Travaux Pratiques (Noté)

Informatiser une gestion de stock

Intervenant : Nicolas DAGNAS

Durée du TP : 10h

Objectif du TP

- Ce TP a pour objectif la création d'une application de gestion d'un catalogue de fournitures de bureau par l'intermédiaire d'une base de données de type SQLite à l'aide de Microsoft Visual Studio .

- Il va vous être demandé trois fonctionnalités :
  - Intégrer un fichier « csv » pour alimenter la base de données.
  - Permettre la gestion de la base de données via des écrans clairs et explicites.
  - Permettre l'export de toutes les données dans un fichier « csv » au même format que le fichier « csv » fournis pour la première demande.

- Une série de contraintes seront explicitées au fur et à mesure de l'avancé de votre projet. Ces contraintes sont là pour vous donner un aperçu « soft » de ce que l'on trouve dans le monde professionnel. Le respect de ses contraintes représenteront une part importante de votre note finale, suivez-les rigoureusement.

TP – Les règles de codage

1. Votre code devra être commenté et documenté (Intellisense).
2. Les noms des classes, structures, attributs, méthodes et variables devront suivre le même schéma, **première lettre de chaque mot en majuscule** comme présenté ci-après (Ex: OnLoad, ObjetLeft, etc…). Les variables trop courtes de type « i » ou « Id » sont proscrites.
3. Espacez vos lignes de codes (ni trop, ni trop peu) pour l'éclaircir.

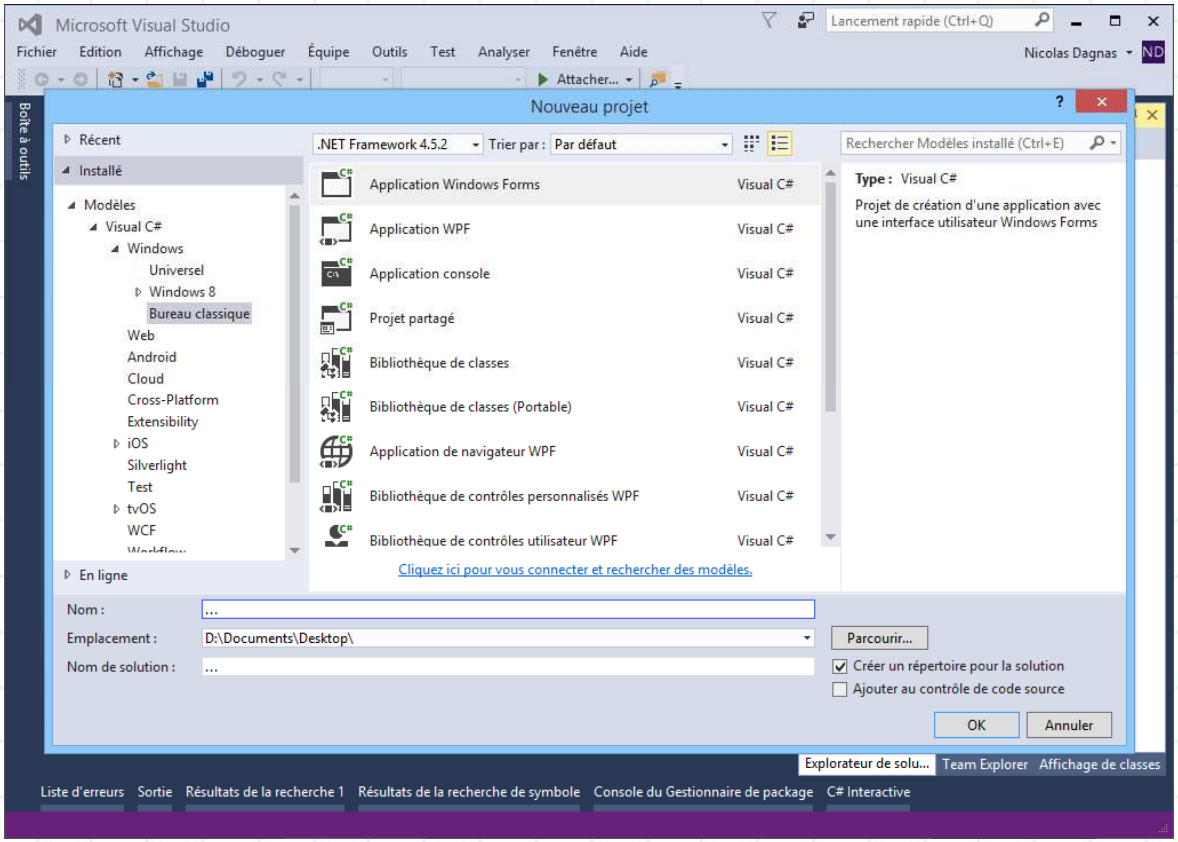
```
/// <summary>
/// Initialise une nouvelle instance de l'objet <b>FrmMain</b>.
/// </summary>
public FrmMain ()
    {
    this.InitializeComponent ();

    string AppPath = "...";

    this.AppConfiguration = ConfigurationManager.OpenExeConfiguration ( AppPath );
    }
/// <summary>
/// Déclenche l'événement Load.
/// </summary>
/// <param name="Args"><b>EventArgs</b> qui contient les données de l'événement.</param>
protected override void OnLoad ( EventArgs Args )
    {
    // ...

    // Récupère la valeur de la propriété d'application 'Left'

    var LeftSetting = this.AppConfiguration.AppSettings.Settings["Left"];

    // Si on a une valeur, on l'applique

    int Left;

    if ( LeftSetting != null && Int32.TryParse ( LeftSetting.Value, out Left ) )

    //...
    }
```

Le rendu du TP

- Le TP sera à rendre au plus tard le vendredi 8 avril 2022.
- Le rendu devra se faire à l'aide d'une archive contenant votre projet sans les dossiers « bin », « obj » et « packages » (avec ces dossiers, l'archive serait inutilement grosse).
- Chaque binôme/trinôme ne fournira qu'un seul rendu.
- N'oubliez pas de préciser les **noms et prénoms** de chaque intervenant dans le nom de l'archive.
- La propreté du code et la simplicité d'utilisation de vos fenêtres représentes la majeure partie de votre note. Donc inutile d'allez trop vite, soignez votre travail car vous pouvez avoir une meilleure note avec un travail incomplet mais propre qu'avec un travail bâclé même si fonctionnel. Inspirez-vous de vos outils habituels.
- Je testerais votre projet avec les fichiers « Csv » et « SQLite » **originaux**, il est donc fortement déconseillé de modifier le fichier SQLite par exemple ; sous peine de perdre des points pour disfonctionnement de certaines fonctions**.**

TP – Nouveau projet

- Créez maintenant un nouveau projet de type « Application Windows Form **(.Net Framework)** » en version « 4.5 ou 4.7 » et que vous nommerez « Hector ».
- Renommez le nom de la fenêtre principale « Form1 » en « FormMain » via un clic droit sur le nom de l'objet dans l'explorateur de solution.



TP – Fenêtre principale

- Ajoutez un menu « Fichier » à votre fenêtre principale.
  - Ajoutez à votre menu les sous-menus : « Actualiser », « Importer » et « Exporter ».
- Ajoutez un objet « StatusStrip » qui se placera en bas de votre fenêtre principale.
- Ajoutez ensuite un objet « SplitContainer » qui devrait remplir le reste de votre fenêtre principale, si besoin configurez-le pour. Configurez aussi son comportement pour que la partie de gauche ne se réduise pas en dessous de 200 pixels, testez son comportement en exécution.
- Ajoutez un objet « TreeView » dans la partie gauche de votre objet « SplitContainer » qu'il devra remplir.
- Ajouter enfin un objet « ListView » dans la partie droite de votre objet « SplitContainer », même chose pour le remplissage. L'objet devra être en mode de vue « détails ».
- Le contenue de la fenêtre doit refléter le contenue de votre base de données à son lancement.

TP – Sous-menu « Importer »

- Pour exploiter le fichier « Hector.SQLite » fournis, vous utiliserez la librairie « System.Data.SQLite » également fournis.
  - Attention : puisque je testerais avec le fichier « SQLite » original, vous ne pouvez pas modifier la structure de la base de données.
  - L'utilisation de la librairie peut nécessiter la modification du fichier App.config de cette façon :
    - <startup useLegacyV2RuntimeActivationPolicy="true">
- Pour visualiser votre base de données, utilisez 'SQLite Browser' disponible ici https://sqlitebrowser.org/dl/ aussi bien pour Pc que pour Mac.
- Ajoutez le fichier « Hector.SQLite » fourni à votre projet en le configurant pour qu'il soit copié dans le répertoire de sortie quand il est modifié (Rappelez-vous du fichier txt dans le TD).
- Dans votre code, le chemin d'accès à votre BDD devra se faire d'une manière intelligente. Pas de chemin relatif et pas de chemin absolue codé en dur. Le chemin absolue doit être évaluer en fonction de l'emplacement de votre .exe puisque le fichier BDD que vous utiliserez sera la copie crée lors de la compilation comme spécifié au point précédent
- Maintenant que c'est fait, codez votre sous-menu « Importer » pour qu'il ouvre une fenêtre modale et de taille fixe (centrée par rapport à la fenêtre principale) contenant :
  - Un bouton permettant de sélectionner un fichier « csv » (le fichier csv fournis ne contient aucune « quantité », cela est volontaire).
  - Un champ contenant le nom du fichier sélectionné.
  - Un bouton permettant de lancer l'intégration en mode écrasement (la Bdd devra donc être vidée avant import).
  - Un bouton permettant de lancer l'intégration en mode ajout (L'import devra mettre à jour les éléments divergents et ajouter les nouveaux).
  - Une barre de progression représentant l'intégration des données. Qui devra s'actualiser **PENDANT** l'intégration.
- **Attention, lisez la suite avant de commencer le codage de l'intégration.**

TP – Sous-menu « Importer »

- Je répète : N'hésitez pas à vous inspirer d'interfaces existantes pour vous fenêtres.
- La fenêtre ne devra pas se figer pendant le traitement.
- Le résultat de l'intégration, nombre d'articles ajoutés, anomalies, devra être affiché via un MessageBox.
- Comme on dit toujours, qui peut le plus, peut le moins. Donc pour l'utilisation des données, vous utiliserez exclusivement les objets « SQLiteConnection », « SQLiteCommand » et « SQLiteDataReader », « SQLiteParameter » est aussi autorisé. **Vous n'utiliserez aucun schéma simplifiant l'utilisation des données, en gros vous ferez vos propres requêtes à la main, « DataTable » est par exemple interdit.**
- La base de données vous étant fournis, à vous de l'analyser pour voir comment intégrer les données. Les champs IDs sont en auto-incrément, pensez à correctement les gérer.
- Une astuce : ne faites pas une intégration « à l'arrache », soyez plus malin ;) Par exemple, la création et l'utilisation de classes ou de structures reflétant les tables de la base de données pourrait être une bonne piste.
- NB : les indexes des tables « Familles », « SousFamilles » et « Marques » sont en auto-incréments, il vous faudra aussi les gérer correctement.
- Une précision qui a toute son importance, cette fonction est là pour **INITIALISER** votre base de données. Elle n'a pas pour but d'être régulièrement utilisée.
- Même si nous utilisons ici un fichier csv, sa lecture devra se faire sans l'utilisation d'une librairie annexe. Privilégiez l'utilisation d'un StreamReader.

TP – Sous-menu « Exporter »

- Vous allez maintenant coder votre sous-menu « Exporter ». C'est bien entendu la même chose que l'intégration mais à l'envers, enfin presque puisque l'on n'a pas besoin d'options car cette fonction exportera toutes nos données.

- Cela devra se faire tout naturellement via une fenêtre modale et de taille fixe (toujours centrée par rapport à la fenêtre principale).

- Attention, le format des données du fichier « csv » généré devra être rigoureusement le même que le format du fichier fournis, c'est-à-dire qu'on doit être capable d'intégrer ce fichier via le sous-menu « Intégrer » sans anomalie.

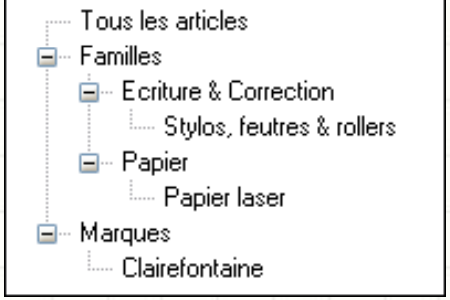
- Vous pouvez ajouter le champ « Quantité » en fin de ligne si vous le souhaitez, mais il ne sera évidement pas géré dans l'import.

TP – Fenêtre principale (suite)

- Dans votre TreeView (Partie gauche) vous ajouterez plusieurs éléments qui permettront de charger votre ListView (Partie droite) selon certains critères.

- Nœuds racines (Nœuds de premier niveau) :
  - Nœud « Tous les articles » : Chargera la liste complète des articles dans le ListView.
  - Nœud « Familles » : Chargera la liste des familles dans le ListView.
  - Nœud « Marques » : Chargera la liste des marques dans le ListView.

EXEMPLE



- Le nœud « Familles » devra disposer de la liste complète des familles » en tant que nœuds enfants, et chaque nœud enfant « famille » devra disposer des « sous-familles ».
  - Un clic sur un nœud « famille » chargera la liste des « sous-familles » dans le ListView.
  - Un clic sur un nœud « sous-famille » chargera la liste des « articles » appartenant à cette sous famille dans le ListView.

- Le nœud « Marques » devra avoir en nœuds enfants la liste complète des marques.
  - Un clic sur un nœud « marque » chargera la liste des « articles » appartenant à cette marque dans le ListView.

- Dans chacun des cas, vous devrez gérer le trie des données à l'écran (exclusivement dans l'objet « ListView », n'utilisez pas de requête) via un clic sur les entêtes de colonne. Cette option devra permettre de trier de manière ascendante et descendante et je le répète : sans recharger les données.

TP – Les listes

- Les entêtes de colonnes de l'objet « ListView » devront être les suivantes en fonction de ce qui est chargé :
  - Si on a chargé une liste « d'Articles » :
    - « Description »
    - « Familles »
    - « Sous-familles »
    - « Marques »
    - « Quantité »
  - Si on a chargé une liste de « Familles », de « Sous-familles » ou de « Marques » :
    - « Description »

TP – Les groupes

- En fonction de ce qui sera chargé dans l'objet « ListView », vous devrez gérer les groupes d'éléments :

- https://docs.microsoft.com/fr-fr/dotnet/framework/winforms/controls/how-to-group-items-in-a-windows-forms-listview-control

- Quand les articles sont chargés, vous devrez grouper les articles selon le trie en cours qui correspond au trie visible sur les entêtes de colonnes (Visualisez cela comme un explorateur de fichier).

  - Si le trie est sur la colonne « description », groupez par la première lettre.
  - Si le trie est sur la colonne « familles », groupez par famille.
  - Si le trie est sur la colonne « sous-familles », groupez par sous-famille.
  - Si le trie est sur la colonne « marque », groupez par marques.

- Quand se sont des « familles », « sous-familles » ou des « marques » qui sont chargés, ne groupez que par la première lettre puisque l'on ne doit avoir que des descriptions de chargées dans le ListView (Slide précédent).

TP – Interactions

- Les interactions suivantes avec l'objet « ListView » sont à coder :
  - Les touches « Entrée » et « Espace » sur un élément sélectionné tout comme un double clic sur cet élément ouvrira la fenêtre de modification de l'élément associé.
  - La touche F5 rechargera la liste des éléments tout comme le sous-menu « Actualiser ».
  - La touche « Supp » demandera la suppression de l'élément sélectionné.
  - Le clic droit devra ouvrir un menu contextuel permettant d'ajouter un élément, de modifier ou de supprimer l'élément sélectionné. Pensez à correctement gérer l'état de ces menus en fonction de ce qui est sélectionné. Par exemple, quand rien n'est sélectionné, les menues de modification et de suppression doivent être grisés.
  - Un clic droit dans le « ListView » en dehors de tout élément devra également afficher le menu contextuel.

TP – Fenêtres Ajout/Modif.

- Les fenêtres d'ajout et de modification d'un élément devront toutes êtres modales, de taille fixe et centrée par rapport à la fenêtre principale.
- Dans le cas de la fenêtre d'ajout/modification d'un article et uniquement dans celle là, vous utiliserez des objets « ComboBox » pour sélectionner les « marques », « familles » et/ou « sous-famille » en mode « DropDownList ».
- Vous contrôlerez aussi l'existence, le format, ou la taille de ce qui est sélectionné/saisie pour éviter toute erreur (on ne peut valider la fenêtre si un « ComboBox » est vide par exemple ou si une référence est trop grande ou une quantité invalide.
- La tabulation devra être correctement configurée pour permettre à l'utilisateur de passer d'un champ à l'autre en partant du haut vers le bas de la fenêtre.
- Toutes vos fenêtres devront respecter les règles basiques du design. Les champs doivent être alignés et la tabulation doit suivre un chemin logique, leurs apparences respectives doivent être confortable à l'œil et simple à utiliser.
- En gros, vous devez assister l'utilisateur finale pour qu'il n'ait pas à trop réfléchir lors de l'utilisation de vos fenêtre, envisagez l'utilisateur comme quelqu'un n'y connaissant strictement rien en informatique et découvrant l'outil.
- La suppression d'une « famille », d'une « sous-famille » ou d'une « marque » ne devra être possible que si aucun « article » ne l'utilise.
- L'ajout, la modification ou la suppression d'une « famille », d'une « sous-famille » ou d'une « marque » devra être répercutée dans l'objet « TreeView » sans tout recharger.

TP – Compléments

- Assurez-vous de n'avoir aucune **anomalie** ni **avertissement** à la compilation de votre projet.

- Enregistrez la position de votre fenêtre principale quand vous la fermez. Pour cela, utilisez « ConfigurationManager » comme vous pouvez le voir dans le *slide* 3, le statut « Maximisé » ne doit pas être oublié. « AppPath » doit être renseigné avec le chemin complet de l'exécutable final et de manière dynamique, utilisez « Assembly… ».

- Puisque maintenant nous avons cette information, replacez votre fenêtre principale lors de son lancement à son dernier emplacement/état tout en évitant les débordements. La fenêtre doit rester dans la zone de l'écran à son ouverture.

- Nous l'avons oublié, mais nous avons un objet « StatusStrip » sur notre fenêtre principale, et bien placez-y le nombre d'articles, de familles, de sous-familles et de marques actuellement dans notre base de données et mettez-le à jour à chaque fois que cela est nécessaire ;)