de empleados y pedidos. La base de datos se llama Northwind, y es utilizada por Microsoft en muchos de sus ejemplos. La aplicación que vamos a realizar deberá basarse en un modelo de datos multicapa con entidades usando Entity Framework. Para esto fíjate en el ejemplo disponible en Aules.

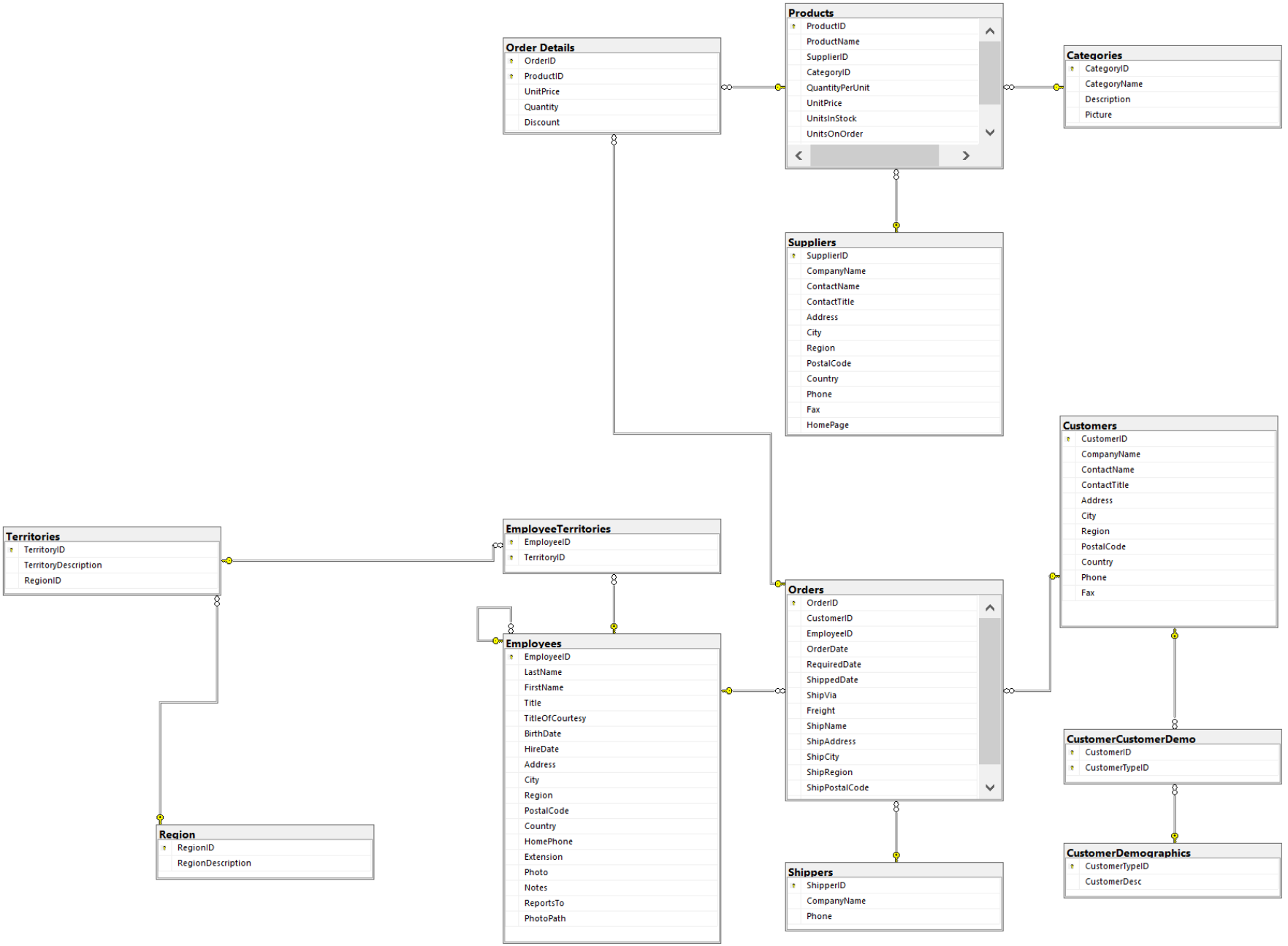
Pasos Previos:

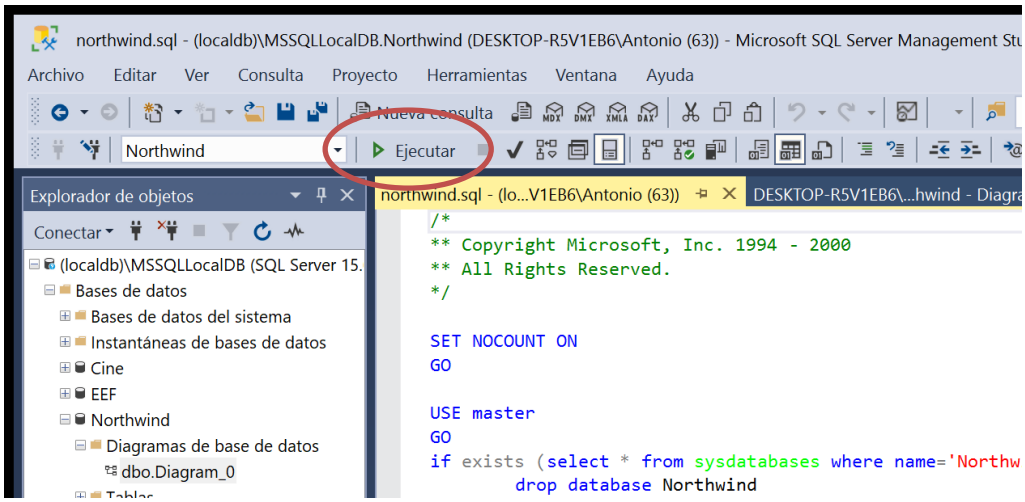
1. **Instalación SQL Server**
   A la hora de realizar la instalación de Visual Studio la opción de desarrollo de Escritorio .NET instala la versión Express 2019 de SQL Server. Al seleccionar el Almacenamiento y procesamiento de datos, y eso nos instala las SQL Server Data Tools, que permite la integración con Editor de Visual Studio.

2. **Instalación Cliente**
   Existen muchos clientes para poder operar con SQL Server. En este curso se deja libre elección de cliente para interactuar con la base de datos. Visual Studio integra una cliente para interactuar con SQL Server y la versión Express que se instala a la vez que Visual Studio. Otra opción recomendable es Management Studio de SQL Server que se puede descargar de forma gratuita desde la siguiente URL

   https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

3. **Creación de la base de datos de Northwind**
   Abrimos el script proporcionado con la base datos y lo ejecutamos pulsando sobre el botón de triángulo verde (Ejecutar) o pulsaremos F5. Al terminar de ejecutar el script tendremos creada la base de datos Northwind con datos de ejemplo.


Products
ProductID
ProductName
SupplierID
CategoryID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
Order Details
OrderID
ProductID
UnitPrice
Quantity
Discount
Categories
CategoryID
CategoryName
Description
Picture
Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
HomePage
Customers
CustomerID
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
Territories
TerritoryID
TerritoryDescription
RegionID
EmployeeTerritories
EmployeeID
TerritoryID
Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
Employees
EmployeeID
LastName
FirstName
Title
TitleOfCourtesy
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhone
Extension
Photo
Notes
ReportsTo
PhotoPath
CustomerCustomerDemo
CustomerID
CustomerTypeID
Region
RegionID
RegionDescription
Shippers
ShipperID
CompanyName
Phone
CustomerDemographics
CustomerTypeID
CustomerDesc



Una vez que ya tenemos todo el entorno configurado para poder realizar la práctica.

Enunciado
La práctica consiste en la creación de una aplicación llamada VENTASDAM utilizando una arquitectura por capas, en este caso usaremos el modelo con entidades usando Entity Framework Core 6.0.

En esta primera entrega tenéis que realizar la capa de datos de y la capa de entidades para el modelo de da datos, en concreto únicamente usaremos las tablas que vamos a utilizar son: Categories, Customers, Employees, Order Details, Orders, Products, Shippers, Suppliers.

Como ya tenemos la base de datos creada hay que utilizar la ingeniería inversa para obtener el contexto y las entidades creadas mediante anotaciones. Para esto deberás utilizar el comando siguiente desde la consola de administración de paquetes:

```
Scaffold-DbContext 'Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Northwind' -Provider Microsoft.EntityFrameworkCore.SqlServer -DataAnnotations -Tables Categories, Customers, Employees, "Order Details", Orders, Products, Shippers, Suppliers
```

Cada una de las entidades es un reflejo directo de la base de datos. Como se muestra en el ejemplo, habrá que desplazar la correspondiente clase a la capa de entidades, y crear en la capa de acceso a datos la correspondiente clase DAO (ADO).

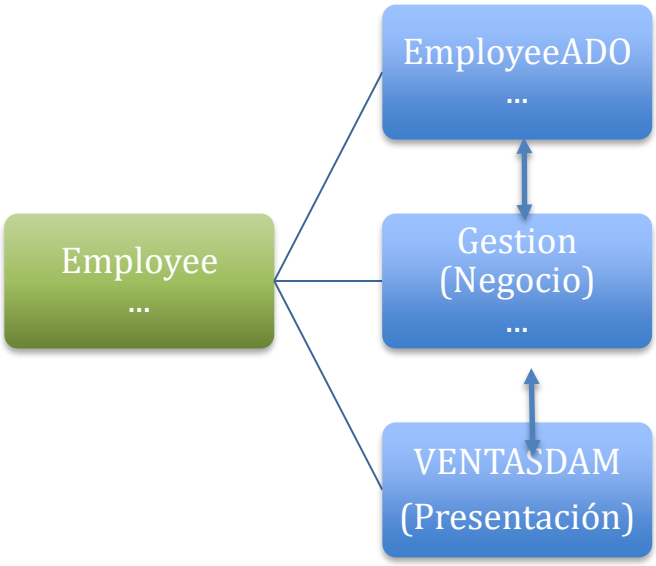
- Todas las clases deberán poseer un constructor únicamente con los parámetros que sean obligatorios.
- Las clases employee y Order deberán tener un constructor de copia.
- Impementar el interfaz Icompareble en la clase Employee ordenando por Firstname y después por Lastname.
- En estas clases deberemos implementar el interfaz IDisposable para poder utilizarlas con la instrucción using.
- Sobreescribe el método ToString, este deberá crear una cadena de texto separando todos y cada uno de los atributos por el carácter #.

Algunos atributos pueden ser nulos. Para que una variable o atributo en C# admita nulos debemos añadir el símbolo "?" después del tipo.
int? Es un tipo int que puede contener el valor null.

Mas información en https://learn.microsoft.com/es-es/dotnet/csharp/language-reference/builtin-types/nullable-value-types

En la capa de Negocio, se implementará la clase Gestión, que accederá a la capa de Entidades y de Datos proporcionando servicios para la capa de presentación.

El esquema de la aplicación con solo una de las entidades quedaría de la siguiente forma:



Realiza la clase **Gestion**
Esta clase contendrá al menos los siguientes atributos y métodos:

- List<Employee> Empleados, una lista de todos los empleados.
- List<Order> Pedidos, una lista de todos los pedidos.
- List<Asignatura> Asignaturas, una lista de las asignaturas.
- Podéis implementar los atributos que consideréis necesarios
- Propiedades para todos y cada uno de los atributos.
- Constructor sin argumentos.
- Constructor con todos los atributos.
- Constructor de copia (ojo al copiar las listas).
- ToString, creará una cadena de texto separando todos y cada uno de los atributos por el carácter #.

Crea los métodos siguientes en la clase **Gestion**
ListarPedidosEmpleado(int EmployeeID): Cargará todos los pedidos realizados por un empleado.
DatosPedido(int OrderID): Cargará todos los datos del pedido, incluyendo sus Order_details. Se deberá mostrar la información más relevante del producto, no solo su ID.
ListarPedidosCliente(int CustomerID): Obtendrá todos los pedidos de un cliente, llamando a la función Datos Pedido.

No se especifica tipo de retorno, para que el alumno lo adecue a sus necesidades.

Sería conveniente también crear algún formulario que nos permita modificar/insertar datos en alguna de las clases. Aunque todo esto se programará de forma más exhaustiva en el resto de entregas del curso, sería MUY RECOMENDABLE probar todas las clases creadas para evitar errores futuros. Es muy importante realizar pruebas y ara probar esta arquitectura por capas

crearemos un formulario en la capa de presentación que nos liste todos los pedidos de forma similar a como se hace en el ejemplo proporcionado en el aula virtual.

Resumiendo lo que hay que hacer en cada una de las capas más o menos en detalle:

Capa de Presentación:

- Mostrar la información de forma ordenada usando criterios de usabilidad.
- Introducir y validar que se cumplan los requisitos del dominio de los datos. (validación de formulario)
- En caso de error debe indicársele al usuario de forma clara.
- Una que vayamos a introducir datos deberá informar del resultado de la misma.
- Enviar y Pedir datos a la capa de negocio.

Capa de Negocio

- Contendrá las funcionalidades en concreto de nuestro programa.
- Debemos revisar que no se puedan producir errores en el uso de los datos, que no puedan ser controlados en la capa de datos.
- Un ejemplo de esto sería que un jugador al que se le sacar más de dos tarjetas amarillas porque sería expulsado. El programa deberá indicarle que ha habido un error a la hora de introducir la tercera. (Ejemplo ilustrativo)

Capa de Datos

- Esta se encargará de la gestión con EEF.
- Debemos validar que se cumplan los requisitos del dominio de los datos (base de datos).
- En caso de error debe informarse de forma clara a la capa de negocio.
- Manejar los datos de la aplicación, en el caso de la aplicación que nos ocupa realizar los SELECT/INSERT/UPDATE/DELETE.

Capa de Entidades

- Tendrá las definiciones de los objetos, así como métodos auxiliares para su funcionamiento, pero el procesamiento de los datos deberá estar en la capa de datos.