CÉGEP DE SAINT-JÉRÔME

420-4C4-JR_H22

Évolution des applications

ÉVALUATION SYNTHÈSE
VOLET A

Travail individuel

30% de la note finale

Dates limites de remise :

Remise 1 : 14 mai 2022 à 23h59min

Remise 2 : 25 mai 2022 à 23h59min

Faites les remises ***sur Léa***

Pour chaque date de remise :

Des pénalités de 10% par jour de retard seront imposées.
En cas de retard, Faites la remise par message Teams.

Présentation

Dans ce travail, vous allez continuer le développement d'une application WPF présentant les statistiques de l'évolution des ventes des différents types de véhicules neufs au Canada[1], à travers les provinces, pour les années s'étalant entre 2011 et 2021.

1- Base de données

On vous a fourni les 3 fichiers « ventes.sql », « provinces.sql » et « vehicules.sql ». Chacun de ces fichiers contient les données qui doivent être insérées dans les 3 tables « ventes », « provinces » et « vehicules » à créer dans une nouvelle base de données relationnelle que vous nommez « donneesVentes ».

Assurez-vous de bien définir la clé primaire de chaque table, les clés étrangères nécessaires et de garantir l'intégrité référentielle.

2- Améliorations

L'application WPF déjà fournie contient juste la classe « Vente » alors que la base de données contient plutôt trois tables. Refactorisez le code de la classe « Vente » et ajoutez au besoin d'autres éléments de la POO (classes, attributs, propriétés, etc.) pour correspondre au schéma relationnel de la base de données. Vos classes doivent informer leurs clients de tout changement dans leurs propriétés.

En plus, la classe « Vente » doit avoir une propriété présentant le prix moyen de vente d'un type de véhicule pour une année et une province donnée. La valeur correspondante à cette propriété dans un objet de la classe « Vente » est obtenu en divisant le « Mntx1000 » par le « NbUnites ». Le calcul doit se faire dans une méthode à part.

3- Architecture en 3 couches

Utilisez une architecture en 3 couches pour séparer les fonctions de « présentation », de « la logique métier » et « d'accès aux données » de l'application.

Ces 3 couches doivent être bien séparées. La couche de « présentation » ne devrait communiquer qu'avec la couche de « logique métier ». La couche de « logique métier » ne devrait communiquer qu'avec la couche « d'accès aux données ».

4- Fenêtre principale (MainWindow)

La seule fenêtre (Window) WPF de l'application est « MainWindow ». Tous les autres conteneurs utilisés dans l'application doivent être des UserControls.

La fenêtre « MainWindow » contient **un menu** avec 4 items :

- « *Lister les ventes* » : En cliquant sur cet item, le UserControl « UCListerLesVentes » est affiché dans la grille principale (Grid) de « MainWindow ».
- « *Ventes par province* » : En cliquant sur cet item, le UserControl « UCVentesParProvince » est affiché dans la grille principale de « MainWindow ».

[1] Source des données : Statistique Canada et Wikipédia

- « *Évolution des ventes* » : En cliquant sur cet item, le UserControl « UCEvolutionVentes » est affiché dans la grille principale de « MainWindow ».
- Un item pour arrêter le programme. En cliquant sur cet item, une boite de message permet de valider si l'utilisateur est sûr de vouloir arrêter le programme.

5- Les contrôles utilisateurs (UserControl)

Votre application doit contenir les 4 UserControls suivants :

1- Le UserControl « UCAccueil » : C'est le UserControl affiché par défaut dans la grille principale de « MainWindow ».
« UCAccueil » doit contenir : Le texte « ÉvaluationSynthèse_Volet A », un nom que vous choisissez pour votre application, le texte Réalisé par : Votre *Nom et prénom* et votre *Matricule,* une image d'arrière-plan. En bas de la page : *Source de données : Statistique Canada et Wikipédia*.

2- Le UserControl « UCListerLesVentes » : Affiché dans la grille principale de la fenêtre « MainWindow » lorsqu'on clique sur l'item « Lister les ventes » du menu.
Ajoutez-y un DataGrid affichant les données des ventes.
Utilisez un ComboBox pour permettre à l'utilisateur de choisir de trier l'affichage soit :
   - dans l'ordre décroissant des années, ou
   - dans l'ordre décroissant des « NbUnites », ou
   - dans l'ordre décroissant des prix moyens de vente.

3- Le UserControl « UCVentesParProvince » : Affiché dans la grille principale de la fenêtre « MainWindow » lorsqu'on clique sur l'item « Ventes par province » du menu.

   Ce UserControl permettrait à l'utilisateur d'afficher, dans une ListBox, les noms des provinces (sans doublons) dans sa première colonne et la *somme* des ventes de chaque province dans sa deuxième colonne pour une période d'années (année de début et année de fin) et un type de véhicule choisi par l'utilisateur.

   Pour cela, « UCVentesParProvince » doit contenir (en plus du ListBox) :
   - Un ComboBox contenant une liste sans doublons des années. Ses items doivent être remplis dynamiquement (c'est-à-dire ne doivent pas être insérés manuellement) à partir des données de la table « ventes ». L'utilisateur sélectionne dans ce ComboBox l'année de début.
   - Un deuxième ComboBox pour sélectionner l'année de fin. Il est rempli dynamiquement après la sélection d'une année dans le premier ComboBox. Sa valeur minimale est l'année sélectionnée dans le premier ComboBox pour ne pas permettre à l'utilisateur de sélectionner une année antérieure.
   - Un ComboBox pour sélectionner le type de véhicule. Ses items sont remplis dynamiquement à partir de données de la table « véhicules ».
   - Un bouton pour valider les choix de l'utilisateur et afficher le résultat dans le ListBox.

   Par exemple, pour l'année de début « *2013* », l'année de fin « *2017* » et le type de véhicule « *Voitures* », on aura le résultat ci-dessous dans la ListBox :

Province	Somme en CAD (x1000)
Alberta	8408255
Colombie Britannique	11427216
île-du-Prince-Édouard	419206
Manitoba	2207275
Nouveau-Brunswick	2111438
Nouvelle-Écosse	3049108
Ontario	42230467
Québec	29225724
Saskatchewan	1626496
Terre-Neuve-et-Labrador	1492435

4- Le UserControl « UCEvolutionVentes » : Affiché dans la grille principale de la fenêtre « MainWindow » lorsqu'on clique sur l'item « Évolution des ventes » du menu. Ce UserControl utilise la bibliothèque externe « LiveCharts.Wpf » pour afficher un histogramme présentant l'évolution du prix moyen de vente d'un type de véhicule donné pour une province choisie. L'axe des abscisses doit contenir les années et l'axe des ordonnées les prix moyens de vente.

   En plus du graphique, « UCEvolutionVentes » doit avoir les contrôles suivants :
   - Un ComboBox rempli dynamiquement avec une liste sans doublons des noms de provinces.
   - Un ComboBox rempli dynamiquement avec une liste sans doublons des types de véhicules.

   Le graphique devrait se mettre à jour chaque fois que l'utilisateur change les items sélectionnés dans les ComboBox.

Chacun des UserControls « UCListerLesVentes », « UCVentesParProvince » et « UCEvolutionVentes » doit contenir un contrôle affichant un titre. Vous pouvez aussi y ajouter tous les contrôles que vous jugez nécessaires pour une meilleure présentation et expérience d'utilisation.

La conception des UserControls doit respecter les règles d'ergonomie et doivent être facile à comprendre et à utiliser. Le code-behind doit être clair, bien structuré selon les principes de la programmation orienté objet.

6- Styles et personnalisation

- Définissez au moins 3 styles globaux accessibles par tous les UserControls de l'application. Parmi ces styles, un doit avoir une clé.
- Définissez au moins 2 styles locaux par « UserControl ».
- Utilisez un « ControlTemplate » pour personnaliser l'apparence par défaut d'un contrôle de votre choix.
- Ajoutez une icône à votre application.

7- Implémentation d'un patron de conception

Le fichier « connexion.json », fourni avec les fichiers de travail, contient des identifiants et leurs mots de passe respectifs. On veut améliorer l'application en ajoutant un module d'authentification de l'utilisateur.

Dans le UserControl « UCAccueil », ajoutez les contrôles nécessaires pour la connexion d'un utilisateur. Le menu dans « MainWindow » doit être ***désactivé***. C'est juste après avoir saisi un nom d'utilisateur et un mot de passe correctes que le menu devient ***actif*** à l'utilisateur et une boite de message s'affiche pour attester la réussite de l'authentification. En cas d'échec de connexion, une boite de message affiche un message d'erreur.

Utilisez la bibliothèque externe « Newtonsoft.Json » pour désérialiser les données du fichier « connexion.json » dans des collections d'objets et en respectant les critères de la POO.

Utilisez le patron de conception ***Singleton*** pour être sûr qu'un seul compte utilisateur peut être connecté à la fois. Pour cela, créez une classe « Connexion » qui implémente ce patron de conception et ayant en plus les propriétés « Id » et « Passwd ».

8- Test unitaire

On veut s'assurer du bon fonctionnement de la méthode qui calcule le prix moyen de vente d'un type de véhicule. Créez un nouveau projet dans la même solution pour tester la méthode avec NUnit.

Consignes pour les remises

Pour chacune des deux remises (remise 1 et remise 2) :

- Au début de chaque fichier au format « .cs », « .xaml » ou « .xaml.cs », ajoutez un commentaire contenant votre nom et votre matricule.
- Dans PhpMyAdmin, faites « ***Exporter*** » dans un fichier d'extension « .sql » votre Base de données. Ce fichier doit aussi être remis sur Léa avec votre travail.
- Les interfaces utilisateurs doivent être claires et respecter les standards habituels pour un GUI dans l'environnement Windows. Toutes les fonctionnalités doivent être faciles à comprendre et à utiliser.
- Le code doit être clair, respecte les standards de la POO, n'implémente pas des anti-patrons de conception et respecte les contraintes de programmation d'une application en 3 couches.
- Les parties de code les plus intéressantes doivent être bien commentées.
- La remise doit être faite sur Léa. Remettez un fichier « .zip » contenant le projet en entier et le fichier exporté d'extension « .sql » de la base de données.

  NB : En cas d'incapacité de remise dans Léa à cause de dépassement de la taille maximale permise ou après la date limite de remise, envoyez-moi votre travail dans un message Teams.

Barème

Remise 1	1- Base de données - Création des 3 tables - Définition des clés primaires et clés étrangères, intégrité référentielle - Insertion des données dans les tables	5
	2- Améliorations - Refactorisation correcte du code de la classe « Vente ». - Définition correcte de nouveaux éléments de la POO - Prise en compte par les clients des classes de tous les changements des propriétés.	5
Remises 1 & 2	3- Architecture en 3 couches - Création correcte des 3 couches - Séparation correcte du code des 3 couches - Ajouts corrects des références entre les couches - Installation adéquate de bibliothèque externe	10 x 2
Remise 1	4- Fenêtre « MainWindow » - Création du menu dans la fenêtre « MainWindow » - Définition de la grille principale	5

Remise 1	5- Les contrôles utilisateurs 5-1- UserControl « UCAccueil » : - Contient les contrôles nécessaires - Fonctionnel et s'affiche correctement.	5
	5-2- UserControl « UCListerLesVentes » : - Contient les contrôles nécessaires - Fonctionnel et s'affiche correctement selon les choix de l'utilisateur - DataGrid remplit avec les données et binding réalisé	15
Remise 2	5-3- UserControl « UCVentesParProvince » : - Contient tous les contrôles nécessaires - Fonctionnel et s'affiche correctement. - Remplissage dynamique du ComboxBox de l'année de début - Remplissage dynamique du ComboBox de l'année de fin - Remplissage dynamique du ComboBox du type du véhicule - Bouton de validation fonctionnel - Affichage des résultats dans la ListBox	13
	5-4- UserControl « UCEvolutionVentes » : - Contient les contrôles nécessaires - Fonctionnel et s'affiche correctement - Installation et utilisation de la bibliothèque externe « LiveCharts.Wpf » - Remplissage dynamique des 2 ComboBox - Affichage correcte du graphique - Le graphique se met à jour quand on change la sélection dans un Combox	12
Remises 2	6- Styles et personnalisation - Au moins un style global - Au moins 2 styles par UserControl - Personnalisation d'un contrôle avec un ControlTemplate - Icône pour l'application	5
Remise 2	7- Patron de conception - Contrôles nécessaires dans UCAccueil - Désérialisation correcte du fichier « connexion.json » - Implémentation correcte du patron Singleton - Menu invisible au début et visible après authentification - Utilisation de boites de message	7
	8- Création du projet NUnit - Méthode de test fonctionnelle	3
Remises 1 & 2	- Remise de la base de données créée dans un fichier « .sql » - Interfaces utilisateurs claires et respectent les standards pour un GUI - Code bien commenté	2,5 x 2

NB :

- Attention à la qualité du français, des pénalités peuvent être appliquées jusqu'à concurrence de 10% de la note.