# TP1 : CEGES

## **Consignes**

* Lisez toutes les instructions et la grille avant de commencer
* Le projet doit mettre en œuvre les Data Layers : Areas et Multi-projets
* La logique métier doit être gérer dans les **Services**
* Les branches doivent être identifiées correctement (ne pas les supprimer après le *merge*)

## **Étude de cas CEGES**

L’étude de cas est présentée dans le fichier joint « TP1 Projet CEGES.docx »

## **Sprint zéro**

* Créez un repo privé et donnez accès à votre enseignante (valerieturgeon)
* Le projet de départ **CEGES\_Depart.zip** se trouve dans Teams :
  + Le projet .NET MVC **(CEGES\_MVC)** avec des vues et des contrôleurs sans logique d'affaire (logique métier)
  + Un projet « Util » **(CEGES\_Util)** avec quelques fonctions pratiques
* Faites l’analyse de l’étude de cas et discutez de votre modèle de classe avec un coéquipier
* Créez le diagramme de classe et ajoutez-le dans le repo. Faites au moins un commit pour le diagramme. Vous pouvez le faire approuver avant de continuer si vous le souhaitez.
* Ajoutez les projets Visual Studio requis pour le modèle de données (**CEGES\_Models)**, l'accès aux données (**CEGES\_DataAccess)** ainsi que le projet des services **(CEGES\_Services)** à la solution de départ.

## **Sprint 1**

### Récits utilisateurs

Voir la liste des récits utilisateurs du sprint 1.

## **Grille synthèse**

|  |  |  |
| --- | --- | --- |
| Critère d’évaluation | Note et pondération | |
| Qualité du code et fonctionnalités |  | / 20 |
| Gestion des données (Data Layers et services asynchrones) |  | / 25 |
| Classes du modèle (relations, annotations, validations, propriétés de navigation) |  | / 20 |
| Vues et contrôleurs (validations côté client et serveur, menu, etc.) |  | / 15 |
| Gestion du code source, documentation |  | / 10 |
| Qualité du français, usage des termes techniques |  | / 10 |
| Total |  | / 100 |

| **Grille de correction: TP 1** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Composantes de la tâche et critères d’évaluation** | **Très bien** | **Bien** | **Passable** | **Insuffisant** | **Incomplet** | **Note** |
| Qualité du code et fonctionnalités   * Le code compile * L'application est utilisable * Les fonctionnalités demandées sont fonctionnelles | L'application ne présente aucune erreur de compilation ni avertissements  L'application peut être utilisée sans générer d'erreurs à l'exécution  Toutes les fonctionnalités sont présentes telles que demandées. | L'application ne présente aucune erreur de compilation mais certains avertissements ne sont pas corrigés.  L'application peut être utilisée mais des erreurs se produisent rarement lors de l'exécution  Toutes les fonctionnalités sont présentes mais certains détails ont été omis | L'application présente des erreurs de compilation qui sont facilement corrigées  L'application peut être utilisée mais des erreurs se produisent à l'occasion lors de l'exécution  Plusieurs fonctionnalités sont présentes mais des détails significatifs ont été omis. | L'application présente des erreurs de compilation majeures  Au moins une fonctionnalité de l'application ne peut être utilisée en raison des erreurs à l'exécution.  Au moins une fonctionnalité n'est pas présente. | L'application présente des erreurs de compilation majeures  Plusieurs fonctionnalités de l'application ne peuvent être utilisées en raison des erreurs à l'exécution.  Plusieurs fonctionnalités demandées ne sont pas présentes. | /20 |
| 20-17 | 16-13 | 12 | 11-6 | 5-0 |
| Commentaires : | | | | | | |
| Mise en place d’une application respectant le patron MVC   * DataLayers * Type de projets * Repository Patterns | L’application est répartie en plusieurs projets utilisant le type adéquat et correctement identifiés.  L’application met en place le Repository Patterns.  L’application comporte un structure d’*Areas* correctement implémentée.  L'étudiant démontre une très bonne maîtrise des bonnes pratiques. | L’application est répartie en plusieurs projets utilisant le type adéquat et sont plus ou moins identifiés correctement.  L’application met en place le Repository Patterns.  L’application comporte une structure d’*Areas* correctement implémentée sauf quelques erreurs mineures.  L'étudiant démontre une bonne maîtrise des bonnes pratiques. | L’application est répartie en plusieurs projets utilisant le type adéquat et sont plus ou moins identifiés correctement.  L’application ne met pas en place le Repository Patterns correctement.  L’application comporte une structure d’*Areas* avec plusieurs erreurs.  L'étudiant démontre une maîtrise minimum des bonnes pratiques. | L’application est répartie en plusieurs projets ou ceux-ci n’utilisant pas le type adéquat et ne sont pas correctement identifiés correctement.  L’application ne met pas en place le Repository Patterns correctement.  L’application ne comporte pas une structure d’*Areas* ou elle n’est pas correctement implémentée.  L'étudiant ne démontre pas une maîtrise suffisante des bonnes pratiques. | L’application n’est pas répartie en plusieurs projets ou ceux-ci n’utilisant pas le type adéquat et ne sont pas correctement identifiés correctement.  L’application ne met pas en place le Repository Patterns.    L’application ne comporte pas une structure d’*Areas.*  L'étudiant ne démontre pas une maîtrise suffisante des bonnes pratiques. | /15 |
| 15-12 | 11-10 | 9 | 8-5 | 4-0 |
| Commentaires : | | | | | | |
| Modèle de classe   * Classes * Annotations * Relations * Validations complexes personnalisées | Le modèle de données contient toutes les classes et attributs requis.  Les laisons entre les classes reflète les besoins d’affaires.  Les annotations et les validations personnalisées sont complètes et correspondent aux règles d’affaires.  L'étudiant démontre une très bonne maîtrise de la conception de modèle de données | Le modèle de données contient toutes les classes et attributs requis.  Les laisons entre les classes reflète les besoins d’affaires.  Les annotations et les validations personnalisées sont presque complètes et correspondent aux règles d’affaires.  L'étudiant démontre une bonne maîtrise de la conception de modèle de données | Le modèle de données contient toutes les classes et attributs requis malgré quelques erreurs.  Les annotations et les validations personnalisées sont incomplètes et/ou ne correspondent pas toutes aux règles d’affaires.  L'étudiant démontre une maîtrise acceptable de la conception de modèle de données | Le modèle de données ne contient pas toutes les classes ou les attributs requis ou contient plusieurs erreurs.  Les annotations et les validations personnalisées sont incomplètes et/ou ne correspondent pas aux règles d’affaires.  L'étudiant ne démontre pas une connaissance minimum du EF Core | Le modèle de données ne contient pas toutes les classes ou les attributs requis et/ou contient plusieurs erreurs majeures.  Les annotations et les validations personnalisées sont incomplètes et/ou ne correspondent pas aux règles d’affaires.  L'étudiant ne démontre pas une connaissance suffisante du EF Core. | /20 |
| 20-17 | 16-13 | 12 | 11-6 | 5-0 |
| Commentaires : | | | | | | |
| Mise en place d’une application respectant le patron MVC   * Controller (task asynchrones) * Views * Validation côté client et côté serveur * Menu | Les controllers implémentent adéquatement les *Actions* GET/POST correspondantes aux *Views.*  Les *Views* demandées sont présentes et adéquatement définies.  L'étudiant démontre une très bonne maîtrise des concepts MVC. | Les controllers implémentent adéquatement les *Actions* GET/POST correspondantes aux *Views* malgré quelques erreurs*.*  Les *Views* demandées sont présentes et adéquatement définies malgré quelques erreurs.  L'étudiant démontre une bonne maîtrise des concepts MVC. | Les controllers implémentent les *Actions* GET/POST correspondantes aux *Views* et comporte plusieurs erreurs*.*  Les *Views* demandées sont présentes et ne sont pas toutes adéquatement définies.  L'étudiant démontre une maîtrise minimum des concepts MVC. | Les controllers n’implémentent pas toutes les *Actions* GET/POST correspondantes aux *Views* ou comporte plusieurs erreurs*.*  Les *Views* demandées ne sont pas toutes présentes ou comportent plusieurs erreurs majeures.  L'étudiant ne démontre pas une maîtrise suffisante des concepts MVC. | Les controllers n’implémentent pas toutes les *Actions* GET/POST correspondantes aux *Views* et/ou comporte plusieurs erreurs*.*  Les *Views* demandées ne sont pas toutes présentes et/ou comportent plusieurs erreurs majeures.  L'étudiant ne démontre pas une maîtrise suffisante des concepts MVC. |  |
|  | 15-12 | 11-8 | 9 | 8-5 | 4-0 | /15 |
| Commentaires : | | | | | | |
| Gestion de code source et documentation   * *Commit* documentés * Branches demandées * Identification des objets/fichiers/branches   Le suivi dans Azure DevOps   * Avancement des US * Documentation * Référence entre les récits et les branches. | La documentation accompagnant les *commit* est claire et adéquate et les branches sont correctement effectuées.  Les objets, fichiers et branches sont tous correctement identifiés.  Le suivi dans DevOps (en cours de projet) est représentatif de la réalité et la répartition des sous-tâches est équitable.  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 et branches sont tous correctement identifiés.  Le suivi dans DevOps (en cours de projet) est plutôt représentatif de la réalité et la répartition des sous-tâches est équitable.  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.  Les objets, fichiers et branches sont presque tous correctement identifiés.  Le suivi dans DevOps (en cours de projet) est moyennement représentatif de la réalité et la répartition des sous-tâches est équitable.  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.  Les objets, fichiers et branches sont presque tous correctement identifiés.  Le suivi dans DevOps (en cours de projet) n’est pas représentatif de la réalité ou la répartition des sous-tâches est inéquitable.  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.  Les objets, fichiers et branches sont incorrectement identifiés.  Le suivi dans DevOps (en cours de projet) n’est pas représentatif de la réalité ou absent, et/ou la répartition des sous-tâches est inéquitable.  L'étudiant ne démontre pas une connaissance suffisante du gestionnaire de code source. | /10 |
| 10-9 | 8-7 | 6 | 5-3 | 2-0 |
| Commentaires : | | | | | | |

| **Composantes de la tâche et critères d’évaluation** | **Très bien** | **Bien** | **Passable** | **Insuffisant** | **Incomplet** | **Note** |
| --- | --- | --- | --- | --- | --- | --- |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Travail collaboratif**  Mise en pratique du buddy programming  Révisions de codes avec des « pull requests » et des commentaires | L’étudiant démontre une excellente compréhension et maîtrise les techniques de travail collaboratif  Les « pull requests » sont toujours utilisés et bien commentés. | L’étudiant démontre une bonne compréhension et utilise bien les techniques de travail collaboratif  Les « pull requests » sont utilisés et parfois commentés. | L’étudiant démontre une compréhension correcte et utilise parfois les techniques de travail collaboratif  Les « pull requests » sont parfois et parfois commentés. | L’étudiant démontre une piètre compréhension et utilise peu les techniques de travail collaboratif  Les « pull requests » sont peu utilisés et peu commentés | Les techniques de travail collaboratif ne sont pas utilisées  Les « pull requests » ne sont pas utilisés | /10 |
|  | 10-9 | 8-7 | 6 | 5-3 | 2-0 |
| Commentaires : | | | | | | |

| **Composantes de la tâche et critères d’évaluation** | **Très bien** | **Bien** | **Passable** | **Insuffisant** | **Incomplet** | **Note** |
| --- | --- | --- | --- | --- | --- | --- |
| Qualité du français écrit  Termes techniques  Code orthographique et grammatical | L’étudiant démontre une très bonne maîtrise du français écrit. | L’étudiant démontre une bonne maîtrise du français écrit. | L’étudiant démontre une maîtrise minimum du français écrit. | L’étudiant ne démontre pas une maîtrise suffisante du français écrit. | L’étudiant ne démontre pas une maîtrise minimum du français écrit. |  |
| 10-9 | 8-7 | 6 | 5-3 | 2-0 | /10 |
| Commentaires : | | | | | | |

| **Grille de correction: TP 1** | |
| --- | --- |
| Tâche : Application CEGES. Correction négative | |
| 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** | **\_\_/100** |