Travail Demandé

On vous demande de réaliser l'analyse et l'implémentation de l'application JUNIOR ENTREPRISE avec au minimum 2 modules parmi :

- Création, édition et validation des conventions,
- Suivi des étudiants,
- Gestion et remboursement des frais,
- Facturation,
- Indemnisation des étudiants

L'analyse sera effectuée en UML et l'implémentation avec un langage de votre choix (e.g., Java/Oracle, PHP/MySQL, etc.). On veillera à bien scinder les 3 couches : Présentation, Métier et Persistance des données, de manière à permettre un changement d'interface ou de BD, sans avoir à modifier tout l'ensemble.
Couche Présentation : elle concerne toutes les interactions avec les utilisateurs.
Couche Métier : elle concerne tous les processus métiers à effectuer pour les différents modules.
Couche Données : stockage des données persistantes dans des tables relationnelles ORACLE (ou MySQL). On veillera à faire une classe de connexion indépendante avec un fichier de propriétés pour les paramètres de connexion.

Au niveau de la modélisation, il faudra décrire toutes les classes participant à l'application.
Et bien sûr, toutes les gestions usuelles seront également à implémenter : création/modification/ suppression des différentes entités de l'application (étudiants, frais, conventions, etc.).

L'annexe 1 propose un guide méthodologique qui devrait vous permettre de mieux cadrer le déroulement de votre projet. En Annexes 2 et 3 on spécifie clairement ce qu'on attend des deux rapports à rendre (en analyse/ conception puis en programmation).

ANNEXE 1 – GUIDE METHODOLOGIQUE

- L'utilisation d'un AGL est obligatoire pour la génération automatique de code et la rétro conception (reverse engineering), et l'environnement NetBeans est fortement conseillé pour la programmation Java (et le développement web).
- Le module a une durée de 44h, l'analyse requiert environ 3 séances (12h), la programmation 7 séances (28h). Les dernières 4h sont consacrées à l'évaluation du travail réalisé par chaque binôme sous la forme de démonstrations/exposés.
- Le travail concerne deux parties bien distinctes : l'analyse/conception et l'implémentation de l'application. Vous devrez donc **rendre deux dossiers distincts**, l'un concernant l'analyse/conception, et l'autre la programmation. Ce dernier comportera en outre un bilan et une réflexion a posteriori du travail d'analyse effectué, avec mention des *points forts* : qu'est-ce qui a permis d'aller plus vite, de faciliter la construction des classes, etc. et des *points faibles* : erreurs commises, incomplétudes, éléments non utilisés.
- Pour éviter toute erreur de débutants (écrasement de fichiers) et faciliter le développement à plusieurs, il est conseillé d'utiliser un système de gestion des versions type CVS (*Concurrent Versions System*). **Dans NetBeans, CVS est intégré à l'IDE et on peut l'utiliser sans avoir à initialiser quoique ce soit.** Si vous souhaitez connaître un CVS autonome, voir par ex. *Subversion* (http://subversion.tigris.org/).
- On veillera à constituer un **planning prévisionnel des tâches** pour l'étape programmation, avec durée et affectation aux membres du binôme. Outre les tâches de codage des différentes parties, vous penserez aux tâches annexes comme la formation à un nouvel outil (ex. CVS), la création des tables, les tests, la préparation de la démonstration, etc.

Vous noterez ensuite le temps passé à réaliser chacune des activités pour pouvoir, en fin d'implémentation, **construire un planning réel** et comparer avec le prévisionnel.

ANNEXE 2 – RAPPORT D'ANALYSE

Le dossier d'analyse/conception est un dossier complet. Il doit comprendre tous les éléments explicatifs afin de ressembler à un **rapport professionnel** (surtout ne pas reprendre l'énoncé fourni tel quel !). Vous vous aiderez pour cela de la génération automatique de rapport d'analyse à partir de l'AGL (à paramétrer ! tout rapport de plus de 50 pages ne sera pas lu). Il comportera au minimum :

- La présentation de l'application complétée et présentée selon vos soins : vous détaillerez ce que vous avez compris à partir de l'énoncé et des explications que l'on vous aura fournies en tant que "client".
- Les **Packages** que vous avez identifiés et leurs relations de dépendance;

Puis par package :

- le **Diagramme des Cas d'utilisation** avec les principaux acteurs et les rôles que ces acteurs vont jouer en interaction avec le système;
- Puis, par **cas d'utilisation** jugé intéressant :
  - un **Diagramme d'Activités** illustrant le fonctionnement du Cas ;
  - les **scénarii** associés aux différents cas de figure (écrits dans les propriétés du Cas dans l'AGL, onglet Spécifications), et éventuellement un *diagramme d'activités* montrant l'enchaînement des scénarios ;
  - les **diagrammes de séquence**s (ou de communication) intéressants qui synthétisent des scénarii;
- Le **diagramme de classes du package**, avec explicitation des attributs et opérations (arguments, types, donner éventuellement les diagrammes d'activités des opérations complexes).

Le dossier incorpore aussi une **partie IHM** : vous fournirez les dessins d'écran et les schémas d'enchaînement des fenêtres, pour illustrer le fonctionnement choisi. On donnera également:

- les Diagrammes de Classes de Conception (enrichis des éléments de conception) ;
- le schéma de la BD et le dictionnaire de données ;
- le script de création des tables ;
- les squelettes des classes Java/MySQL issues de la génération automatique de code.

TRES IMPORTANT : les diagrammes doivent être **intégrés** dans le corps du rapport (ne pas les mettre en Annexe !) et **complétés d'explications textuelles** : ils participent à la compréhension générale de l'analyse autant que le texte qui est là pour guider la lecture.

ANNEXE 3 –RAPPORT DE PROGRAMMATION

La programmation donnera lieu à une évaluation qui tient compte de la réalisation (démonstration) et de son rendu via le dossier de programmation. Ce dernier comportera deux volets : un volet Technique et un volet Gestion de projet.

3.1. Éléments techniques

- **État d'avancement** : ce qui marche, ce qui reste à améliorer, ce qui n'a pas été commencé.
- **Description de l'architecture en 3 couches** : *présentation* (choix pour les IHM, classes frontières), *métier* (classes d'application, classes techniques) et *données* (argumenter les choix pour le stockage des données).
- **Retour à l'analyse** : l'objectif du module étant de développer une application de synthèse Conception & Programmation, il est important de réfléchir à ce lien une fois le codage terminé. Vous donnerez donc les éléments qui vous permettent d'en évaluer les avantages et inconvénients (détails envisagés en analyse qui s'avèrent inutiles ou irréalisables lors du codage, contraintes de codage qui conditionnent l'analyse, modification de l'analyse en cours de codage et justifications…).
- **Points d'amélioration / extensions** : vous énumérez ici les extensions que l'on pourrait envisager pour cette application, compte tenu des idées (ou des limites) qui sont apparues en la développant.
- On pourra aussi faire une **rétro-conception** du code développé et comparer avec le modèle d'analyse qui avait été proposé.

3.2. Éléments de gestion de projet

- Donner les **2 plannings des tâches** (prévisionnel et réel) et effectuer la comparaison. Ce travail devrait vous permettre de mieux chiffrer ensuite les durées des tâches.
- Préciser aussi le **partage des tâches** au sein du binôme (cela peut être mentionné dans le planning réel) ;
- On rendra compte de **l'utilisation d'un système de gestion des versions** du projet, type Subversion ;
- Expliquer le **protocole de tests effectués pour valider l'application**, avec l'utilisation par ex. de JUnit sous NetBeans (démarche, jeux de tests, tests unitaires, tests d'intégration).