Bloc 4.1

Logiciel de gestion Negosud

Jean ROBERT - Théo NORMAND

Table des matières

I - Introduction

Dans le cadre de notre formation, nous devons appréhender les bases de la conception et production logiciel lourd et assimiler des connaissances en C#. Le CESI nous a donc donné un travail collaboratif à présenter le Vendredi 21 Octobre 2022.

1. Contexte

La société Negosud, négociante en vin situé en Gascogne, va ouvrir un entrepôt pour favoriser la vente et la découverte de produit du terroir. Le gérant possédant un ordinateur Windows, il souhaiterait un logiciel permettant la gestion de ses stocks.

2. Organisation

Nous avons commencé par faire un point sur les compétences actuelles de chaque membre du groupe pour trouver un équilibre dans l'apprentissage et la production de la solution. Mais le point le plus important que nous avons fixé, c'est l'assimilation des concepts de la programmation de logiciel lourd et de C#.

2.1- Jean ROBERT

Etant le plus avancé en C#, il a attaqué ASP .NET pour créer l'API. Pour la partie logiciel back, son rôle était de créer la page principale, la partie article, les commandes de produits et les familles de vin.

2.2- Théo NORMAND

Commençant la programmation en C#, dans un but pédagogique, il s'est concentré exclusivement sur l'apprentissage de C# et WPF, par la suite, son rôle est de faire la partie client et fournisseur.

II - Conception

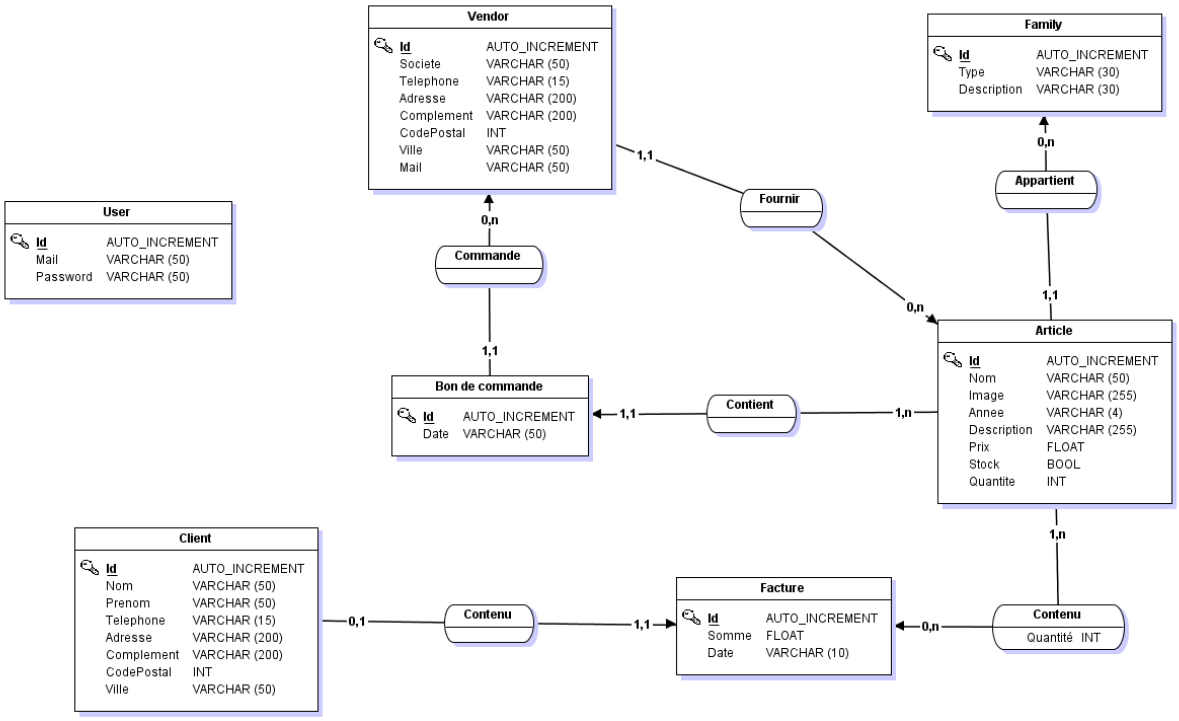
La conception est passé par plusieurs parties.

1. Priorisation des taches

Nous avons commencé par identifier les différentes fonctionnalités du logiciel et de l'API et nous avons commencé à prioriser les tâches à effectuer :

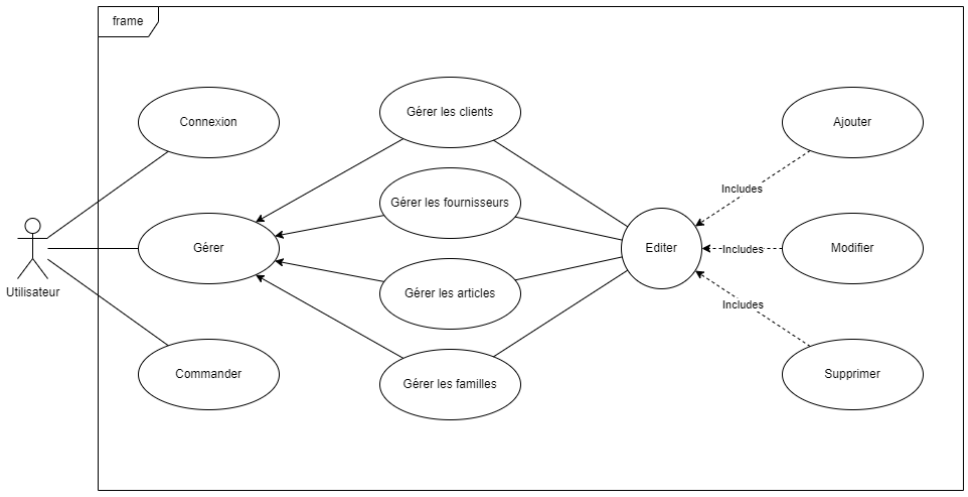
Tâche	Poids de priorité /4
Repérage des différentes fonctionnalités	4
Réalisation du MCD	4
Réalisation du MLD	3
Réalisation des maquettes	4
Réalisation du diagramme de cas d'utilisation	3
Réalisation du diagramme de classe	4
Conception et production de l'API	4
Mise en place de la base de données	4
Déploiement de l'API sur un serveur	4
Réaliser les interfaces graphiques de l'application back	4
création des modèles des entités	4
Réalisation de la logique de la page d'accueil	4
Réalisation de la page article	3
Réalisation de la page famille	3
Réalisation de la page commander	2
Réalisation de la page client	3
Réalisation de la page fournisseur	3
Test logiciel	4
Création d'un installateur pour le back	1
Réalisation d'un cahier mode d'emploi pour l'API et le back	2
Rédaction du rapport de bloc	4

2. Réalisation du MCD



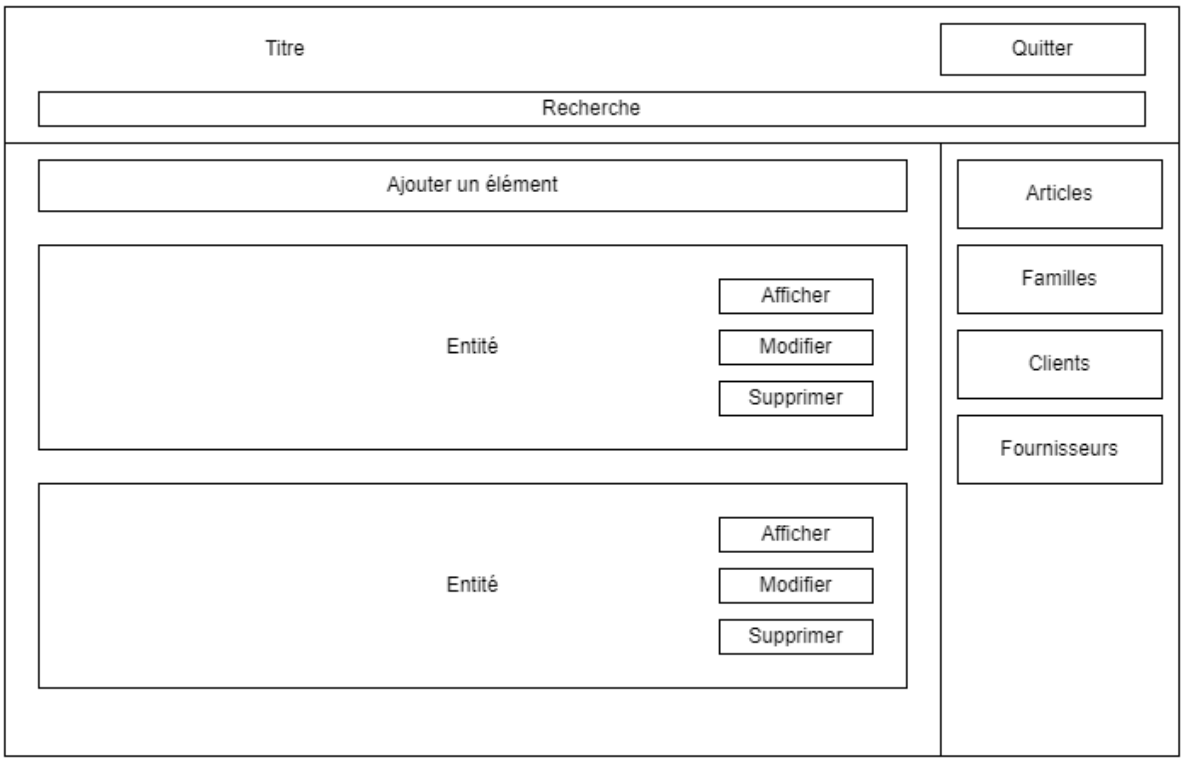
Le MCD représente le logiciel de gestion de gestion qui contient principalement 7 entités, l'utilisateur, les fournisseurs, les clients, les factures, les bons de commande, les articles et les familles de vin.

3. Réalisation du diagramme de cas d'utilisation

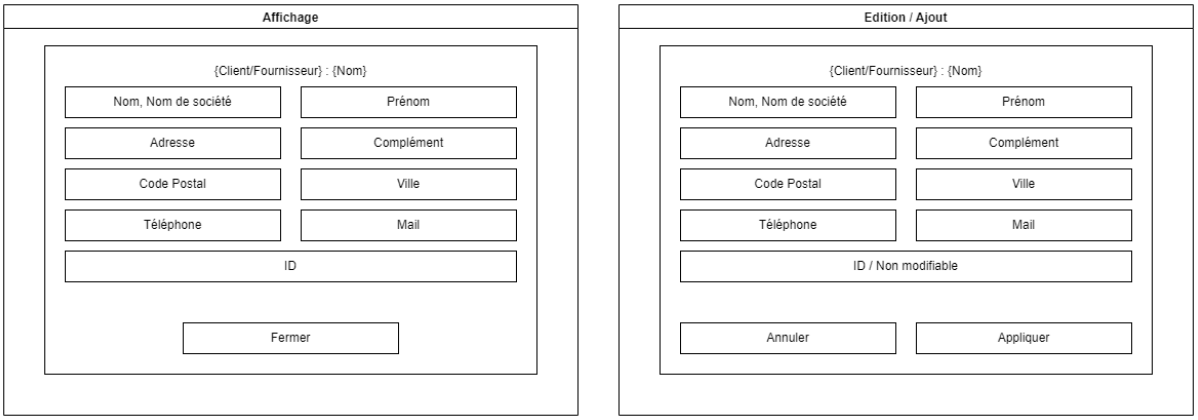


Les fonctionnalités principales que l'utilisateur peut faire sont de se connecter, de commander chez les fournisseurs et de gérer les clients, les fournisseurs, les articles et les familles.

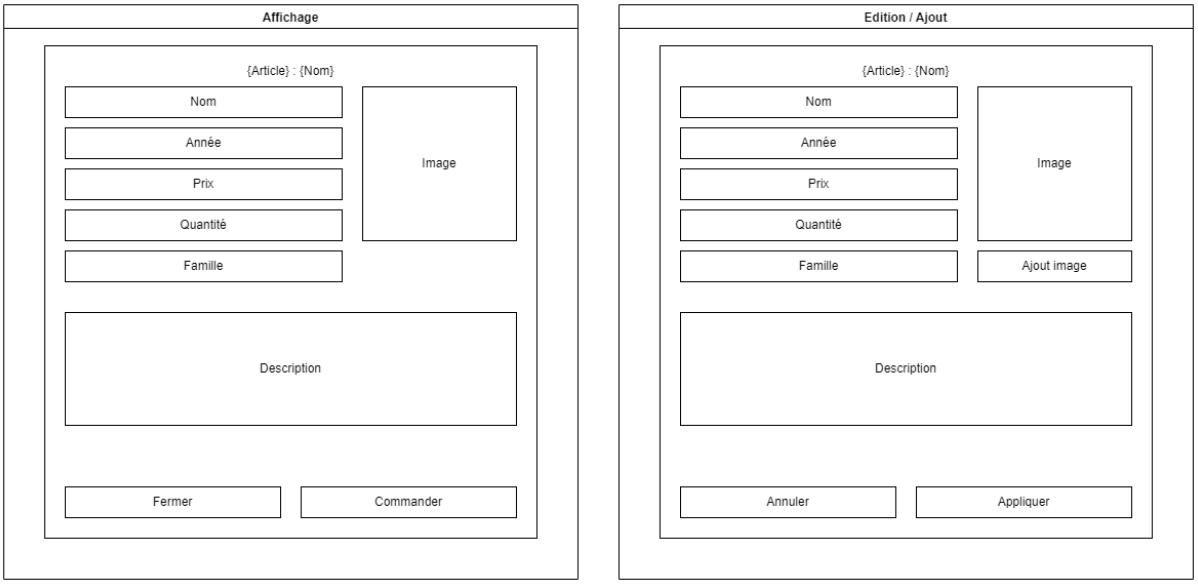
4. Réalisation des maquettes graphiques



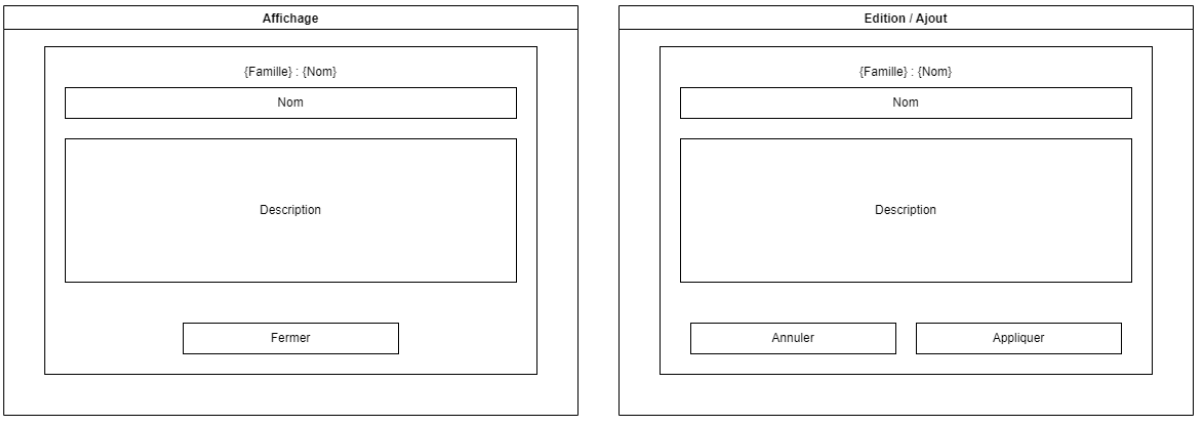
Ceci représente l'interface principale, elle possèdera une partie recherche par nom, au centre, il y aura la génération de blocs qui contiendra les informations et la possibilité de gérer les informations selon la section sélectionnées à droite.



Cette partie représente la fenêtre de gestion des clients et des fournisseurs.

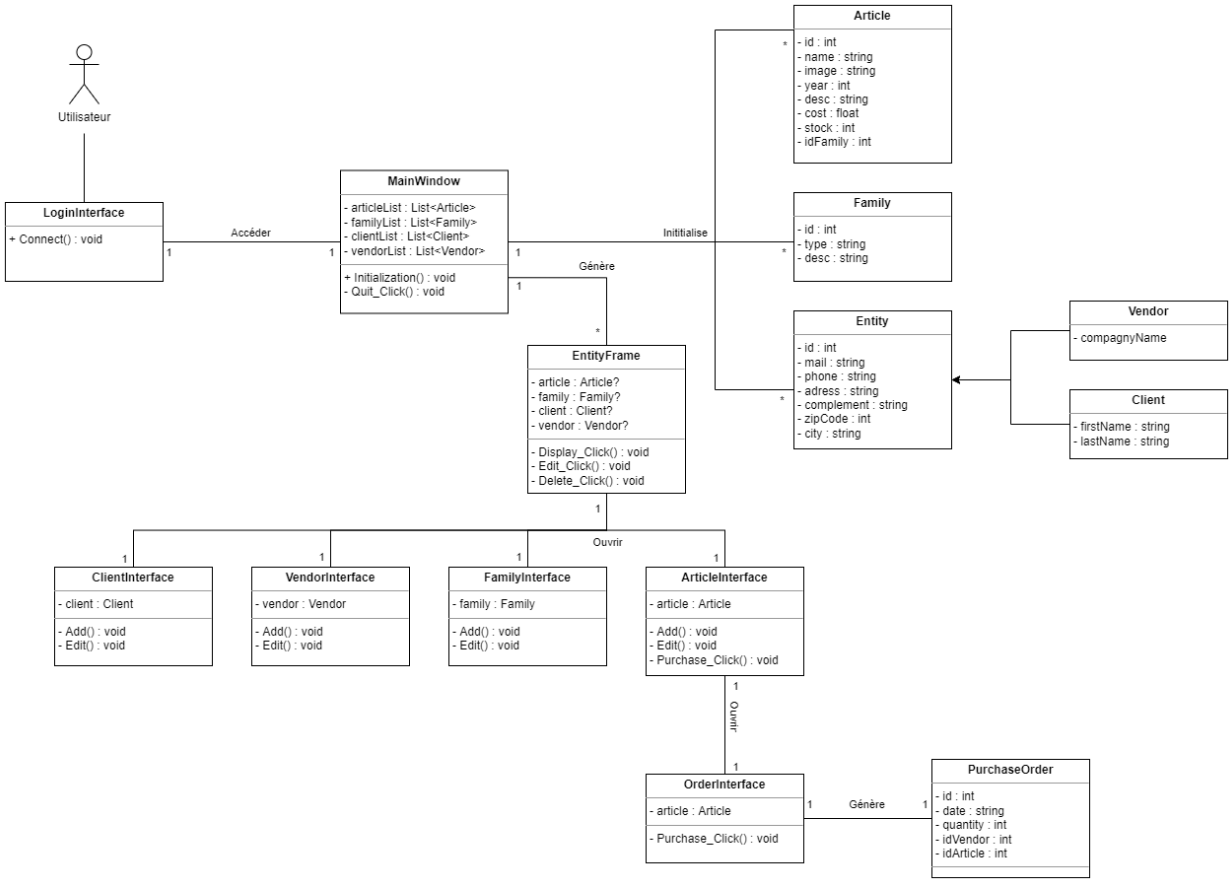


Ici représente la fenêtre de gestion des articles. Le bouton d'ajout d'image est censé apparaitre à l'édition et le bouton commander à valider.



Cette fenêtre représente la gestion des familles de vin qui prendrons un nom et une description.

5. Réalisation du diagramme de classe



L'utilisateur arrive sur l'interface de connexion, une fois connecté, il arrive sur la page principale qui initialise les articles, les familles, les fournisseurs et les clients dans des entity frame, qui peuvent ouvrir les différentes interfaces, selon le contenu de la frame.

6. Choix des technologies à utiliser

Pour la base de données, il a été choisi d'utiliser comme gestionnaire MySQL car nous savons utiliser ce gestionnaire et que notre apprentissage sur ce bloc devait se tourner vers la programmation logicielle et la logique back. Mais si on avait plus de temps, nous aurions utilisé SQL Server qui est plus adapté pour un logiciel ASP .NET et bien plus performant.

Pour le développement, C# était obligatoire pour cette partie et ASP .NET était le plus adapté pour créer l'API demandé. Pour la communication entre l'API et la base de données, nous avons utilisé EntityFramework pour créer un ORM (Object Relation Mapping), qui permet de communiquer avec la base de données non pas en requête SQL mais en objet.

Pour l'interface, on utilisera WPF.

III - Production

Deux programmes sont à développer, une API et un logiciel back. L'API est donc une priorité car le bloc étant divisé en deux, elle est commune aux deux parties. Donc pour assurer le fonctionnement des deux blocs, la priorité est de réaliser l'API en premier pour accéder à la base de données.

1. Production de l'API

Pour créer l'API, nous avons installé sur Visual Studio le paquet ASP .NET et nous avons créé un projet ASP NET core API vierge. Par la suite, nous avons installé les paquets EntityFrameworkCore, EntityFrameworkCore.Tools et MySQL.EntityFrameworkCore pour créer l'ORM avec comme base de données MySQL.

1.1- Créer les models

L'API génèrera les tables de la base de données en Code First via les modèles que l'on aura créés. Les modèles représenteront les entités de la base de données, il s'agira donc des articles, des familles, des bons de commande, des factures, des clients et des fournisseurs. Pour définir les tables, les colonnes et les clés étrangères, on a utilisé les data annotation pour une meilleure visibilité et pour alléger le code.

1.2- Créer et configurer le data context

Le data context va gérer les différents modèles et la connexion au gestionnaire de la base de données. Pour cela on a créé une nouvelle classe DataContext qui hérite de DbContext D'EntityFramework, on a ensuite créé le constructeur qui hérite du constructeur de la classe mère, puis on a créé des propriétés DbSet pour chaque modèle.

Ensuite, on a créé les informations de connexion à la base de données dans le fichier appsettings.

Pour finir, nous avons renseigné dans la classe principale le DataContext dans le builder avec un addContext et le fichier dans lequel sont stockées les informations de connexions.

1.3- Créer les repository et les interfaces

Pour gérer l'évolution de l'API, nous avons décidé de gérer les méthodes par injection d'interface, qui permettra d'éviter des problèmes de dépendances entre les classes selon l'évolution de l'api. Nous avons donc pour commencer créer les repository qui contiendront les méthodes de chaque contrôleur.

Les repository possèdent trois méthodes get, une qui retourne toute une liste, une qui retourne des éléments par mot-clé, une qui retourne par id, une méthode post qui créera des données dans la base, une méthode put qui modifiera des données et une méthode delete qui supprimera des données, sauf user qui n'a qu'un get.

Par la suite, on a créé les interfaces de chaque repository, inséré les méthodes des repository et on a implémenté les interfaces dans les repository correspondants.

1.4- Création des contrôleurs

Les contrôleurs sont les points d'entrées de l'API, nous avons donc créé pour chaque repository un contrôleur qui gèrera les entrées selon le lien.

Nous avons injecté les interfaces correspondant aux contrôleurs et nous avons créé les méthodes get, post, put, delete et nous avons indiqués le type de requête par les annotations HttpGet, HttpPost, HttpPut, HttpDelete.

1.5- Création du web.config

Un problème s'est posé lors des tests de l'API sur le serveur IIS, nous avons les requêtes Put et Delete qui n'étaient pas autorisées. Pour débloquer la situation, nous avons créé un fichier web config qui nous a permis d'autoriser l'utilisation de ces requêtes une fois configurées.

1.6- Déploiement sur un serveur IIS

Pour mettre en marche l'API, nous l'avons publié sur un serveur local IIS, idéal pour des projets ASP.NET. Une fois installé, nous avons réalisé des tests via Postman, un logiciel qui nous a permis de réaliser des tests des différentes requêtes http de l'API.

2. Production du logiciel

Le logiciel permet à l'utilisateur de gérer son stock de vin ainsi que ses clients et ses fournisseurs. Il pourra aussi gérer les bons de commande et les factures.

2.1- Créer les pages et les fenêtres

On a commencé le logiciel par créer les différentes interfaces avec XAML selon les maquettes créées.

2.2- Créer les models

Une fois les pages créées, nous avons créé les modèles des différentes entités que le logiciel va générer, les articles, les familles, les clients, les fournisseurs et les bons de commande.

2.3- Réaliser la logique d'EntityFrame

EntityFrame est une page qui sera contenue dans une frame sur la page principale pour afficher les informations d'une des entités. On y ajoutera les informations de l'entité stockées, ainsi que trois boutons, un pour afficher les informations, un pour éditer l'entité, un pour supprimer l'entité. Il a aussi une fonctionnalité qui permet de recharger la liste dans page principal en cas de suppression de l'entité.

2.4- Réaliser la logique de la page connexion

Une fois la fenêtre prête, nous avons créé une fonction connexion lié au bouton de connexion dans lequel ont créé un objet HttpClient et on y effectue une requête get à l'API au niveau de user qui renverra l'utilisateur et le mot de passe s'il existe dans la base de données, puis le logiciel vérifiera le mot de passe et si tout est correct, il fermera la page de connexion et ouvrira la page principale.

2.5- Réaliser la logique de la page principale

Lors de l'initialisation de la page principale, elle devient la fenêtre principale du projet pour permettre l'interaction des autres fenêtres sur elle.

On s'occupe ensuite de la méthode d'initialisation des différents éléments qui prendra en paramètre une chaine de caractère qui définira le type d'entité à renvoyer.

L'initialisation enverra une requête get à l'API concerné et renverra toutes les entités mises en paramètre. Par défaut, il s'agira des articles. Le bouton ajouter changera de contenu et de tag selon le contenu affiché pour ajouter le contenu selon la liste sélectionnée.

Par la suite, on a créé la partie logique des boutons pour switcher la liste entre les différentes entités.

2.6- Réaliser la logique de la page article

La page article est composée de boîte texte de nom, année, prix, quantité, famille et description ainsi que d'un champ pour une image. Lorsque l'on est sur le mode affichage, toutes les boîtes sont en lecture seule et le bouton valider permet de commander un produit. En mode édition, les boîtes deviennent éditables, un bouton envoyer une image apparaît et le bouton commander devient valider et en mode ajout, la boîte apparaît vierge.

2.7- Réaliser la logique de la page famille

La logique est à peu près la même que la page article, les différences sont que la fenêtre possède une boîte texte de nom et description, ainsi que d'un bouton valider qui disparaît sur le mode affichage.

2.8- Réaliser la logique de la page fournisseur/client

Le principe est le même mais la fenêtre est commune, les boîtes contiennent le nom, prénom (Qui disparait pour le mode fournisseur), adresse, complément, code postal, ville, téléphone et mail, ainsi qu'une boîte id non modifiable dans tous les cas.

2.9- Réaliser la logique de la page commande

Pour la page commande, elle est composée d'une boîte quantité ainsi que de la liste des fournisseurs. une fois validé, cela va générer un bon de commande et le stocker dans la base de données.

2.10- Passer par une phase de test

La phase de test nous a permis de tester la viabilité de notre projet et d'ajuster la structure de notre programme et de l'API. Par la suite, nous avons généré le projet en exécutable.

2.11- Créer un installateur

Pour faciliter l'installation de la solution, nous avons décidé de générer un installateur de notre projet. Mais n'ayant pas encore fait, nous nous sommes renseignés sur les différentes solutions et nous en avons retenu deux, clickonce et visual studio installer project.

IV - Conclusion

Le projet était bien plus lourd que ce qu'on avait fait jusqu'à maintenant, mais nous avons pu adapter selon les compétences de chacun. Il est aussi vrai que c'était une véritable course contre la montre, assimiler autant d'information en si peu de temps n'était pas évident. Le prochain bloc concernera l'application web, avec nos connaissances actuelles, nous pourrons les utiliser avec ASP.NET. Il faudra aussi adapter l'API avec le site.

1. Difficultés rencontrées

La plus grosse difficulté à laquelle on s'est frotté est évidemment le délai ultra compressé d'un mois pour réaliser le projet entrecoupé des séminaires et autres cours qui nous ont freinés dans notre avancé.

Le langage C# est aussi une difficulté par apport au fait que c'est bien plus bas niveau que ce que l'on a vu jusqu'à maintenant et que l'on a dû aussi assimiler énormément de concept auquel nous ne sommes forcément habitués.

2. Fonctionnalités futures

Pour le futur, nous pouvons prévoir d'y insérer dans un premier temps les fonctionnalités que nous n'avions pas le temps de mettre :

- Gestion des factures.
- Ajout des produits dans le stock et non par modification de la fiche article.
- Affichage des familles de vin sur les articles.
- Gestion des bons de commande.
- Ajouter les alertes et les notifications.

Pour d'autres fonctionnalités, nous suggérons :

- Créer un mini logiciel pour gérer les utilisateurs ou l'inclure au logiciel via une combinaison de touches et un compte super-administrateur.
- Ajouter de la gestion de compte client.
- Ajouter la gestion de rédaction d'article sur le futur site.