Chapitre 3 : MODELISATION DETAILLEE

SECTION I : PRESENTATION DU LANGAGE DE MODELISATION UTILISE

1. Présentation D’UML

UML (Unified Modeling Language) est un langage de modélisation graphique normalisé utilisé dans le domaine du développement logiciel. Créé dans les années 1990 par Grady Booch, James Rumbaugh et Ivar Jacobson (souvent appelés "les trois amis"), UML est devenu un standard adopté par l'Object Management Group (OMG) en 1997.

UML permet de visualiser, spécifier, construire et documenter les artefacts d'un système logiciel. Il offre une notation standardisée et des techniques de modélisation pour représenter les différents aspects d'un système, tant sur le plan structurel que comportemental.

Les principaux avantages d'UML sont :

* Une notation standardisée facilitant la communication entre les différents acteurs d'un projet
* Une approche indépendante des langages de programmation et des méthodologies de développement
* La capacité à représenter différentes vues d'un système à différents niveaux d'abstraction
* La possibilité de documenter l'architecture logicielle de manière complète et cohérente

UML est aujourd'hui à sa version 2.5 et reste un outil fondamental dans l'ingénierie logicielle, particulièrement dans les approches orientées objet.

1. Les principaux diagrammes d'UML

UML propose 14 types de diagrammes, répartis en deux catégories principales :

**Diagrammes comportementaux** (représentant le comportement dynamique du système) :

* **Diagramme de classes** : représente la structure statique du système en termes de classes et de relations
* **Diagramme d'objets** : montre des instances de classes et leurs relations à un moment précis
* **Diagramme de composants** : illustre les composants logiciels et leurs dépendances
* **Diagramme de déploiement** : décrit l'architecture physique du système en termes de nœuds et de connexions
* **Diagramme de paquetages** : organise les éléments du modèle en groupes logiques
* **Diagramme de structures composites** : montre la structure interne des classes
* **Diagramme de profils** : permet d'étendre UML pour des domaines spécifiques

**Diagrammes comportementaux** (représentant le comportement dynamique du système) :

* **Diagramme de cas d'utilisation :** capture les exigences fonctionnelles du système
* **Diagramme d'activité :** décrit les flux de travail ou les processus métier
* **Diagramme de séquence :** illustre les interactions entre les objets dans un scénario particulier
* **Diagramme de communication :** montre les interactions entre les objets avec un accent sur les liens
* **Diagramme d'états-transitions :** représente les différents états d'un objet et les transitions entre ces états
* **Diagramme de temps :** montre l'évolution des états d'un objet au fil du temps
* **Diagramme global d'interaction :** combine les diagrammes d'activité et de séquence

Dans la pratique, les diagrammes les plus couramment utilisés sont les diagrammes de cas d'utilisation, de classes, de séquence et d'activité, qui seront détaillés dans les sections suivantes de ce document.

SECTION II : MODELISATION DU FUTUR SYSTEME

1. Analyse Des Besoins Fonctionnels
2. Les Acteurs Du Système

Les acteurs représentent les entités externes qui interagissent avec le système. Un acteur peut être une personne physique, un autre système informatique, ou même un dispositif matériel. Les acteurs sont identifiés lors de l'analyse des besoins fonctionnels et constituent le point de départ pour la modélisation des cas d'utilisation.

Dans un système d'information, on distingue généralement plusieurs types d'acteurs :

* **Étudiant :** Utilisateur principal qui consulte les cours, reçoit des notifications et interagit avec les professeurs. L'étudiant est caractérisé par son numéro de matricule et sa filière d'études.
* **Professeur :** Utilisateur responsable de la création et de la gestion des cours, des notifications d'absence et des interactions avec les étudiants. Le professeur possède des attributs spécifiques comme sa discipline enseignée et ses horaires de disponibilité.
* **Administrateur :** Utilisateur chargé de la gestion globale du système, notamment l'administration des comptes utilisateurs, la modération des contenus et la publication d'annonces officielles.

L’identification précise des acteurs permet de délimiter clairement les frontières du système et d'établir les interactions que celui-ci devra gérer.

1. Rôles des acteurs du système

Chaque acteur identifié joue un rôle spécifique dans le système et dispose de responsabilités clairement définies :.

**Rôle de l'Étudiant** :

* Accéder aux cours et aux documents partagés par les professeurs
* Recevoir et consulter les notifications (notamment les absences des professeurs)
* Interagir avec les professeurs en posant des questions et en consultant les réponses
* Utiliser la messagerie interne pour communiquer avec les professeurs et d'autres étudiants
* S'authentifier pour accéder au système

**Rôle du Professeur :**

* Créer, modifier et supprimer des cours
* Partager des documents pédagogiques avec les étudiants
* Déclarer ses absences et en informer les étudiants concernés
* Répondre aux questions des étudiants
* Consulter les notifications le concernant
* Utiliser la messagerie interne pour communiquer avec les étudiants
* S'authentifier pour accéder au système

**Rôle de l'Administrateur :**

* Gérer les comptes utilisateurs (création, modification, suppression)
* Attribuer des rôles aux utilisateurs
* Modérer les contenus publiés sur la plateforme
* Publier des annonces officielles
* Gérer la messagerie interne
* S'authentifier pour accéder au système

Ces rôles déterminent les privilèges et les responsabilités de chaque acteur dans le système et guident la conception des interfaces utilisateur et des fonctionnalités associées.

1. Diagramme Des Cas D'utilisation
2. Objectif

Le diagramme de cas d'utilisation a pour objectif principal de représenter graphiquement les fonctionnalités offertes par le système du point de vue de ses utilisateurs. Il permet de :

* + Capturer les exigences fonctionnelles du système
  + Illustrer les interactions entre les acteurs et le système
  + Définir les limites et le contexte du système
  + Servir de base pour la communication entre les parties prenantes du projet
  + Guider le développement des autres modèles et la conception du système

Le diagramme de cas d'utilisation sert de fondement pour la compréhension des besoins des utilisateurs et oriente les phases ultérieures de conception et d'implémentation.

1. Présentation des concepts du diagramme des cas d'utilisation

Le diagramme de cas d'utilisation utilise plusieurs concepts clés pour représenter les fonctionnalités d'un système :

* + **Acteur** : entité externe (personne, système, etc.) qui interagit avec le système.
  + **Cas** **d'utilisation** : fonctionnalité ou service offert par le système à ses utilisateurs.
  + **Relations** : différents types de liens entre les éléments du diagramme.
  + **Frontière du système** : délimitation entre le système modélisé et son environnement.

1. Acteur

* Administrateur
* Professeurs
* Etudiants

1. Cas d'utilisation

Les cas d'utilisation représentent les fonctionnalités offertes par le système. Dans le diagramme fourni, nous pouvons identifier plusieurs cas d'utilisation essentiels :

**Pour l'Étudiant :**

* + Accéder aux cours
  + Accéder aux documents partagés
  + Interagir avec les professeurs
  + Poser des questions
  + Consulter les réponses
  + Recevoir les notifications
  + Utiliser la messagerie interne
  + S'authentifier

**Pour le Professeur :**

* + Gestion des cours (créer, modifier, supprimer)
  + Partager des documents
  + Répondre aux questions des étudiants
  + Déclarer une absence
  + Consulter les notifications
  + Utiliser la messagerie interne
  + S'authentifier

**Pour l'Administration :**

* + Gérer les comptes utilisateurs (création, modification, suppression)
  + Attribuer des rôles aux utilisateurs
  + Modérer les contenus publiés
  + Publier des annonces officielles
  + Gérer la messagerie interne
  + S'authentifier

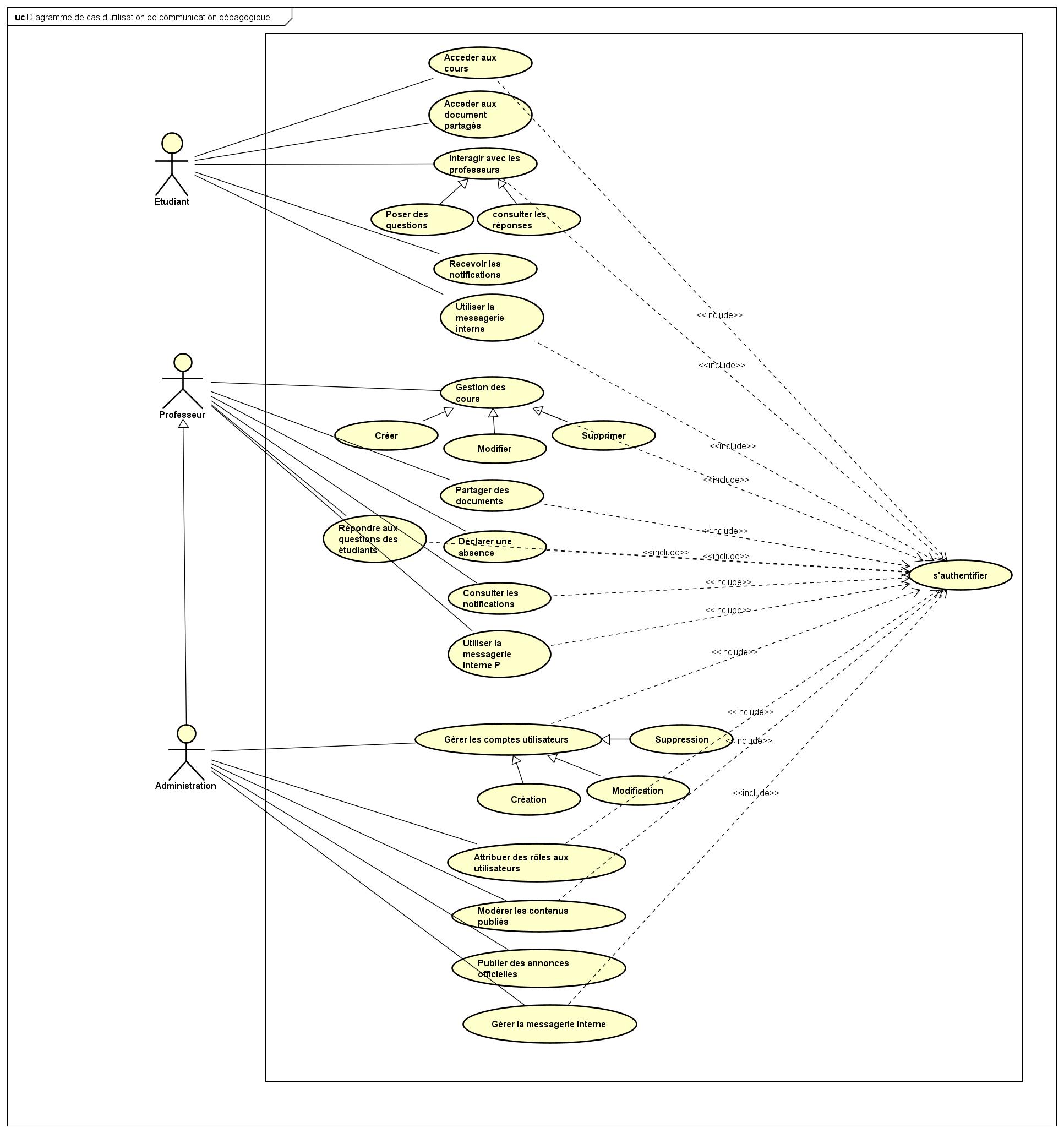
1. Relation

Le diagramme de cas d'utilisation présente plusieurs types de relations entre les éléments :

* + Association : relation de base entre un acteur et un cas d'utilisation, représentée par une ligne simple. Par exemple, l'association entre l'Étudiant et le cas d'utilisation "Accéder aux cours".
  + Inclusion (<<include>>) : relation entre deux cas d'utilisation indiquant que le premier incorpore le comportement du second. Dans le diagramme, plusieurs cas d'utilisation incluent le cas "S'authentifier", indiquant que l'authentification est un prérequis pour ces fonctionnalités.
  + Extension (<<extend>>) : relation indiquant qu'un cas d'utilisation peut étendre le comportement d'un autre sous certaines conditions. Cette relation n'est pas clairement visible dans le diagramme fourni.
  + Généralisation : relation hiérarchique entre cas d'utilisation ou entre acteurs. Dans le diagramme, on peut voir que certains cas d'utilisation comme "Créer", "Modifier" et "Supprimer" sont des spécialisations du cas "Gérer les comptes utilisateurs".

Ces relations permettent de structurer le diagramme et d'exprimer les dépendances entre les différentes fonctionnalités du système.

1. Modélisation du diagramme des cas d'utilisation



1. Description textuelle des cas d'utilisation

Pour illustrer la description textuelle des cas d'utilisation, prenons l'exemple du cas "Déclarer une absence" pour le Professeur :

**Cas d'utilisation** : Déclarer une absence

**Acteur principal** : Professeur

**Préconditions** :

* Le professeur est authentifié dans le système
* Le professeur a des cours programmés

**Scénario principal** :

1. Le professeur accède à son espace personnel
2. Il sélectionne l'option "Déclarer une absence"
3. Le système affiche un formulaire de déclaration d'absence
4. Le professeur renseigne la date, la raison de l'absence et sélectionne les cours concernés
5. Le professeur valide sa déclaration
6. Le système enregistre l'absence et génère des notifications pour les étudiants concernés
7. Le système confirme l'enregistrement de l'absence

**Scénarios alternatifs** :

* Si le professeur annule sa déclaration, le système revient à l'espace personnel sans enregistrer d'absence
* Si la date sélectionnée est dans le passé, le système affiche un message d'erreur

**Postconditions** :

* L'absence est enregistrée dans le système
* Les étudiants concernés sont notifiés de l'absence du professeur

Cette description détaillée permet de comprendre précisément le déroulement du cas d'utilisation et sert de base pour le développement de la fonctionnalité correspondante.

1. Diagramme des classes
2. Objectif

Le diagramme de classes est l'un des diagrammes UML les plus utilisés dans la modélisation orientée objet. Dans le cadre de notre projet de gestion des interactions pédagogiques, ses objectifs sont :

* Représenter la structure statique du système en identifiant les classes d'objets et leurs relations
* Définir les attributs et les opérations de chaque classe
* Établir les fondations du modèle de données de l'application
* Servir de base pour l'implémentation du système
* Documenter l'architecture du système pour faciliter sa maintenance

1. Présentation des concepts du Diagramme des classes

Le diagramme de classes utilise plusieurs concepts fondamentaux pour modéliser la structure statique d'un système :

* **Classe** : représentation d'un ensemble d'objets ayant des propriétés et des comportements similaires
* **Attribut** : propriété ou caractéristique d'une classe
* **Association** : relation structurelle entre classes
* **Agrégation/Composition** : relations spéciales d'association indiquant des liens de type tout-partie
* **Généralisation/Spécialisation** : relation hiérarchique entre classes

1. Classe

Nous pouvons identifier plusieurs classes essentielles pour notre système de gestion des interactions pédagogiques :

* **Utilisateur** : classe abstraite représentant tous les utilisateurs du système avec des attributs communs comme id, nom, email, motDePasse et rôle, ainsi que des opérations comme Authentifier() et changerMotDePasse().
* **Professeur** : classe spécialisée d'Utilisateur avec des attributs spécifiques comme matricule et des opérations comme creerCours(), annoncerAbsence() et partagerFichier().
* **Etudiant** : classe spécialisée d'Utilisateur avec des attributs comme matriculeEtudiant et filière, et des opérations comme consulterCours() et recevoirNotification().
* **Admin** : classe spécialisée d'Utilisateur avec des opérations de gestion comme gererCompte() et modererContenu().
* **Cours** : classe représentant les cours avec des attributs comme id, titre, description et dateCreation, et des opérations comme ajouterFichier() et supprimerFichier().
* **Message** : classe représentant les messages échangés entre utilisateurs.
* **Publication** : classe représentant les contenus publiés par les utilisateurs.
* **Notification** : classe représentant les notifications envoyées aux utilisateurs.
* **Absence** : classe représentant les absences déclarées par les professeurs.
* **Fichier** : classe représentant les documents partagés.
* **Filière** : classe représentant les filières d'études.
* **Groupe** : classe représentant les groupes d'étudiants.

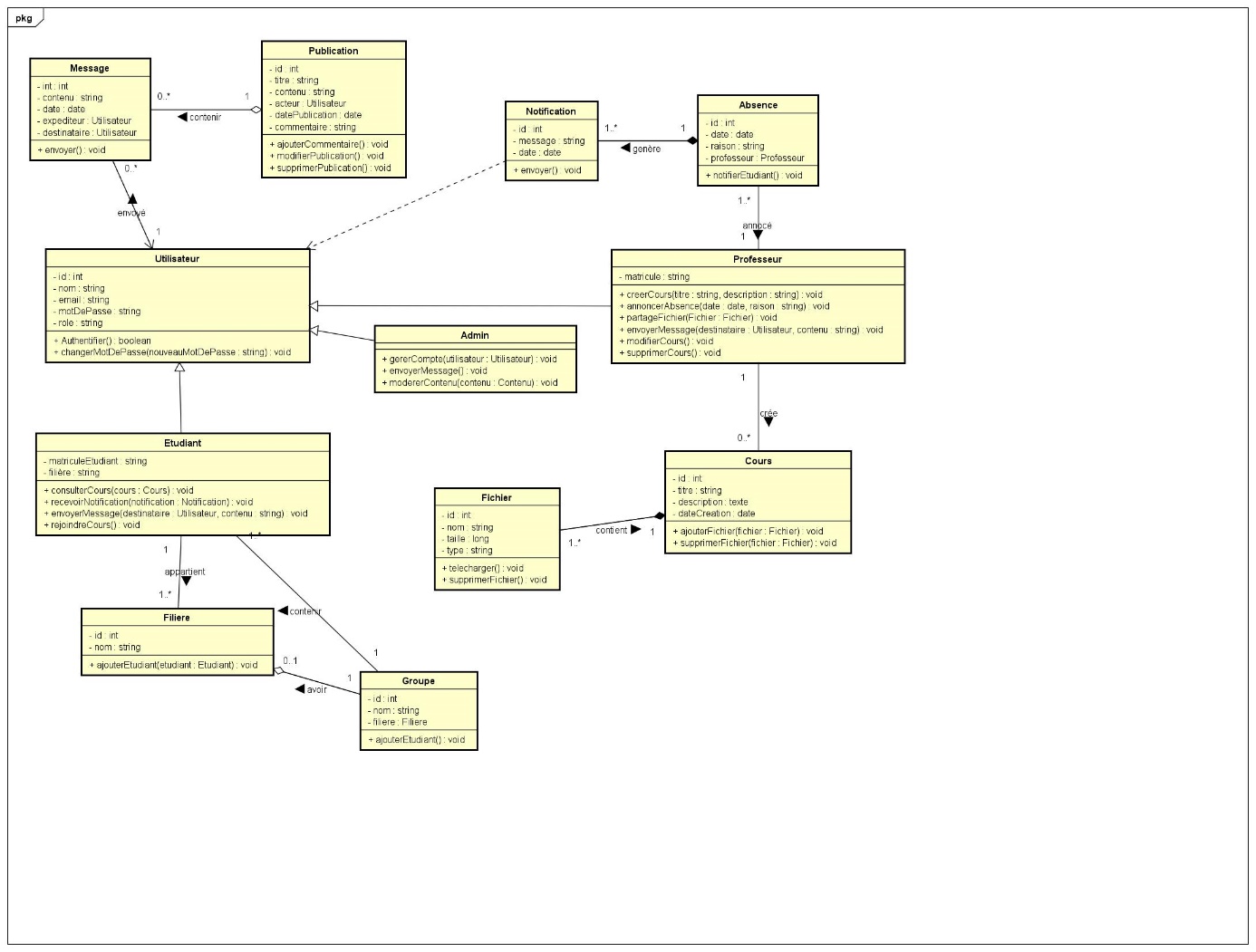
1. Relation

Le diagramme de classes présente plusieurs types de relations entre les classes :

* **Association** : relation structurelle entre classes, représentée par une ligne simple. Par exemple, la relation entre Etudiant et Groupe.
* **Agrégation** : relation de type "est une partie de", représentée par une ligne avec un losange vide. Par exemple, la relation entre Groupe et Filière.
* **Composition** : relation forte de type "est composé de", représentée par une ligne avec un losange plein. Par exemple, la relation entre Cours et Fichier.
* **Généralisation** : relation hiérarchique entre une classe générale et une classe spécialisée, représentée par une flèche avec une pointe triangulaire. Par exemple, la relation entre Utilisateur et ses sous-classes Professeur, Etudiant et Admin.
* **Dépendance** : relation indiquant qu'une classe utilise une autre classe, représentée par une flèche en pointillés. Par exemple, la relation entre Publication et Notification.

1. Modélisation du diagramme des classes

* Identification des classes principales à partir des besoins fonctionnels et des cas d'utilisation
* Définition des attributs et des opérations pour chaque classe
* Établissement des relations entre les classes
* Spécification des cardinalités pour chaque relation
* Définition des hiérarchies de classes (par exemple, la hiérarchie Utilisateur - Professeur/Etudiant/Admin)
* Organisation du diagramme pour maximiser sa lisibilité



1. Diagramme De Séquence
2. Objectif

Le diagramme de séquence est un diagramme d'interaction qui met l'accent sur l'ordre chronologique des messages échangés entre les objets. Dans le cadre de notre projet de gestion des interactions pédagogiques, ses objectifs sont :

* Illustrer comment les objets collaborent pour réaliser un scénario particulier
* Montrer l'enchaînement temporel des interactions entre les acteurs et le système
* Détailler la mise en œuvre des cas d'utilisation identifiés précédemment
* Valider la pertinence des opérations définies dans le diagramme de classes
* Servir de base pour l'implémentation des fonctionnalités du système

Les diagrammes de séquence complètent les diagrammes de cas d'utilisation et de classes en ajoutant une dimension dynamique à la modélisation du système.

1. Présentation des concepts du Diagramme des séquences

Le diagramme de séquence utilise plusieurs concepts fondamentaux pour représenter les interactions entre objets :

* **Ligne de vie** : représentation verticale d'un objet, montrant sa durée de vie dans le scénario
* **Acteur** : utilisateur ou système externe qui initie le scénario
* **Message** : communication entre objets, représentée par une flèche horizontale
* **Activation** : période pendant laquelle un objet exécute une opération, représentée par un rectangle vertical sur la ligne de vie
* **Fragment combiné** : structure permettant de représenter des comportements conditionnels, des boucles, etc.

Ces concepts permettent de représenter de manière précise et détaillée le déroulement d'un scénario d'interaction.

1. Acteur

Dans un diagramme de séquence, l'acteur est représenté par un pictogramme (similaire à celui du diagramme de cas d'utilisation) ou par un rectangle en haut du diagramme. Sa ligne de vie s'étend verticalement pour montrer sa participation tout au long du scénario.

Dans le contexte de notre application de gestion des interactions pédagogiques, les acteurs des diagrammes de séquence correspondent aux acteurs identifiés dans le diagramme de cas d'utilisation :

* Étudiant
* Professeur
* Administrateur

1. Message

Les messages représentent les communications entre objets et sont figurés par des flèches horizontales entre les lignes de vie. Dans le contexte de notre application, plusieurs types de messages peuvent être identifiés :

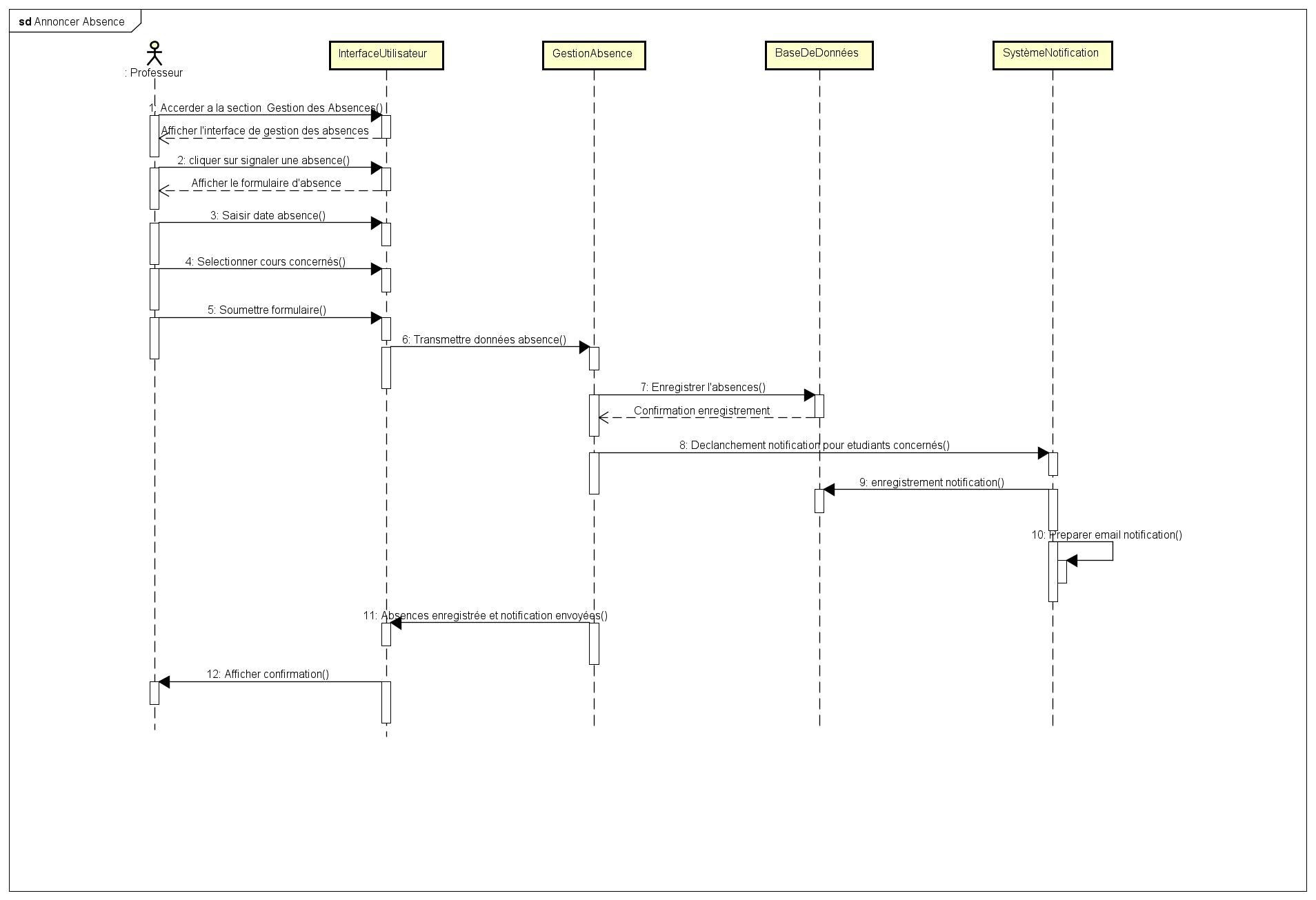
* **Message synchrone** : l'émetteur attend une réponse avant de poursuivre. Par exemple, l'appel à la méthode Authentifier() de la classe Utilisateur.
* **Message asynchrone** : l'émetteur continue son exécution sans attendre de réponse. Par exemple, l'envoi d'une notification aux étudiants suite à la déclaration d'une absence.
* **Message de retour** : réponse à un message synchrone. Par exemple, la confirmation de l'enregistrement d'une absence.
* **Message réflexif** : message qu'un objet s'envoie à lui-même. Par exemple, un contrôle de validation interne.

1. Modélisation du diagramme des séquences

Pour illustrer la modélisation d'un diagramme de séquence, prenons l'exemple du cas d'utilisation "Déclarer une absence" :

* L'acteur Professeur initie le scénario en demandant à accéder au formulaire de déclaration d'absence.
* Le système affiche le formulaire de déclaration.
* Le Professeur remplit le formulaire et le soumet.
* Le système crée une nouvelle instance de la classe Absence avec les informations fournies.
* Le système associe l'absence au Professeur.
* Le système génère des notifications pour tous les étudiants concernés.
* Le système confirme l'enregistrement de l'absence au Professeur.

Représentation du diagramme de séquence, cas de la Création d’une absence



Conclusion

La modélisation UML a précisément défini la structure et le comportement de notre application web de gestion pédagogique à l'IUG. Le diagramme de cas d'utilisation a identifié les fonctionnalités clés pour les trois acteurs principaux. Le diagramme de classes a établi le modèle de données avec les entités principales et leurs relations. Le diagramme de séquence a illustré les interactions dynamiques entre objets. Ces modèles constituent une documentation technique essentielle qui guidera l'implémentation en code selon une approche orientée objet.