420-W1T-SW - Projet 2

Pondération : 50%

Durée et remise

Délai accordé

Selon le calendrier présent, vous avez un total de 16 jours (incluant 7 séances en classes) pour compléter le projet. Vous avez donc jusqu'au 13 juin à 23h59 pour remettre le travail. Après cette date, une pénalité de 10% par jour de retard est appliqué conformément à la politique du département.

Remise de code

Par Git. Vous devrez remettre votre code de façon continue tout le long de ce projet.

Vous devez vous assurer que l'enseignant a accès à votre code source à la date de remise.

NB. Vous pouvez faire autant de commits que vous voulez. Sauf demandes spécifiques, le dernier commit avant la date limite sera considéré pour la correction.

Vous devez inclure des informations **spécifique au cours** afin de pouvoir facilement retrouver votre travail.
Par exemple : 420-W1T-SW - 0123456 - Projet2 (no.cours - no étudiant - Projet1)
Inclure au moins :

- Le nom OU le code du cours (420-W1T-SW)
- Votre nom OU votre matricule d'étudiant

Vous devez créer un nouveau dépôt (repository) **spécifique à ce projet**.

Conseils avant de débuter

Prenez le temps de **bien lire et comprendre le(s) énoncé(s)**.

CÉGEP SHAWINIGAN
Du savoir et des gens

Règles et directives

- C'est un travail à effectuer en équipe de 2 à 3, mais il est également possible de l'accomplir seul. Les règles usuelles concernant le plagiat s'appliquent. **Toutefois, l'usage des IA EST permise** pour vous assister dans le développement.
- Le code en français ou en anglais est accepté, mais vous devez être CONSTANTS dans votre choix. Si vous codez en français, faites le partout. Si vous codez en anglais, faites le partout.
- Indiquez toujours vos références lorsque c'est possible. Par exemple, si vous trouvez un snippet de code sur internet, vous devez en indiquer la provenance en commentaires et expliquer son fonctionnement détaillé.
- Indiquez toujours vos références lorsque c'est possible.
- Vous devez **respecter les standards et conventions**! Exemples :
  - Code bien indenté
  - Code uniforme (Ex: Code appliqué de la même façon dans la solution)
    - Sauts de lignes
    - Tabulations vs espaces
  - Nomenclature du bon format (PascalCase, camelCase, ...)
  - Nomenclature adéquate (Noms appropriés et pertinents)
    Ex. : "Toto" pour un compteur n'est pas très pertinent.

BON SUCCÈS

Énoncé

L'objectif de ce projet est de concevoir et implanter une base de données complète (relationnelle ou documentaire) pour un système de gestion de bibliothèque, avec toutes les composantes suivantes :

- Un modèle de données clair, structuré, et documenté.
- Une implémentation ORM/ODM via Entity Framework Core ou MongoDB.Driver.
- Des classes DAO dédiées, injectées et testées.
- Des requêtes simples et avancées : filtres, tri, jointures, agrégations.
- Une API REST **minimale** servant uniquement de point d'entrée pour les tests.

Mise en situation

Votre travail consiste à créer un système de gestion de médiathèque personnalisée (basée sur IMDb).

Vous devez d'abord télécharger les fichiers de IMDB disponible légalement à cette adresse :
https://datasets.imdbws.com/
Et utiliser leur documentation pour adapter le format des données à vos besoins :
https://developer.imdb.com/non-commercial-datasets/

Attention ! Les fichiers sont volumineux. Vous devez faire un travail d'élagage pour conserver seulement quelques centaines de données (Les IA peuvent vous y aider 😉)

Contexte

Chaque utilisateur peut :

- Créer un compte (aucun système d'authentification requis — données seulement)
- Effectuer les opérations de CRUD basique sur l'ensemble des éléments.
- Rechercher des films ou séries :
  - Recherche par mots-clés dans les champs texte.
  - Filtres inclusifs/exclusifs par genre.
- Marquer des œuvres comme :
  - **Vues**
  - **À voir plus tard**
  - **Favoris**
- Consulter ses **statistiques personnalisées**, telles que :
  - Nombre de films vus.
  - Genres les plus fréquents.
  - Moyenne de durée des films vus.
  - Activité par année.

📂 Données à importer (IMDb)

Ce que vous devez faire :

1. Télécharger les fichiers de données IMDb à l'adresse suivante :
   🔗 https://datasets.imdbws.com/
2. Lire leur documentation pour comprendre les fichiers et les colonnes :
   🔗 https://developer.imdb.com/non-commercial-datasets/
3. Sélectionner **un sous-ensemble pertinent** :
   - Films et séries entre **2000 et 2022**
   - Max. **2000 enregistrements**
4. Nettoyer les données :
   - Supprimer les colonnes inutiles
   - Adapter les champs au format de votre modèle
   - Convertir en **CSV** ou **JSON**
5. Intégrer les données dans votre seed :
   - Soit par code (`HasData()` ou `InsertMany`)
   - Soit en important un fichier via un utilitaire dans votre projet.

Entités minimales recommandées

Entité	Description
`User`	Compte personnel
`MediaItem`	Film ou série (tiré d'un fichier fourni)
`MediaStatus`	Table de liaison entre `User` et `MediaItem` avec champ `Status` (`Seen` , `ToWatch` , `Favorite`)
`Genre`	Liste de genres associés à un `MediaItem`
`CastMember`	Acteurs ou réalisateurs (optionnel)

Relations minimales recommandées

- `User` 1 → N `MediaStatus`
- `MediaItem` N ←→ N `Genre`
- `MediaItem` N ←→ N `CastMember` (optionnelle)

Votre application (via l'API minimale ou des tests) doit permettre d'exécuter les requêtes suivantes :

- ✅ CRUD de base sur l'ensemble des éléments.
- ✅ Ajouter une œuvre **et** ses genres liés dans **une seule transaction**.
- ✅ Lister tous les films vus par un utilisateur donné.
- ✅ Rechercher des éléments selon un mot-clé ou un genre.
- ✅ Requête d'agrégation : nombre de films vus par année.
- ✅ Requête d'agrégation : moyenne de durée des films vus.
- ✅ Obtenir les genres les plus fréquents pour un utilisateur.
- ✅ Requête avec projection : afficher uniquement le titre et l'année.

Contraintes techniques

- Vous pouvez utiliser **MySQL (EF Core)** ou **MongoDB (MongoDB.Driver)** au choix.
- Aucune interface utilisateur n'est requise (pas de frontend).
- L'API REST ne sera **pas évaluée sur sa complexité**, mais doit permettre de **valider vos requêtes**. Des points seront attribués pour sa présence, sa structure minimale, et son bon fonctionnement.
- Votre solution doit inclure :
  - Des **entités correctement modélisées**.
  - Des **DAO** pour toutes les entités.
  - Des **requêtes avancées** (filtrage, agrégation, projection).
  - Un **seed de données réalistes**.
  - Un **README clair** expliquant votre structure et vos choix techniques.

Livrables obligatoires (Git)

- Code source complet (solution Visual Studio ou .NET Core).
- Fichier(s) de seed (`.json` , `.csv` , ou code).
- Fichier Postman (`.json`) avec les tests de démonstration.
- Fichier `README.md` contenant :
  - Technologies utilisées.
  - Structure des entités.
  - Instructions pour exécuter votre projet.
  - Exemples de requêtes (objectifs démontrés).
  - (Facultatif) Difficultés rencontrées ou éléments non complétés.

Jalons et objectifs

Afin de vous aider à mesurer votre progression, voici quelques jalons :

- Cours 1 : Git, nouveaux projets, configuration des librairies, récupération des fichiers.
- Semaine 1 : Définitions, modélisation + extraction des données
- Semaine 2 : DAO + requêtes + tests

Rappel pour la remise

La remise s'effectue sur Bitbucket ou GitHub. Vous devez créer un nouveau dépôt pour ce projet dans le Workspace du cours.

L'utilisation du git est obligatoire. D'ailleurs, des points sont accordés à la régularité de vos commits **(minimum un par cours)**. Donc, dans ce cas-ci, il y a un minimum de **4** commits à faire dans votre dépôt.

Le dernier commit avant la date de remise sera pris pour la correction. Si vous désirez qu'un autre commit soit pris après la date de remise, veuillez en faire part à l'enseignant le plus tôt possible. Toutefois, vous serez pénalisé à la hauteur de ce qui est mentionné pour les travaux remis en retard dans le plan de cours.