420-5W5

Projet

Super Cartes Infinies (aka Super HeartStone Infini)

# Grille de correction – 20%

## Fonctionnalités /10 – 5% (Évaluation d’équipe)

|  |  |  |  |
| --- | --- | --- | --- |
| **Le projet est excellent.**  Les points énumérés dans l’énoncé ont pratiquement tous été complétés.  Les fonctionnalités ont été réalisés avec une grande qualité. | **Le projet est bien.**  Les points énumérés dans l’énoncé ont été complétés en majorité et les fonctionnalités ont été réalisés avec une grande qualité.  Ou les points énumérés dans l’énoncé ont pratiquement tous été complétés, mais le fonctionnement pourrait être mieux. | **Le projet est passable.**  Les points énumérés dans l’énoncé essentiels ont été complétés et les fonctionnalités ont été réalisés avec une grande qualité.  Ou les points énumérés dans l’énoncé ont été complétés en majorité, mais le fonctionnement pourrait être mieux. | **Le projet est très faible.**  Les points énumérés dans l’énoncé essentiels n’ont pas été complétés.  Ou la qualité des fonctionnalités était insatisfaisante. |
| 10 | 8 à 9 | 6 à 7 | 0 à 5 |

## Travail d’équipe et utilisation des outils /10 – 3% (Évaluation d’équipe)

##### Points importants pour l’évaluation

* DevOps
  + Les features (US) sont présentes sur DevOps.
  + Les tâches sont présentes et attribuées aux membres de l’équipe.
  + Sprint est à jour avec les tâches et les US complétées.
* Git
  + Git est utilisé selon les bons principes (1 branche par feature).
  + Les commentaires des commits sont présents et descriptifs du changement dans le code.
* Déploiement
  + Le projet est déployé.
  + Le projet est mis à jour automatiquement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Le travail d’équipe et l’utilisation des outils sont excellents.**  Les points énumérés ont tous été complétés et sont de grande qualité. | **Le travail d’équipe et l’utilisation des outils sont bien.**  Les points énumérés ont été complétés en majorité avec une grande qualité.  Ou les points énumérés ont pratiquement tous été complétés, mais la qualité pourrait être mieux. | **Le travail d’équipe et l’utilisation des outils sont passables.**  Les points énumérés essentiels ont été complétés avec une grande qualité.  Ou les points énumérés ont été complétés en majorité, mais la qualité pourrait être mieux. | **Le travail d’équipe et l’utilisation des outils sont très faibles.**  Les points énumérés essentiels n’ont pas été complétés.  Ou la qualité était insatisfaisante. |
| 10 | 8 à 9 | 6 à 7 | 0 à 5 |

## Correction du code /10 – 12% (Évaluation individuelle)

##### Instructions particulières pour la correction du code

Durant les 3 sprints vous devrez vous faire corriger 1 fois pour chaque partie (Angular, API et MVC). Vous pouvez choisir la partie que vous voulez pour chaque sprint, mais chaque partie sera corrigée qu’une fois durant la session.

Exemple, vous vous faire corriger en Angular au sprint 1, pour les sprints 2 et 3, vous devrez vous faire corriger API et MVC.

Pour la correction, vous devrez remettre une fonctionnalité, pour qu’une fonctionnalité soit considérée suffisante pour la correction, nous demandons qu’elle ait un minimum de validation. Il sera important aussi de valider l’enseignant la fonctionnalité que vous souhaitez vous faire corriger.

##### Points importants pour l’évaluation

### Angular

* Découplage vue code
  + Il n'y a pas de traitement dans les vues
* Structure du projet client
  + Appels HTTP regroupés (service)
  + Structure et noms homogènes et standards
  + Lisibilité du code (1 fonction < 1 page, etc.)
* Stabilité
  + Crash application
  + Données toujours à jour
* Sécurité
  + Gestion de l'utilisateur
  + Utilisation des données du joueur
  + Manipulation des données de la partie
* Interface utilisateur
  + Messages d'erreur (indique clairement une solution)

### Web API

* Persistance des données
  + Utilisation d'Entity Framework et des services
  + Modélisation des données
  + Gestion des droits d'accès
  + Accès aux données
* Code serveur
  + Validation des données
  + Définition d'exceptions appropriées
* Tests serveur
  + Stratégie de test
  + Couverture complète de la fonctionnalité
  + Tester les cas limites du type et de la logique
  + Tester les exceptions
* Structure du code / standards
  + Découpage en couches
  + Lisibilité du code (1 fonction < 1 page, commentaires etc.)
  + Structure et noms homogènes et respect des standards
  + Code factorisé (Réutilisation du code, encapsulation, dépendances minimales)
  + Utilisation adéquate de l'injection de dépendance
  + Utilisation appropriée des types abstraits et des interfaces

### MVC

* Persistance des données
  + Utilisation d'Entity Framework et des services
  + Gestion des droits d'accès
* Code serveur
  + Validation des données
  + Définition d'exceptions appropriées
* Tests serveur
  + Stratégie de test
  + Couverture complète de la fonctionnalité
  + Tester les cas limites du type et de la logique
  + Tester les exceptions
* Structure du code / standards
  + Découpage en couches
  + Cohabitation avec Web API
  + Lisibilité du code (1 fonction < 1 page, commentaires etc.)
  + Structure et noms homogènes et respect des standards
  + Code factorisé (Réutilisation du code, encapsulation, dépendances minimales)
  + Utilisation adéquate de l'injection de dépendance
  + Utilisation appropriée des types abstraits et des interfaces
* Découplage vue code
  + Il n'y a pas de traitement dans les vues
* Stabilité
  + Crash application
  + Données toujours à jour
* Interface utilisateur
  + Messages d'erreur (indique clairement une solution)

|  |  |  |  |
| --- | --- | --- | --- |
| **La qualité du code est excellente.**  Les points énumérés ont tous été complétés et sont de grande qualité. | **La qualité du code est bien.**  Les points énumérés ont été complétés en majorité avec une grande qualité.  Ou les points énumérés ont pratiquement tous été complétés, mais la qualité pourrait être mieux. | **La qualité du code est passable.**  Les points énumérés essentiels ont été complétés avec une grande qualité.  Ou les points énumérés ont été complétés en majorité, mais la qualité pourrait être mieux. | **La qualité du code est très faible.**  Les points énumérés essentiels n’ont pas été complétés.  Ou la qualité était insatisfaisante. |
| 10 | 8 à 9 | 6 à 7 | 0 à 5 |

Remise du TP

* Vous devrez faire 2 remises sur Léa
* **Une remise par équipe**, vous devrez faire un zip avec :
  + Angular
  + C#
  + URLs (frontend et backend) du projet déployé
* Vous devrez ajouter votre enseignant à Azure DevOps
  + jeanmichel.nadeau@gmail.com
* Vous devrez également ajouter votre enseignant à vos repos GitHub
  + jeanmichel.nadeau@gmail.com ou jmnadeau
* **Une remise individuelle**, vous devrez faire un zip avec :
  + Un document texte avec la description de votre feature, vous devrez dire quelle page ou quel contrôleur vous voulez que votre enseignant corrige.
  + L’application à corriger (client ou backend), votre application pourrait être différente de la version de l'équipe

Pour la remise individuelle

Il doit avoir de la validation à votre partie de code pour qu’elle soit suffisante pour la correction.

Voici quelques exemples de features qui seraient suffisantes pour la correction :

* En Angular
  + Enregistrement et connexion
  + Consommer les évènements de la partie
  + Gestion de la boucle de jeu
    - (Alterner entre le polling pour mettre à jour le match et jouer une carte de sa main)
* En Web API
  + Les combats et la génération des évènements
  + AccountController,
    - Si vous faites corriger AccountController, le contrôleur doit également inclure l'initialisation du joueur
* MVC
  + Les outils d’administration
    - (Si vous avez fait une page pour arrêter les parties en cours, voir l’utilisateur en attente d’une partie, etc.)
  + Gestion des cartes de départ

Tests unitaires

* Tester uniquement le backend
* Tester uniquement le / les services utilisé par la feature
  + - Utiliser une InMemoryDatabase pour réaliser les tests
* Les tests doivent couvrir 100% des méthodes utilisées dans les services