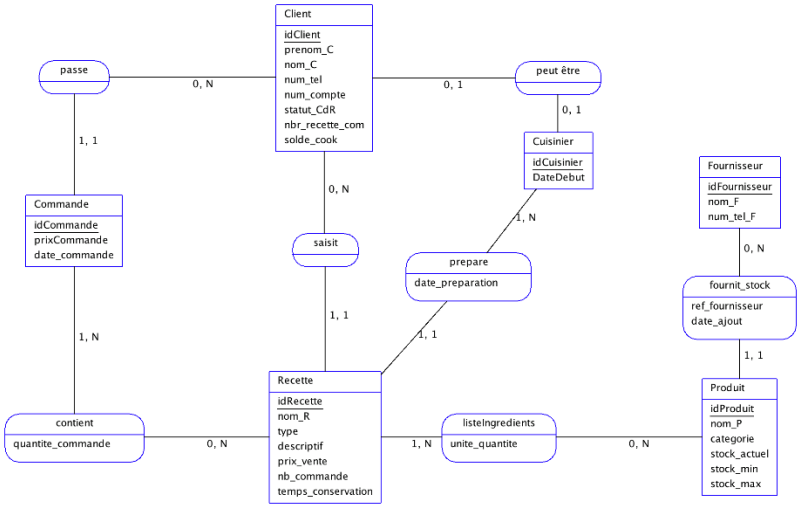
Rapport Final

Afin de répondre aux besoins de la startup Cooking, nous proposons une solution structurée de la manière suivante :

- Une base de données « Cooking » sur SQL.
- Une solution C# comportant toutes les méthodes nécessaires à une utilisation facile, agréable et surtout complète de l'interface.

La base de données sur SQL :

Nous nous sommes basées sur le modèle entités-associations suivant pour créer les tables de notre projet :



Hypothèses faites lors de la réalisation du diagramme E/A :

-Une recette peut être cuisinée par un unique cuisinier

-Chaque produit est fourni par un unique fournisseur

-Chaque commande est unique

-La liste d'ingrédients est unique pour chaque recette

-Il y a l'attribut « quantité » dans l'association « Contient » car on suppose qu'on puisse commander plusieurs fois la même recette dans 1 commande

-L'attribut « num_compte » est rajouté dans l'entité « Client » car cela permet d'accéder à ses coordonnées bancaires (il pourra convertir son argent en Cook)

La base de données créée dans le premier script contient alors 10 tables : Client, Cuisinier, Recette, Produit, Fournisseur, Fournit_Stock, Contient, Commande, ListeIngredients et Prépare, qui **contiennent toutes les clés nécessaires à la validation de la section « Les informations détaillées » du sujet** ainsi que d'autres clés que nous avons jugé utile d'avoir.

Chacune de ces tables est remplie par des données factices dans le deuxième script SQL.

Le code C# :

Avec Visual Studio, nous avons codé en langage C# une solution qui permet à l'utilisateur d'accéder à toutes les fonctionnalités de « notre petite cuisine » depuis la console. Cette solution demande une connexion au serveur SQL afin d'utiliser toutes les données de la base.

Notre solution contient les classes Client, Commande, Recette et ListeIngredients. Elles se présentent toutes de la manière suivante :

- Déclaration des variables-attributs
- Premier constructeur C#
- Deuxième constructeur de récupération des données SQL
- Propriétés
- Méthode d'affichage
- Méthode *public List<Class> X()* qui renvoie la totalité des éléments de la table SQL sous forme de liste

Les méthodes propres à chaque classe ainsi que les méthodes du Program sont explicitées ci-dessous.

Note : dans les sections suivantes le mot « modification » et ses synonymes sous-entendent toujours une modification dans la base de données.

1-Class Client :

Cette classe comporte, en plus de la structure citée précédemment, les méthodes :

- *public string IdentificationClient()* qui demande à l'utilisateur de rentrer son identifiant (en vérifiant son existence) puis le renvoie.
- *public Client BDClient(string IDC)* qui prend en entrée un identifiant client et renvoie le client associé avec toutes ses données.
- *public Client CreationClient()* qui sert à créer un nouveau compte client. Cette méthode demande à l'utilisateur de rentrer toutes ses données afin de l'insérer dans la BD et renvoie donc en sortie ce nouveau client.
- *public void IdentificationCdr(string IDC)* qui prend en entrée un identifiant client et dans un premier temps vérifie son statut CDR. Si le statut est false il y a possibilité de modifier le statut, sinon la méthode ouvre l'accès à la saisie d'une nouvelle recette (méthode SaisieRecette, section 2), à la consultation du solde et à l'affichage des recettes déjà entrées (méthode AffichageListe, section 2).
  Cette méthode valide le point « Pour les créateurs de recettes » page 3 du sujet.

2-Class Recette :

On y trouve les méthodes :

- *public Recette BDRecette(string IDR)* qui prend en entrée un identifiant recette et renvoie la recette associée avec toutes ses données.
- *public void ModifSiCommande(string IDRec, int q)* qui prend en entrée un identifiant recette et une quantité. Elle sert à effectuer la modification du prix de la recette en fonction de son nombre de commandes, à rémunérer le CdR et à incrémenter son nombre de recettes commandées.
- *public Recette SaisieRecette(string idc)* sert à créer une nouvelle recette. Cette méthode demande au client CdR dont l'id est en entrée de communiquer toutes les données afin d'insérer la recette dans la BD et la renvoyer ensuite en sortie.
- *public void AffichageListe(string idc)* qui prend en entrée un identifiant client et affiche toutes ses recettes.

3-Class Commande :

Cette classe possède la méthode *public void DecrementationSolde()* qui permet le paiement d'une commande en décrémentant le solde cook du CdR (si il est suffisant) et en procédant à une conversion en euro si le CdR n'a pas (assez) de cooks ou si le client n'est pas un CdR (un client qui n'est pas CdR n'a pas de cooks par définition).
Nous avons posé arbitrairement 1 cook = 1€.
Cette méthode valide la partie financière de la section « Besoin » du sujet.

4-Class ListeIngredients :

Cette classe contient les méthodes :

- *public void DecrementationStock(string idRec, int q)* qui prend en entrée un identifiant de recette et la quantité commandée de cette recette. Elle recherche dans toutes les listes d'ingrédients les produits et leur quantité nécessaire à la conception de la recette et modifie le stock du produit en fonction du paramètre q.
- *public bool DecrementationStockTest(string idRec, int q)* est la version test de la fonction précédente, elle permet de déterminer si la commande d'une recette est possible car renvoie false si le stock des produits deviendra négatif à la suite de cette commande. Cette méthode ne modifie pas le stock.
- *public ListeIngredients liste(string idr)* prend en entrée l'identifiant d'une recette. Cette méthode permet de rajouter un ingrédient à une recette et elle retourne la nouvelle liste d'ingrédients ainsi créée.

5-Class Program :

a. Région Accueil Client
Permet la création d'un client si souhaité, sinon permet à l'utilisateur de s'identifier afin d'accéder à ses informations et passer des commandes.

b. Région Passage Commande
Elle est constituée de 4 méthodes qui permettent de renvoyer les variables à garder en mémoire pour le passage d'une commande.
- *public static string GenerationID()* renvoie un identifiant de commande généré aléatoirement et non déjà existant dans la BD.
- *public static string ChoixRecette()* affiche les recettes disponibles, demande à l'utilisateur l'identifiant de la recette qu'il souhaite commander et renvoie l'id de cette recette choisie.
- *public static int ChoixQuantite(string IDR)* renvoie la quantité de la recette souhaitée par l'utilisateur en limitant la saisie au stock disponible grace à la méthode DecrementationStockTest.
- *public static int DetPrix(string a, int b)* prend en entrée l'id de la recette souhaitée et sa quantitée afin de renvoyer le prix final de la commande.

Ces deux régions répondent aux attentes de la partie « Pour les clients » pages 2 et 3 du sujet.

c. Région Gestionnaire Cooking
Note : Sauf indication, les méthodes citées ci-dessous sont des public static void.
Cette région contient les méthodes suivantes :
- *CdRSemaine()* indique le CdR dont les recettes ont été le plus commandées cette semaine, *Top5_recettes()* affiche les 5 recettes les plus commandées depuis la création de Cooking.
  public static string CdROr() indique le CdR dont les recettes ont été le plus commandée depuis la création de la plateforme et renvoie son identifiant puis *BestRecettes(string id)* donne les 5 recettes les plus commandées de ce CdR d'or.
  Ces 4 méthodes sont ensuite appelées dans la méthode *TableauBordSemaine()* pour l'affichage.
- *MajQuantite()* qui affiche les produits non commandés depuis plus de 30 jours et demande à l'utilisateur s'il veut procéder à la diminution des stocks min et max.
- *Public static bool Write_XML()* qui permet d'afficher la liste des produits à commander sous un format XML. Ces produits à commander ont un stock actuel inférieur à leur stock minimal et la quantité à commander est égale au stock maximal du produit soustrait à son stock actuel. Si la liste est vide elle renvoie false, sinon true.
- *Read_XML(bool ok)* qui affiche la liste des produits à commander et les fournisseurs respectifs si et seulement si elle prend en entrée un booléen true. On appellera Read_XML(Write_XML).
- Suppression de recette (*SuppressionRecette(string idR)*) ou de cuisinier (*SuppressionCuisinier()*).
- *GestionnaireCooking()* qui permet l'appel et l'affichage des fonctionnalités de gestion.

Comme demandé dans la partie « Pour le gestionnaire de Cooking » page 3 du sujet.

d. Région Démo
On y trouve les méthodes *InfosClients(), InfosCdR(), InfosRecette(), public static List<string> InfosProduits() et InfosCombinees(List<string> prods)* qui correspondent respectivement aux actions demandées dans le **mode démo dans la partie « Validation de votre travail par l'évaluateur » page 3 du sujet**. Ces fonctions sont appelées par la méthode *InfosUtilisateur()*.

e. Méthode Accueil
Cette méthode sert d'accueil de console, elle demande à l'utilisateur s'il veut s'identifier, s'il veut accéder au gestionnaire ou s'il veut passer en mode démo.

6-Tests unitaires :

Nous avons réalisé des tests unitaires pour chaque fonction où cela était faisable. En effet, les méthodes *static void* qui ne renvoient rien ou les méthodes qui récupèrent une valeur suite à une demande à l'utilisateur (*Console.WriteLine("")*) ne peuvent être vérifiées.