|  |  |
| --- | --- |
| **420-4B6-LI Applications Web I**  **Cégep Limoilou**  **Département d'Informatique**  **Hiver 2020** | **Tp3 (30 %)**  **Application Web ASP.NET MVC** |

OBJECTIFS

* Créer une application Web en lien avec une base de données
* Planifier le travail à faire et identifier les principales fonctions, objets et interactions à développer.
* Programmer la logique applicative côté client.
* Programmer les interfaces côté client en exploitant Bootstrap.
* Programmer la logique applicative côté serveur avec le framework ASP.NET MVC.
* Programmer en orienté-objet avec C#.
* Utiliser et exploiter l’utilisation d’une base de données à l’aide du modèle Entity Framework.
* Utiliser et exploiter la mise à jour des vues partielles avec AJAX.
* Valider le fonctionnement de l’application.

**DESCRIPTION**

Vous allez repartir de votre application inspirée de Netflix réalisée dans le Tp2. Dans un premier temps, vous allez analyser le travail et distribuer les tâches. Ensuite, vous allez adapter votre solution afin de travailler à partir d’une base de données et ajoutez quelques fonctionnalités.

Finalement, vous allez ajouter quelques fonctionnalités d’administration à votre application.

Votre application adaptera ou ajoutera les fonctions suivantes :

* S’enregistrer
* Obtenir de l’information sur un produit.
* Voir un extrait.
* Voir au complet.
* Ajouter ou retirer dans une liste de favoris.
* Afficher la liste de favoris.
* Effectuer des opérations d’administration (ajouter, modifier, supprimer des films et des utilisateurs).

INSTRUCTIONS

PREMIÈRE PARTIE

1. Faire une copie de votre TP2 à partir de laquelle vous allez créer votre TP3.
2. Faire un document (tableau) en répertoriant toutes les tâches que vous pensez nécessaires de faire pour réaliser ce Tp.
3. Pour chacune des tâches, décomposez si nécessaire en sous-tâche (apprentissage, site à consulter), évaluez le temps requis et ajoutez le nom du responsable principal de la tâche.
4. Déposez sur GIT ce document.

DEUXIÈME PARTIE

1. Organisez la persistance de vos données pour que celles-ci soient enregistrées dans une base de données. À partir d’Entity Framework Code First, générez votre base de données et vos tables pour la gestion de vos médias et de vos utilisateurs.

**Ici, vous aurez à choisir de quelle façon vous conservez vos images (image réelle ou nom de fichier) dans votre BD et aussi de quelle façon vous conservez le mot de passe des utilisateurs pour les encoder (partie plus complexe que vous pouvez adapter un peu plus loin, voir section sur ce sujet plus bas).**

1. Ajoutez les utilisateurs (2-3) dont 1 administrateur à votre base de données et aussi quelques médias. Faites l’ajout selon la méthode que vous désirez. Vous pouvez différez le mot de passe encoder, ça fera une belle pratique de migration par la suite.
2. Générez le script SQL pour créer les tables de votre base de données et l’ajout des données de chacune des tables dans un seul fichier script que vous placerez à la racine de votre solution. Pensez à une procédure qui vous permettrait de mettre à jour ce script régulièrement pour vos coéquipiers et pour moi aussi.
3. Effectuez une migration initiale de votre base de données et créez de nouvelles migrations si vous apportez des changements par la suite. (je dois voir dans votre solution que vous avez réalisé au moins 2 migrations).
4. Adaptez les actions de vos contrôleurs afin que vos données proviennent de requêtes à la base de données et non du fichier XML. Vous pouvez retirer tous les fichiers XML de votre solution. Donc, pour le moment vous avez à adapter le login de l’utilisateur, l’affichage des médias et la liste des favoris de l’utilisateur. Assurez-vous que ces fonctions sont présentes et au besoin améliorez-les :
5. Les nouveautés de cette partie sont :
   1. améliorer ou ajoutez les actions suivantes s’il y a lieu :
      1. Obtenir de l’information sur un produit.
      2. Voir un extrait.
      3. Voir au complet.
      4. Ajouter ou retirer dans une liste de favoris
      5. Afficher la liste de favoris.
   2. sauvegarder le mot de passe de l’utilisateur de façon adéquate dans la BD (voir annexe)
   3. ajouter les actions nécessaires pour la gestion de l’utilisateur
      1. déconnecter,
      2. modifier mot de passe
      3. Récupérer mot de passe
   4. ajouter une action pour rechercher un film dans la liste des films ou des favoris(selon un bout de titre).

TROISIÈME PARTIE

Cette partie permet d’administrer le site, seul l’administrateur y a accès.

Il pourra :

1. Ajouter des utilisateurs dans la base de données en leur fournissant un mot de passe temporaire.
2. Modifier ou supprimer des utilisateurs.
3. Ajouter, modifier, supprimer des films dans la base de données.

Prévoir les routes, vues, actions et menus adaptés à l’administrateur.

**À remettre**

Le travail se réalise en équipe et demande une démonstration pour les 2 dernières parties. Le Tp comporte trois remises sur GIT et les dépôts doivent être documentés et alimentés par tous les équipiers.

|  |  |
| --- | --- |
| **Dates de remise :** | **10 avril 2020 – Première partie**  **7 mai 2020 – Deuxième partie**  **21 mai 2020 – Troisième partie** |

# Annexe

**Conservation du mot de passe dans la base de données**

Plusieurs solutions possibles et librairies (ex. : Identity) du côté d’ASP.NET sur le sujet.

Une solution est de ‘hasher’ votre mot de passe et conserver le sel dans la BD.

Voici 2 références, il en pleut, mais cherchez pour des solutions ASP.NET MVC avec Identy.

[Hashing Passwords using ASP.NET's Crypto Class - CodeProject](https://www.codeproject.com/Articles/844722/Hashing-Passwords-using-ASP-NETs-Crypto-Class)

[Registartion Page with ASP.net MVC5, Entity Framework, Code First(Part 2) with Password Encryption - YouTube](https://www.youtube.com/watch?v=D4adL6b2aAQ)

Ici un exemple d’authentification maison

<https://openclassrooms.com/fr/courses/1730206-apprenez-asp-net-mvc/2098931-gerer-l-authentification>

Mais, honnêtement, je préfèrerais que vous utilisiez Identity, je vais vous donner un exemple complet cette semaine.

EN CONSTRUCTION – À ADAPTER

420-4B6-LI Développement Web 1

**Grille de correction Tp3 – 30 %**

**Noms :**

**/100**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Capacité 1** | **Excellent 100** | **Bien 85** | **Suffisant 60** | **Incomplet 30** |
| Programmer la logique applicative côté client | Le document d’analyse est complet et parfaitement détaillé.  Les vues sont bien exploitées et utilisent des instructions Razor, HTML et CSS.  Au moins 2 dispositions différentes sont utilisées.  Au moins un élément utilise un effet Javascript ou Jquery.  Les barres de navigations sont bien présentées.  Les images et les éléments multimédias sont efficaces et bien présentés.  Le code est bien structuré, propre et complet en respect avec les instructions fournies et les décisions prises dans l’analyse.  Il est facile de se repérer dans l’application et elle est bien documentée.  Les bonnes pratiques au niveau du positionnement des éléments dans les interfaces sont bien utilisées.  L’application est adaptable à différentes résolutions.  L’esthétisme est remarquable.  Toutes les instructions sont respectées. | Le document d’analyse est complet et suffisamment détaillé.  Les vues sont bien exploitées et utilisent des instructions Razor, HTML et CSS.  Au moins 2 dispositions différentes sont utilisées.  Au moins un élément utilise un effet Javascript ou Jquery.  Les barres de navigations sont bien présentées.  Les images et les éléments multimédias sont efficaces et bien présentés.  Le code est bien structuré, propre et complet en respect avec les instructions fournies et les décisions prises dans l’analyse.  Il est généralement facile de se repérer dans l’application et elle est bien documentée.  Les bonnes pratiques au niveau du positionnement des éléments dans les interfaces sont généralement bien utilisées.  L’application est généralement adaptable à différentes résolutions.  L’esthétisme est très bien.  La plupart des instructions sont respectées. | Le document d’analyse est minimal.  Les vues sont bien exploitées et utilisent des instructions Razor, HTML et CSS la plupart du temps  Au moins 1 disposition est utilisée.  Au moins un élément utilise un effet Javascript ou Jquery.  Les barres de navigations sont présentes.  Les images et les éléments multimédias sont présentes.  Le code est généralement bien structuré, propre mais peut présenter une ou deux lacunes mais ne mettant pas en péril l’exécution.  Il peut être difficile de se repérer dans l’application et elle est documentée sommairement.  Les bonnes pratiques au niveau du positionnement des éléments dans les interfaces sont à améliorer.  L’application ne s’adapte pas à différentes résolutions.  L’esthétisme est correct.  La plupart des instructions sont respectées. | Le document d’analyse est absent.  Les vues utilisent des instructions Razor, HTML et CSS la plupart du temps  Au moins 1 disposition est utilisée.  Les barres de navigations sont présentes.  Les images et les éléments multimédias sont présentes.  Le code est généralement bien structuré, propre mais peut présenter plusieurs lacunes.  Il peut être difficile de se repérer dans l’application et elle n’est pas documentée suffisamment.  Les bonnes pratiques au niveau du positionnement des éléments dans les interfaces sont à améliorer.  L’application ne s’adapte pas à différentes résolutions.  L’esthétisme est rudimentaire.  Plusieurs des instructions ne sont pas respectées. |
| **Capacité 2** | **Excellent 100** | **Bien 85** | **Suffisant 60** | **Incomplet 30** |
| Programmer la logique applicative côté serveur | Les contrôleurs sont bien structurés et les actions sont bien définies et complètes en lien avec les instructions.  La navigation est bien soutenue par les routes.  Les fichiers sont bien utilisés pour récupérer les données et sauvegarder les modifications.  Il y a au moins 2 fichiers de produits et 2 fichiers utilisateurs.  Le code est bien structuré, propre et complet en respect avec les instructions fournies et les décisions prises dans l’analyse.  Il est facile de se repérer dans l’application et elle est bien documentée.  Les modèles sont bien définis et sont bien utilisées.  Les outils de débogage sur un navigateur web sont utilisés de manière autonome.  Des cas de tests pour les classes et pour les principales fonctions ont été fournis et appliqués.  L’exécution est impeccable.  Les choix (algo, structure de données, affichage) sont judicieux et efficaces.  Le travail est fait de façon autonome.  Les dépôts sur GIT sont faits de façon régulière, parfaitement repérables et les différentes versions sont clairement identifiées et documentées. | Les contrôleurs sont bien structurés et les actions sont bien généralement définies et complètes en lien avec les instructions.  La navigation est généralement bien soutenue par les routes.  Les fichiers sont généralement bien utilisés pour récupérer les données et sauvegarder les modifications.  Il y a au moins 2 fichiers de produits et 2 fichiers utilisateurs.  Le code est généralement bien structuré, propre et complet en respect avec les instructions fournies et les décisions prises dans l’analyse.  Il est généralement facile de se repérer dans l’application et elle est bien documentée.  Les modèles sont généralement bien définis et sont bien utilisées.  Les outils de débogage sur un navigateur web sont généralement utilisés de manière autonome.  Généralement, des cas de tests pour les classes et pour les principales fonctions ont été fournis et appliqués.  L’exécution est impeccable la plupart du temps.  Les choix (algo, structure de données, affichage) sont généralement judicieux et efficaces.  Le travail est fait de façon autonome la plupart du temps.  Les dépôts sur GIT sont faits de façon régulière, généralement repérables et les différentes versions sont clairement identifiées et documentées. | Les contrôleurs sont bien structurés et les actions sont assez bien définies en lien avec les instructions (il peut manquer 1 ou 2 éléments).  La navigation peut comporter certaines lacunes au niveau de la définition des routes.  Les fichiers sont utilisés pour récupérer les données et sauvegarder les modifications, mais peuvent présenter des lacunes.  Il y a au moins 1 fichier de produits et 1 fichier utilisateurs.  Le code est structuré, mais est à améliorer.  Il peut être difficile ile de se repérer dans l’application et elle est documentée sommairement.  Les modèles sont à améliorer.  Les outils de débogage sur un navigateur web sont utilisés avec de l’aide.  Les cas de tests pour les classes et pour les principales fonctions ont été fournis sommairement et appliqués sommairement.  L’exécution présente des lacunes (moins de 3).  Les choix (algo, structure de données, affichage) sont à raffiner.  Le travail est fait avec de l’aide constante.  Les dépôts sur GIT sont faits de façon irrégulière et peu documentés. | Les contrôleurs sont structurés et les actions sont peu définies en lien avec les instructions (il peut manquer plus de 2 éléments).  La navigation peut comporter certaines lacunes au niveau de la définition des routes ou être absente.  Les fichiers sont utilisés pour récupérer les données et sauvegarder les modifications, mais peuvent présenter plusieurs lacunes.  Il y a au moins 1 fichier de produits et 1 fichier utilisateurs.  Le code est peu structuré.  Il est difficile ile de se repérer dans l’application et elle est peu documentée.  Les modèles sont à améliorer.  Les outils de débogage sur un navigateur web sont peu utilisés.  Les cas de tests pour les classes et pour les principales fonctions ne sont pas fournis.  L’exécution présente des lacunes (plus de 3).  Les choix (algo, structure de données, affichage) sont peu raffinés.  Le travail est fait avec de l’aide constante.  Les dépôts sur GIT sont peu fréquents e et peu documentés. |