**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 de suivi de la vie associatives 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. 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 des 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 de suivi de la vie associatives 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 étudiantes :

* 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 d’évaluation des étudiants**

Pour sa gestion des étudiants à distance l’école a une plateforme qui comporte plusieurs modules. Chaque module nous permet de faire une tache spécifique de l’enseignement, nous pouvons noter le module d’enseignement qui nous permet de pouvoir faire des cours à distance, le module d’évaluation qui permet d’évaluer les étudiants à distance, le module du calendrier qui nous permet de savoir les différentes activités programmées etc…

Concernant les évaluations ceux-ci mettent en avant deux acteurs le professeur et l’étudiant.

Le professeur programme les évaluations afin de permettre aux étudiants de pouvoir les faire.

Avant de programmer une évaluation le professeur s’assure de définir les informations relatives à celle-ci, c’est-à-dire qu’il doit avoir :

* La classe
* Le questionnaire
* La date de debut
* La date de fin

Après que le professeur a eu toutes les informations nécessaires à l’évaluation, il doit maintenant inscrire tout ça dans la plateforme.

Pour ce faire, il se connecte sur plateforme, renseigne les informations nécessaires à l’évaluation, vérifie si celle-ci sont conformes et valide l’évaluation.

L’étudiant fait les évaluations programmées au préalable par le professeur.

**1.2.2.2 CRITIQUE DE L’EXISTANT**

Nous avons pu constater que la centralisation des informations relatives aux étudiants dans une seule plateforme présente beaucoup de problème qui sont :

* Un risque élevé de voir paralyser les enseignements en cas de bug
* Un manque d’ergonomie
* Une absence de suivi des évaluations après que ceux-ci en déjà été fait
* Une absence d’historique des notes relatives aux évaluations passées déjà effectué.

**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 d’évaluation à distance et elle aura comme avantages :

* D’être totalement indépendant de la plateforme principale d’enseignement à distance, ce qui réduit le risque de voir paralyser les enseignements en cas de bug
* Une interface utilisateur pensée et conçu pour les évaluations, ce qui rendra plus ergonomique la plateforme
* Un module qui permettra de gérer les notes ainsi que leurs historiques
* Cette solution facilitera également la gestion du suivi des évaluations déjà effectué.

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 DE LA SOLUTION**

**II. CHOIX DE LA MÉTHEDO 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.

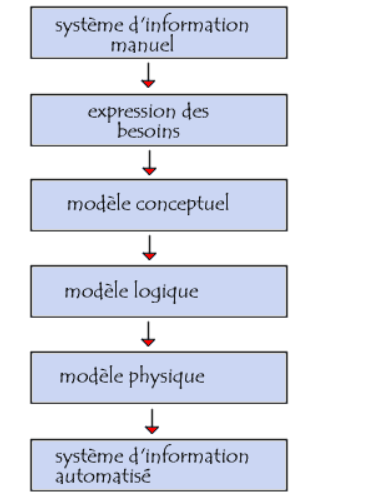
La méthode MERISE date de 1978-1979, et fait suite à une consultation nationale lancée en 1977 par le ministère de l'Industrie dans le but de choisir des sociétés de conseil en informatique afin de définir une méthode de conception de systèmes d'information.

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" \o "Méthode 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.

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

**2.2.1. NOTION DE DIAGRAMME**

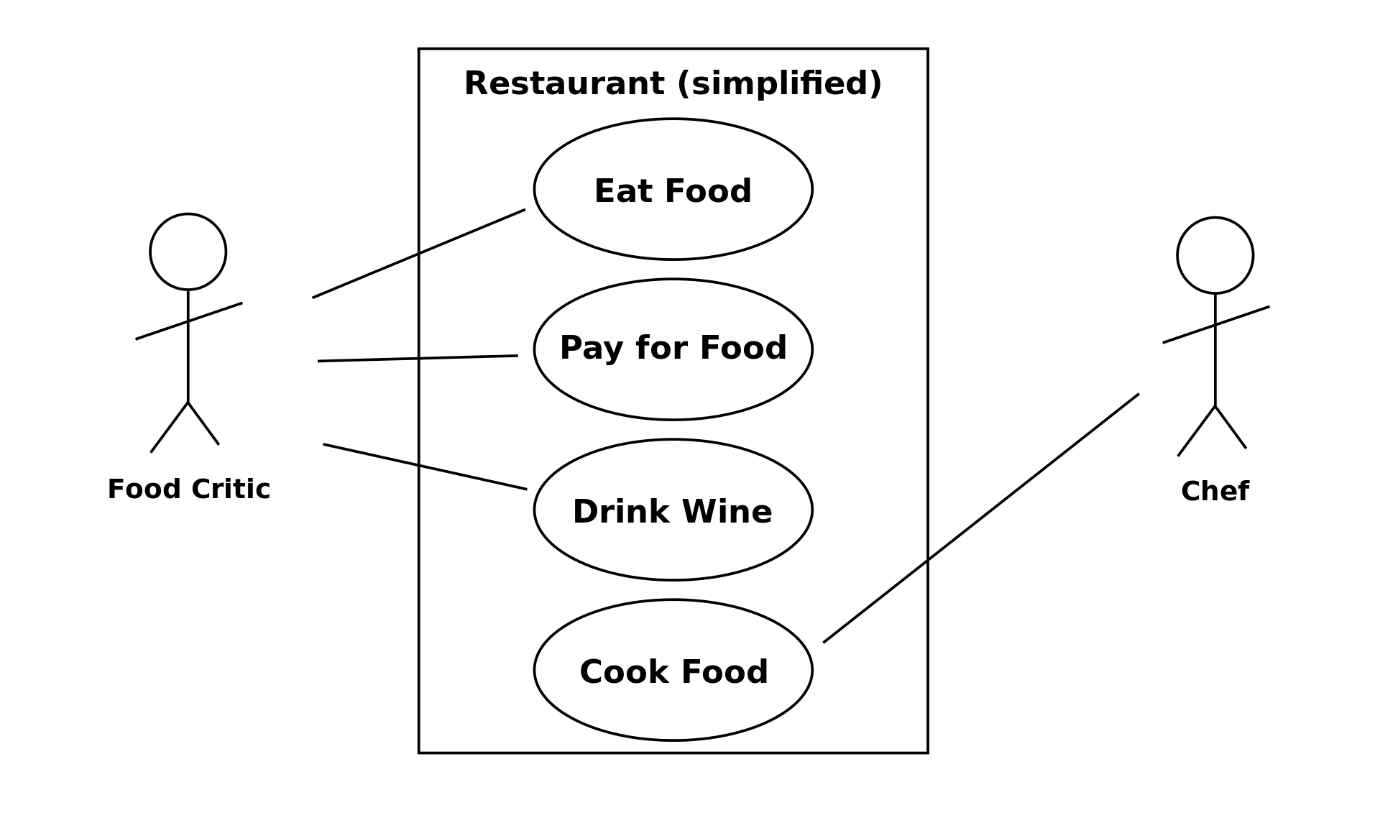
**2.2.1.1. DIAGRAMME DE CAS D’UTILISATION**

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" \o "Unified Modeling Language).

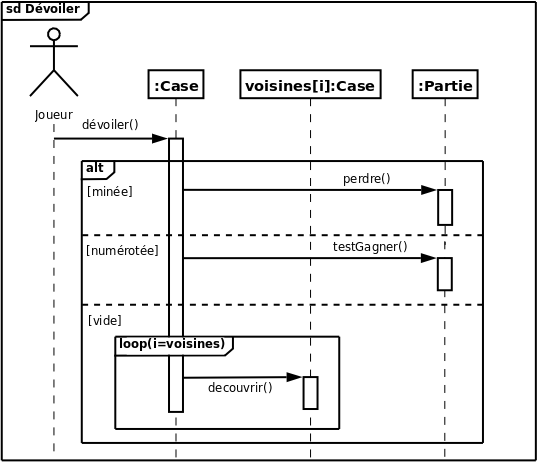
Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un [scénario](https://fr.wikipedia.org/wiki/Sc%C3%A9nario_(informatique)) d'un [Diagramme des cas d'utilisation](https://fr.wikipedia.org/wiki/Diagramme_des_cas_d%27utilisation). Dans un souci de simplification, on représente 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).

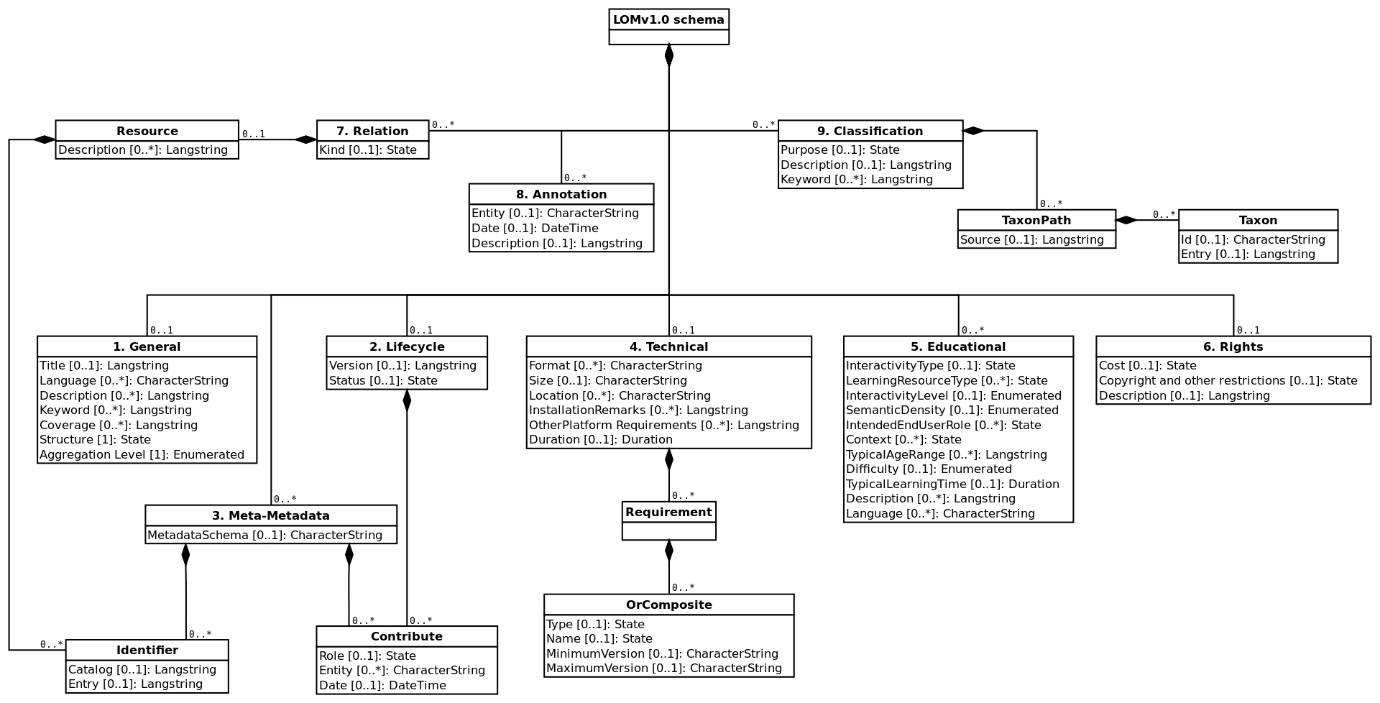
Une [classe](https://fr.wikipedia.org/wiki/Classe_(informatique)) décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les [instances](https://fr.wikipedia.org/wiki/Instance_(programmation)) de la classe.

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. 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.

**Coordonnateur**: son rôle est de gérer les informations relatives aux différents alumnis, il effectue les taches suivantes :

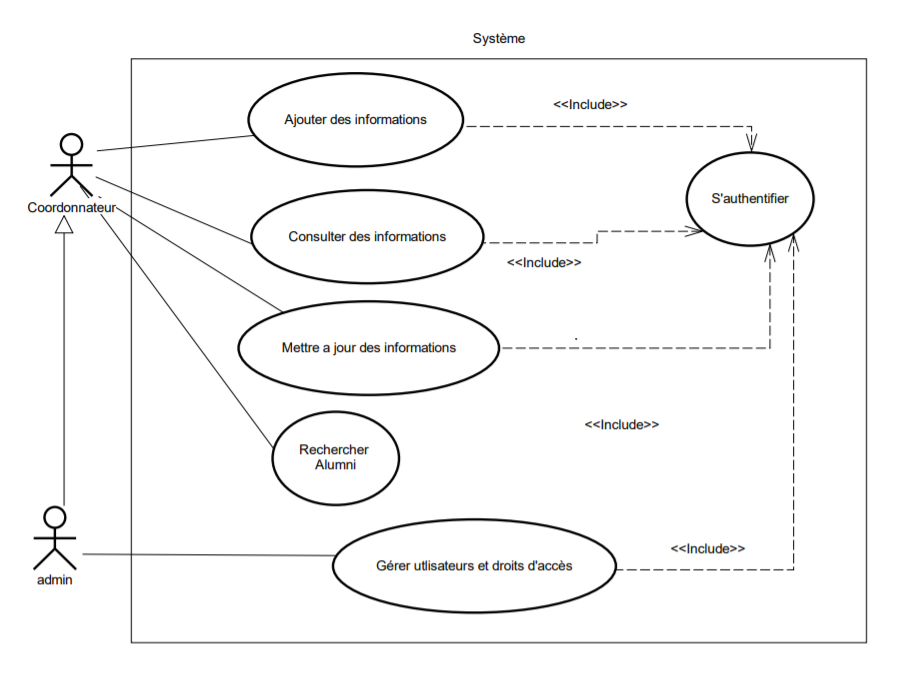
Consulter les informations relatives aux alumnis

Mettre à jour les informations relatives aux alumnis

Ajouter les informations relatives aux alumnis

**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**

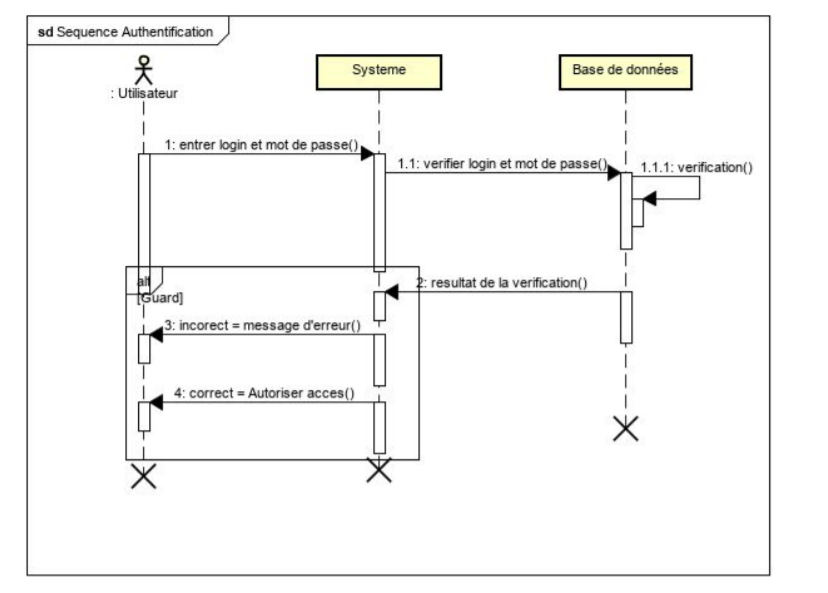
Dans cette partie nous nous focaliserons sur les cas d’utilisation les plus important de notre système, ainsi nous allons les décrire de façon détaillée à l’intention de l’acteur lorsqu’il utilise le système et les séquences qu’il est susceptible d’effectuer.

**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 :** Consulter  **Résumé :** Ce cas d’utilisation permet au coordonnateur de consulter des informations  **Acteurs :** Coordonnateur |
| **Description des scénarios :**  **Préconditions** : Le coordonnateur s’authentifie  **Scénario** nominal :   1. Demande la liste des différents alumnis 2. Le système lui renvoie la liste des alumnis   **Post condition :** La liste a été afficher avec succès. |

**Tableau 2 : Sommaire d’identification Consulter les informations**

|  |
| --- |
| **Identification**  **Titre :** Ajouter des informations  **Résumé :** Ce cas d’utilisation permet au coordonnateur d’ajouter des informations concernant un alumni.  **Acteurs :** Coordonnateur |
| **Description des scénarios :**  Préconditions : Le coordonnateur s’authentifie  Scénario nominal :   1. Consulte la liste des alumnis 2. Sélectionne l’alumni dont les informations doivent etre ajouter 3. Le coordonnateur demande le formulaire d’ajout d’informations 4. Le coordonnateur saisie les informations liées à l’alumnis 5. Le système vérifie si les données sont saisies 6. Le système notifie au coordonnateur que les informations ont été ajoutés avec succès.   **Enchainement alternatif :**  A1 : Un problème est survenu lors de la demande du formulaire d’ajouts d’informations, reprendre le scénario nominal (1)  **Enchainement d’erreur :**  E1 : Après la saisie si les données sont mal saisies reprendre le scénario nominal (4.).  **Enchainement d’exception :**  **E1 :** Annulation de l’ajout des informations.  **Post condition :** Les informations ont été ajouté avec succès |

**Tableau 3 : Sommaire d’identification Ajouter des informations**

|  |
| --- |
| **Identification**  **Titre :** Mise à jours informations  **Résumé :** Ce cas d’utilisation permet au coordonnateur de mettre à jour les informations concernant un alumni.  **Acteurs :** Coordonnateur |
| **Description des scénarios :**  Préconditions : Le coordonnateur s’authentifie  Scénario nominal :   1. Consulte la liste des alumnis 2. Sélectionne l’alumni dont les informations doivent etre ajouter 3. Le coordonnateur demande le formulaire de mise à jour des informations. 4. Le coordonnateur saisie les informations liées à l’alumnis 5. Le système vérifie si les données sont saisies 6. Le système notifie au coordonnateur que les informations ont été ajoutés avec succès.   **Enchainement alternatif :**  A1 : Un problème est survenu lors de la demande du formulaire de mise a jour des informations, reprendre le scénario nominal (1)  **Enchainement d’erreur :**  E1 : Après la saisie si les données sont mal saisies reprendre le scénario nominal (4.).  **Enchainement d’exception :**  **E1 :** Annulation de la mise à jour des informations.  **Post condition :** Les informations ont été modifié avec succès |

**Tableau 4 : Sommaire d’identification Mise à jour des informations**

**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**

