TP 1 : Le Bataillon 2 octobre 2020

Étude de cas

La compagnie Bataillon a été fondée en 2001 par trois frères Justin, Marc et Thierry Riopelle. Les trois administrateurs gèrent chacun un secteur. Bataillon possède quatre terrains (2 extérieurs et 2 intérieurs) de catégorie facile-intermédiaire et trois terrains experts (2 intérieurs et 1 extérieur).

En plus des périodes de batailles normales, l'entreprise propose des événements thématiques spéciaux pour ses membres des forfaits anniversaires. La gestion des événements spéciaux est sous la responsabilité de Thierry tandis que les forfaits anniversaires et les périodes de bataille normales sont gérés par Marc.

Un membre peut créer une équipe et y inscrire 6 à 12 membres via leur adresse courriel, ce membre est automatiquement nommé capitaine de l'équipe. Le capitaine est le seul à pouvoir modifier la liste des membres de l'équipe et inscrire l'équipe à une partie ayant lieu dans les périodes de bataille. (la gestion des périodes de bataille sera effectuée ultérieurement).

Le capitaine inscrit son équipe à une partie qui a lieu à une période de bataille (date et heure) précise, contre une autre équipe. Lorsque la partie est terminée, l'équipe gagnante est déterminée et indiquée dans la partie. (elle sera éventuellement déterminée en fonction des résultats de tirs).

Le tableau de bord des membres avec leurs résultats personnels, en équipe, les prochaines parties etc. sera créé ultérieurement à partir d'un service fourni répertoriant les tirs.

Consignes (15% de la note finale)

Section 1 (pour le 22 septembre)

1. Créez un Repo sur GitHub **LeBataillon** et partagez votre repo avec votre enseignant
   a) Créez une branche à partir de la branche master **TP1_Depart**
   b) Documentez vos modifications (push) de façon pertinente
2. Utilisez les fichiers fournis dans *TP1_section_1_fichiers_complementaires.zip*
   a) Classe *Player*
   b) Classe *Team*
   c) Classe *Game*
   d) Le layout *_Layout.cshtml*
   e) Copiez les Seeds pour initialiser des données de départ dans la BD
3. Utilisez EF Core - rameworks 3.1
4. Créez et nommez les éléments :
   a) La solution **LeBataillon**
   b) Le projet **LeBataillon.Database**
   c) Le projet MVC **LeBataillon.Web**
      i) Ajoutez/remplacez le contenu du dossier **wwwroot**
      ii) Ajoutez/Remplacez les fichiers du dossier **Home**
   d) Créez la base de données **LeBataillon**
5. Faites deux *migrations* et *update*
   a) **InitialCreate**
      i) pour les classes *Player* et *Team*
      ii) Clés primaires
      iii) Liaisons clés étrangères
      iv) Ajoutez les annotions dans les deux classes selon les règles d'affaires fournies
      v) Ajoutez les *seeds* de chaque classe

NOTE : Vous pouvez avoir plus de migrations et updates si nécessaire, mais vous devez avoir au moins les deux ci-dessus.

   b) **AddGame**
      i) La classe *Game*
      ii) Assurez-vous d'avoir la clé primaire et liaison clé étrangère
      iii) Ajoutez les annotions dans les deux classes selon les règles d'affaires fournies
      iv) Avec le *seed* de chaque classe
6. Générez les *Views* (Create, Delete, Update, Details) pour les trois classes
7. Générez la *view* **_topNav**
   a) Modifiez les formulaires pour avoir des listes
   b) Cachez les ID
8. Générez les *Controllers*
9. Testez les *views* et les *controllers*.
10. Si ce n'est pas déjà fait, faites un *push* de vos modifications sur GitHub

Section 2 (pour le 25 septembre)

1. Ajoutez/Remplacez le fichier **_topNav**
2. Ajoutez le nécessaire pour naviguer dans votre site de façon fluide et sans cul de sac.
3. Créez un projet test xUnit **LeBaillon.tests**
4. Ajoutez un test pour chaque classe, si nécessaire, référez-vous aux règles d'affaires fournies précédemment
   a) *Mockez* les classes *Player, Team, Game*
   b) Méthodes
   c) *Controllers*
5. Testez l'accessibilité de votre site
   a) Générez rapport initial de *Chrome Lighthouse* et enregistrez la version initiale en pdf **rapportInitial,** ajoutez-le dans votre repo GitHub
      i) Best pratrices
      ii) Accessibility
      iii) SEO
      iv) Desktop
   b) Effectuez les corrections nécessaires
   c) Générez rapport de nouveau et enregistrez la version en pdf **rapportv2** ajoutez-le dans votre repo GitHub.
6. Si ce n'est pas déjà fait, faites un *push* de vos modifications sur GitHub

Section 3 (pour le 2 octobre)

1. Ajoutez/Remplacez les fichiers du dossier **Repository**
2. Ajoutez des validations simples de saisies utilisateur pour les trois niveaux, si nécessaire, référez-vous aux règles d'affaires *TP1_section_3_fichiers_complementaires.zip*
   a) Niveau du serveur
   b) Niveau de la page
   c) Niveau de la base de données
   d) Affichez des messages clairs
3. Si ce n'est pas déjà fait, faites un *push* de vos modifications sur GitHub
4. Créez une branche **TP1_RemiseFinale** à partir de la branche **TP1_Depart** comprenant la version à corriger

NOTE : cette branche sera finale (comme une version en production), vous repartirez de la branche principale pour le prochain TP.

Grille de correction: TP 1						
Composantes de la tâche et critères d'évaluation	**Très bien**	**Bien**	**Passable**	**Insuffisant**	**Incomplet**	**Note**
Mise en place d'une application respectant le patron MVC • Type de projets • Contenu (fichiers fournis)	L'application est composée d'un projet Database et d'un projet Web utilisant le type adéquat et correctement identifiés. Les projets contiennent tous les fichiers fournis correctement positionnés. L'étudiant démontre une très bonne connaissance du framework .NET Core	L'application est composée d'un projet Database et d'un projet Web utilisant le type adéquat et sont plus ou moins identifiés correctement. Les projets contiennent tous les fichiers fournis plus ou moins correctement positionnés. L'étudiant démontre une bonne connaissance du framework .NET Core	L'application est composée d'un projet Database et d'un projet Web utilisant le type adéquat et sont plus ou moins identifiés correctement. Les projets contiennent tous les fichiers fournis plus ou moins correctement positionnés. L'étudiant démontre une connaissance du framework .NET Core	L'application n'est pas composée d'un projet Database et d'un projet Web utilisant le type adéquat ou ne sont pas identifiés correctement. Et/ou les projets ne contiennent pas tous les fichiers fournis correctement positionnés. Et/ou l'étudiant ne démontre pas une connaissance minimum du framework .NET Core	L'application n'est pas composée d'un projet Database et d'un projet Web utilisant le type adéquat et ne sont pas identifiés correctement. Et les projets ne contiennent pas tous les fichiers fournis correctement positionnés. Et l'étudiant ne démontre pas une connaissance du framework .NET Core.	
	5	4	3	2	0-1	/5

Grille de correction: TP 1						
Composantes de la tâche et critères d'évaluation	**Très bien**	**Bien**	**Passable**	**Insuffisant**	**Incomplet**	**Note**
Mise en place d'entity framework Core • Connexion à la Base de données • DbContext • Annotations • Migrations et updates • Repository • *Seeds*	L'application contient une connexion fonctionnelle à la BD et un DbContext adéquat. Les annotations sont complètes et correspondent aux règles d'affaires. Le projet contient au moins les deux migrations demandées. L'étudiant démontre une très bonne connaissance du EF Core	L'application contient une connexion fonctionnelle à la BD et un DbContext adéquat. Les annotations sont presque complètes et correspondent aux règles d'affaires. Le projet contient au moins les deux migrations demandées. L'étudiant démontre une bonne connaissance du EF Core	L'application contient une connexion fonctionnelle à la BD et un DbContext comportant quelques erreurs. Ou les annotations sont incomplètes et/ou ne correspondent pas toutes aux règles d'affaires. Le projet contient au moins les deux migrations demandées. L'étudiant démontre une connaissance du EF Core	L'applicationne ne contient pas une connexion fonctionnelle à la BD et/ou un DbContext comportant plusieurs erreurs ou manques. Et/ou les annotations sont incomplètes et/ou ne correspondent pas aux règles d'affaires. Et/ou le projet ne contient pas les deux migrations demandées. Et/ou l'étudiant ne démontre pas une connaissance minimum du EF Core	L'application ne contient pas une connexion fonctionnelle à la BD et un DbContext comportant plusieurs erreurs ou manques. Et/ou les annotations sont incomplètes et/ou ne correspondent pas aux règles d'affaires. Et le projet ne contient pas les deux migrations demandées. Et l'étudiant ne démontre pas une connaissance suffisante du EF Core.	
	17-20	13-16	12	6-11	0-5	/20

Grille de correction: TP 1						
Composantes de la tâche et critères d'évaluation	**Très bien**	**Bien**	**Passable**	**Insuffisant**	**Incomplet**	**Note**
Navigation et présentation Projet de tests xUnit • Tests • Mocking Accessibilité • *Lighthouse* Rapports (2) • Conformité aux normes	Le projet de tests contient tous les tests requis. Le projet test contient un moq couvrant les scénarios possibles en lien avec les règles d'affaires. Les deux rapports de conformité *lighthouse* sont présents. Le projet est conforme aux règles d'accessibilité. L'étudiant démontre une très bonne maîtrise des tests xUnit.	Le projet de tests contient presque tous les tests requis. Ou le projet test contient un moq couvrant presque tous les scénarios possibles en lien avec les règles d'affaires. Les deux rapports de conformité *lighthouse* sont présents. Le projet est conforme aux règles d'accessibilité. L'étudiant démontre une bonne maîtrise des tests xUnit.	Le projet de tests ne contient qu'une partie des tests requis. Et/ou le projet test contient un moq couvrant presque tous les scénarios possibles en lien avec les règles d'affaires. Les deux rapports de conformité *lighthouse* sont présents. Et/ou le projet n'est pas entièrement conforme aux règles d'accessibilité. L'étudiant démontre une maîtrise correcte des tests xUnit.	Le projet de tests ne contient qu'une partie des tests requis. Et/ou le projet test contient un moq couvrant presque tous les scénarios possibles en lien avec les règles d'affaires. Et/ou lne es deux rapports de conformité *lighthouse* sont pas présents. Et/ou le projet n'est pas entièrement conforme aux règles d'accessibilité. L'étudiant démontre une maîtrise suffisante des tests xUnit.	Le projet de tests ne contient qu'une partie des tests requis. Et/ou le projet test ne contient pas le moq couvrant presque tous les scénarios possibles en lien avec les règles d'affaires. Et/ou lne es deux rapports de conformité *lighthouse* sont pas présents. Et le projet n'est pas entièrement conforme aux règles d'accessibilité. L'étudiant ne démontre pas une maîtrise suffisante des tests xUnit.	
	13-15	10-12	9	5-8	0-4	/15

Grille de correction: TP 1						
Composantes de la tâche et critères d'évaluation	**Très bien**	**Bien**	**Passable**	**Insuffisant**	**Incomplet**	**Note**
Gestion de code source et documentation • Utiliser GitHub • Documenter les modifications • Identification des objets/fichiers	La documentation accompagnant les *commit* est claire et adéquate et les branches sont correctement effectuées. Les objets/fichiers sont tous correctement identifiés. L'étudiant démontre une très bonne maîtrise du gestionnaire de code source.	La documentation accompagnant les *commit* est claire et adéquate, à l'exception de quelques ambiguïtés et/ou les branches sont presque toutes correctement effectuées. Les objets/fichiers sont tous correctement identifiés. L'étudiant démontre une bonne maîtrise du gestionnaire de code source.	La documentation accompagnant les *commit* plusieurs ambiguïtés et/ou les branches sont presque toutes correctement effectuées. Et/ou les objets/fichiers sont presque tous correctement identifiés. L'étudiant démontre une maîtrise correcte du gestionnaire de code source.	La documentation accompagnant les *commit* plusieurs ambiguïtés et/ou les branches ne sont pas toutes correctement effectuées ou sont absentes. Et/ou les objets/fichiers sont presque tous correctement identifiés. Et/ou l'étudiant ne démontre pas une connaissance suffisante du gestionnaire de code source.	La documentation accompagnant les *commit* plusieurs ambiguïtés et/ou les branches ne sont pas toutes correctement effectuées ou sont absentes. Et les objets/fichiers sont incorrectement identifiés. Et/ou l'étudiant ne démontre pas une connaissance suffisante du gestionnaire de code source.	
	9-10	7-8	6	3-5	0-2	/10

Grille de correction: TP 1	
Tâche : Application du Bataillon. Correction négative	
Qualité de la langue (PIEA) : Conformément à la Politique institutionnelle d'évaluation des apprentissages (PIEA) en vigueur au Collège, la note obtenue sera diminuée au maximum de 10 % pour tout manquement à la qualité de la langue (0,5 % par erreur). Une erreur répétée est considérée comme une faute additionnelle, sauf dans le cas de fautes d'orthographe. Maximum – 5pts	
Retard : La note obtenue sera diminuée de 10 % (5pts) par jour de retard (tel que stipulé dans le plan de cours et la PIEA) pour un maximum de 5 jours, après ce délai le devoir sera considéré comme non remis et obtiendra la note de 0%	
Note finale	**__/50**