DOCUMENT DE CONCEPTION

PLAN

I-OBJECTIFS ET DESCRIPTION DU PROJET

1.Description

Le domaine de l'éducation universitaire est un vaste réseau d'interactions et de processus, allant des admissions à l'évaluation des étudiants. Historiquement, les inscriptions et préinscriptions ont toujours été effectuées en personne, nécessitant que l'étudiant récupère sa fiche d'inscription ou de préinscription ainsi que d'autres documents certifiés, pour ensuite les déposer à la scolarité ou dans le bureau administratif concerné. Cela entraîne souvent de longues files d'attente, car de nombreux étudiants cherchent à s'inscrire ou à se préinscrire en même temps et doivent déposer leur dossier dans le même bureau. De plus, cela surcharge le personnel administratif qui doit gérer tous ces dossiers. Pour résoudre ce problème, nous proposons une application de gestion universitaire qui permettra de numériser entièrement le processus d'inscription et de préinscription. Ainsi, l'étudiant pourra envoyer son dossier d'inscription ou de préinscription en ligne, directement via la plateforme, facilitant ainsi le traitement par le personnel administratif.

2.Méthodologie

Pour gérer efficacement ces interactions, un système modulaire, évolutif et fiable est requis. Et donc, l'objectif est de mettre en place une architecture de microservices qui facilite le développement, le déploiement et la gestion des diverses fonctionnalités du système universitaire.

3.Outils et Techniques

- *Outils des Microservices :*

  - **API Gateway** :Sert de point d'entrée pour toutes les demandes externes et gère l'authentification, la limitation du débit et le routage vers les microservices appropriés.
  - **Kafka ou RabbitMQ** :pour des événements tels que les notifications, les annonces et la mise à jour des données.
  - **Circuit Breaker :** Prévenir la cascade des défaillances entre les services.
  - **Prometheus et Grafana** :pour la surveillance en temps réel des microservices.
- *Environnement de developpement :* **Système d'exploitation Ubuntu**
- *Editeurs de texte :* **Visual studio code, Intellij idea**
- *langage de programmation* : **Java, Javascript,Node.js,Dart**
- *Framework :* **Flutter(Front-end), Spring boot(Back-end)**

II. ARCHITECTURE DU SYSTÈME

les services de cette plateforme seront :

- **Service d'Admission**

-Gère l'inscription des étudiants, les candidatures et l'enregistrement des pièces
justificatives.

-Base de données : Relationnelle (pour stocker les détails des étudiants et des candidatures).

- **Service des Cours**

Gestion, planification des cours et attribution des ressources.

- Base de données : NoSQL (pour une structuration flexible des cours et des horaires).

- **Service d'Évaluation**

- Examen, notes et évaluations des étudiants.

- Base de données : Relationnelle (pour la précision et l'intégrité des notes).

- **Service des Utilisateurs**

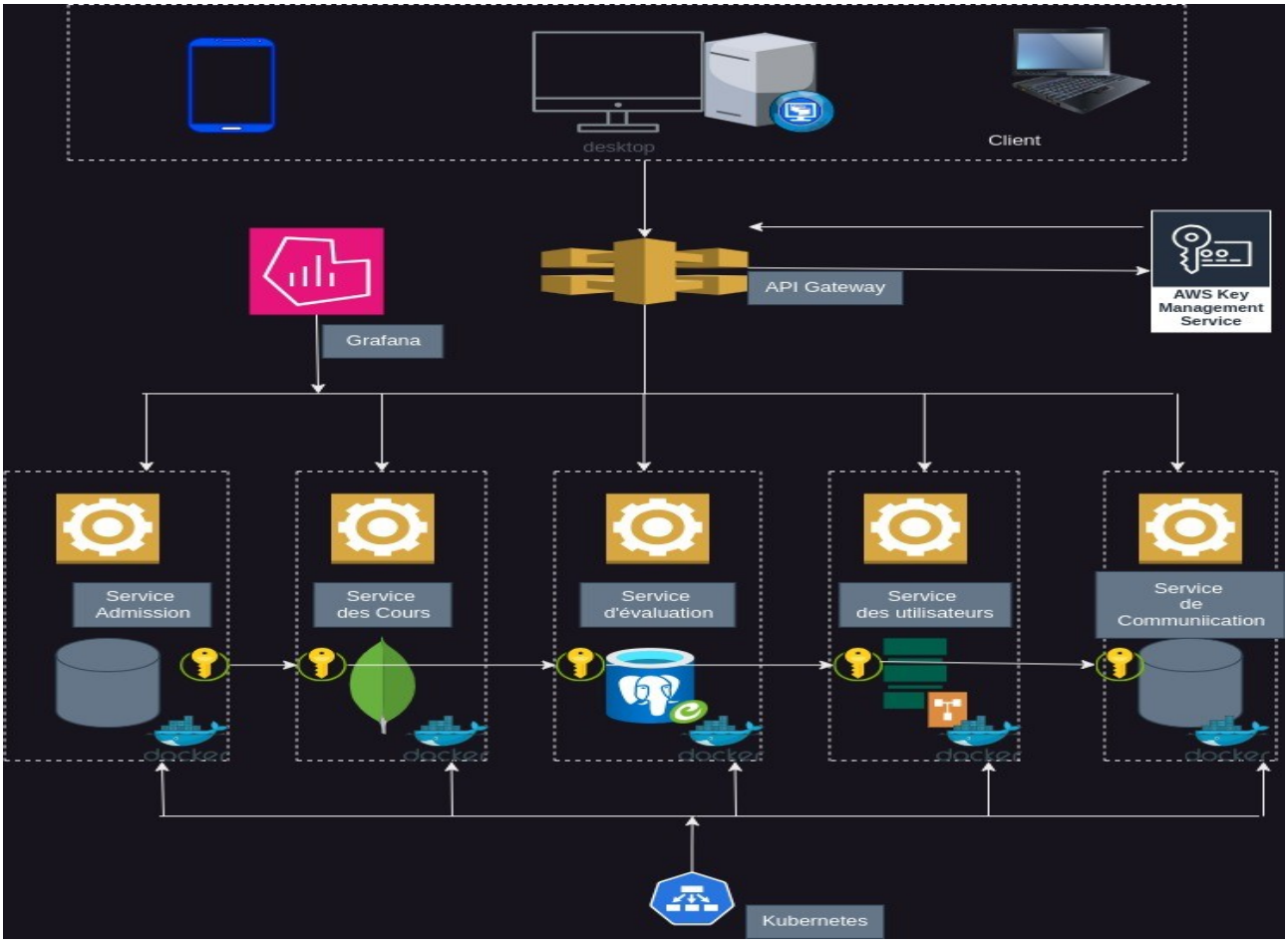
-Profils des étudiants, des enseignants et du personnel administratif.

- Base de données : Relationnelle (pour les relations entre les étudiants, les cours et les
enseignants).

- **Service de Communication**

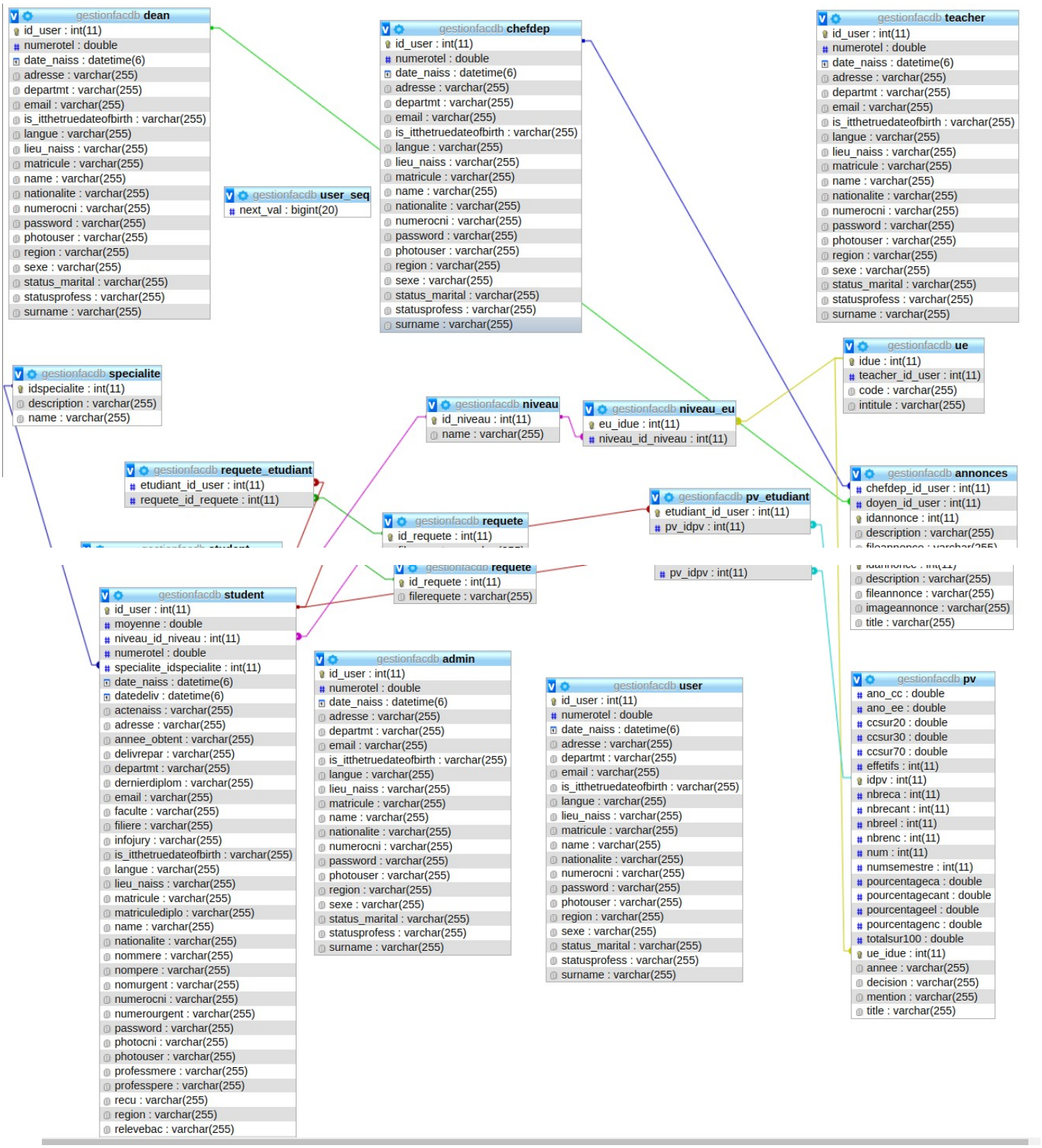
- Portail d'annonces, système de messagerie intégré.

- Base de données : NoSQL (pour la flexibilité des formats de communication).



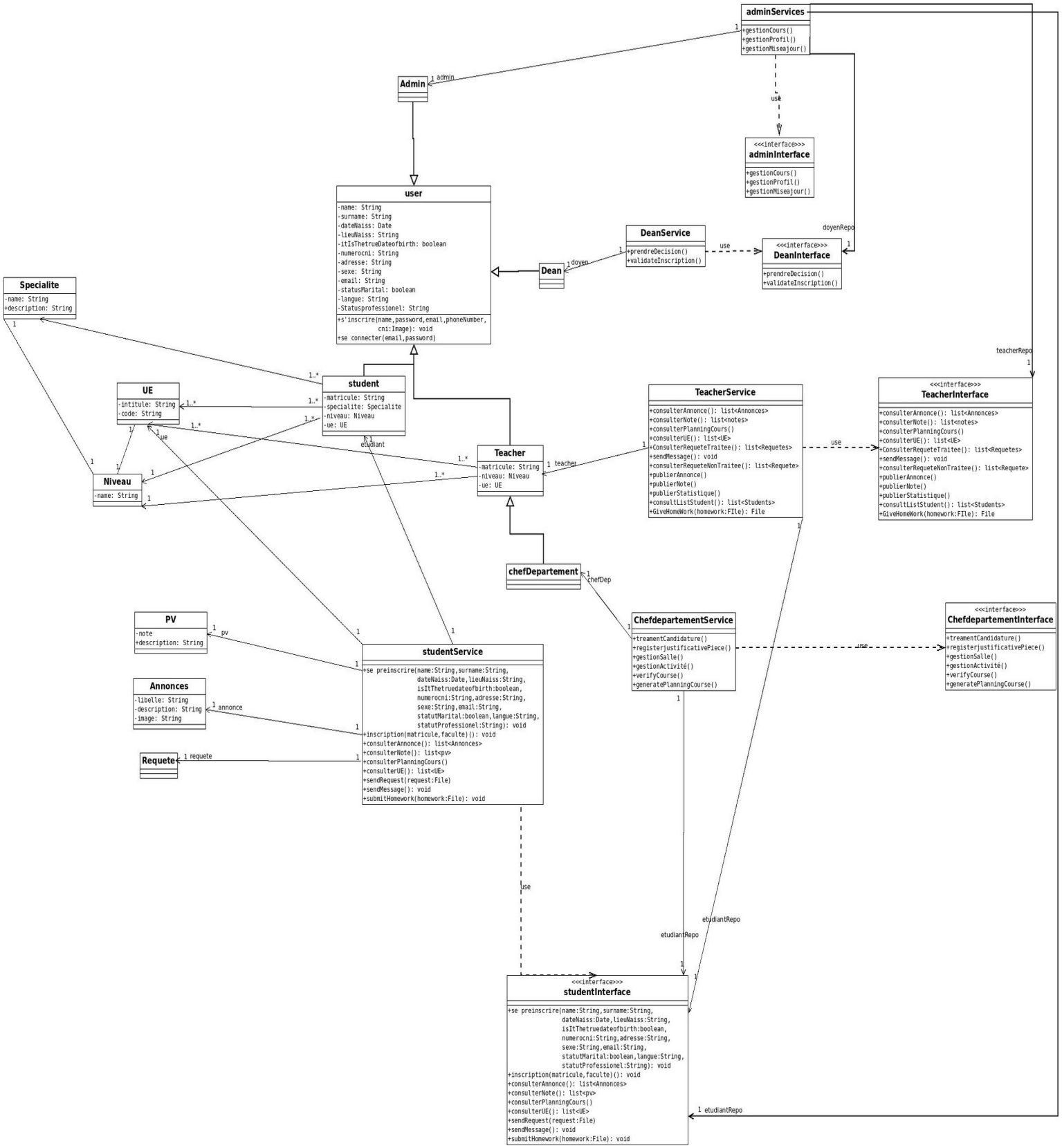
III. CONCEPTION DE DONNÉES

la base de donnée contient les tables suivantes : **dean**(le doyen),**chefdep**(le chefdedépartement),**teacher**(l'enseignant),student,**niveau**,**specialite**,**requet e**, **annonces, PV**(pour les procès-verbaux), a**dmin,**



IV. CONCEPTION DÉTAILLÉE

1.Diagramme de classe



Description et presentation

Dans le diagramme de classe présenté plus haut, nous avons :

- ✔ une classe appelée **user** : qui est une classe abstraite qui contient les informations communes à chaque acteur de l'application,elle transmet ces informations par héritage, elle contient les attributs suivants :
  -name : qui est de type String et qui représente le nom de l'utilisateur
  -surname : qui est de type String et qui représente le prénom de l'utilisateur
  -dateNaiss : qui est de type Date et qui représente la date de naissance de l'utilisateur
  -lieuNaiss : qui est de type String et qui représente le lieu de naissance de l'utilisateur
  -itIsThetrueDateofbirth :qui est de type booléen et qui prendra la valeur true lorsque l'utilisateur aura confirmé que la date de naissance spécifiée est verirable sa date de naissance
  -numerocni : de type String et qui représente le numéro de cni de l'utilisateur
  -photouser : qui est de type String et qui contient la photo de l'utilisateur
  -adresse : de type String et qui représente l'adresse de l'utilisateur
  -sexe : de type String qui represente le genre de l'utilisateur
  -matricule: de type String et qui represente matricule ;
  -email :de type String qui représente l'adresse émail de l'utilisateur
  -password : de type String et qui represente le mot de passe
  -statusMarital :de type String qui représente le statut marital de l'utilisateur(célibataire ou marié(e))
  -langue : de type string qui représente la langue de l'utilisateur(Français,anglais ...)
  -Statusprofess: de type String et qui représente le métier que l'utilisateur fait dans la vie.
  -numerotel:numéro de téléphone
  -nationalite : la nationalité qui est de type String
  -region:la région
  -departmt:le departement
  Ces méthodes sont :
  -se connecter(email,password):pour se connecter
  -s'inscrire():pour s'inscrire
  Ces classes filles qui sont également les acteurs de l'application sont : **Dean,Student,chefdepartement,Teacher et admin.**

- ✔ La classe **Student :** elle contient les attributs et les données relatives à un étudiant,et puisqu'elle hérite de **user**, elle aura tout ses attributs avec en plus les attributs suivant :
  -matricule : qui est un string et qui permet d'identifier de manière unique un étudiant
  -spécialité: qui est un objet de type **Spécialité** qui est une autre classe utilisée par la classe **Student** pour définir le domaine de l'étudiant, sa filière.

-niveau : qui est un objet de type **Niveau** qui est une autre classe utilisée par la classe **Student** pour définir le niveau de l'étudiant

-ue :qui est un objet de type **UE** qui est une autre classe utilisée par la classe **Student** pour définir les matières de l'étudiant.

- ✔ La classe **Teacher** : qui contient les attributs relatifs à un enseignant d'université
- ✔ La classe **chefdepartement** : qui contient les attributs relatifs à un chef de departement dans une université.
- ✔ la classe **Dean** : qui contient les attributs relatifs à un doyen d'université.
- ✔ La classe **admin**:qui contient les données sur le gestionnaire de l'application

A chacune de ces classes principales, est associé une interface contenant les différentes méthodes que ces classes implémenteront et une classe de transition qui leur permet d'interagir avec les interfaces. Et donc :

=>pour la classe **Student** : nous avons **studentService** et **studentInterface** qui est l'interface.

=>pour la classe **Teacher** : nous avons **TeacherService** et **TeacherInterface** qui est l'interface.

=>pour la classe **Dean** : nous avons le **DeanService** et le **DeanInterface** qui est l'interface.

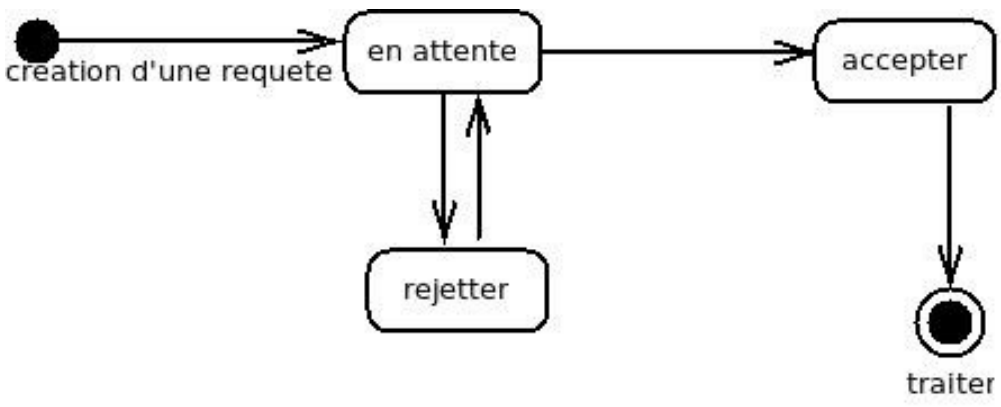
=>pour la classe **chefdepartement** : nous avons **chefdepartementService** et le **chefdepartementInterface** qui est l'interface.

=>pour la classe admin : nous avons **adminService** et le **adminInterface** qui est l'interface.

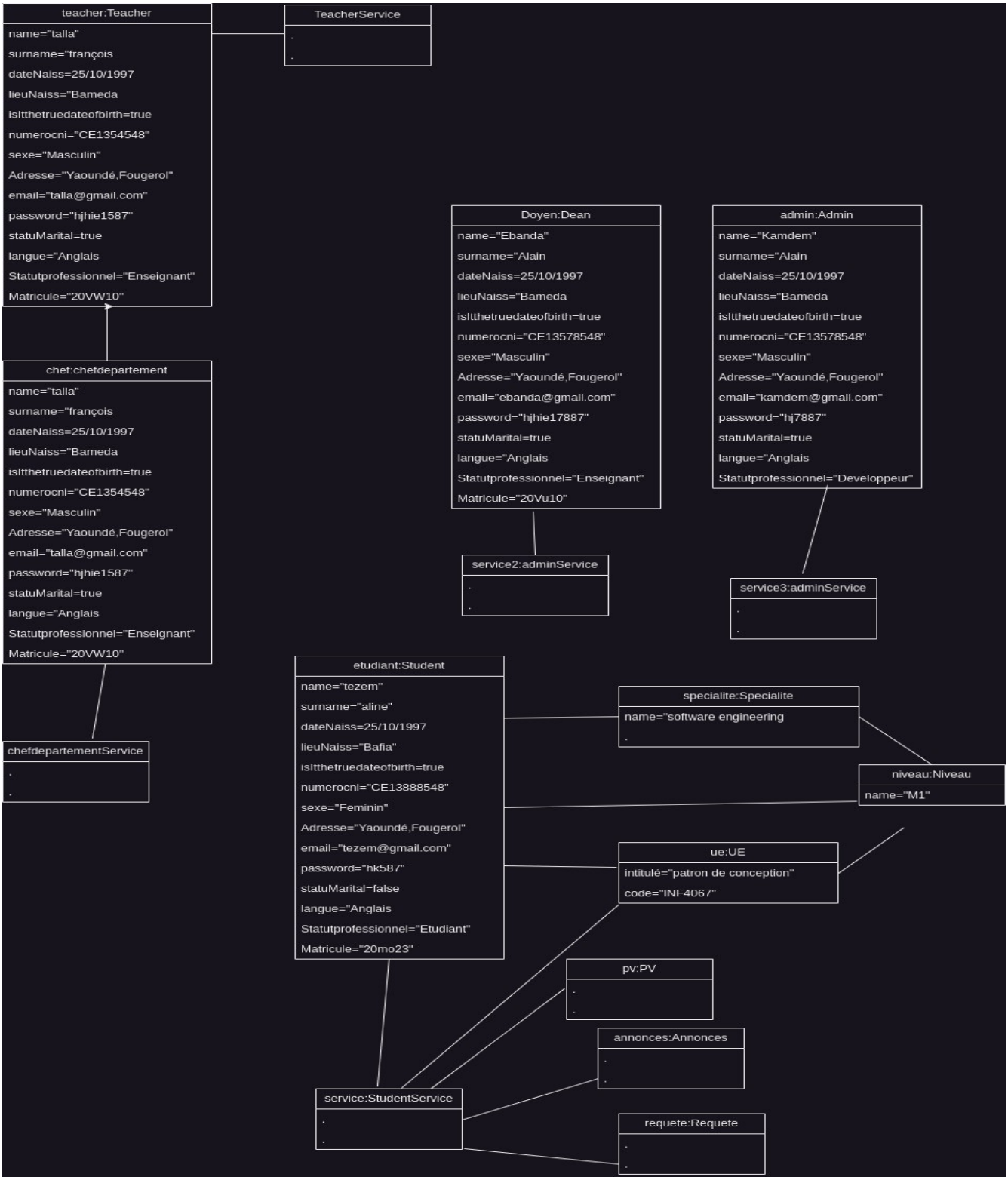
- ✔ La classe **PV** : qui est utilisée par la classe **StudentService** pour avoir les informations concernant les notes des etudiants pour une ue donné
- ✔ La classe **Annonce** : qui est utilisée par la classe **StudentService** pour pouvoir consulter les différentes annonces.
- ✔ La classe **requête :** qui est utilisée par la classe **StudentService** pour pouvoir gérer l'envoie des requêtes par le Student.

2-Diagramme d'état transition pour le traitement d'une requêtes

Lorsque les étudiants rencontrent des problèmes académiques, tels que des problèmes de notes ou de matricule, ils peuvent adresser une requête à l'administration. Cette requête peut être adressée à un enseignant, au chef de département ou au doyen. La requête de l'étudiant peut être en attente, acceptée et traitée, ou rejetée si tous les documents justificatifs nécessaires n'ont pas été fournis. Ce processus est illustré dans le diagramme d'état transition suivant avec les états **en attente**,**rejetter**,**accepter**



3-Diagramme d’objet

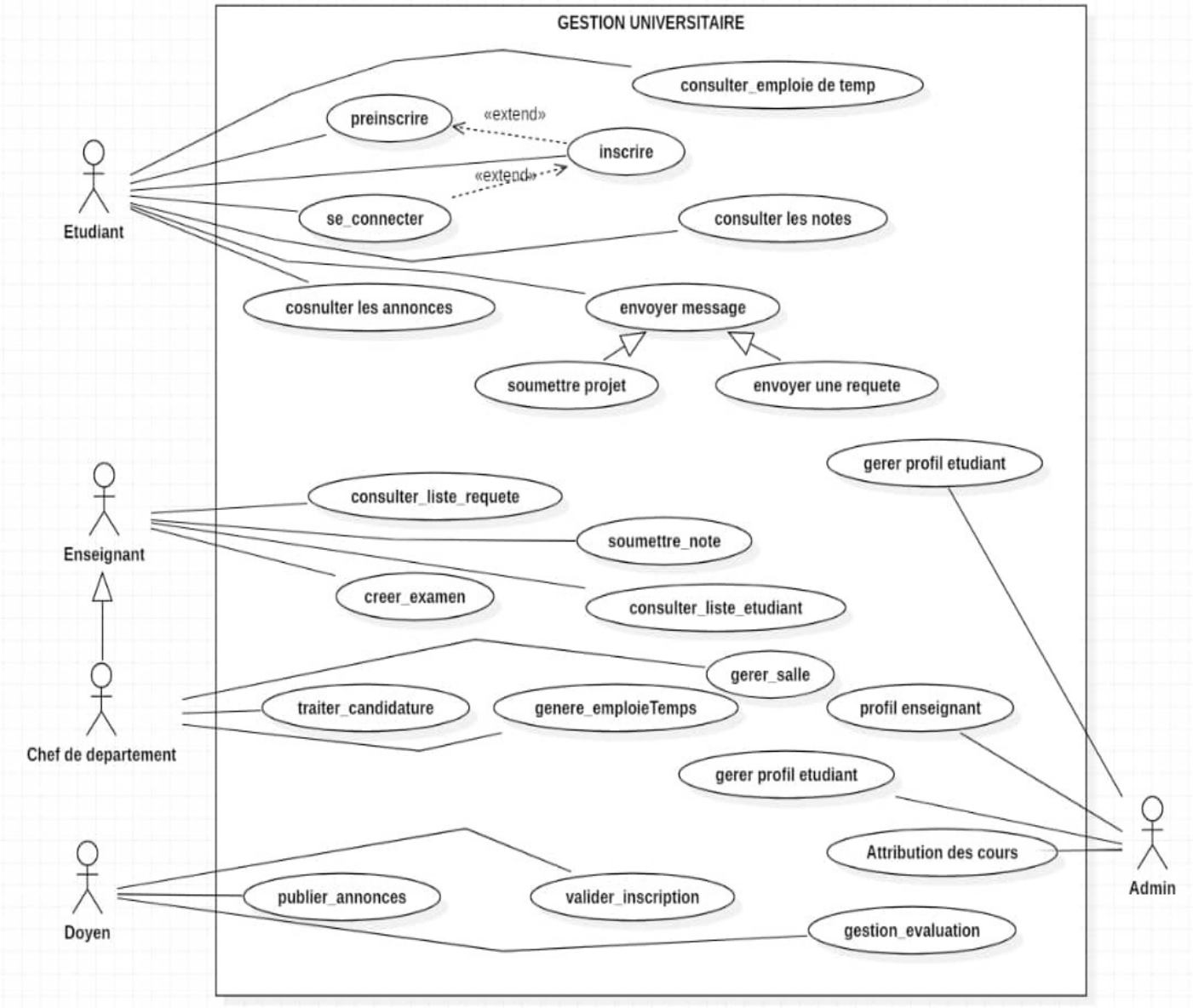


Ce diagramme décrit la communication entre les différentes instances du diagramme de classe.

4-Diagramme de cas d’utilisation

le diagramme de cas d’utilisation est constitué des acteurs suivants :

-**Enseignant** : qui est l’enseignant
-**Doyen :** pour designer le doyen de la faculté
-**Etudiant :** pour designer les étudiants
-**Chef de departement :** pour designer le chef de département
-A**dmin:**qui est le gestionnaire de la plateforme



V. CONCEPTION D’INTERFACE EXTERNE