PROJET COMPLET

1. Le projet

Problématique :

Monsieur X se lance dans la création d'une entreprise destinée à vendre et à livrer des denrées alimentaires (produits frais) aux restaurateurs locaux. Son objectif est de proposer à ces clients potentiels des produits frais achetés chez les producteurs locaux avec lesquels il est en contact. Afin de vendre ses produits frais dans les meilleures conditions, il souhaiterait mettre en place un système complet pour démarrer son entreprise avec les éléments suivants :

- Une base de données permettant de stocker tous ses produits, ses commandes, ses clients, etc.
- Une application installable sous forme d'un exécutable pour gérer les stocks c'est-à-dire les entrées et sorties de produit. Cette application sera installée sur plusieurs postes dans ses entrepôts et accédera à la base de données.
- Une application web pour permettre à ses futurs clients de passer commande rapidement et facilement et de suivre l'évolution de la commande.
- Un service web pour « communiquer » avec les personnes en charge des livraisons qui sont équipées d'un PDA. Ce service connecté à la base permettra à Monsieur X de suivre le livreur et de déterminer les commandes livrées et à livrer.

Solution envisagée :

Afin de répondre à tous les besoins de Monsieur X en utilisant exclusivement des technologies Microsoft, voici la solution à mettre en place :

- Une base de données SQL Server
- Une bibliothèque de classe pour gérer la couche d'accès aux données (J1).
- Des tests unitaires pour ajouter des cas de test à la couche de données (J1) et assurer la pérennité de la couche de données.
- Une application WPF pour concevoir une application client lourd permettant de gérer les stocks de marchandises (J2).
- Un service WCF pour gérer la synchronisation entre la base de données et une application mobile « virtuelle » (J3).
- Une application Web pour offrir au grand public un portail permettant la commande de produit (J4).

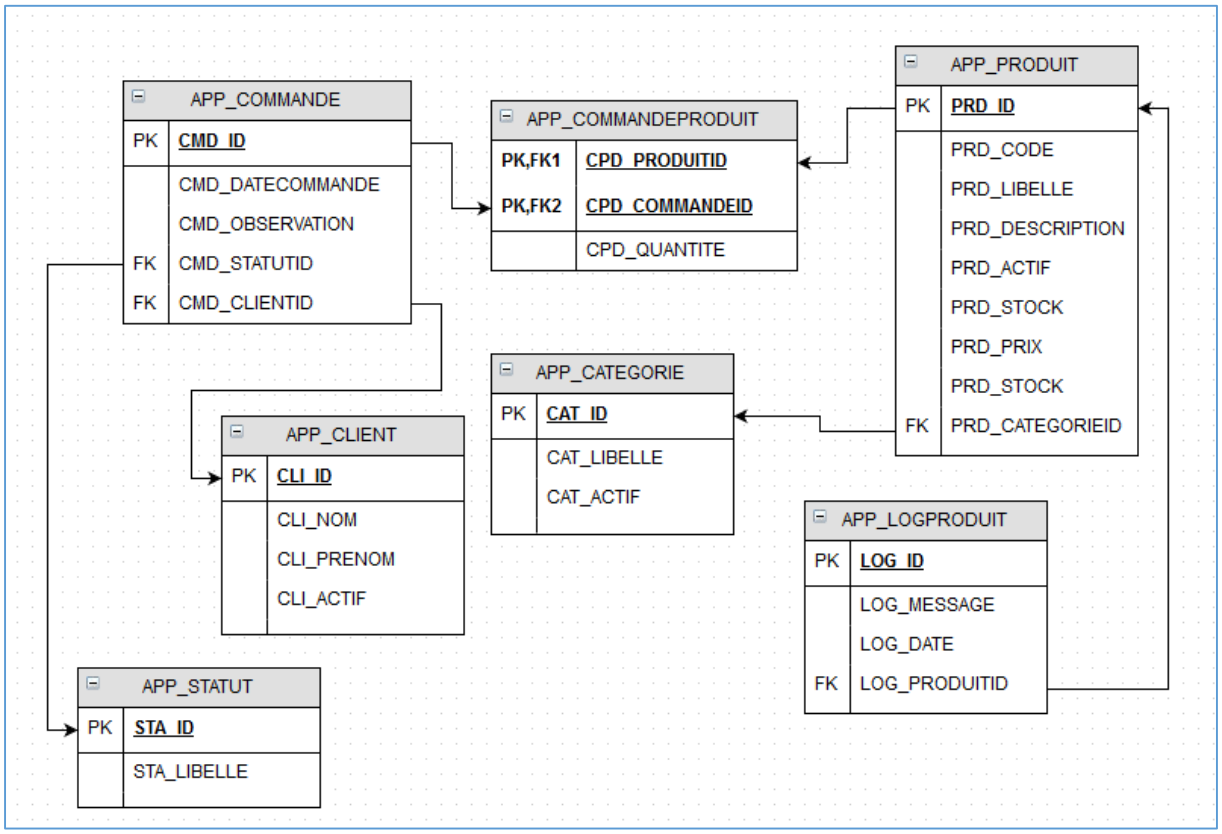
2. Préambule

La base de données :

Afin de faciliter le développement et le déploiement de l'application, la base de données sera volontairement au format **« mdf »**. Nous ne travaillerons pas sur une base de données hébergée sur un serveur SQL. Cette méthode ne change en rien la méthode de développement et surtout évite l'installation et la configuration d'un serveur SQL. Ce fichier au format mdf sera à rendre avec la solution complète à la fin du TP.

Le schéma de la base de données :

Voici la base de données que Monsieur X a imaginée :



NB : Ce schéma pourra être amené à être complété au fur et à mesure du projet mais représente une base de travail indispensable à réaliser pour la suite du projet.

L'architecture de la solution :

A des fins de maintenance, Monsieur X souhaite que l'ensemble du projet soit stocké dans une seule et unique solution Visual Studio. La solution comprendra donc plusieurs « sous-projets ». Le découpage des sous-projets conseillé est le suivant :

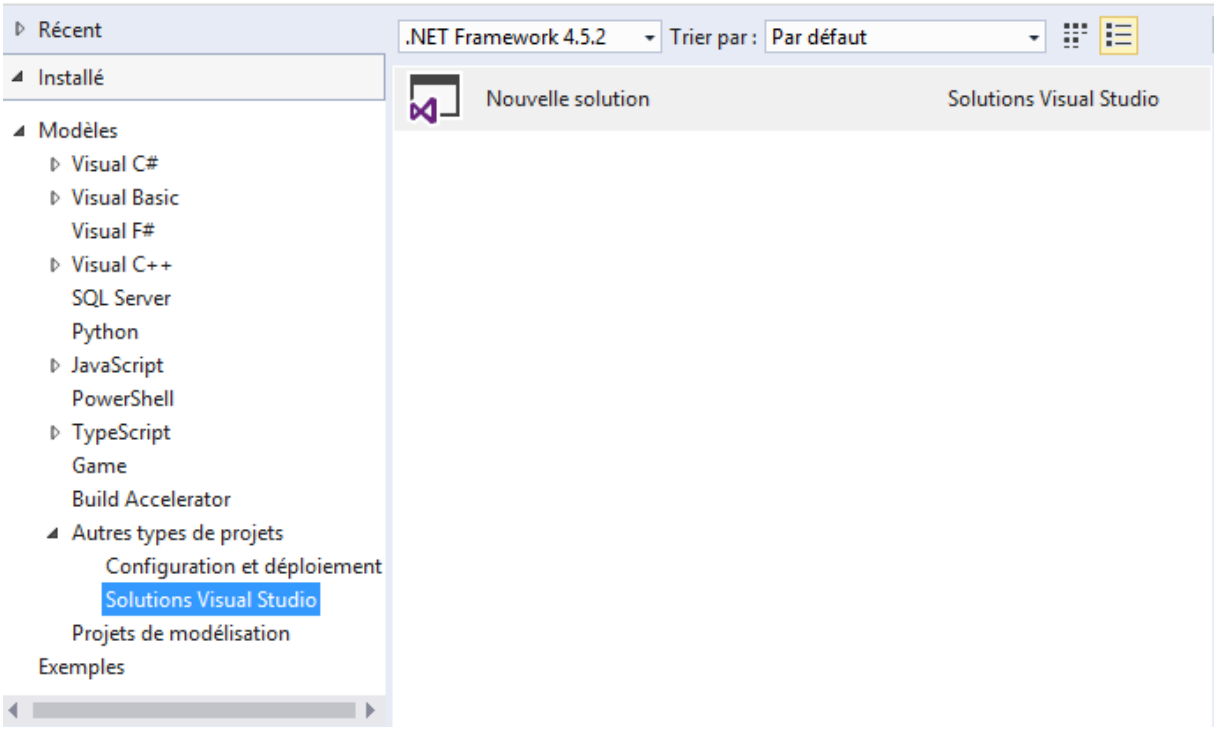
- Un projet de type Bibliothèque pour la couche d'accès aux données
- Un projet de type application WPF pour l'application permettant de gérer les stocks
- Un projet de type application WCF pour créer le service permettant la synchronisation de ses données
- Un projet de type ASP.NET MVC pour créer l'application web permettant de vendre les produits sur internet

- Autant de projet de type Test Unitaire qu'il y aura de test à réaliser

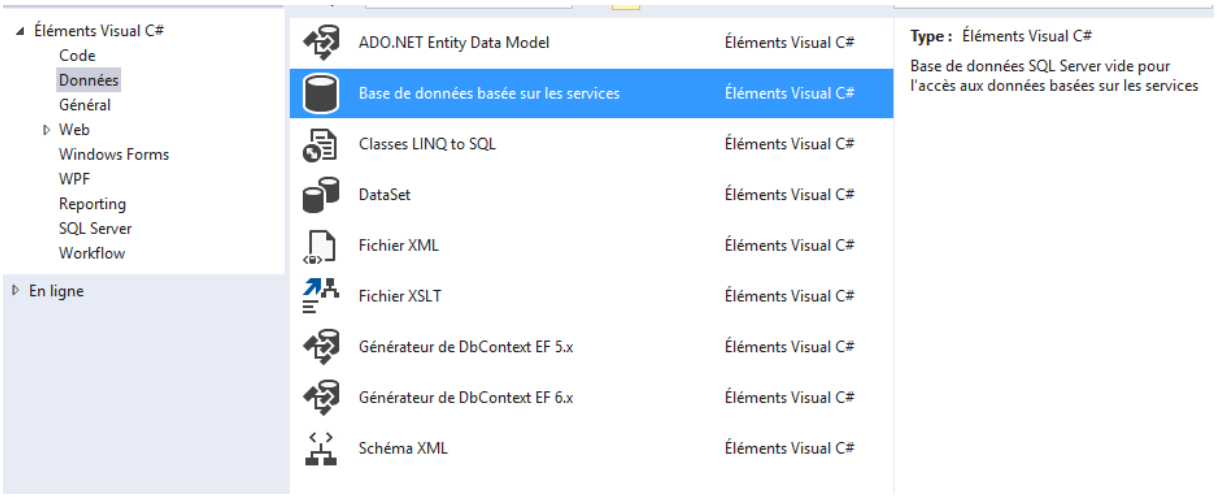
D'autres projets peuvent être ajoutés à cette solution pour compléter certains projets (exemple : un projet de type console pour tester la couche d'accès aux données dans un premier temps). La base de données au format présenté précédemment se placera dans le projet de la couche d'accès.

J0 : Création de la solution

1) Créer un projet de type solution sous Visual Studio. Le nommage de ce projet devra être le suivant : NomProjet.NomBinome1.NomBinome2



2) Ajouter les projets liés au jalon 1 (vous ajouterez le reste des projets au fur et à mesure) c'est-à-dire au minimum une bibliothèque de classe et un projet de test unitaire (il est conseillé d'ajouter un projet console pour commencer)
3) Ajouter la base de données au format MDF dans votre bibliothèque de classe.



NB : Le Framework ciblé est 4.0 ou 4.5

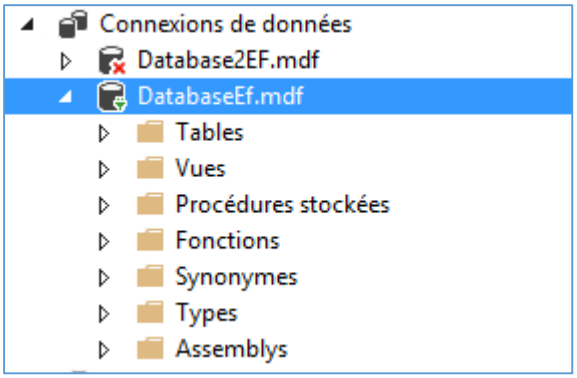
J1 : Création de la couche d'accès aux données, tests unitaires

Objectifs de ce jalon :

- Installer Entity Framework 6 dans les projets nécessaires.
- Créer un répertoire pour stocker vos entités. L'idéal est de séparer la base de données (fichier mdf), les entités et le mapping (pour ceux qui utiliseront Fluent).
- Créer les entités nécessaires conformément au modèle de données fourni par Monsieur X (page 2) pour créer votre modèle.
- Réaliser le mapping entre les entités et la base de données. Ce mapping pourra se faire de deux façons (vous choisirez la méthode qui vous convient le mieux) :
  - En utilisant les Data Annotation.
  - En utilisant l'API Fluent **(conseillé)**.
- Créer un contexte Entity Framework pour avoir accès à votre base de données.
- Tester unitairement chaque entité de votre modèle en réalisant les opérations suivantes :
  - Un ajout.
  - Une modification.
  - Une suppression.
  - Une liste.

Quelques conseils :

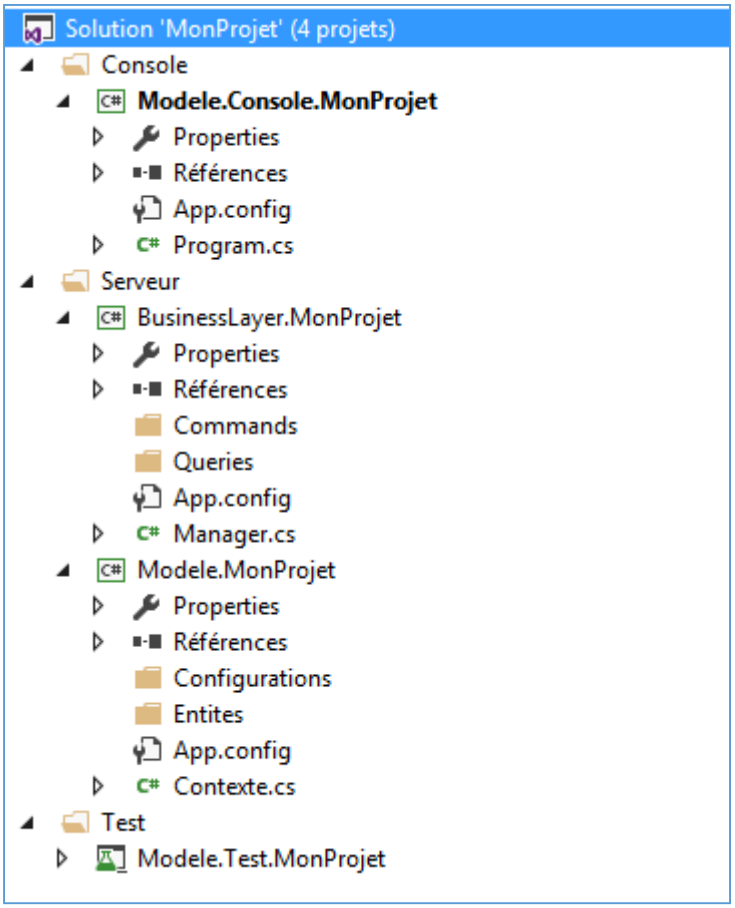
- L'installation d'EF est beaucoup plus facile en passant par le gestionnaire de package Nuget (voir le cours).
- Créer un projet de type Console pour tester au fur et à mesure votre mapping. Vous pouvez ajouter des données manuellement dans votre base de données pour vos premiers tests (via l'explorateur de serveur).



- Suivre le lien fourni en cours pour accéder à la documentation complète d'Entity Framework.
- Pour les tests unitaires, vous pouvez créer une seconde base (pour ne pas polluer la première qui vous servira tout au long de l'application).
- Des exemples utilisant Entity Framework sont disponibles dans le cours.
- Une classe par fichier .cs si possible.
- Pour ceux qui utilisent l'API Fluent, vous pouvez ajouter toutes vos classes de mapping (en une seule ligne) en utilisant les Assembly.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    modelBuilder.HasDefaultSchema("dbo");
    modelBuilder.Configurations.AddFromAssembly(Assembly.GetExecutingAssembly());
}
```

Exemple d'architecture souhaitée :



NB : Veillez à revoir les noms des projets pour que le tout soit cohérent.

Sorties du jalon :

- Un modèle commenté relié à une base de données.
- Le mapping doit permettre d'instancier un contexte Entity Framework valide.
- Une série de test unitaire que l'on peut exécuter à tout moment du projet. Les tests doivent bien évidemment être « success ».