Ecole Supérieure Polytechnique

Département Génie Informatique

Sujet :

Conception et implémentation d'un système permettant d'alimentation une base de données des enregistrements de cours.

Présenté par:

Sokhna Maimouna Mbacké Mboup

Victorine Sady Ndiaye

Khady Mbacké Diop

Anta Diop Mbaye

Marième Aidara

Année universitaire : 2020/2021

PLAN

Chapitre1 : Présentation Générale

Les années 2020 et 2021 sont marquées par la progression du COVID 19. Les mesures de distanciation sociale devant être prises en compte dans les enseignements, elles ont précipité/forcé l'usage de l'enseignement à distance dans les structures académiques. Cependant, du fait de cette précipitation les outils d'évaluation de la qualité des interactions pendant le cours manquent. A cet effet, une université sollicite des développeurs pour la conception et le développement d'un système permettant d'alimenter une base de données des enregistrements de cours, ainsi que de noter l'évolution de la qualité des séances.

Pour mieux cerner notre sujet, nous allons présenter notre travail en trois parties. Dans le premier chapitre, nous procéderons à la présentation du sujet soumis à notre réflexion. La deuxième partie sera consacrée à l'analyse et à la conception. La dernière partie traitera la réalisation et la mise en œuvre de la solution.

Chapitre2 : Analyse et Conception

I. Diagrammes d'UML

UML (Unified Modeling Language) se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML unifie à la fois les notations et les concepts orientés objet. Il ne s'agit pas d'une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage. UML unifie également les notations nécessaires aux différentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis l'expression de besoin jusqu'au codage. Dans ce cadre, un concept appartenant aux exigences des utilisateurs projette sa réalité dans le modèle de conception et dans le codage. Le fil tendu entre les différentes étapes de construction permet alors de remonter du code aux besoins et d'en comprendre les tenants et les aboutissants. En d'autres termes, on peut retrouver la nécessité d'un bloc de code en se référant à son origine dans le modèle des besoins.

UML 2 s'articule autour de treize types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces types de diagrammes sont répartis en deux grands groupes :

- ✓ Six diagrammes structurels
  - Diagramme d'objets : Il montre les instances des éléments structurels et leurs liens à l'exécution.
  - Diagramme de packages : Il montre l'organisation logique du modèle et les 9 relations entre les packages
  - Diagramme de structure composite : Il montre l'organisation interne d'un élément statique
  - Diagramme de composants : Il montre des structures complexes, avec leurs interfaces fournies et requises.

- Diagramme de déploiement : Il montre le déploiement physique des « artefacts » sur les ressources matérielles.
- Diagramme de classes : Il montre les briques de base statiques classes, associations, interfaces, attributs, opérations, généralisations etc.

✓ Sept diagrammes comportementaux

- Diagramme de cas d'utilisation : Il montre les interactions fonctionnelles entre les acteurs et le système à l'étude.
- Diagramme de vue d'ensemble des interactions : Il fusionne les diagrammes d'activités et de séquences pour combiner les fragments avec des décisions et des flots.
- Diagramme de séquences : Il montre la séquence verticale des messages passés entre objets au sein d'une interaction.
- Diagramme de communication : Il montre la communication entre objets dans le plan au sein d'une interaction.
- Diagramme de temps : Il fusionne les digrammes d'états et de séquences pour montrer l'évolution de l'état d'un projet au cours du temps.
- Diagramme d'activité : Il montre l'enchaînement des actions et décisions au sein d'une activité.
- Diagramme d'états : Il montre les différents états et transitions possibles des objets d'une classe.

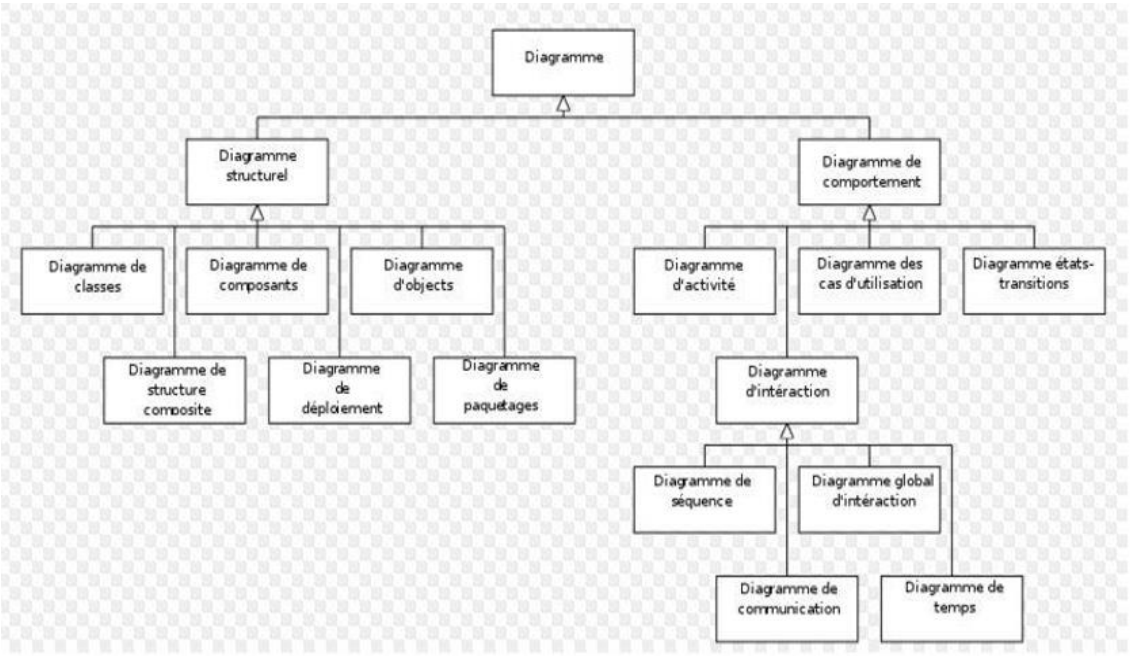

Figure1 : Hiérarchie des diagrammes d'UML

II.1. Acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. Il peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

Dans ce système, il existe cinq acteurs à savoir Professeur, Etudiant, Système, Plugin et API

Le **professeur** c'est un acteur principal du système.

Il s'authentifie, choisit une classe et une matière. Ainsi une liste de présences est détectée pour que la base de données soit alimentée.

Le **plugin** est l'acteur principal du cas détecter la liste des présences.

L'**API**, acteur principal, chargé d'alimenter la base de données en sauvegardant les données sur les participants et celles sur chaque séance.

Le **système**, cinq minutes après la clôture de la séance, envoi aux étudiants un formulaire leur permettant d'évaluer de façon anonyme la qualité de la séance.

Les étudiants, une fois que la séance soit terminée, remplissent le formulaire permettent de noter celle-ci.

II.2. Fonctionnalités (cas d'utilisation)

Un cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable pour un acteur. Il permet de décrire ce que le futur système devra faire sans spécifier comment il le fera. Nous allons lister les différents cas d'utilisations associés aux différents acteurs :

- Professeur
    - Choisir classe
    - Choisir matière
- Etudiant
    - Remplir formulaire d'évaluation
- Plug-in
    - Détecter liste des présences
- API
    - Alimenter base de données
    - Sauvegarder données participants
    - Sauvegarder données séance
- Système
    - Envoyer formulaire d'évaluation

II.3. Diagramme des cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, ils interagissent avec les cas d'utilisation (use cases).

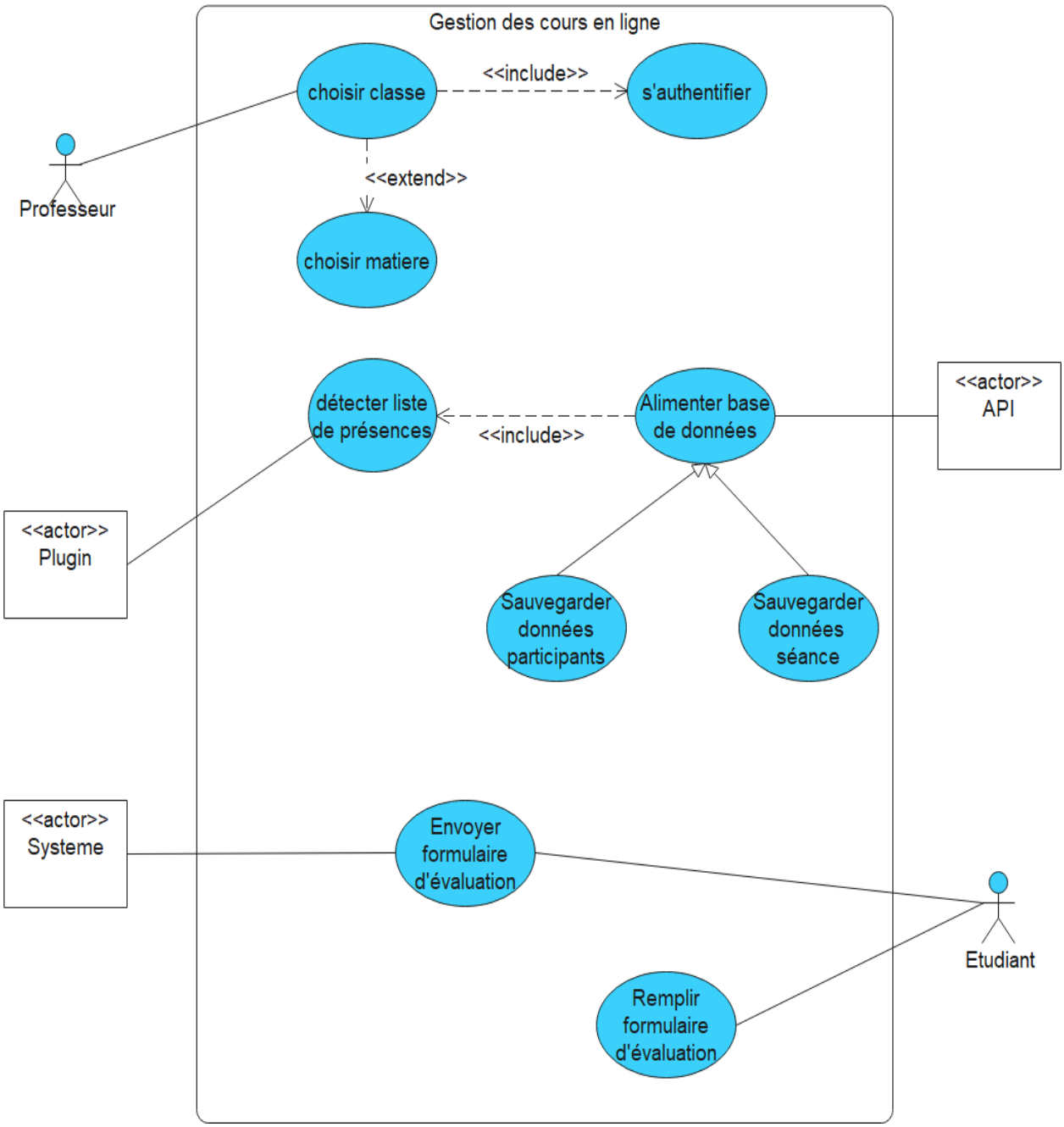

Figure2 : Le diagramme des cas d'utilisation

II.4. Fiches textuelles et diagrammes de séquence

Dans ce qui suit, nous décrirons de façon détaillée certains cas d'utilisation identifiés précédemment en recensant de façon textuelle les interactions entre les acteurs et le système. Nous compléterons certaines descriptions textuelles par des diagrammes de séquences UML.

Le diagramme de séquence décrit les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.

Scénario : une liste d'actions qui décrivent une interaction entre un acteur et le système.

Interaction : un comportement qui comprend un ensemble de messages échangés par un ensemble d'objet s dans un certain contexte pour accomplir une certaine tache.

Un message : est une transmission d'information unidirectionnelle entre deux objets, l'objet émetteur et l'objet récepteur.

Alt : c'est un opérateur qui contient une liste des fragments dans laquelle se trouvent d'autres séquences de messages et une seule séquence peut se produire à la fois.

Loop : c'est un opérateur correspondant à une boucle qui permet d'exécuter une séquence d'interaction tant qu'une condition est satisfaite.

Opt : c'est un opérateur qui contient une séquence qui peut se produire ou non.

Message synchrone : il bloque l'expéditeur jusqu'à la réponse du destinataire. Le flot de contrôle s'effectue entre l'émetteur et le récepteur.

Message asynchrone : ce message n'est pas bloquant pour l'expéditeur et le message envoyé peut être pris en compte par le récepteur à tout moment comme il peut être ignoré à tout moment.

Cas d'utilisation : « **S'authentifier** »

Titre	S’authentifier
Acteur	Professeur
Objectif	Le professeur a l’intention de s’authentifier pour accéder aux fonctionnalités de système.
Précondition	Disposer d’un compte
Scenario nominal	1. Le professeur demande la page de connexion 2. Le professeur saisit son login et son mot de passe puis valide 3. Le système vérifie les informations de l’authentification 4. Le système redirige le professeur dans son espace
Scenario alternatif	A1 : Après l’étape 3, si le login et/ou le mot de passe ne sont pas valides alors 3. a. Le système envoie un message d’erreur Donc retour à l’étape 1 du scenario nominal
Post-condition	Authentification réussie

Figure3 : Fiche textuelle du cas « **S’authentifier** »

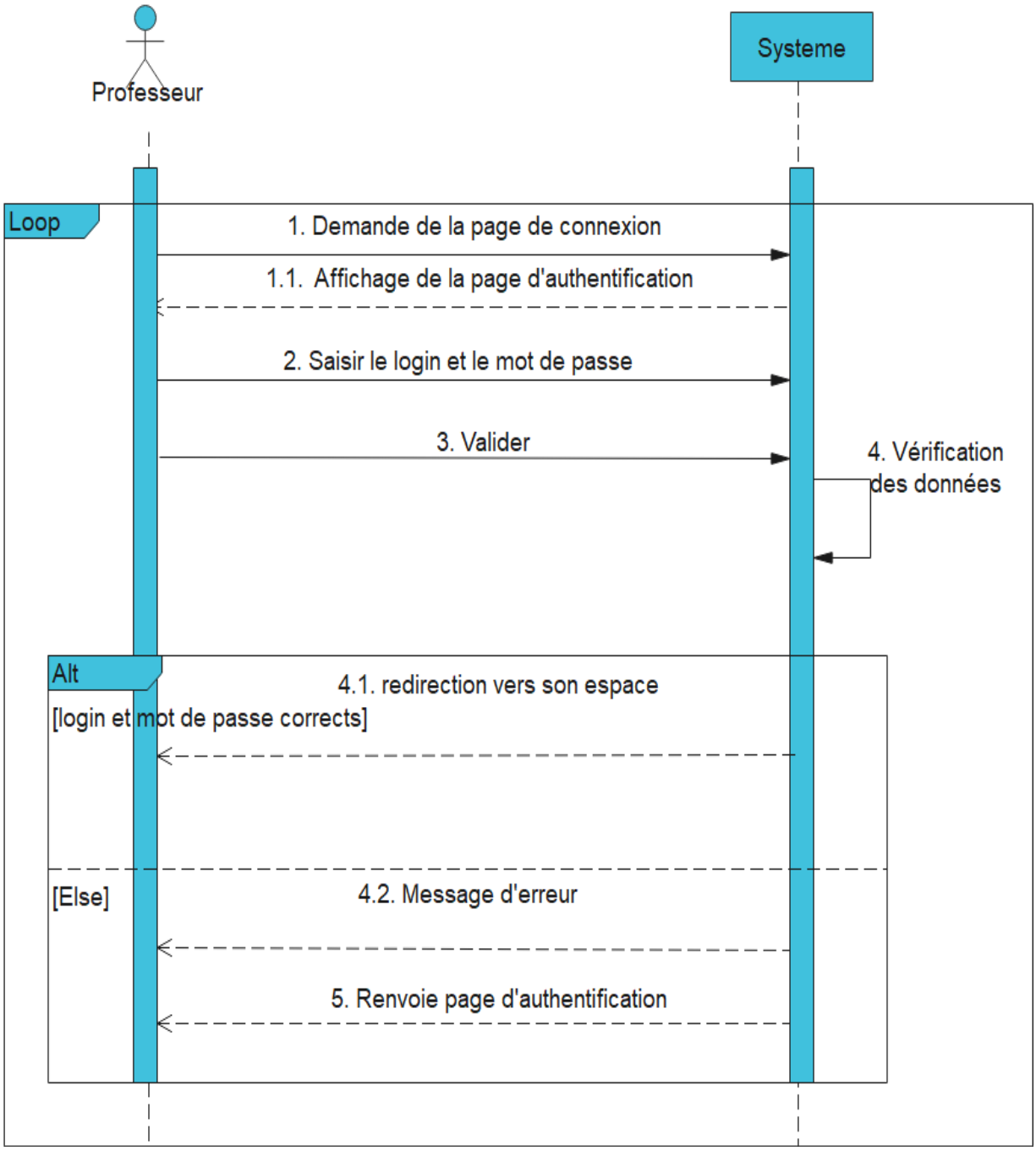

Figure4 : Diagramme de séquence du cas « **S'authentifier** »

Cas d'utilisation : « **Choisir classe** »

Titre	Choisir classe
Acteur	Professeur
Objectif	Le professeur a l'intention de choisir une classe
Précondition	S'authentifier
Scenario nominal	1. Le professeur demande la liste des classes 2. Le système affiche la liste des classes 3. Le professeur choisit une classe 4. Consulter la liste des matières. 5. Le système recherche renvoie la liste des matières. 6. Le professeur choisit la matière. 7. Le système enregistre les informations
Scenario alternatif	Aucun
Post-condition	Affichage de la liste des classes

Figure5 : Fiche textuelle du cas « **Choisir classe** »

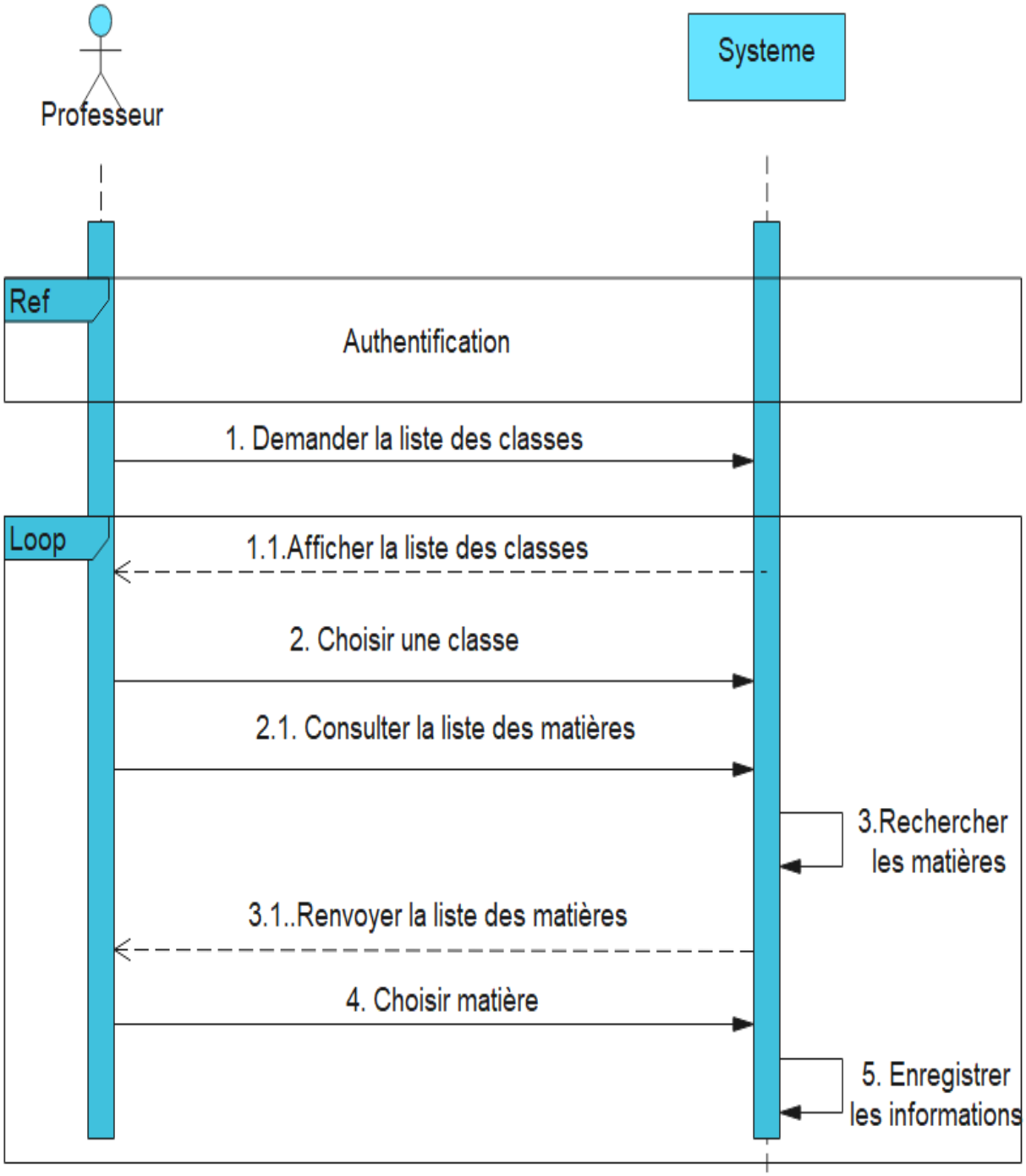

Figure6 : Diagramme de séquence du cas « **Choisir Classe** »

Cas d'utilisation «**Remplir formulaire d'évaluation** »

Titre	Remplir formulaire d'évaluation
Acteur	Etudiant
Objectif	L'étudiant doit gérer l'évaluation de la séance.
Précondition	Clôturer séance
Scenario nominal	1. Le système envoie un formulaire d'évaluation. 2. L'étudiant coche les champs 3. Valider l'évaluation de la séance 4. Enregistrement des données dans la base de données à travers l'API
Scenario alternatif	Aucun
Post-condition	Validation du formulaire l'évaluation

Figure7 : Fiche textuelle du cas « **Remplir formulaire d'évaluation** »

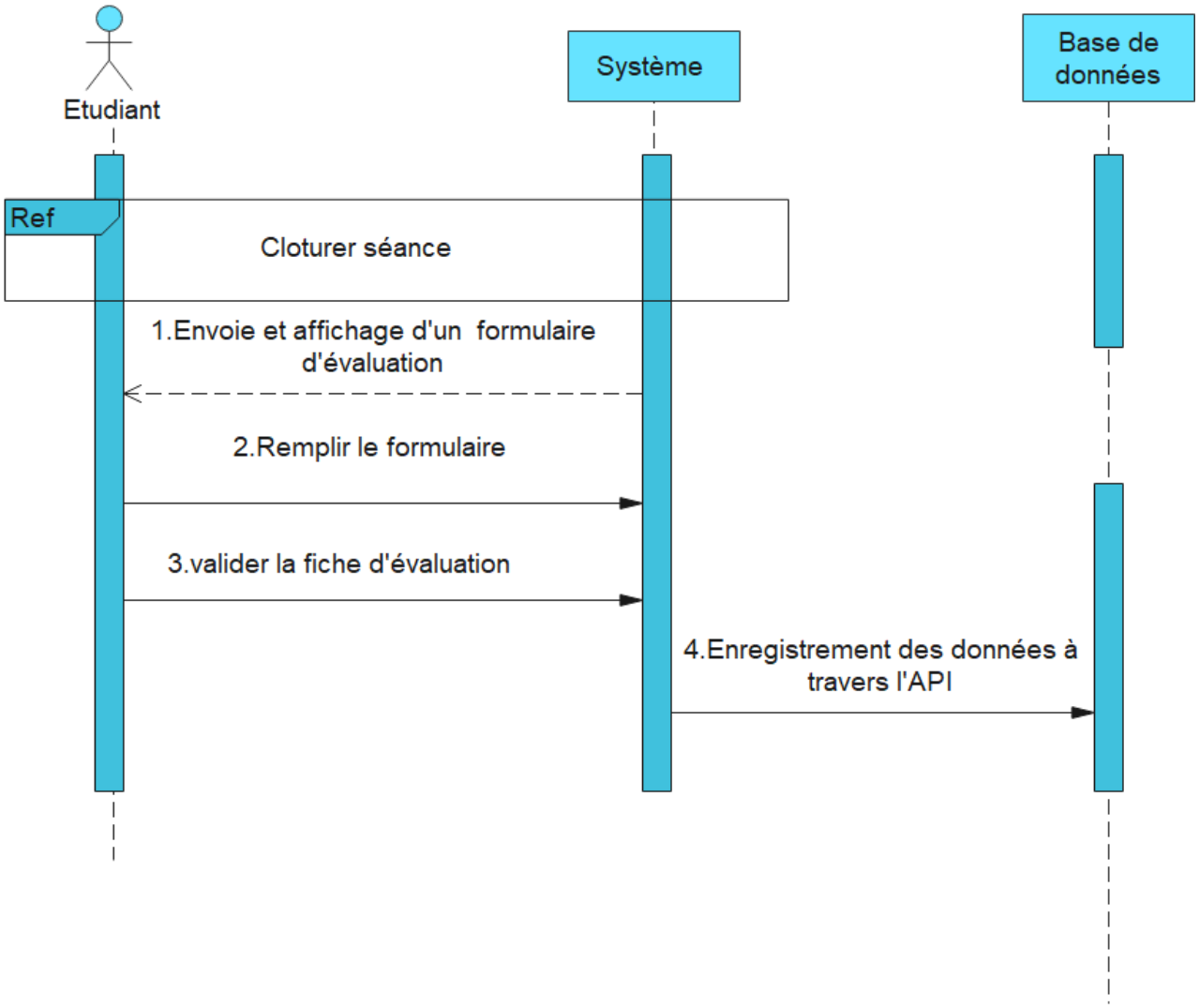

Figure 8 : Diagramme de séquence du cas « **Remplir formulaire d'évaluation** »

Cas d'utilisation : « **Détecter liste présence** »

Titre	Détecter liste présence
Acteur	Plugin
Objectif	Le plugin a pour rôle de détecter la liste des présences et d'alimenter la base.
Précondition	Après authentification du professeur et du choix de la classe et de la matière.
Scenario nominal	1. Le plugin récupère la liste de présences. 2. Le système lui envoi la liste des présences 3. Enregistrement de la liste de présences dans la base de données par le système.
Scenario alternatif	A1. Si aucune présence n'est détectée alors le scénario reste à l'étape 1.
Post-condition	Mise à jour de la liste des présences dans la base de données.

Figure 9: Fiche textuelle du cas « **Détecter liste présence** »

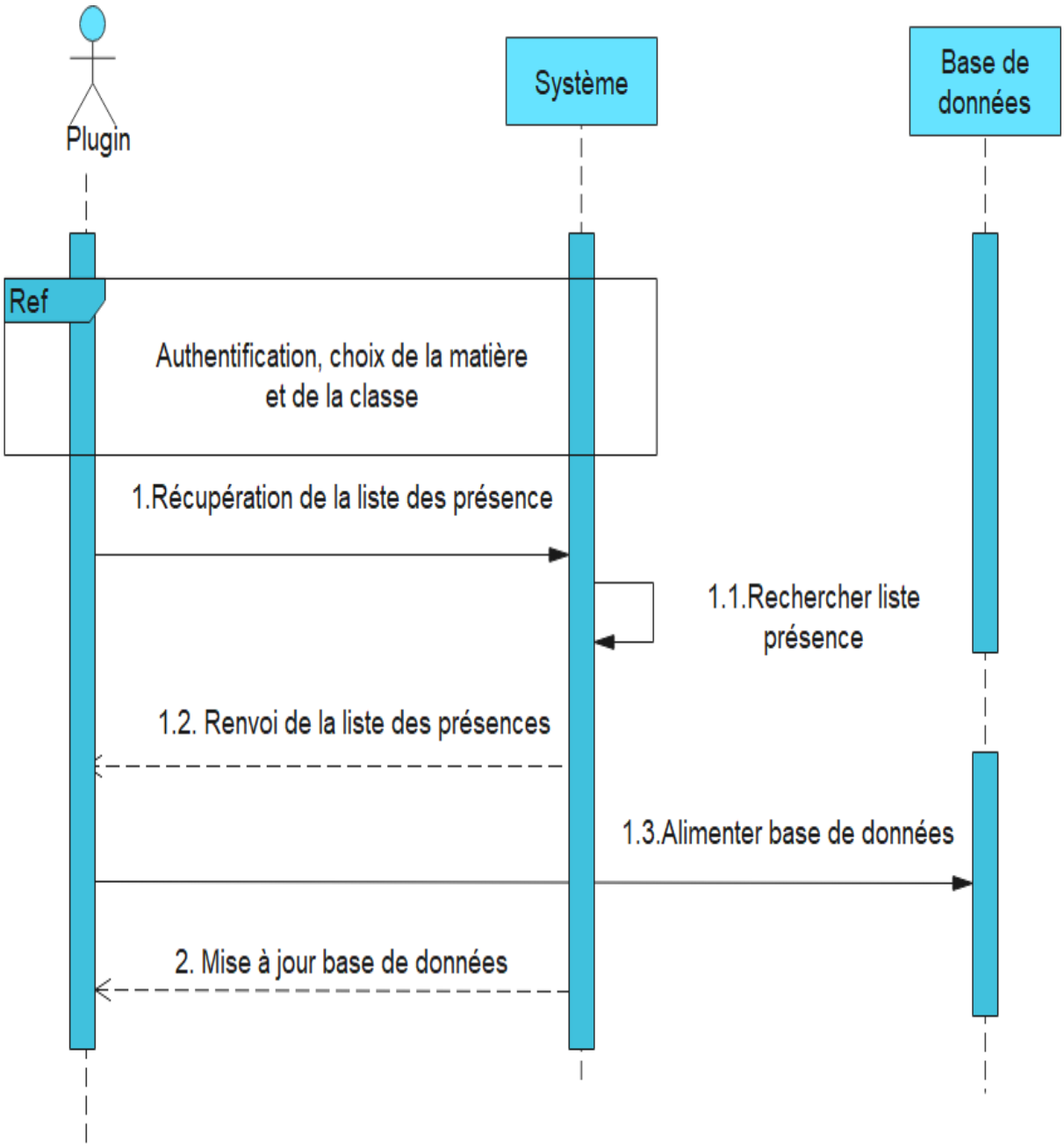

Figure 10: Fiche textuelle du cas « **Détecter liste présence** »

II.5. Diagramme de classes

Le diagramme de classes permet de représenter les classes intervenant dans le système. Le diagramme de classes est une représentation statique des éléments qui composent un système et de leurs relations. Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettant de créer des objets ayant ces propriétés. Une classe est une description d'un groupe d'objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations) et de relations avec d'autres objets (les associations et les agrégations). La classe est définie par son nom, ses attributs et ses opérations.

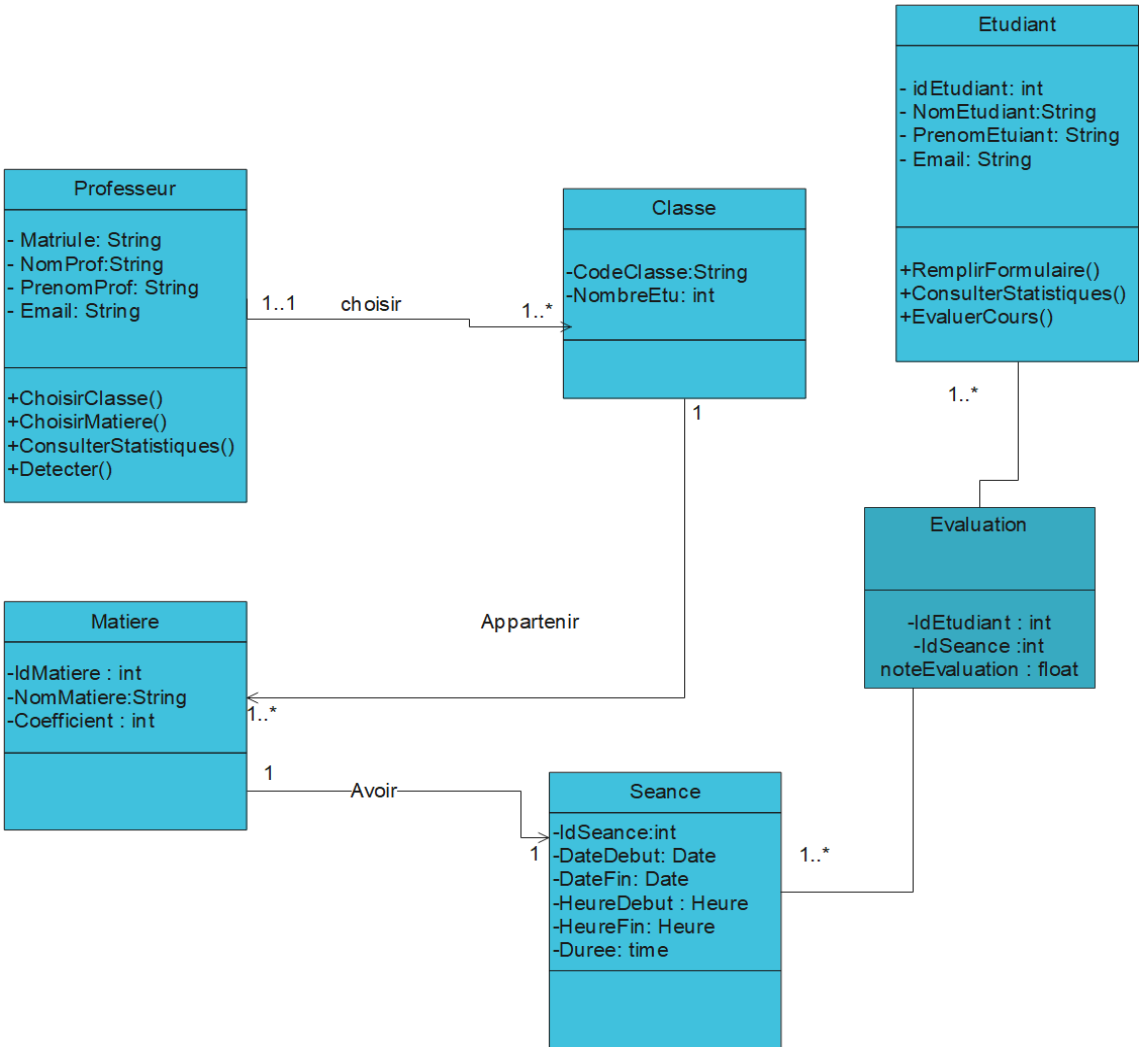

Figure 11: Diagramme de classes

II.6. Modèle relationnel

```
Professeur
(Matricule, NomProf, PrenomProf, Email, #CodeClasse)
Etudiant (idEtudiant, nomEtudiant, prenomEtudiant, Email)
Classe (CodeClasse, NombreEtu, #idMatiere)
Matière (idMatiere, NomMatiere, Coefficient)
Seance (idSeance, DateDebut, DateFin, HeureDebut, HeureFin, Duree)
Evaluation (#idEtudiant, #idSeance, noteEvaluation)
```

II. Modèle physique de base de données

Le MPD (Modèle physique des données) est la dernière étape de l'analyse. Le MPD n'est autre qu'une liste de tables avec pour chacune d'elle les colonnes faisant partie de cette table.

Formalisme

Chaque nom de table est suivi d'une liste de colonnes entre parenthèses.

- Les clés primaires sont soulignées
- les clés étrangères sont précédées par un dièse.

Une entité devient une table, tous ses attributs deviennent des colonnes et son identifiant devient la clé primaire.

Une association aux cardinalités de type …, n des deux côtés (donc de type plusieurs à plusieurs), devient aussi une table :

- Les attributs de cette association deviennent des colonnes.
- Les clés primaires des tables se trouvant de part et d'autres de l'association sont importées sous forme de clés étrangères.
- La concaténation de ces clés étrangères devient la clé primaire.

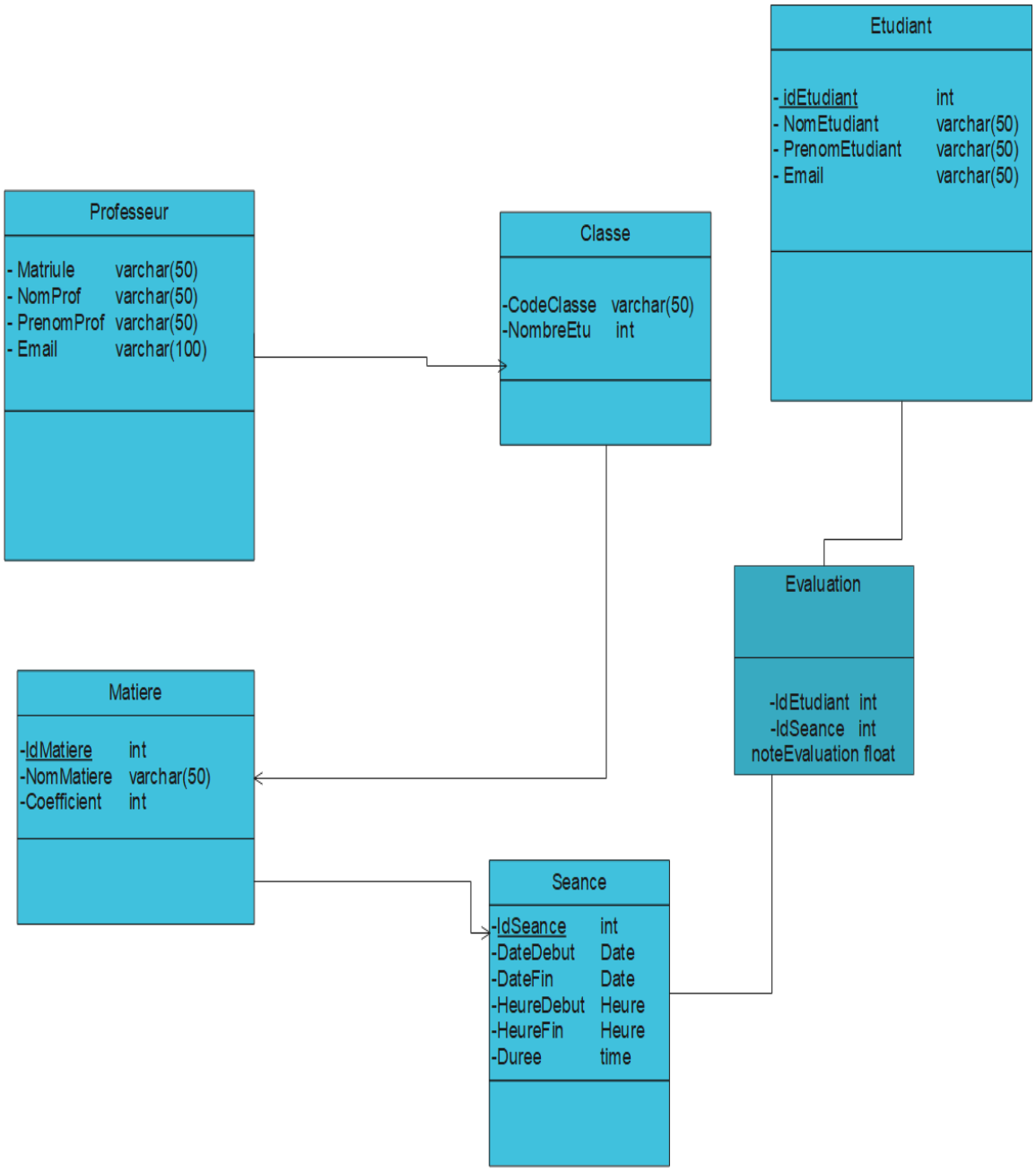

Figure12 : Modèle physique de données

IV. Architecture Technique