Projet – Bottin de téléphone

Livrable 2 (20%)

2022-08-15

Objectif

Mettre en pratique les notions de base de données à l'intérieur d'une application.
Transposer la théorie des requêtes SQL dans une programmation orienté objet.
Comprendre le mécanisme sous-jacent d'un ORM classique.

Énoncé

À la suite du livrable 1 qui ajoute les fonctionnalités de gestion de votre base de données on vous demande de créer l'affichage de la liste de vos contacts.

Besoins non-fonctionnels

- L'application doit être développé en C#
- L'application est une application console
- Information de base de données obligatoire :
    - Username : root
    - Password: dev
    - Dbname: bottin
- Vous devez utiliser Mysql.Data.MysqlClient
- L'utilisation de Entity Framework ou autre n'est pas permise
    - Vous devez créer une classe par table qui hérite d'une classe Model
- L'importation d'un script ou d'un fichier SQL n'est pas permise
- Votre branche principale doit être : **master**
- Vous devez travailler dans une branche qui se nomme : **livrable2**

Besoins fonctionnels

Liste des besoins fonctionnels de votre application

- Pour donner suite au livrable 1 vous devez valider l'ordre des paramètres de lancement et s'assurer que cela soit fonctionnel peu importe l'ordre que l'utilisateur l'inscrit sur la ligne de commande.
- L'application sans option s'exécute à l'infini jusqu'à ce que l'utilisateur écrit la commande « *quitter* »
- L'application affiche les 10 premiers contacts au démarrage et attend une commande d'utilisateur.
    - Les commandes possibles sont :
        - Suivant : affiche les 10 prochains contacts s'il y en a. Si nous sommes à la dernière page l'option suivant n'est pas disponible.
        - Précèdent: Affiche les 10 contacts précédents. Si nous sommes à la première page l'option précédent n'est pas disponible.
        - Afficher {id} : affiche le détail du contact dont l'id a été spécifié
            - Afficher un message d'erreur si le contact n'existe pas
        - Supprimer {id}: Supprime le contact dont l'id a été spécifié
            - Afficher un message d'erreur si le contact n'existe pas
            - Afficher un message de succès si le contact a bel et bien été supprimé.
        - Quitter

1. Si l'utilisateur entre une option inexistante le programme affiche un message d'information :
    a. Message: Cette commande n'existe pas.
2. Tous autres options doivent faire en sorte que le menu redemande une option et affiche les options disponibles.

La liste de contact doit afficher :

Critères :

- Doit afficher le nombre total de contacts
- Doit afficher le numéro de la page en cours
- Doit afficher le nombre de page au total
- Doit afficher les informations par contact :
    - Id, nom complet, téléphone.
- Doit afficher les options disponibles
- Le programme attend une commande

Exemple d'interface :

Nombre de contacts total : 2 (Ceci doit être le nombre de contacts dans votre base de données)

Id | Nom complet | Téléphone principal |
1 | Karl Provencher | (819) 123-6969
2 | Patrick Bellerive | (819) 345-7878

Page suivante : 2 / nombre de page

Options disponibles: Suivant, Précèdent, Afficher (Id), Supprimer (id)
> commande :

L’affichage d’un contact:

Id : 1
Nom complet: Karl Provencher
Téléphones :

- Principal : (819) 123-6969
- Maison : (819) 123-6968
- Chalet : (819) 123-6967

Adresses :

- Principal : 123 ma rue, Shawinigan
- Chalet : 123 ta rue, St-Mathieu

Notes :

- Note 1
- Note 2
- Note 3

Informations additionnelles:
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Appuyer sur une touche pour revenir en arrière …

Critère de corrections

Les besoins non-fonctionnels sont respecté	1%
L'ordre des paramètres pour la création est fonctionnel en tout temps.	1%
L'option pour la page suivante est présente et fonctionnelle	2%
L'option pour la page précédente est présente et fonctionnelle	2%
L'option pour la suppression est présente et fonctionnelle	2%
L'option pour afficher un contact est présente et fonctionnelle	2%
Les instructions SQL sont dans le programme	1%
Chaque table doit être représenté par une classe modèle qui exécute les fonctions de recherche	1%
Chaque classe qui sert représente une table doit être contenu dans le dossier « ***models*** »	2%
Les fichiers superflus ne sont pas présents sur bitbucket • Seulement le code est les fichiers de configuration (gitignore)	2%
Travail du livrable 2 dans sa propre branche avec un « *pull request* » vers master	1%
L'affichage du nombre de contacts est fonctionnel	1%
Les options disponibles sont les seules qui fonctionnent et les messages adéquats s'affiche	1%
Les informations de la base de données sont les bons	1%

Bonus (note maximale de 100%)

1. Remaniement des classes de migration (1%)
    a. Vous devez avoir une classe de base : « *BaseMigration* »
    b. « *BaseMigration* » doit contenir un attribut « *tableName* » qui représente le nom de la table.
    c. Base migration contient une fonction « *migrate* » (attributs)
    d. Le tableau d'attributs contient une liste d'attribut, chaque attribut contient
        i. Nom de l'attribut
        ii. Son type
        iii. Si c'est une clé primaire
    e. La fonction « migrate » construit la requête adéquate pour créer ladite table.
    f. Chaque classe de migration hérite de « *BaseMigration* »
    g. Les classes de migration doivent seulement affecter l'attribut « *tableName* »

2. Remaniement de la connexion (1%)
    a. Mettre en place un « *FactoryMethod* » pour permettre le changement de moteur de base de données (https://en.wikipedia.org/wiki/Factory_method_pattern#C#)
    b. Pour le livrable on demande seulement de supporter Mysql.
    c. La classe de gestion de votre base de données doit hériter d'une interface. L'interface demande les méthodes :
        i. Open
        ii. Close
        iii. « *GetConnection* »
        iv. La variable connexion est privée.

Remise :

Date de remise : Mercredi 7 septembre 23:59

- Le livrable du code est fait par un « *pull request* » *vers master*
- Une remise régulière sur Bitbucket est exigée tout au long du projet. (Minimum tous les cours.)

Bitbucket :

- Je tiens pour acquis que les connaissances de gestionnaire de version sont bien acquises étant donné qu'un vidéo devrait vous avoir été fourni dans le cours de web.
- Le projet contient seulement les fichiers de votre code (exclure dossiers bin, idea etc....)

Références :

- https://dev.mysql.com/doc/connector-net/en/connector-net-connections-string.html
- https://dev.mysql.com/doc/connector-net/en/connector-net-programming-mysqlcommand.html
- https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-sql-command.html
- https://docs.microsoft.com/en-us/dotnet/api/system.console?view=net-6.0
- https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-console?view=vs-2022
- https://en.wikipedia.org/wiki/Factory_method_pattern#C#