

Suivi des étudiants dans une formation à la carte.

L3 Miage, projet de programmation

1 Présentation générale

1.1 Contexte

L'application à réaliser se place dans le contexte d'une Université qui proposerait des **formations** à la carte dans lesquels **les étudiants choisissent les enseignement qu'il veulent suivre**. Les enseignements sont régis par des **notions de prérequis**.

Un étudiant peut décrocher un **diplôme** lorsqu'il a validé tous les enseignements nécessaires pour faire un **parcours** complet.

1.2 Objectif

L'objet de l'application est de **suivre les étudiants** : les inscrire aux enseignements, voir où ils en sont de leur parcours et saisir les enseignement validés.

2 Le Logiciel

Cette section décrit l'application demandée. Cela constitue la base à partir de laquelle des évolutions seront demandées ultérieurement.


Dans un premier temps, nous définissons les concepts qui seront manipulés par l'application.

Une année universitaire est à cheval sur deux années civiles (par ex. 2020-2021). Elle est divisée en deux semestre : le semestre impair (début de l'année universitaire) et le semestre pair (fin de l'année universitaire)

2.1 Les formations et les UE

Les **formations sont organisées par mention puis sous-divisées en parcours** (par exemple mention MIASHS parcours Miage).

Les Unités d'Enseignement (UE) sont les cours proposés par les différentes mentions. **Chaque UE est associée à une mention, à l'exception de certaines UE qui sont des UE d'ouverture. Une UE possède un code d'identification et rapporte des crédits ECT (3 ou 6).**

Dans une formation à la carte, toutes les UE ont potentiellement un ou des prérequis qui sont d'autres UE et qui doivent être validées antérieurement (par exemple on **peut** imaginer que pour suivre "Prog Objet 2" il faut avoir validé "algorithmique" et "Prog Objet 1"). Les UE qui n'ont pas de prérequis peuvent être choisies dès l'arrivée en première année. 

2.2 Les étudiants


Un étudiant est défini par son numéro d'étudiant. Il possède un prénom et un nom et il est inscrit à un **parcours d'une mention**.

Il faut effectuer le suivi pour chaque étudiants des UE qu'il a suivies, en quelle année universitaire il a validé cette UE ainsi que le semestre (pair ou impair) où cette validation a eu lieu. Il faut de plus mémoriser si l'étudiant a

validé ou non l'UE à l'issue de ce semestre. Par exemple un étudiant peut suivre une UE et échouer sur le semestre pair de 2020-2021 puis suivre et valider l'UE le semestre impair de 2021-2022.

Il faut également suivre les UE auxquelles l'étudiant est inscrit pour le semestre en cours.

2.3 Objectifs

À l'ouverture, l'application doit charger les données concernant les mentions, parcours, UE et la liste des étudiants à partir de fichiers .csv. Ces données doivent par défaut être chargées à partir d'un répertoire Données de l'application. On vous demande toutefois de proposer également à l'utilisateur de charger les données à partir d'un répertoire de son choix. 

L'application doit permettre de choisir un rôle parmi : directeur d'étude, secrétariat pédagogique, bureau des examens.

L'application propose alors plusieurs fonctionnalités dépendant du rôle choisi. L'ordre de présentation des fonctionnalités dans ce sujet représente un ordre de priorité de développement.

2.3.1 Visualisation et conseil

Cette fonctionnalité est proposée aux directeurs d'étude. Ils accèdent à une liste des étudiants, triables par mention et parcours. Ils peuvent alors sélectionner un étudiant. Les données relatives aux UE suivies par cet étudiants (par le passé ou en cours) sont alors chargées et l'utilisateur accède à un écran qui liste :

- les UE que l'étudiant a déjà validé
- les UE que l'étudiant suit actuellement (potentiellement vide en inter-semestre)
- les UE pour lesquels l'étudiant a les prérequis. Cette liste doit pouvoir être filtrée par mention et parcours et doit également permettre de voir les UE d'ouverture.

L'objet de cette fonctionnalité est pour le directeur d'étude de pouvoir avoir accès aux données lors d'un entretien avec l'étudiant afin de pouvoir le conseiller sur ses choix d'UE futurs.

2.3.2 Inscriptions

Cette fonctionnalité est proposée au secrétariat pédagogique. Elle donne accès à une liste d'étudiants. Lorsqu'un étudiant est sélectionné, ces données sont chargées et la liste des UE pour lequel l'étudiant a les prérequis est alors affichée. Il faut pouvoir filtrer et trier cette liste comme pour la fonctionnalité précédente.

Le secrétariat doit alors pouvoir inscrire l'étudiant à des UE puis sauvegarder ou non les modifications effectuées.

L'objet de cette fonctionnalité est d'inscrire efficacement les étudiants dans les UE qu'ils auront choisies sous réserve qu'ils en aient les prérequis.

2.3.3 Saisie des résultats

Cette fonctionnalité est proposée au bureau des examens. Elle donne accès à une liste des UE (là encore, triable et filtrable). Lorsqu'une UE est sélectionnée, tous les étudiants inscrits à l'UE apparaissent et il est possible de saisir si l'étudiant a validé l'UE ou échoué à l'UE. Il est ensuite possible de sauvegarder les modifications.

Le chargement des données lors du choix d'une UE est laissé libre. Dans un premier temps il peut s'agir d'un parcours des informations de tous les étudiants. Dans un deuxième temps il peut être utile d'avoir un fichier qui liste tous les étudiants inscrits à une UE.

3 Consignes

L'application devra être réalisée en Java. Les données seront chargées à partir de fichiers .csv.

3.1 Interface et ergonomie

Il est important que votre logiciel propose une interface lisible et efficace. L'ergonomie sera particulièrement importante pour les traitements les plus fréquents et répétitifs (inscription des étudiants, saisies des résultats). Le nombre d'actions nécessaire pour ces traitements devra être le plus faible possible.

Il est fortement recommandé de faire assez tôt des prototypes de l'interface.

3.2 Conception, Code et documentation

Il est recommandé de prévoir une conception modulaire de votre projet. Cela sera un atout lors de l'implémentation des évolutions du logiciel. Utilisez les interfaces et classe abstraites proposées par le langage.

L'utilisation de bibliothèque externes est autorisée. Il n'est pas judicieux de redévelopper des fonctionnalités qui existent déjà (par exemple la manipulation de fichiers .csv).

Il vous est également demandé de produire une documentation sous forme de javadoc. Écrivez les commentaires javadoc à chaque fois que vous ajoutez une classe ou une fonctionnalité.

3.3 Travail collaboratif

Il vous est demandé d'utiliser GIT pour ce projet. Les autres aspects du travail collaboratif sont laissés libres.

3.4 Rendez-vous intermédiaire

Un rendez-vous intermédiaire est prévu pour ce projet. Vous devrez y faire une démonstration de la première version du logiciel. Suite à cette démonstration, il sera demandé des évolutions qui devront intégrer la version finale du logiciel.

Vous devrez déposer des documents la veille du rendez-vous. La nature de ces documents vous sera précisée ultérieurement dans un document qui précisera également l'ordre de passage.

3.5 Rendu

À l'issue du projet vous ferez une démonstration du logiciel qu'il faudra soigneusement préparer. Vous devrez également déposer les éléments suivants :

- tout votre code (pas de projet Eclipse ou Netbeans)
- des fichiers de données permettant de tester extensivement les fonctionnalités de l'application.
- la javadoc de votre application
- un diagramme de classes.
- un fichier .jar permettant le déploiement de l'application.

La livraison d'un fichier jar fonctionnel est essentielle. Ne négligez pas cet aspect qui n'est pas trivial et doit être envisagé assez tôt dans le développement.

3.6 Remarques

Un canal TEAMS a été créé pour ce projet. En particulier il vous est demandé d'y mentionner votre groupe de projet si ce n'est déjà fait. Surveillez ce canal, des informations sur le projet pourront y être postées.

N'hésitez pas à poser des questions sur les aspects du sujet qui ne seraient pas clair. Pour JavaFX vous pouvez contacter Racim Fahssi et pour Maven et Git, Patrice Torguet et Cédric Teyssié.