

# TP Projet « Kaddem »

## TP étude de cas Projet « Kaddem »

UP ASI  
Bureau E204

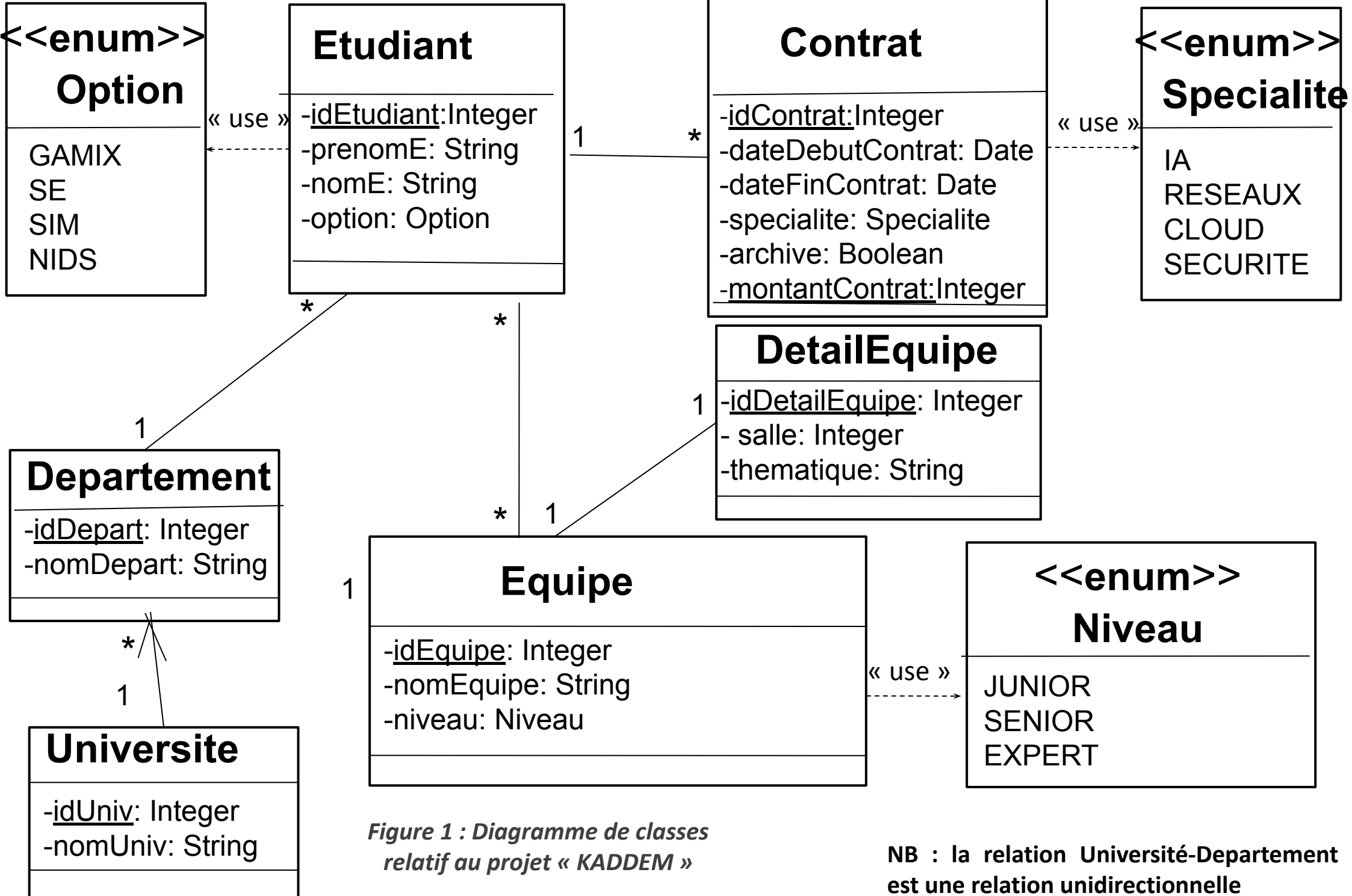


# TP Projet « Kaddem »

- **« Kaddem », c'est quoi?:** C'est un projet qui vise à encourager les jeunes étudiants à améliorer leurs compétences professionnelles notamment dans le cadre des nouvelles tendances du monde de l'informatique.
- **Objectif:** Dans le but de préparer les étudiants aux nouvelles exigences du marché d'emploi, nous proposons de travailler sur le projet « KADDEM ».
- **Donc concrètement, qu'est-ce-qu'on veut faire?**
- On désire créer une application de gestion des contrats d'étudiants dans le cadre du projet « Kaddem ».
- Ce projet définit dans chaque département des universités adhérentes un programme de répartition des étudiants par équipes.
- Chaque équipe aura l'un des niveaux (junior/senior ou expert) dans l'une des spécialités suivantes: IA, réseaux, sécurité, cloud.
- Ce projet propose donc aux étudiants des contrats selon la spécialité et le niveau.

# TP Projet « Kaddem »

- Chaque étudiant obtient un contrat s'il adhère à une équipe,  
Dans le cas où il sera affecté à plusieurs équipes, il aura un contrat pour chaque activité avec une équipe.
- Le contrat constitue simplement un engagement moral entre l'étudiant et son université □ il doit donc honorer son engagement en participant activement à l'évolution de son équipe.



# Travail à faire

## **Partie 1 Spring Data JPA – Première entité**

- Créer les entités se trouvant dans le diagramme des classes (sans les associations) et vérifier qu'ils ont été ajoutés avec succès dans la base de données.

# Travail à faire

## **Partie 2 Spring Data JPA – Le mapping des différentes associations**

- Supprimer les tables existantes dans la base de données.
- Créer les associations entre les différentes entités.
- Générer la base de données de nouveau et vérifier que le nombre de tables créées est correct.

# Travail à faire

## Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités** indiquées dans les slides suivants en respectant les signatures suivantes

### Entité Etudiant

**List<Etudiant>** retrieveAllEtudiants();

**Etudiant** addEtudiant (Etudiant e);

**Etudiant** updateEtudiant (Etudiant e);

**Etudiant** retrieveEtudiant(Integer idEtudiant);

**void** removeEtudiant(Integer idEtudiant);

# Travail à faire

## Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités** indiqués dans les slides suivants en respectant les signatures suivantes

### Entité Contrat

**List<Contrat>** retrieveAllContrats();

**Contrat** updateContrat (Contrat ce);

**Contrat** addContrat (Contrat ce);

**Contrat** retrieveContrat (Integer idContrat);

**void** removeContrat(Integer idContrat);



# Travail à faire

## Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

### Entité Département

**List<Département>** retrieveAllDepartements();

**Département** addDépartement (Département d);

**Département** updateDépartement (Département d);

**Département** retrieveDépartement (Integer idDepart);

# Travail à faire

## Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités** indiqués dans les slides suivants en respectant les signatures suivantes

### Entité Equipe

**List<Equipe>** retrieveAllEquipes();

**Equipe** addEquipe(Equipe e); // ajouter l'équipe avec son détail

**Equipe** updateEquipe (Equipe e);

**Equipe** retrieveEquipe (Integer idEquipe);

# Travail à faire

## Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

### Entité Université

**List<Université>** retrieveAllUniversites();

**Université** addUniversite (Université u);

**Université** updateUniversite (Université u);

**Université** retrieveUniversite (Integer idUniversite);

**NB:** Pour l'ajout de l'universite, il faut créer en même temps les départements de cette université (l'entité associée Departement )

# Travail à faire

## **Partie 4 Spring MVC**

Exposer les services implémentés dans la partie 3 avec Postman et/ou Swagger pour les tester.

# Travail à faire

## **Partie 5 : Services avancés**

On désire affecter un étudiant à un département.  
Créer un service permettant l'assignation d'un étudiant à un département et exposer le en respectant la signature suivante :

**public void assignEtudiantToDepartement (Integer etudiantId, Integer departementId) ;**

# Travail à faire

## **Partie 5 : Services avancés**

On désire affecter une université à un département.

Créer un service permettant l'assignation d'une université à un département et exposer le en respectant la signature suivante :

```
public void assignUniversiteToDepartement(Integer idUniversite, Integer  
idDepartement)
```

# Travail à faire

## **Partie 5 : Services avancés**

On désire ajouter et affecter un étudiant à une équipe et un contrat en utilisant une seule méthode.

Créer un service permettant l'affectation d'un étudiant à une équipe et un contrat et exposer le en respectant la signature suivante :

**Etudiant addAndAssignEtudiantToEquipeAndContract(Etudiant e, Integer idContrat, Integer idEquipe);**

# Travail à faire

## **Partie 5 : Services avancés**

On désire affecter un contrat à un étudiant en vérifiant que l'étudiant n'a pas dépassé la limite autorisée de 5 contrats actifs.

Créer le service adéquat et exposer le en respectant la signature suivante :

**Contrat affectContratToEtudiant (Contrat ce, nomE:String,prenomE:String);**



# Travail à faire

## **Partie 5 : Services avancés**

On souhaite récupérer les étudiants d'un département donné.  
Créer le service adéquat en respectant la signature suivante :

**List<Etudiant> getEtudiantsByDepartement (Integer idDepartement);**

# Travail à faire

## **Partie 5 : Services avancés**

On souhaite récupérer le ou les départements d'une université donnée.  
Créer le service adéquat en respectant la signature suivante :

**public List<Departement> retrieveDepartementsByUniversite(Integer idUniversite)**

# Travail à faire

## Partie 5 : Services avancés

Nous souhaitons calculer le montant à payer par les universités entre deux dates triés par spécialité.

Créer un service permettant de faire le calcul en respectant la signature suivante et le type de l'affichage ci dessous :

**public float getChiffreAffaireEntreDeuxDate(Date startDate, Date endDate)**

**PS :** le montant à payer correspond à la somme des règlements (montantContrat) entre deux dates correspondant aux contrats non archivés.

Le règlement:

Pour un contrat dont la spécialité est IA: 300Dt/mois

Pour un contrat dont la spécialité est RESEAUX: 350Dt/mois

Pour un contrat dont la spécialité est CLOUD: 400Dt/mois

Pour un contrat dont la spécialité est SECURITE: 450Dt/mois

# Travail à faire

## **Partie 5 : Services avancés**

Nous souhaitons calculer le nombre de contrats encore valides entre deux dates.  
Créer un service permettant de faire le calcul en respectant la signature suivante :

**Integer nbContratsValides(Date startDate, Date endDate);**

Un contrat est considéré comme valide s'il n'a pas encore été archivé

# Travail à faire

## **Partie 6 : Spring Scheduler**

Nous souhaitons créer un service schedulé ( programmé automatiquement) permettant d'avertir le responsable de la gestion des contrats dont la date de fin est prévue pour les 15 prochains jours afin de vérifier s'il faut renouveler ou mettre fin au contrat de l'étudiant concerné. Le résultat permet d'afficher le contrat en question et les informations associées (dateFin, spécialité,étudiant concerné)

Ce meme service nous permet d'afficher les contrats concernés tous les jours à 13h et de changer l'état du contrat à "archive=true" une fois arrivé à la date indiquée en respectant la signature suivante :

**String retrieveAndUpdateStatusContrat();**

**NB:** Pour des raisons de test, vous pouvez modifier l'horaire selon l'heure affichée sur votre machine. Le message sera affiché simplement sur console.

# Travail à faire

## **Partie 6 : Spring Scheduler**

Nous souhaitons faire passer des équipes au niveau supérieur (junior → senior ou senior → expert) selon les conditions suivantes:  
si l'équipe concernée a 3 membres ou plus ayant dépassé 1 an avec contrat et si le niveau de l'équipe en question est junior ou bien senior.

Créer un service permettant de modifier le niveau des équipes selon les conditions indiquées ci-dessus en respectant la signature suivante :

**void faireEvoluerEquipes()**

# TP Projet « Kaddem »

Si vous avez des questions, n'hésitez pas à nous contacter :

**Département Informatique**  
**UP ASI (Architectures des Systèmes d'Information)**

Bureau E204