Travail pratique 2

Application console d'interaction avec la base de données

Consignes

- Le travail doit se faire individuellement
- Il est composé de 2 parties:
    - le design de la base de données
    - la conception de l'application

Parti 1: design de la base de données

- Vous devez concevoir une bd répondant à une série de critères. Cette bd sera ensuite utilisée pour la partie 2 du tp. Voir les critères ci-bas.
- Une revue du design sera faite au début du cours du 19 février.
- Notez que si votre design n'est pas complété au début du cours, il ne sera pas revu.
- Vous pourrez corriger les erreurs notées durant la revue avant d'en faire la remise finale sur Léa
- Vous devez remettre le DEA en format **pdf** sur Léa avant le **lundi 19 février 2024 23:59:59**

**note: pour faire votre diagramme vous pouvez utiliser draw.io, visual paradigm community edition, ou tout autre outil de diagramme, ou même mermaid (Word si vous voulez vraiment ;))

Partie 2: conception de l'application

- Vous devez concevoir une application en C# en utilisant Entity Framework pour interroger la bd conçue dans la partie 1 du tp
- Le code doit répondre aux fonctionnalités demandées (voir ci-bas)
- Le travail doit être remis sur **GitLab ou Github.**
    - Sur **Github**, l'utilisateur **BenoitDesrosiers** doit être ajouté en tant que **collaborator**
    - Sur **Gitlab**, l'utilisateur **BenoitDesrosiers** (image de Wall-e) doit être ajouté en tant que **developer.**
- Sur **LÉA**, il faut remettre un fichier **texte** contenant le lien du dépôt git.
    - ce fichier texte doit être remis dans **Léa** avant le vendredi **22 mars 2024 23:59:59**.
- Le dépôt doit contenir le code complet de l'application
- Le dépôt **doit aussi contenir un document contenant le dictionnaire de données** correspondant à la bd finale (pas nécessairement tel que le DEA). Voir les critères ci-bas.
- La date du dernier **COMMIT** dans **GitLab** sera la date de remise du projet.

Énoncé du travail pour la partie 2

Vous devez créer une application console pour interagir avec la base de données dont vous avez fait la conception.

- L'application doit être développée en C# en utilisant les méthodes vues dans le cours.
- L'interface sera faite en mode console.
    - Vous devez donc avoir un menu permettant d'accéder aux opérations demandées.
    - Ce menu ne sera pas corrigé.
- La communication avec la base de données doit se faire à l'aide de l'***ORM*** **Entity Framework Core**.
- Vous devez utiliser la technique ***Code First*** pour générer votre contexte et vos modèles.
- La **chaine de connexion** doit être codée **directement dans le context.** (je ne dois pas avoir à chercher comment me connecter à votre bd. Ce n'est pas une solution recommandée en entreprise, mais elle me sauvera beaucoup de temps de correction. Nous verrons une autre façon de le faire plus tard)
- L'application doit utiliser le **serveur départemental.**
- Vous devez **utiliser la technique de migration de Entity Framework**. Vous pouvez utiliser autant de migrations que nécessaire.
- Vous devez insérer des données dans la bd en utilisant la méthode démontrée dans la section **insertion des données d'entity framework** des notes de cours.
- Vous **devez créer et utiliser une classe utilitaire** pour valider les entrées de l'utilisateur. Par exemple, si la valeur demandée doit être un entier, il faut poser la question de nouveau si l'utilisateur entre une lettre.
- Vous devez **séparer l'application en plusieurs projets** tel que démontré durant le cours (.EF, CRUD)
- Vous devez **utiliser les méthodes d'extensions** pour faire l'affichage des données.
- Le code doit être documenté
    - documentation des fonctions et des arguments des fonctions
    - documentation des classes
    - documentation de tout code "obscure" ou "complexe" (au lieu d'avoir du code complexe, vous devriez le diviser en fonctions plus simples ayant un nom significatif.

Critères pour la base de données

1. Votre bd devra être entreposée **sur le serveur départemental**. Vous devez lui donner le nom **eDA_4N1_TP2** (en changeant DA pour votre DA)
2. Vous ne pouvez pas réutiliser la bd du cours de web 4
3. Vous devez utiliser les techniques vues dans la section Entity framework des notes de cours afin de créer les migrations et les données de base (seed). (**ne pas utiliser Bogus** pour ce projet)
4. Vous devez utiliser la technique **Code First**
5. La bd doit contenir un minimum **de 5 tables**
6. Vous devez utiliser un **maximum de contraintes**
7. Une de ces tables doit être **une table de jonction n-m**
8. La table de jonction doit avoir **au moins un champ autre que les clés étrangères** des tables associées
9. Il ne doit pas y avoir de **table isolée**.
10. Il doit y avoir au moins **2 relations 1-n** (autre que la table de jonction n-m).
11. Les tables doivent avoir au **moins 3 champs autres que la clé principale** et les clés étrangères.
12. Chaque table devra contenir **au moins 5 enregistrements** (créés lors du seed)
13. Vous devez avoir des données de **plusieurs types**: int, date, char, float….
14. Les données et les relations **doivent faire du sens**.

Critères pour le dictionnaire de données

1. Chaque table doit être définie:
    a. son nom
    b. son rôle dans le projet (ce qu'elle contient)
2. Chaque champ doit comporter:
    a. son nom,
    b. son type,
    c. sa longueur (si applicable)
    d. son unicité,
    e. sa définition
3. La définition doit inclure :
    a. le rôle de ce champ
    b. les valeurs acceptées (si applicable) (contraintes)
    c. les relations et dépendances (si applicable) (FK)
    d. les règles de calcul (si applicable)
    e. les règles de validation (si applicable) (contraintes)

Les fonctionnalités devant être réalisées par l'application

1) **CRUd simple version query**
    a. L'application doit permettre de faire un CRUD pour une entité simple. Simple veut dire qu'elle ne dépend pas d'une autre entité (pas de FK)
    b. Les requêtes doivent utiliser la **syntaxe query**.
    c. L'application doit **demander et valider** les valeurs à mettre dans la table.
    d. Il doit y avoir **au moins 2 champs** de types différents dans la table utilisée.
    e. Vous n'avez pas à faire la suppression (le D).
    f. L'update ne doit pas permettre de changer la clé primaire, seulement les autres champs
2) **cRud simple version lambda**
    a. Vous devez refaire la **partie R du CRUD** précédent, mais vos accès à la bd doivent utiliser la **syntaxe lambda**
3) **CRUD Complexe**
    a. L'application doit permettre de faire un CRUD pour une entité associée à une autre par une clé étrangère (relation 1-n) (pas la table n-m).
    b. Vous devez demander et valider l'information pour créer l'entité du côté 1 de la relation, et ensuite afficher les valeurs disponibles pour la table associée et permettre à l'usager de choisir une valeur dans celle-ci pour faire l'association.
    c. Pour la partie D (effacer), vous **devez avertir l'usager** si l'effacement de la donnée entrainera une cascade d'effacement. Si l'usager confirme, l'effacement peut avoir lieu.
    d. Utilisez la syntaxe que vous préférée (query ou lambda, ou un mix)
4) **Affichage de données multi-tables.**
    a. Vous devez afficher les données provenant des tables qui sont des deux côtés de la relation n-m.
    b. Pour ce faire, vous devez afficher les données d'une des tables et demander quelle entité afficher. Vous devez ensuite afficher cette entité ainsi que toutes celles qui y sont attachées dans l'autre table en passant par la table de relation. Vous devez aussi afficher l'information contenue dans la table de relation.
    c. Utilisez du **eager loading**
    d. Les données multiples doivent **être triées alphabétiquement** sur un des champs.
    e. Utilisez la syntaxe que vous préférée (query ou lambda, ou un mix)
5) **Agrégation**
    a. Vous devez faire une agrégation (somme, moyenne, compte)
    b. Pour ce faire, vous devez aller chercher une information dans une table et faire un calcul sur les données associées dans une autre table.
    c. Vous n'avez pas à demander quelle information afficher à l'usager, cette sélection peut être "hard coded".
    d. Utilisez la syntaxe que vous préférée (query ou lambda, ou un mix)

Critères d'évaluation

La première partie du travail pratique vaut pour 5% de la note.

Elle est à remettre le **19 février 2024 23:59:59**

Évaluation	**Pondération**
1- DEA conforme aux spécifications	5
Total	**5**
Retard : 10% par jour de retard pour un maximum de 10 jours.	-10% par jour

La deuxième partie du travail pratique vaut pour 35% de la note finale du cours.

Elle est à remettre le **22 mars 2024 23:59:59**

Ce travail est évalué sur 100 points.

Pour chacune des fonctionnalités, le code doit respecter l'architecture et les techniques présentées en classe.

Évaluation	**Pondération**
1) Dictionnaire de données correspondant à la bd tel qu'utilisée dans l'application et répondant aux critères.	10
2) Migrations	10
3) Seed	5
4) Fonctions de validation des données entrées	5
5) CRUD simple query	20
6) CRUD simple lambda	5
7) CRUD multi-tables	20
8) Affichage de données multi-tables	10
9) Agrégation	5
10) Utilisation de méthodes d'extension pour l'affichage	5
11) Documentation et mise en forme du code	5
Total	**100**
Français : Pénalité accordée au français. -1% par faute	1 point par faute jusqu'à 15 points
Retard : 10% par jour de retard pour un maximum de 10 jours.	-10 points par jour