**I. PRÉSENTATION GÉNÉRALE**

**1.1. PRÉSENTATION DE L’ESTM**

**1.1.1. DOMAINE D’ACTIVITÉ**

L’ESTM est une école d’Ingénieurs et de Spécialistes du Management à la pointe des dernières technologies, s’appuyant sur un corps professoral très expérimenté.  
 Ils cultivent la passion, l’excellence, le savoir pratique, le professionnalisme, grâce à leurs excellentes relations internationales concrétisées par des partenariats privilégiés avec des institutions leaders dans leurs domaines respectifs.  
 L’objectif de l’ESTM n’est pas seulement de délivrer des formations reconnues, innovantes et de qualité à nos étudiants, mais c’est aussi de les accompagner dans leur insertion professionnelle et dans la construction d’une belle carrière.

**1.1.2. STRUCTURE ORGANIQUE ET FONCTIONNELLE**

**1.1.2.1. STRUCTURE ORGANIQUE**

ESTM a une structure organique qui est représenté comme suit :

* Un directeur général (D.G) ;
* Un directeur des études (D.E) ;
* Un service financier ;
* Un service chargé de communication et développement ;
* Un corps professoral ;

**1.1.2.2. STRUCTURE FONCTIONNELLE**

Elle explique la manière dont la structure organique fonctionne de manière détaillée

* Le D.G : Il est le gérant de l’entreprise, il donne des ordres au différents services en vue d’une bonne marche du déroulement des activités de l’entreprise.
* Le D.E : il organise, planifie et s’assure du bon déroulement des activités de pédagogique de l’école.
* Un service financier : Son rôle est de maintenir saine la santé financière de l’école, s’assurer que l’école réalise un bon investissement en prévenant les risques de pertes possibles.
* Un Service chargé de communication et développement : ce service comporte 2 départements, le premier et celui de la communication qui se charge de la communication et la promotion de l’école, et le second est le développement qui se charge des partenariats et du développement de l’école envers les organismes et institutions étrangère.
* Un corps professoral : Leur rôle est de dispenser les enseignements de qualités en fonction des différents domaines d’expertise de chacun.

**1.2. PRÉSENTATION DU SUJET**

**1.2.1 DÉFINTION DU SUJET ET OBJECTIFS**

**1.2.1.1 DÉFINITION DU SUJET**

Notre sujet portera sur la mise en place d’une application web de suivi de la vie associative des étudiants. Cette application permettra de gérer les informations relatives à la vie associative des étudiants à l’ESTM dans tous leurs aspects, de la création d’informations, mise à jour, consultation, elle devra nous permettre d’avoir une vue d’ensemble sur la vie associatives de l’étudiant. Les données d’entrées et de sorties seront suivis de prêt en mettant une traçabilité.

**1.2.1.2 PROBLÉMATIQUE**

Etant donné que la vie associative d’un étudiant l’aide dans beaucoup les secteurs de sa vie active, nous avons pu constater que l’ESTM dans sa gestion de la vie associative des étudiants rencontre beaucoup de difficulté notamment dans la planification, le suivi, la traçabilité des activités associatives, ceux-ci sont quasiment inexistants et pose des problèmes aux étudiants tel que le manque d’informations concernant les différentes activités des clubs de l’école, ou ceux de l’amicale des étudiants.

La problématique de notre sujet est le besoin, de rendre accessible les informations relatives aux activités associatives des étudiants afin de simplifier le processus d’accès aux clubs et aux différentes activités.

**1.2.1.3 OBJECTIFS**

Notre objectif est la mise en place d’une application web de suivi de la vie associative des étudiants. Cette application permettra de gérer les informations relatives aux activités associatives des étudiants dans tous leurs aspects, de la création d’informations, mise à jour, consultation, elle devra nous permettre d’avoir une vue d’ensemble sur la vie associatives de l’étudiant, pour y arriver nous devons mettre en place une interface administrateur qui nous permettra de gérer plusieurs aspects du suivi des activités des étudiants tel que l’ajout d’informations, la consultation de ces informations ou encore la mise à jour de celle-ci, nous devrons entre autres fournir des interfaces utilisateur pour l’exécution de chaque aspects cité ci-dessus. En plus de ça les données d’entrées et de sorties seront suivis de prêt en mettant une traçabilité.

**1.2.1.4 DÉLIMITATION DU CHAMPS D’ÉTUDE**

L’ESTM de façon générale dispose des champs d’actions suivants pour sa gestion de la vie associatives des étudiants :

* La gestion des clubs et de l’amicale
* La gestion des sorties extra-scolaire

Nous nous focaliserons sur la gestion des clubs et de l’amicale. Ceci veut dire que notre application permettra de gérer tous les aspects liés à la gestion des clubs et de l’amicale. Cependant cette délimitation ne restera pas statique car certaines informations citées ci-dessus pourront nous aider à développer notre réflexion.

**1.2.2 PRÉSENTATION DE L’EXISTANT**

**1.2.2.1 ÉTUDE DE L’EXISTANT**

**Processus de suivi des activités associatives des étudiants**

Pour le suivi des activités associatives de ces étudiants l’ESTM ne dispose pas de méthode informatisée tout est fait de manière assez archaïque avec les fiches et des dossiers.

Les activités associatives à l’ESTM se résume à l’intégration aux différents clubs dont dispose l’école exemple (CEMAD, ANGLAIS, SCIENTIFIQUE, AMICALE etc…) et aux participations aux différentes activités organisé par leur club ou par l’amicale des étudiants telle que (La JPO, Les week-end d’intégration, …).

Pour se faire chaque étudiant doit se faire enregistrer dans un club, le club s’assure d’avoir les informations relatives à l’étudiant c’est-à-dire :

* Nom
* Prénom
* Classe
* Filière
* Niveau

Après que le présidant du club ai eu toutes les informations nécessaires, il doit maintenant accepter l’adhésion de l’étudiant au club.

Les étudiants figurants dans différents clubs se doivent d’effectuer les activités organisées par leur clubs respectifs ou par l’amicale des étudiants, une activité est programmée ainsi pour chaque activité organisé l’école doit pouvoir savoir les informations suivantes :

* + - * Le titre de l’activité (Ex : JPO)
      * Le lieu de déroulement de l’activité
      * La date de déroulement de l’activité
      * Le club organisateur de l’activité
      * La description de l’activité
      * Les différents participants à l’activité

**1.2.2.2 CRITIQUE DE L’EXISTANT**

Nous avons pu constater que le manque de méthode informatisé pour cette gestion de la vie associatives des étudiants présente beaucoup de problèmes tel que :

* Manque d’informations relatives aux différents clubs de l’école
* Manque d’informations sur les activités organiser au sein de l’école
* Absence de suivi de chaque activité organiser

**1.2.2.3 PROPOSITION DE LA SOLUTION**

Pour répondre aux problèmes de notre étude, nous proposons comme solution la mise en place d’une plateforme de suivi des activités associatives des étudiants et elle aura comme avantages :

* Fournir les informations relatives aux différents clubs c’est-à-dire que chaque membre d’un club ou un administrateur pourra facilement avoir accès aux informations de son club en se connectant simplement à l’application
* Fournir les informations relatives aux différentes activités organisées par les clubs ou par l’amicale.
* Accéder à l’historique d’activités effectué par un étudiant.
* Avoir une traçabilité de toutes les activités organisées au sein de l’école.

Cette plateforme sera mise en place à base de nos compétences en programmation et chaque module sera entièrement développé par nous meme.

MAQUETTES D’ÉCRANS ET FONCTIONNALITÉS DE L’APPLICATION

**II. CHOIX DE LA MÉTHODE D’ANALYSE ET DE CONCEPTION**

**2.1. CHOIX DE LA MÉTHODE D’ANALYSE**

En ingénierie, une **méthode d'analyse et de conception** est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client.

Pour ce faire, on part d'un énoncé informel (le besoin tel qu'il est exprimé par le client, complété par des recherches d'informations auprès des experts du domaine fonctionnel, comme les futurs utilisateurs d'un logiciel), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus à traiter par le système se déroulent actuellement chez le client).

**2.1.1. PRÉSENTATION DE LA MÉTHODE MERISE ET DU LANGAGE UML**

**2.1.1.1. PRESENTATION DE LA METHODE MERISE**

MERISE est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.   
 La séparation des données et des traitements assure une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

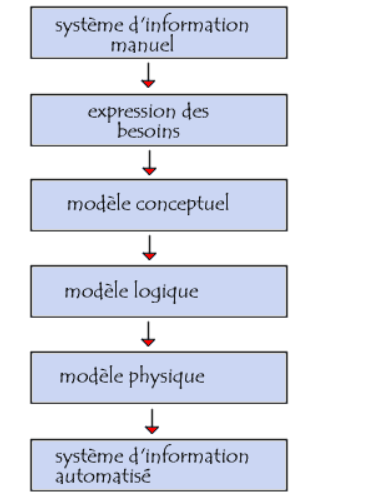
La méthode MERISE date de 1978-1979, et fait suite à une consultation nationale lancée en 1977 par le ministère de l'Industrie dans le but de choisir des sociétés de conseil en informatique afin de définir une méthode de conception de systèmes d'information.

Les deux principales sociétés ayant mis au point cette méthode sont le CTI (Centre Technique d'Informatique) chargé de gérer le projet, et le CETE (Centre d'Etudes Techniques de l'Equipement) implanté à Aix-en-Provence.

La conception du système d'information se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. Il s'agit donc de valider une à une chacune des étapes en prenant en compte les résultats de la phase précédente.

D'autre part, les données étant séparées des traitements, il faut vérifier la concordance entre données et traitements afin de vérifier que toutes les données nécessaires aux traitements sont présentes et qu'il n'y a pas de données superflues.

Cette succession d'étapes est appelée cycle d'abstraction pour la conception des systèmes d’information :



L'expression des besoins est une étape consistant à définir ce que l'on attend du système d'information automatisé, il faut pour cela :

* Faire l'inventaire des éléments nécessaires au système d'information
* Délimiter le système en s'informant auprès des futurs utilisateurs

Cela va permettre de créer le **MCC** (Modèle conceptuel de la communication) qui définit les flux d'informations à prendre en compte.

L'étape suivante consiste à mettre au point le **MCD** (Modèle conceptuel des données) et le **MCT** (Modèle conceptuel des traitements) décrivant les règles et les contraintes à prendre en compte.

Le modèle organisationnel consiste à définir le **MOT** (Modèle organisationnel des traitements) décrivant les contraintes dues à l'environnement (organisationnel, spatial et temporel).

Le modèle logique représente un choix logiciel pour le système d'information.

Le modèle physique reflète un choix matériel pour le système d'information.

**2.1.1.2. PRESENTATION DU LANGAGE UML**

Le **Langage de Modélisation Unifié**, de l'anglais *Unified Modeling Language* (**UML**), est un [langage](https://fr.wikipedia.org/wiki/Langage) de modélisation graphique à base de [pictogrammes](https://fr.wikipedia.org/wiki/Pictogramme) conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en [développement logiciel](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_logiciel) et en [conception orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet).

L'UML est le résultat de la fusion de précédents langages de modélisation objet : [Booch](https://fr.wikipedia.org/wiki/M%C3%A9thode_Booch), [OMT](https://fr.wikipedia.org/wiki/Object_Modeling_Technique), [OOSE](https://fr.wikipedia.org/wiki/OOSE). Principalement issu des travaux de [Grady Booch](https://fr.wikipedia.org/wiki/Grady_Booch), [James Rumbaugh](https://fr.wikipedia.org/wiki/James_Rumbaugh) et [Ivar Jacobson](https://fr.wikipedia.org/wiki/Ivar_Jacobson), UML est à présent un standard adopté par l'[Object Management Group](https://fr.wikipedia.org/wiki/Object_Management_Group) (OMG). UML 1.0 a été normalisé en janvier 1997 ; UML 2.0 a été adopté par l'OMG en juillet 2005.La dernière version de la spécification validée par l'OMG est UML 2.5.1 (2017).

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

* Activité d'un objet/logiciel
* Acteurs
* Processus
* Schéma de base de données
* Composants logiciels
* Réutilisation de composants

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une application logicielle, par exemple en langage Java, à partir des divers documents réalisés. La version actuelle, UML 2.5, propose 14 types de diagrammes dont 7 structurels et 7 comportementaux. A titre de comparaison, UML 1.3 comportait 25 types de diagrammes.

UML n'étant pas une méthode, l'utilisation des diagrammes est laissée à l'appréciation de chacun. Le diagramme de classes est généralement considéré comme l'élément central d'UML. Des méthodes, telles que le processus unifié proposé par les créateurs originels de UML, utilisent plus systématiquement l'ensemble des diagrammes et axant l'analyse sur les cas d'utilisation (« use case ») pour développer par itérations successives un modèle d'analyse, un modèle de conception, et d'autres modèles. D'autres approches se contentent de modéliser seulement partiellement un système, par exemple certaines parties critiques qui sont difficiles à déduire du code.

UML se décompose en plusieurs parties :

* Les vues : ce sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.
* Les diagrammes : ce sont des ensembles d'éléments graphiques. Ils décrivent le contenu des vues, qui sont des notions abstraites. Ils peuvent faire partie de plusieurs vues.
* Les modèles d'élément : ce sont les éléments graphiques des diagrammes.

**2.1.2. COMPARAISON ET JUSTIFICATION DU CHOIX DES APPROCHES MERISE ET UML**

**2.1.2.1. COMPARAISON DES APPROCHES MERISE ET UML**

Merise propose une approche descendante où le système réel est décomposé en activités, elles-mêmes déclinées en fonctions. Les fonctions sont composées de règles de gestion, elles-mêmes regroupées en opérations. Ces règles de gestion au niveau conceptuel génèrent des modules décomposés en modules plus simples et ainsi de suite jusqu'à obtenir des modules élémentaires... Les limites d'une telle approche résident dans le fait que les modules sont difficilement extensibles et exploitables pour de nouveaux systèmes.

Dans UML les fonctions cèdent la place aux cas d'utilisation qui permettent de situer les besoins de l'utilisateur dans le contexte réel. A chaque scénario correspond des diagrammes d'interaction entre les objets du système et non pas un diagramme de fonction... La méthode Merise ressemble à la méthode UML pour la phase de modélisation de la base de données. La différence principale est que Merise est une méthode d'analyse, et UML un langage de modélisation de données. UML ne propose pas de cycle précis : les organisations sont libres de choisir le cycle qui leur convient.

UML fonctionne sur un principe d’itérations qui ne s’oppose pas aux phases définies dans MERISE.

MERISE découpe plus au travers de ses phases l’analyse métier et l’architecture logicielle. Dans UML, l’architecture logicielle a une place prépondérante et est intégrée très en amont dans l’élaboration du système d’information

Dans UML, l’avancement du projet est mesuré par le nombre de cas d’utilisation, de classes... réellement implantées et non par la documentation produite ce qui est le cas dans Merise. Les itérations servent en outre à répartir l’intégration et les tests tout au long du processus d’élaboration du système d’information.

**2.1.2.2. JUSTIFCATION DU CHOIX DE LA MÉTHODE**

Merise et UML ont des caractéristiques voisines au niveau de la modélisation des bases de données mais également des points de divergence. En effet :

* + La méthode MERISE nécessite une démarche par étape qui favorise la qualité de chaque modèle avec ses différents niveaux de validations. Alors que le langage UML n’impose pas de méthode de travail particulière.
  + MERISE présente l'intérêt d'avoir des modèles logiques moins détaillés facilement compréhensibles. En revanche UML conçu pour s'adapter à n'importe quel langage de programmation orientée objet (POO), présente plusieurs modèles (diagrammes) dont leurs compréhensions nécessitent une grande attention.
  + MERISE est moins préférable. Malgré sa clarté, il manque une précision du fait qu'elle est éloignée du langage donc difficile à implémenter alors mais UML intègre les éléments communs des différents langages, sa volonté est d'être fidèle à la réalisation finale. Elle est beaucoup plus complète avec ses différents diagrammes.
  + Pour en finir avec l'exploitabilité, MERISE est une méthode plus généraliste. Elle donne une vue globale de la solution sans autant entrer dans les petits détails. Contrairement à UML qui est conçu pour l'implémentation objet avec ses différents détails et sa portabilité (s'adapte à n'importe quelle plateforme) elle est donc plus exploitable.

L'une ou l'autre présente des avantages et des inconvénients. Il est réservé au concepteur de choisir la méthode la mieux adaptée pour son cas. Si on cherche la précision et l'exploitabilité UML devance MERISE. En revanche si c'est la clarté et l'accessibilité qui sont en question MERISE est préférable. Dans notre cas, on va gérer des données plus complexes, donc il est préférable d’utiliser UML, vu qu’elle rend la modélisation plus simple à implémenter.

**2.2. ANALYSE ET CONCEPTION DE LA SOLUTION**

**2.2.1. NOTION DE DIAGRAMME**

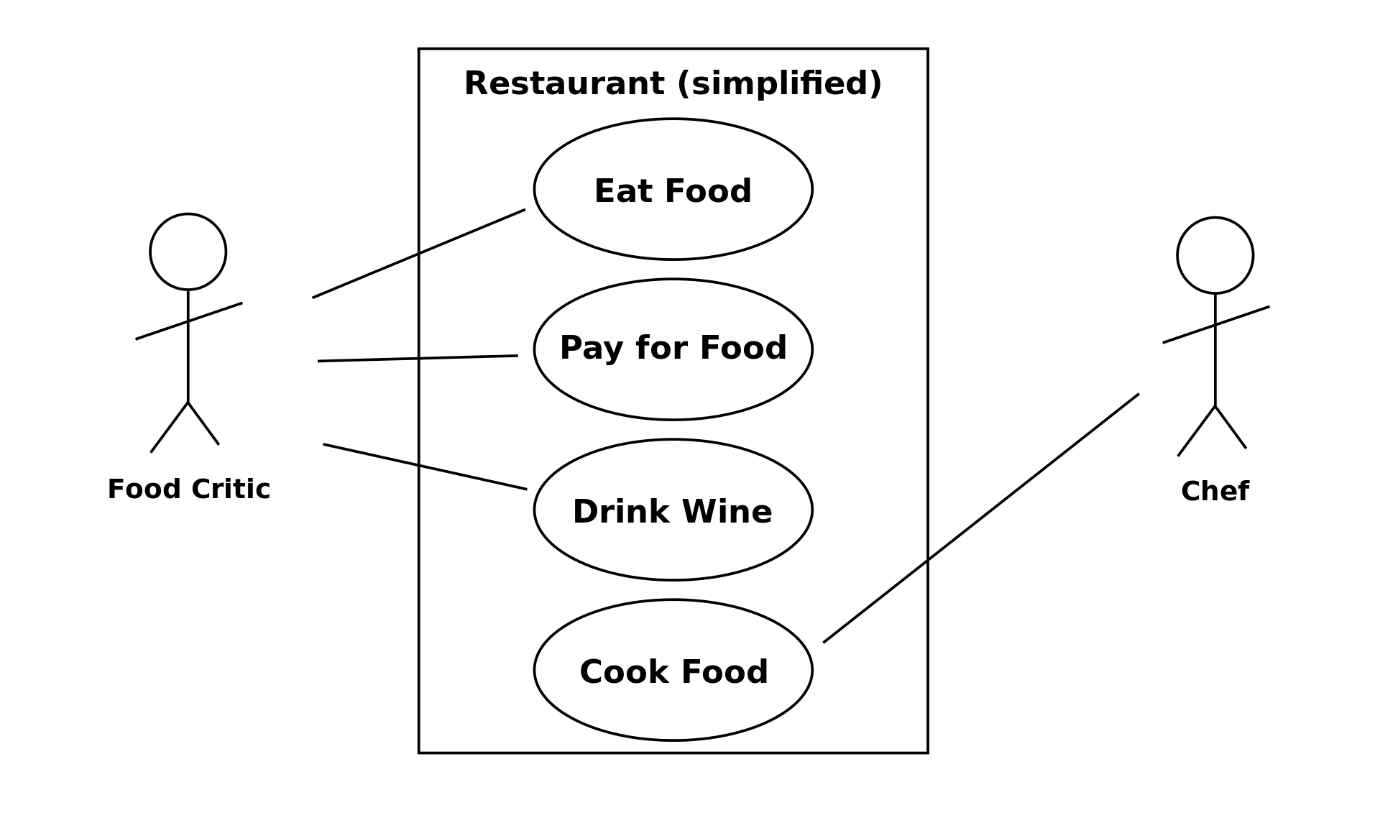
**2.2.1.1. DIAGRAMME DE CAS D’UTILISATION**

Les diagrammes de cas d'utilisation (DCU) sont des [diagrammes UML](https://fr.wikipedia.org/wiki/Unified_Modeling_Language#Les_diagrammes) utilisés pour donner une vision globale du comportement fonctionnel d'un système [logiciel](https://fr.wikipedia.org/wiki/Logiciel). Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les [cas d'utilisation](https://fr.wikipedia.org/wiki/Cas_d%27utilisation) sont plus appropriés. 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 (actors), ils interagissent avec les cas d'utilisation (use cases).

[UML](https://fr.wikipedia.org/wiki/Unified_Modeling_Language) définit une notation graphique pour représenter les cas d'utilisation, cette notation est appelée diagramme de cas d'utilisation. UML ne définit pas de standard pour la forme écrite de ces cas d'utilisation, et en conséquence il est aisé de croire que cette notation graphique suffit à elle seule pour décrire la nature d'un cas d'utilisation. Dans les faits, une notation graphique peut seulement donner une vue générale simplifiée d'un cas ou d'un ensemble de cas d'utilisation. Les diagrammes de cas d'utilisation sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation. Cela permet donc de comprendre qui est l'acteur et ce que le système doit réaliser.

Ils permettent de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système ! Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les use case (cas d'utilisation) sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation (éventuellement le nom est placé dans l'ellipse). Un acteur et un cas d'utilisation sont mis en relation par une association représentée par une ligne.

**Représentation graphique d’un diagramme de cas d’utilisation**



**2.2.1.2. DIAGRAMME DE SÉQUENCE SYSTÈME**

Les diagrammes de séquences sont la représentation graphique des [interactions](https://fr.wikipedia.org/wiki/Unified_Modeling_Language) entre les [acteurs](https://fr.wikipedia.org/wiki/Acteur_(UML)) et le système selon un ordre chronologique dans la formulation [Unified Modeling Language](https://fr.wikipedia.org/wiki/Unified_Modeling_Language).

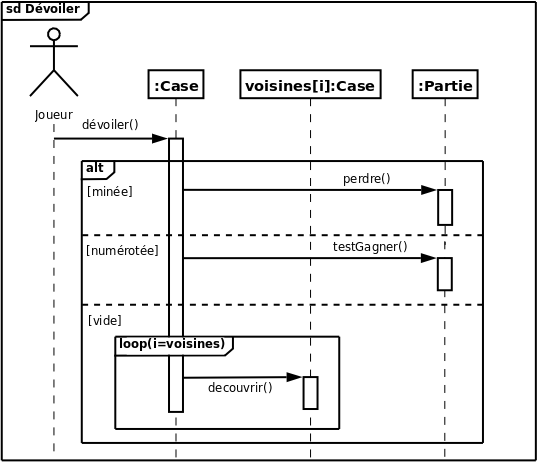
Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un [scénario](https://fr.wikipedia.org/wiki/Sc%C3%A9nario_(informatique)) d'un [Diagramme des cas d'utilisation](https://fr.wikipedia.org/wiki/Diagramme_des_cas_d%27utilisation). Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent à l'aide de messages.

Plusieurs types de messages (actions) peuvent transiter entre les acteurs et objets.

* Message simple : le message n'a pas de spécificité particulière d'envoi et de réception.
* Message avec durée de vie : l'expéditeur attend une réponse du récepteur pendant un certain temps et reprend ses activités si aucune réponse n'a lieu dans un délai prévu.
* Message synchrone : l'expéditeur est bloqué jusqu'au signal de prise en compte par le destinataire. Les messages synchrones sont symbolisés par des flèches barrées.
* Message asynchrone : le message est envoyé, l'expéditeur continue son activité que le message soit parvenu ou pris en compte ou non. Les messages asynchrones sont symbolisés par des demi-flèches.
* Message dérobant : le message est mis en attente dans une liste d'attente de traitement chez le récepteur.

**Représentation graphique d’un diagramme de séquence système**



**2.2.1.3. DIAGRAMME DE CLASSE**

Le diagramme de classes est un schéma utilisé en [génie logiciel](https://fr.wikipedia.org/wiki/G%C3%A9nie_logiciel) pour présenter les [classes](https://fr.wikipedia.org/wiki/Classe_(informatique)) et les [interfaces](https://fr.wikipedia.org/wiki/Interface_(informatique)) des systèmes ainsi que les différentes relations entre celles-ci. Ce [diagramme](https://fr.wikipedia.org/wiki/Diagramme) fait partie de la partie [statique](https://fr.wikipedia.org/wiki/Statique) d'[UML](https://fr.wikipedia.org/wiki/Unified_modeling_language) car il fait abstraction des aspects temporels et [dynamiques](https://fr.wikipedia.org/wiki/Dynamique).

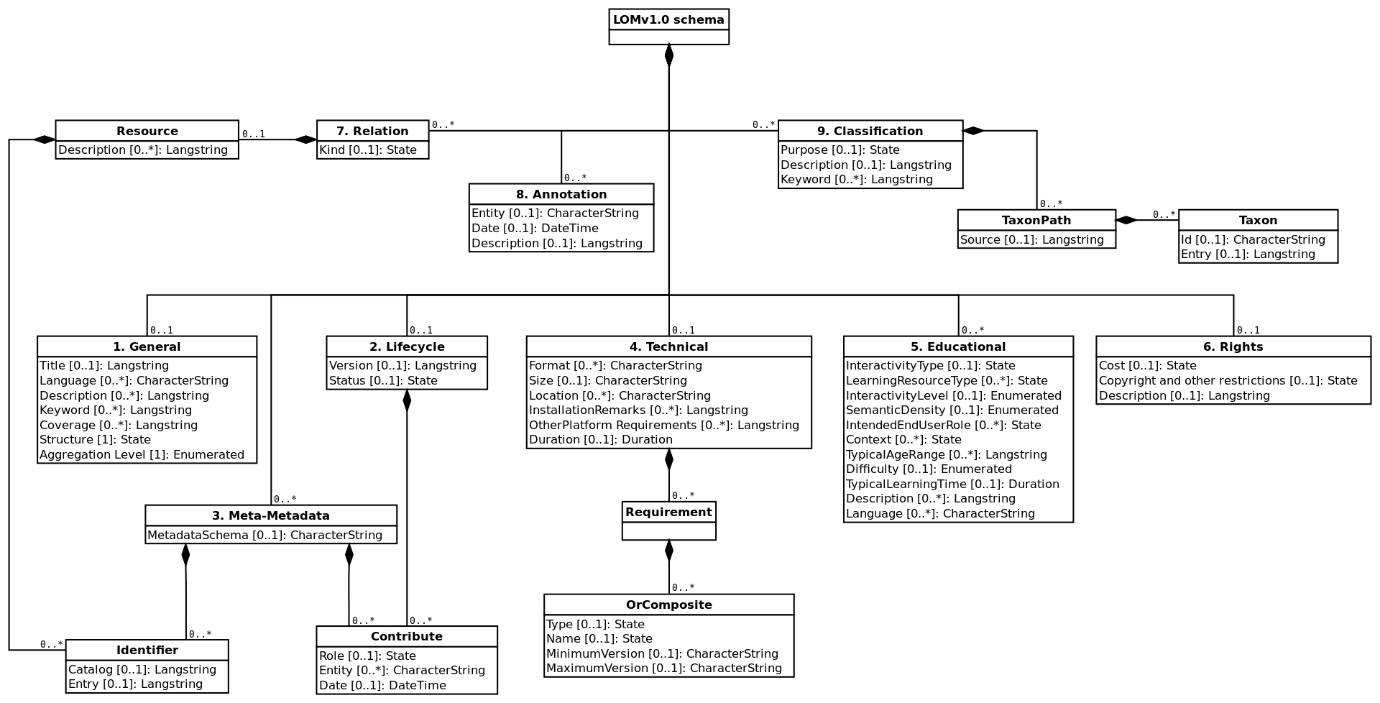
Une [classe](https://fr.wikipedia.org/wiki/Classe_(informatique)) décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les [instances](https://fr.wikipedia.org/wiki/Instance_(programmation)) de la classe.

Une [classe](https://fr.wikipedia.org/wiki/Classe_(informatique)) est un ensemble de [fonctions](https://fr.wikipedia.org/wiki/Fonction_(informatique)) et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la [programmation orientée objet](https://fr.wikipedia.org/wiki/Programmation_objet). Elles permettent de modéliser un [programme](https://fr.wikipedia.org/wiki/Programme_informatique) et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être liées entre elles grâce au mécanisme d'[héritage](https://fr.wikipedia.org/wiki/H%C3%A9ritage_(Informatique)) qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, chacune de ces relations est représentée par un arc spécifique dans le diagramme de classes.

Elles sont finalement instanciées pour créer des objets (une classe est un moule à objet : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).

**Représentation graphique d’un diagramme de séquence système**



**2.2.2 MODÈLISATION**

* **Définition d’un acteur**

Un acteur représente le type de rôle joué par une entité qui interagit avec le système modélisé. L’acteur peut consulter, modifier directement l’état du système, en fournissons ou en recevant des informations susceptible d’etre porteur de données.

* **Identification des acteurs et leurs rôles**

Les acteurs de notre système sont :

**Administrateur** : il se charge du bon fonctionnement de l’application ainsi que la gestion des utilisateurs et des droits d’accès.

**Président de club**: son rôle est de gérer les informations relatives aux différentes activités des clubs, il effectue les taches suivantes :

Ajouter des informations relatives au club

Ajouter les informations relatives aux activités

Accepter les demandes d’adhésion au club

Mettre à jour les informations sur les activités d’un club

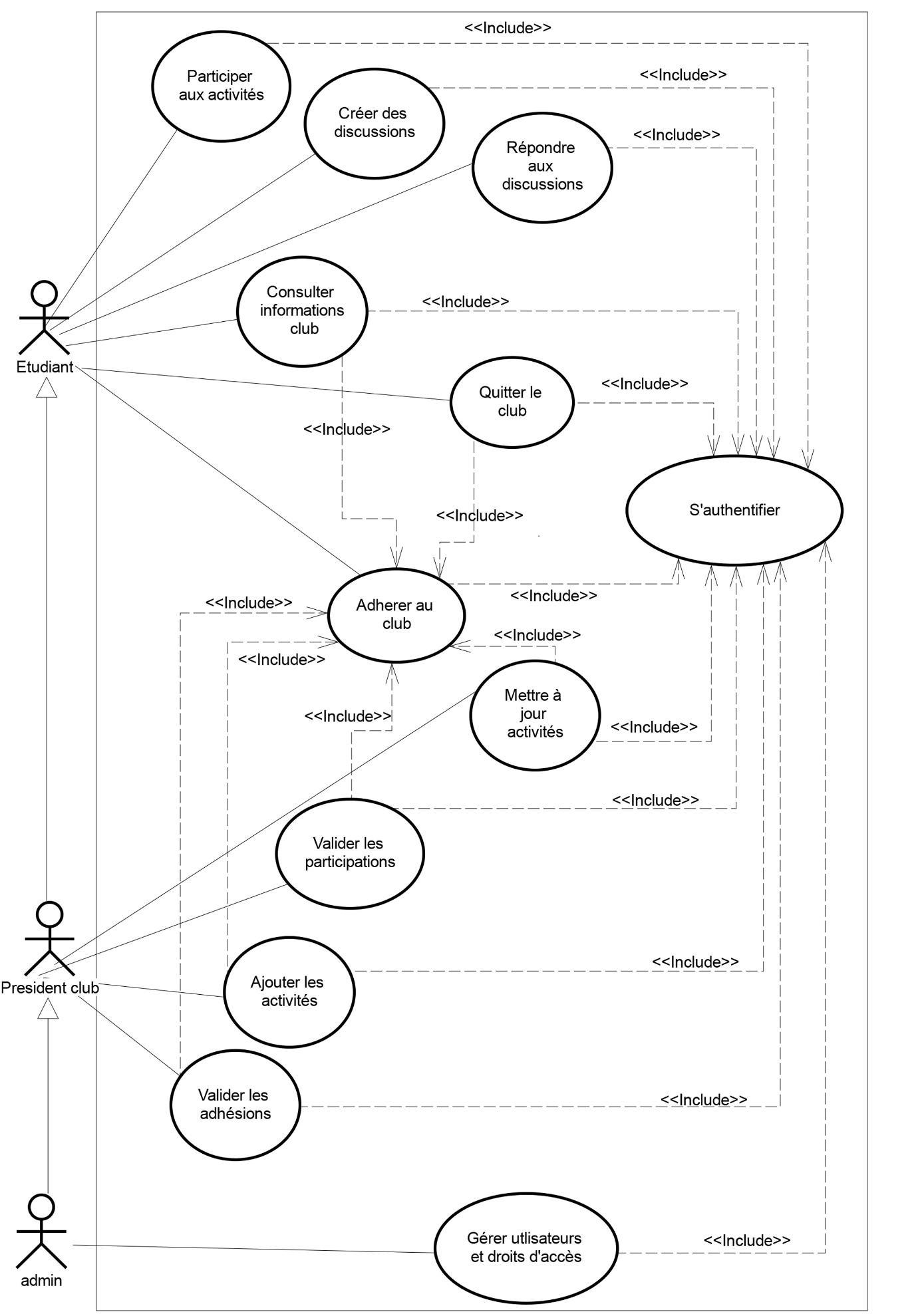
Valider les participations aux activités

**Etudiant :** sonrôle est d’utiliser les différentes fonctionnalités liées à l’application, il effectue les taches suivantes :

* Adhérer à un club
* Participer aux activités du club ou de l’amicale
* Créer des discussions
* Répondre aux discussions
* Consulter les informations d’un club ou de l’amicale
* Quitter un club

**Diagramme des cas d’utilisations du système**

Ce diagramme représente les cas d’utilisations de notre application. Ça nous donne une idée des principales fonctionnalités de notre application.



**Figure 1 : Diagramme de cas d’utilisation du système**

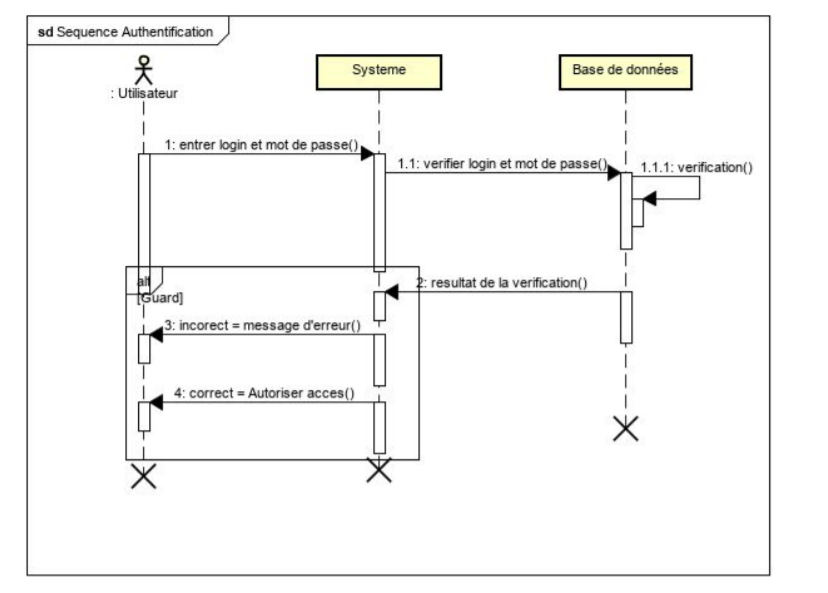
**Diagramme de séquence système**

Dans cette partie nous nous focaliserons sur les cas d’utilisation les plus important de notre système, ainsi nous allons les décrire de façon détaillée à l’intention de l’acteur lorsqu’il utilise le système et les séquences qu’il est susceptible d’effectuer.

**Description des scénarios**

|  |
| --- |
| **Identification**  **Titre :** Authentification  **Résumé :** Ce cas d’utilisation permet à l’utilisateur de se connecter au système  **Acteurs :** Administrateurs, Étudiants, Professeurs |
| **Description des scénarios :**  **Préconditions** : L’utilisateur doit etre crée dans la base de données et connaitre ses identifiants.  **Scénario** nominal :   1. L’utilisateur démarre l’application 2. Le système affiche le formulaire de connexion 3. L’utilisateur saisi son login et son mot de passe 4. Le système vérifie les identifiants de connexion de l’utilisateur 5. Le système renvoi à l’utilisateur la page correspondante à ses droits d’accès   **Enchainement alternatif :**  A1 : Le login ou/et le mot de passe de l’utilisateur est invalide réessayer (4.) le système indique à l’utilisateur la non validation de ses identifiants pour la première ou deuxieme fois.  **Enchainement d’erreur :**  E1 : Les identifiants de l’utilisateur sont définitivement invalides.  Post condition : L’utilisateur est authentifié avec succès. |

**Tableau 1 : Sommaire d’identification Authentification**

**Figure 2 : Diagramme de séquence système authentification**

|  |
| --- |
| **Identification**  **Titre :** Adhérer au club  **Résumé :** Ce cas d’utilisation permet à l’étudiant d’adhérer à un club  **Acteurs :** Etudiant |
| **Description des scénarios :**  **Préconditions** : L’étudiant s’authentifie  **Scénario** nominal :   1. Demande la liste des différents clubs 2. Le système lui renvoie la liste des différents clubs 3. L’étudiant sélectionne le club auquel il veut adhérer 4. Demande le formulaire d’adhésion 5. Le système traite sa demande et lui renvoie le formulaire d’adhésion 6. L’étudiant saisie les informations liées à son adhésion 7. Le système vérifie si les données sont saisies 8. Le système notifie à l’étudiant que sa demande d’adhésion va être traitée   **Enchainement Alternatif :**  A1 : Un problème est survenu lors du renvoie de la liste des différents clubs, reprendre le scénario nominal (1)  A2 : Un problème est survenu lors de la demande du formulaire d’adhésion, reprendre le scénario nominal (1)  **Enchainement d’erreur :**  E1 : Après la saisie, si les données sont mal saisies reprendre le scénario nominal (6)  **Enchainement d’exception :**  E1 : Annulation de la demande d’adhésion  **Post condition :** La demande d’adhésion a été pris en compte. |

**Tableau 2 : Sommaire d’identification Adhérer au club**

|  |
| --- |
| **Identification**  **Titre :** Participer aux activités  **Résumé :** Ce cas d’utilisation permet à l’étudiant de pouvoir participer aux activités  **Acteurs :** Etudiant |
| **Description des scénarios :**  Préconditions : L’étudiant s’authentifie  Scénario nominal :   1. Demande la liste des activités 2. Le système affiche la liste des activités 3. L’étudiant sélectionne l’activités auquel il veut participer 4. Demande à participer 5. Le système traite sa demande 6. Le système notifie à l’étudiant que sa demande de participation a été accepté   **Enchainement alternatif :**  A1 : Un problème est survenu lors de l’affichage de la liste des activités, reprendre le scénario nominal (1)  A2 : Un problème est survenu lors du traitement de la demande, reprendre le scénario nominal (3)  **Enchainement d’exception :**  E1 :Annulation de la participation  **Post condition :** Les informations ont été ajouté avec succès |

**Tableau 3 : Sommaire d’identification Participer aux activités**

|  |
| --- |
| **Identification**  **Titre :** Créer des discussions  **Résumé :** Ce cas d’utilisation permet à l’étudiant de pouvoir créer une discussion.  **Acteurs :** Etudiant |
| **Description des scénarios :**  Préconditions : L’étudiant s’authentifie  Scénario nominal :   1. Demande le formulaire de création de discussion 2. Le système renvoie le formulaire 3. L’étudiant saisie les informations concernant la discussion 4. Le système vérifie si les données sont saisies 5. Le système notifie à l’étudiant que la discussion à été posté avec succès.   **Enchainement alternatif :**  A1 : Un problème est survenu lors de la demande du formulaire de création de discussion, reprendre le scénario nominal (1)  **Enchainement d’erreur :**  E1 : Après la saisie si les données sont mal saisies reprendre le scénario nominal (3.).  **Enchainement d’exception :**  **E1 :** Annulation de la création de discussion.  **Post condition :** La discussion a été posté avec succès |

**Tableau 4 : Sommaire d’identification Créer des discussions**

|  |
| --- |
| **Identification**  **Titre :** Valider les adhésions  **Résumé :** Ce cas d’utilisation permet à au président d’un club de pouvoir valider les adhésions au club.  **Acteurs :** Président d’un club |
| **Description des scénarios :**  Préconditions : Le président du club s’authentifie et accède à son club  Scénario nominal :   1. Demande la liste des adhésions 2. Le système renvoie la liste des adhésions 3. Le président du club valide les adhésions 4. Le système traite les informations 5. Le système notifie au président qu’un nouveau membre a rejoint le club   **Enchainement alternatif :**  A1 : Un problème est survenu lors du renvoie de la liste des adhésions, reprendre le scénario nominal (1)  **Enchainement d’exception :**  **E1 :** Annulation de la validation.  **Post condition :** Un nouveau membre a rejoint le club |

**Tableau 5 : Sommaire d’identification Valider les adhésions**

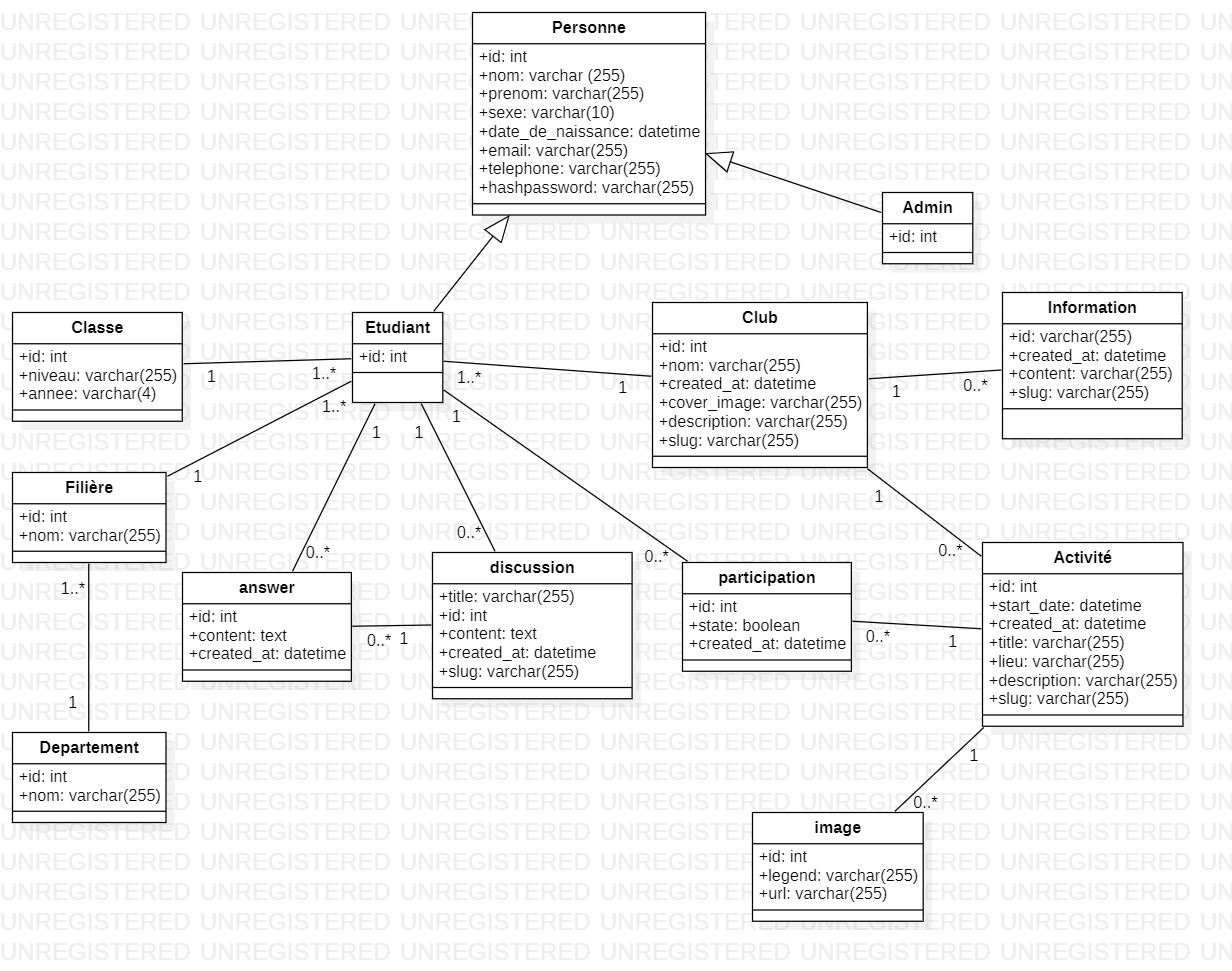
|  |
| --- |
| **Identification**  **Titre :** Valider les participations  **Résumé :** Ce cas d’utilisation permet à au président d’un club de pouvoir valider les participations à une activités.  **Acteurs :** Président d’un club |
| **Description des scénarios :**  Préconditions : Le président du club s’authentifie et accède à son club  Scénario nominal :   1. Demande la liste des activités 2. Le système renvoie la liste des activités 3. Le président du club sélectionne une activité 4. Demande la liste des participations 5. Le système renvoie la liste des participations 6. Le président du club valide ou invalide les participations 7. Le système traite les informations 8. Le système notifie au président et a l’étudiant que la participation a été validé ou invalidé.   **Enchainement alternatif :**  A1 : Un problème est survenu lors du renvoie de la liste des activités, reprendre scénario nominal (1)  A1 : Un problème est survenu lors du renvoie de la liste des participations, reprendre le scénario nominal (3)  **Enchainement d’exception :**  **E1 :** Annulation de la validation.  **Post condition :** Participation a été validé ou invalidé avec succès |

**Tableau 6 : Sommaire d’identification Valider les participations**

**Diagramme des classes du système**

Dans cette partie nous allons représenter toutes les classes qui vont interagir avec notre système.

**Figure 3 : Diagramme de classe du système**



**III. MISE EN OEUVRE**

**3.1 Système de gestion de base de données**

**3.1.1 Présentation**

Un système de gestion de base de données (abr. SGBD) est un logiciel système servant à stocker, à manipuler ou gérer, et à partager des données dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations. Un SGBD (en anglais DBMS pour database management system) permet d'inscrire, de retrouver, de modifier, de trier, de transformer ou d'imprimer les informations de la base de données. Il permet d'effectuer des comptes rendus des informations enregistrées et comporte des mécanismes pour assurer la cohérence des informations, éviter des pertes d'informations dues à des pannes, assurer la confidentialité et permettre son utilisation par d'autres logiciels. Selon le modèle, le SGBD peut comporter une simple interface graphique jusqu'à des langages de programmation sophistiqués. Les systèmes de gestion de base de données sont des logiciels universels, indépendants de l'usage qui est fait des bases de données. Ils sont utilisés pour de nombreuses applications informatiques, notamment les guichets automatiques bancaires, les logiciels de réservation, les bibliothèques numériques, les logiciels d’inventaires, les progiciels de gestion intégrés ou la plupart des blogs et sites web. Il existe de nombreux systèmes de gestion de base de données. En 2008, Oracle détenait près de la moitié du marché des SGBD avec MySQL et Oracle Database. Vient ensuite IBM avec près de 20 %, laissant peu de place pour les autres acteurs. Les SGBD sont souvent utilisés par d’autres logiciels ainsi que les administrateurs ou les développeurs. Ils peuvent être sous forme de composant logiciel, de serveur, de logiciel applicatif ou d’environnement de programmation.

**3.1.2 Etude comparative de quelques SGBD**

Il existe plusieurs types de SGBD (Système de gestion de base des données) dont chacun à des caractéristiques qui le diffère des autres. Nous pouvons citer entre autres :

* **MySQL** : MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur. C'est un logiciel libre, open source, développé sous double licence selon qu'il est distribué avec un produit libre ou avec un produit propriétaire. Dans ce dernier cas, la licence est payante, sinon c'est la licence publique générale GNU (GPL) qui s'applique. Un logiciel qui intègre du code MySQL ou intègre MySQL lors de son installation devra donc être libre ou acquérir une licence payante. Cependant, si la base de données est séparée du logiciel propriétaire qui ne fait qu'utiliser des API tierces (par exemple en C# ou PHP), alors il n'y a pas besoin d'acquérir une licence payante MySQL.
* **Microsoft Office Access :** édité par Microsoft, Microsoft Access (MS Access en abrégé) est une base de données relationnelle composé de plusieurs programmes dont le moteur de base de données Microsoft Jet, un éditeur graphique, une interface de type Query by Example pour interroger les bases de données, et le langage de programmation Visual Basic for Applications. Il permet de configurer, avec des assistants ou librement, des Formulaires et sous-formulaires de saisie, des états imprimables (avec regroupements de données selon divers critères, des totalisations et des sous-totalisations, conditionnelles ou non et éventuellement des sous-états, reliés ou non aux données principales de l'état), des pages html liées aux données d'une base, des macros et des modules VBA. Les formulaires et les états disposent aussi de leurs propres modules VBA. On accède aux objets par un navigateur de l’interface d'Access ou dans des applications programmées (par macro ou VBA) par les formulaires de l’interface homme/machine ainsi construits.
* **PostgreSQL :** c’est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD. Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB et Firebird), ou propriétaires (comme Oracle, MySQL, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.
* SQL Server : Il désigne couramment un serveur de base de données. La définition du SQL server est étroitement liée à celle du langage SQL (Structured Query Language), un langage informatique permettant d'exploiter des bases de données. Concrètement, un SQL server est un outil qui possède toutes les caractéristiques pour pouvoir accompagner l'utilisateur dans la manipulation, le contrôle, le tri, la mise à jour, et bien d'autres actions encore, de bases de données grâce au langage SQL. Le terme désigne également le nom donné au système de gestion de base de données (SGBD) commercialisé par Microsoft, ou plus précisément le nom du moteur de bases de données de ce SGDB produit par le fabricant de produits informatiques américain.
* Oracle DATABASE : fourni par la société Oracle Corporation (Leader mondiale des bases de données), oracle database est un système de gestion de base des données développé par Larry Ellison. Oracle dispose de plusieurs fonctionnalités et nous pouvons énumérer quelques-uns :
  + - SQL (Standard Query Language)
    - PL/SQL, langages de programmation, utilisé pour créer des procédures, des fonctions et des déclencheurs.
    - Java, ce langage de programmation est aussi utilisable pour créer des triggers lors de l'insertion, la modification ou l'effacement d'éléments
    - Montage de la base de données sur plusieurs serveurs (grid en 10g, rac en 9i)
    - Spatial, pour permettre la gestion de données géographiques
    - Partitionnements physiques des données en sous-ensembles pour optimiser les temps d'accès
    - Moteur OLAP intégré, stockant les cubes sous forme de BLOB (Binary Large Objects)
    - Gestion de très grands volumes de données, taille maximale de 65 536 fichiers de 128 Téra octet (To) chacun en utilisant les Big Files (grandes fichiers) de la version 10gR2 ou 10.2
    - Réplication des données selon différents modes synchrones ou asynchrones de tout ou partie d'une base de données : voir Oracle Data Guard (en) ou encore la solution Golden Gate (le couteau suisse d'Oracle tant il permet de faire de choses différentes à la fois).

**3.1.3 Choix et justification**

Le système de gestion de base de données (SGBD) est aujourd'hui au cœur des systèmes opérationnels et analytiques. En effet, si les données constituent l'élément vital de l'entreprise, le SGBD est l'organe qui permet de les stocker, de les administrer, de les sécuriser et de les mettre à la disposition des applications et des utilisateurs. Il existe cependant de nombreux types de SGBD sur le marché, offrant chacun leurs propres avantages et inconvénients. Dans le cadre de notre travail nous utiliserons MySQL pour le stockage et la gestion des données pour sa simplicité et son accessibilité.

**3.2 Langage de programmation**

**3.2.1 Présentation**

Un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire et de significations. Les langages de programmation permettent de décrire d'une part les structures des données qui seront manipulées par l'appareil informatique, et d'autre part d'indiquer comment sont effectuées les manipulations, selon quels algorithmes. Ils servent de moyens de communication par lesquels le programmeur communique avec l'ordinateur, mais aussi avec d'autres programmeurs ; les programmes étant d'ordinaire écrits, lus, compris et modifiés par une équipe de programmeurs. Un langage de programmation est mis en œuvre par un traducteur automatique : compilateur ou interprète. Un compilateur est un programme informatique qui transforme dans un premier temps un code source écrit dans un langage de programmation donné en un code cible qui pourra être directement exécuté par un ordinateur, à savoir un programme en langage machine ou en code intermédiaire, tandis que l’interprète réalise cette traduction « à la volée ».

**3.2.2 Etude comparative**

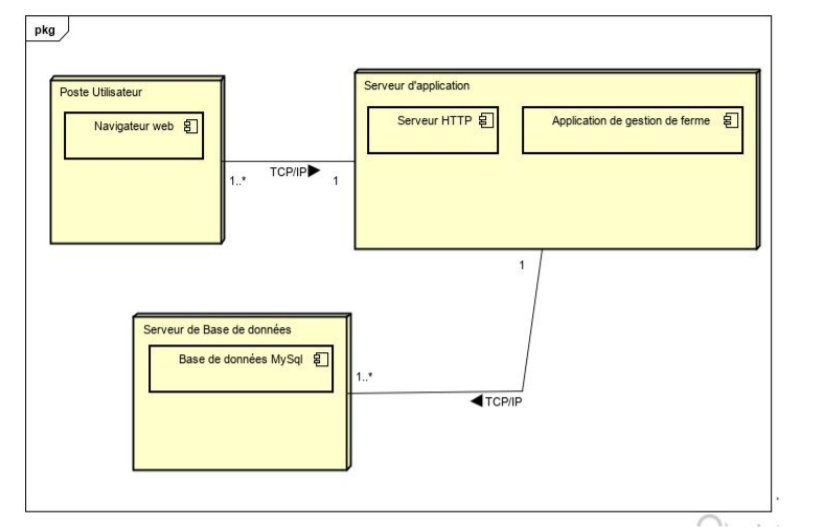
* **Le langage C :** c’est un langage de programmation impératif généraliste, de bas niveau. Inventé au début des années 1970 dans les Laboratoires Bell pour réécrire UNIX, C’est devenu un des langages les plus utilisés, encore de nos jours. De nombreux langages plus modernes comme C++, C#, Java et PHP ou Javascript ont repris une syntaxe similaire au C et reprennent en partie sa logique. C offre au développeur une marge de contrôle importante sur la machine (notamment sur la gestion de la mémoire) et est de ce fait utilisé pour réaliser les « fondations » (compilateurs, interpréteurs…) de ces langages plus modernes. En outre, C propose un éventail de types entiers et flottants conçus pour pouvoir correspondre directement aux types de donnée supportés par le processeur. Il fait un usage intensif des calculs d'adresse mémoire avec la notion de pointeur. Enfin, hormis les types de base, C supporte les types énumérés, composés, et opaques. Il ne propose en revanche aucune opération qui traite directement des objets de plus haut niveau (fichier informatique, chaîne de caractères, liste, table de hachage …).
* **C++ :** c’est un langage de programmation compilé permettant la programmation sous de multiples paradigmes (comme la programmation procédurale, orientée objet ou générique). Ses bonnes performances, et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique. On peut considérer que C++ « est du C » avec un ajout de fonctionnalités. Cependant, certains programmes syntaxiquement corrects en C ne le sont pas en C++, à commencer bien sûr par ceux qui font usage d'identificateurs correspondant à des mots clefs en C++.
* JAVA : Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (Cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld (la société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java). Java permet de développer des applications autonomes mais aussi, et surtout, des applications client-serveur. C'est la garantie de portabilité qui a fait la réussite de Java dans les architectures client-serveur en facilitant la migration entre serveurs, très difficile pour les gros systèmes. Une autre particularité de Java est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (JVM) en faisant abstraction du système d'exploitation.
* **C# :** C# (prononcé C Sharp) est un langage de programmation orientée objet fortement typé, dérivé de C et de C++, ressemblant au langage Java. Il est destiné à développer sur la plateforme Microsoft .NET. Il est utilisé pour développer des applications web, ainsi que des applications de bureau, des services web, des commandes, des widgets ou des bibliothèques de classes. En C#, une application est un lot de classes où une des classes comporte une méthode Main, comme cela se fait en Java. C# apporte un typage sûr, ainsi que les possibilités d'encapsulation, d'héritage et de polymorphisme des langages orientés objet. En C# tous les types sont des objets. Le langage comporte un ramasse miettes et un système de gestion d'exceptions.
* PHP : créé en 1994 par Rasmus Lerdorf, PHP est un langage de script utilisé le plus souvent côté serveur. Dans cette architecture, le serveur interprète le code PHP des pages web demandées et génère du code (HTML, XHTML, CSS par exemple) et des données (JPEG, GIF, PNG par exemple) pouvant être interprétés et rendus par un Navigateur web. PHP peut également générer d'autres formats comme le SVG et le PDF. Il a été conçu pour permettre la création d'applications dynamiques, le plus souvent développées pour le Web. PHP est le plus souvent couplé à un serveur Apache bien qu'il puisse être installé sur la plupart des serveurs HTTP tels que IIS ou Nginx. Il est multi-plateforme : autant sur Linux qu'avec Windows il permet aisément de reconduire le même code sur un environnement à peu près semblable (prendre en compte les règles d'arborescences de répertoires qui peuvent changer).
* **PYTHON :** c’est un langage de programmation interprété, multi- paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Small talk et Tcl.
* JAVASCRIPT : JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js ou de Deno. Il s'exécute à l'aide d'un programme spécial appelé " Moteur Javascript ". C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet de créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe. Le langage supporte le paradigme objet, impératif et fonctionnel. JavaScript est le langage possédant le plus large écosystème grâce à son gestionnaire de dépendances npm, avec environ 500 000 paquets en août 2017. Il ne faut pas confondre JavaScript avec Java car ces deux n’ont rien à voir en commun à part la similitude de leur nom.

**3.2.3 Choix et justification**

Après avoir étudié et comparé les langages de programmation ci-dessus, nous avons choisi PHP comme langage d’implémentation de notre application. Avec son point fort qui est le développement web (dynamique) et aussi ses avantages et sa simplicité, PHP attire notre intérêt.

**3.3 Diagramme de déploiement**

Le langage de modélisation unifié (UML) est le langage standard que de nombreux ingénieurs logiciels et de nombreuses entreprises utilisent pour avoir une vue d'ensemble de systèmes complexes. Un diagramme de déploiement est un des types de diagrammes créés avec ce langage. Dans le contexte du langage de modélisation unifié (UML), un diagramme de déploiement fait partie de la catégorie des diagrammes structurels, car il décrit un aspect du système même. Dans le cas présent, le diagramme de déploiement décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. On appelle artefact l'information qui est générée par le logiciel. Attention, ne confondez cette utilisation du terme avec celle qui existe dans d'autres approches de modélisation comme le BPMN. Les diagrammes de déploiement sont constitués de plusieurs formes UML. Les boîtes en trois dimensions, appelées nœuds, représentent les composants du système, qu'ils soient logiciels ou matériels. Les lignes entre les nœuds indiquent les relations et les petites formes à l'intérieur des boîtes représentent les artefacts logiciels qui sont déployés.



**Figure 4 : Diagramme de déploiement**

**3.4 Présentation de l’application**