
ADRAR PÔLE NUMERIQUE
> INFRASTRUCTURES SYSTÈMES & RÉSEAUX
> CYBERSÉCURITÉ INFRASTRUCTURES & APPLICATIONS
> DEVOPS / SCRIPTING & AUTOMATISATION
> DEVELOPPEMENT WEB & MOBILE
> TRANSFORMATION NUMERIQUE DES ENTREPRISES
www.adrar-numerique.com

Introduction

UML : Unified Modeling Language

Langage graphique qui permet de créer des modèles de systèmes utilisant des **objets**.

2 « styles » de programmation :

- **Programmation procédurale** :
  - Décomposé en fonctions
  - Code structuré mais peu modulaire
- **Programmation orientée objet** :
  - Décomposé en entités (objets)
  - Chaque objet embarque les données et traitements qui lui sont associés
  - Code très modulaire

Introduction

La modélisation ?

Qu'est-ce qu'un modèle ?

- Représentation abstraite et simplifiée d'une entité ou d'un système en vue de le décrire, l'expliquer ou le prévoir. Exemples :
  - Plans et schémas (architecture, électrique, …)
  - Modèles météorologiques

Intérêt de la modélisation :

- Langage commun entre les différents intervenants (MOA et MOE)
- Abstraction de la partie technique -> réutilisabilité
- Réduction de la complexité

Introduction

Pour convenablement concevoir en orienté objet (classes, interfaces, attributs et méthodes) il est nécessaire d'avoir une bonne vision d'ensemble de l'application à développer.

La modélisation permet d'organiser les idées, les documenter afin de réfléchir au mieux à comment les réaliser.

Au milieu des années 90, 3 méthodes de modélisation orientée objet faisaient consensus :

- **OMT** de James Rumbaugh (General Electric) fournit une représentation graphique des aspects statique, dynamique et fonctionnel d'un système
- **OOD** de Grady Booch, définie pour le Department of Defense, introduit le concept de paquetage (package)
- **OOSE** d'Ivar Jacobson (Ericsson) fonde l'analyse sur la description des besoins des utilisateurs (cas d'utilisation, ou use cases).

Dès 1995, les concepteurs de ces 3 méthodes se mirent d'accord et collaborèrent pour définir un langage de modélisation commun : l'UML.

Diagrammes

UML propose 13 types de diagrammes :
Les diagrammes qui nous intéressent le plus en tant que développeur / concepteur sont les diagrammes soulignés suivants :

- **Diagrammes structurels ou diagrammes statiques (UML Structure)**
  - diagramme de classes (Class diagram)

    Considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens.
  - diagramme d'objets (Object diagram)

    Permet d'illustrer un diagramme de classes en l'illustrant par des exemples (des instances).
  - diagramme de composants (Component diagram)
  - diagramme de déploiement (Deployment diagram)
  - diagramme de paquetages (Package diagram)
  - diagramme de structures composites (Composite structure diagram)

Diagrammes

- **Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)**
  - ➢ diagramme de cas d'utilisation (Use case diagram)
    Représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.
  - ➢ diagramme d'activités (Activity diagram)
    Représentation du processus : il montre l'enchaînement des activités qui concourent au processus.
  - ➢ diagramme d'états-transitions (State machine diagram)
    Représente la façon dont évoluent (i.e. cycle de vie) les objets appartenant à une même classe.

Diagrammes

- **Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)**
  - **Diagrammes d'interaction (Interaction diagram)**
    - diagramme de séquence (Sequence diagram)
      Représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre. Dans ce diagramme l'accent est mis sur la chronologie des messages.
    - diagramme de communication (Communication diagram)
      Similaire au diagramme de séquence mais l'accent est mis sur la structure des messages transmis
    - diagramme global d'interaction (Interaction overview diagram)
    - diagramme de temps (Timing diagram)

Les cas d'utilisation

Objectif des cas d'utilisation : Comprendre les besoins du client pour rédiger le cahier des charges fonctionnel.

3 questions :

- Définir les utilisations principales du système : à quoi sert-il ?
- Définir l'environnement du système : qui va l'utiliser ou interagir avec lui ?
- Définir les limites du système : où s'arrête sa responsabilité ?

Eléments de description :

- Diagramme des cas d'utilisation
- Description textuelle des cas d'utilisation
- Diagrammes de séquence des scénarios d'utilisation

Diagramme des cas d'utilisation : Use-Case Diagram

Le diagramme des cas d'utilisation fournit un moyen simple à la maîtrise d'ouvrage (MOA) d'exprimer leurs besoins , ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique.

Il modélise le comportement d'un système, d'un sous-système ou d'une classe tel qu'un utilisateur extérieur le voit.

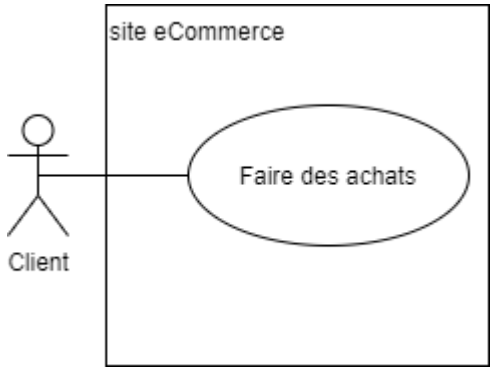
Il convient de déterminer un bon niveau de granularité du schéma ("taille du grain" représente la quantité d'opérations regroupées dans un même UC).

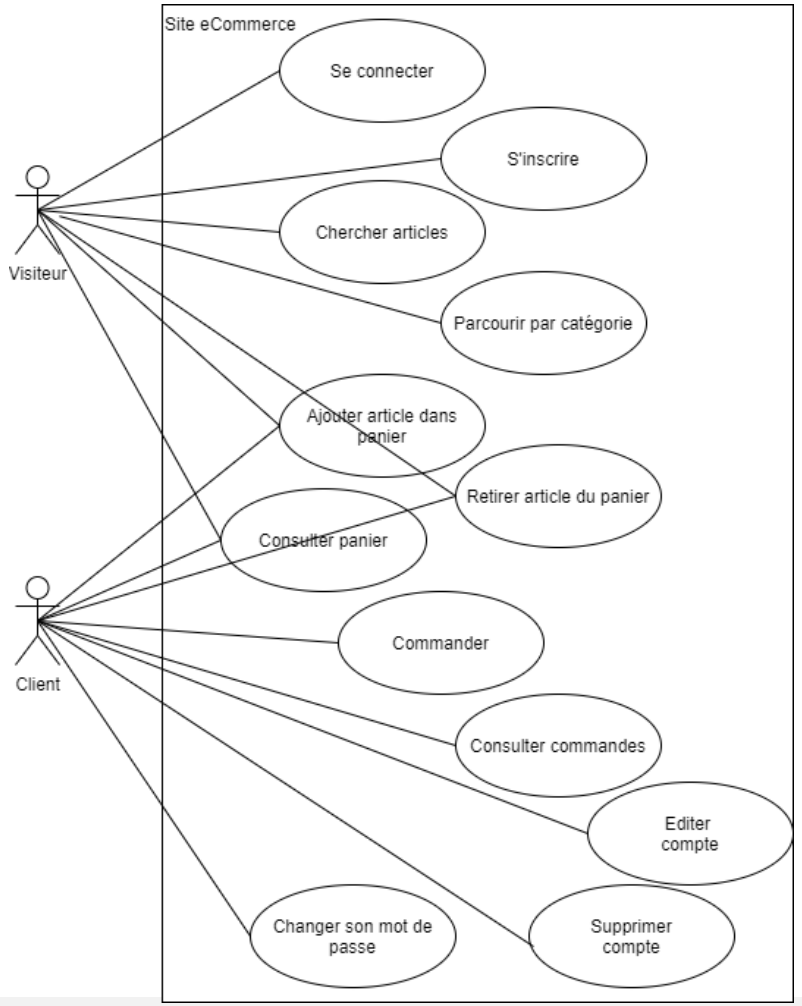
En effet, si les grains sont trop petits, le schéma devient difficilement compréhensible par la multiplication des UC et des associations.

Diagramme des cas d'utilisation : Use-Case Diagram

A droite, un schéma dont la granularité est trop fine, le rendant difficilement lisible.

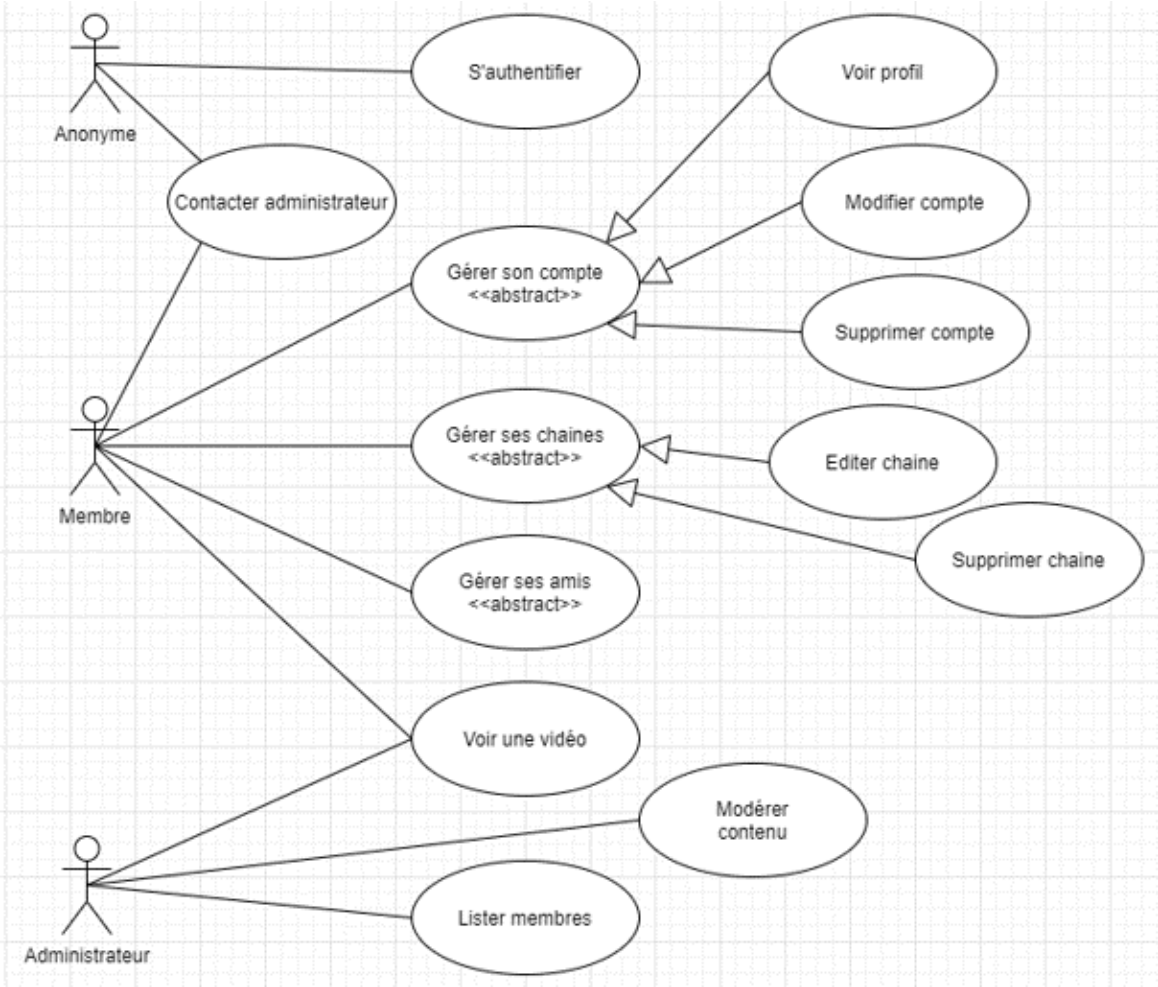
Tandis qu'à l'inverse, en dessous, une granularité trop grossière, de nombreuses fonctionnalités importantes sont masquées.



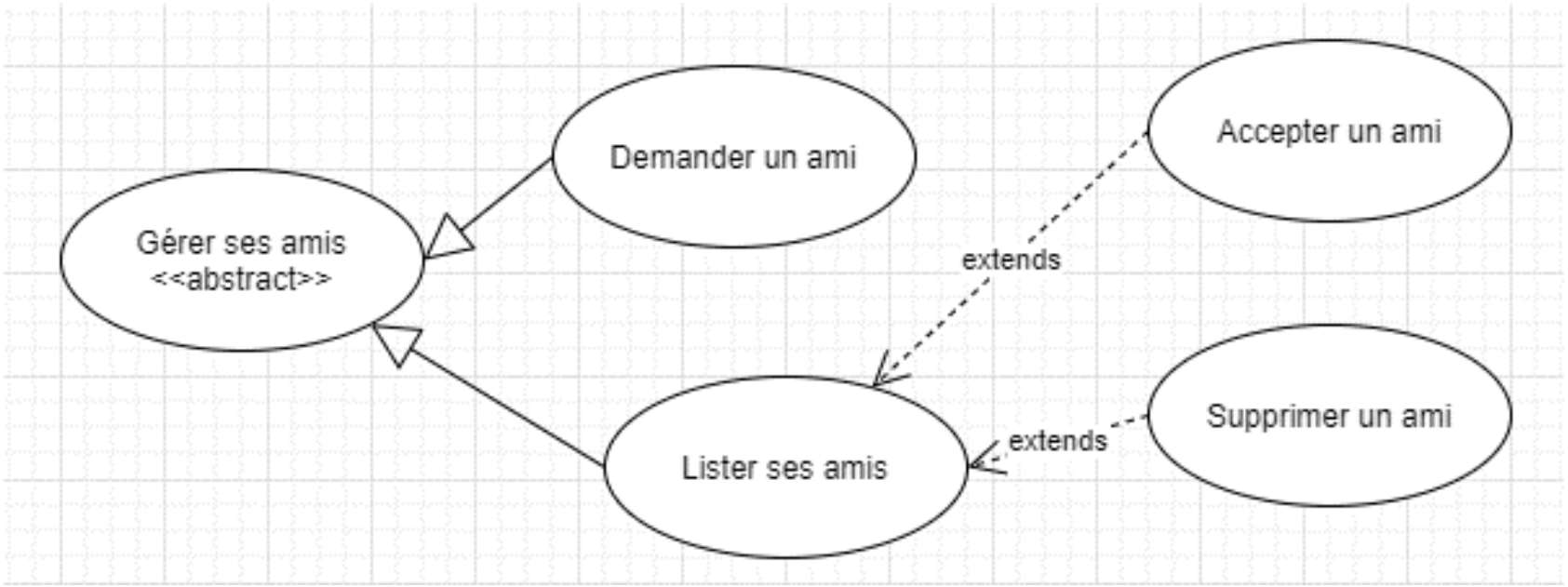


Etude de cas : projet de site web

Use-case diagram global du projet web

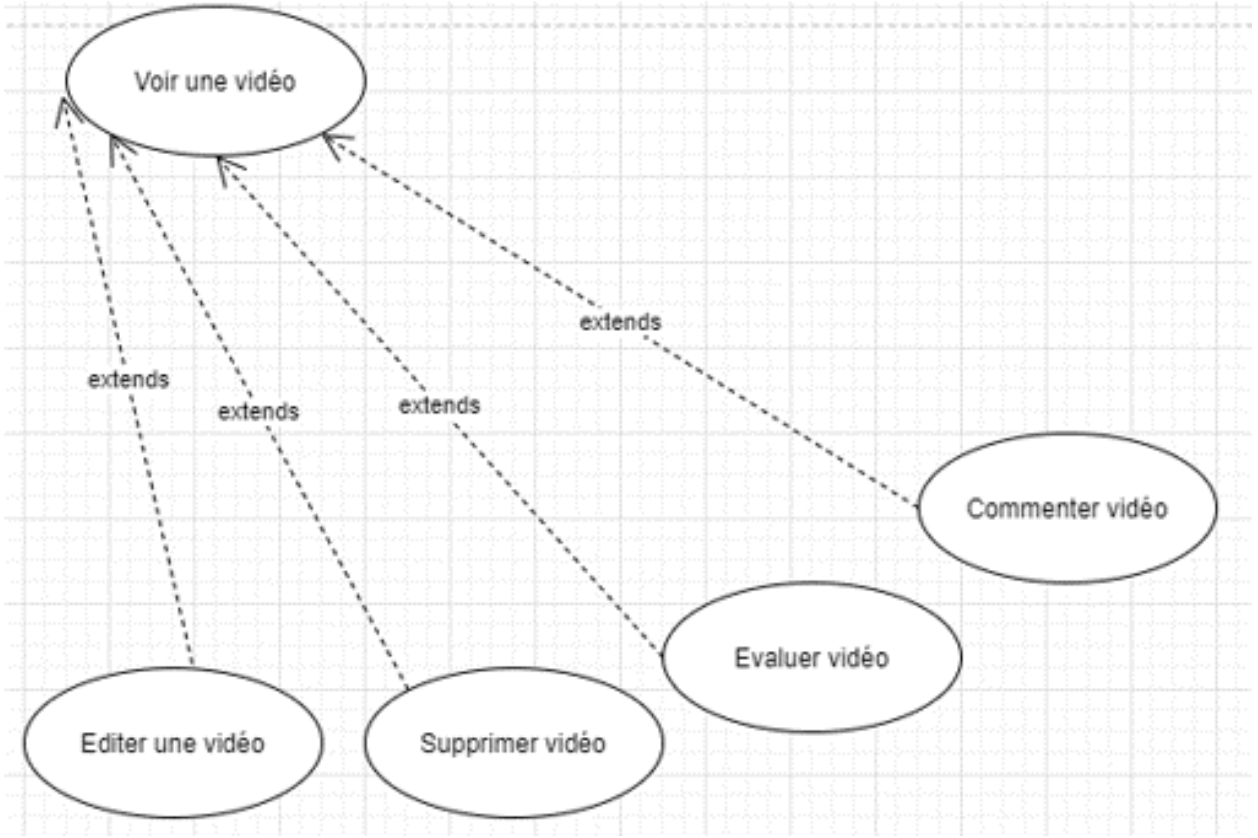


Etude de cas : projet de site web



Use-case diagram : Détails de « gérer ses amis »

Etude de cas : projet de site web



Use-case diagram : Détails de « Voir vidéo »

Etude de cas : projet de site web

Questions :

- En quelques mots, comment décririez-vous l'application décrite ci-dessus ?
- Cette application est-elle destinée à un usage interne d'une entreprise ou à tout internaute ?

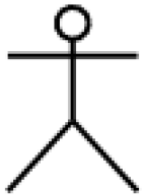
Réponses :

- C'est un réseau social dédié au partage de liste de lecture de vidéos (chaines).
- Bien que la possibilité pour un membre de supprimer son compte soit un indice, le diagramme ne permet pas de répondre à cette question avec certitude. Nous avons besoin de connaitre plus précisément ce que recouvre l'UC "S'authentifier". La description de l'UC que nous verrons ultérieurement permet de lever toute ambiguïté.

Diagramme des cas d'utilisation : Use-Case Diagram

Eléments d'un diagramme des cas d'utilisation :

- **Acteur** : rôle joué par une personne externe au système, un processus, ou une chose qui interagit avec le système (ex : timer, capteur de pression, …).
  Il est représenté par un petit bonhomme avec son nom (rôle) inscrit dessous.



Client

- **Cas d'utilisation** : Représente une fonctionnalité visible de l'extérieur. Il modélise un service rendu par le système, sans imposer le mode de réalisation de ce service.
  Il est représenté par une ellipse contenant le nom du cas (verbe à l'infinitif) et éventuellement un stéréotype.

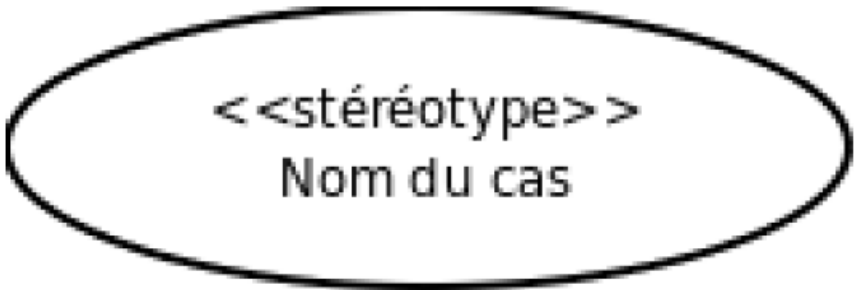


Diagramme des cas d'utilisation : Use-Case Diagram

Libellé d'un cas d'utilisation :

- Le libellé d'un UC doit représenter une action. Il est donc préférable que ce libellé commence par un verbe à l'infinitif suivi d'un complément explicatif en quelques mots.
- Le libellé des UC est à choisir avec soin pour être "le moins ambigu possible". Par exemple: "Imprimer facture" plutôt que "Générer document".
- En fonction de la granularité attendue, il conviendra de détailler ou non certaines actions.
  Par exemple, lorsque l'expression des besoins demandera de « gérer » une entité, il faudra considérer que cela signifie de permettre toutes les opérations CRUD (Create, Read, Update, Delete) et donc, créer ou non les Use-Case correspondant.
- De même, « Editer » sous-entend les opérations Update et Delete.

Diagramme des cas d'utilisation : Use-Case Diagram

Les acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système (ses données à un instant t), en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

Un acteur n'es pas forcément une entité concrète.
Un utilisateur peut bénéficier de plusieurs rôles (exemple: une personne peut être à la fois éditeur de contenus et administrateur d'une plateforme elearning); de même un rôle peut être attribué à plusieurs personnes (c'est d'ailleurs le cas en général).

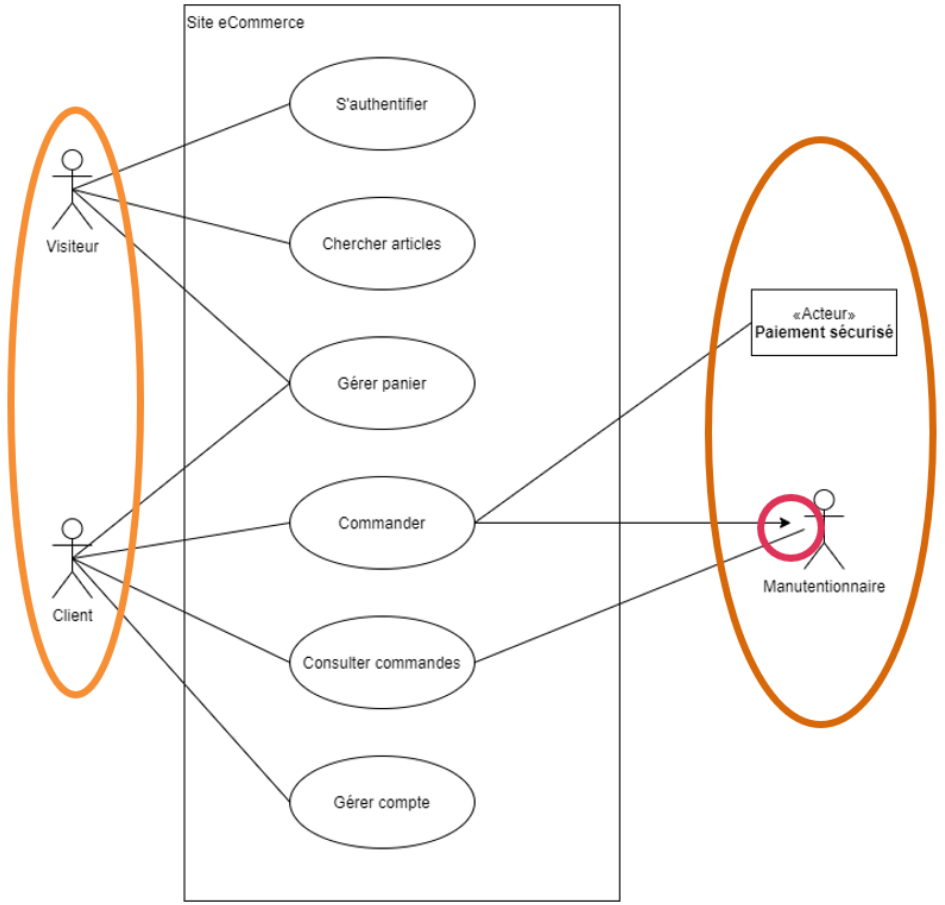
Les Use-Cases associés à un acteur vont donc définir ses privilèges au sein de l'application ou les responsabilités auxquelles il va pouvoir contribuer.

Un acteur **qui possède au moins un rôle/privilège** est un acteur primaire et est placé du **côté gauche** du système.

Un acteur **qui n'a que des responsabilités** (n'a pas vocation à utiliser le système mais est là pour le faire fonctionner) est un acteur secondaire et est placé du **côté droit** du système.

Diagramme des cas d'utilisation : Use-Case Diagram

Acteurs **primaires**
(min 1 privilège)



Acteurs **secondaires**
(uniquement responsabilité(s))

Diagramme des cas d'utilisation : Use-Case Diagram

Association acteur et UC

La liaison entre un acteur et un UC est représentée par un trait simple.

Certains auteurs préconisent l'usage d'une flèche directionnelle entre un UC et un acteur secondaire pour indiquer si celui-ci ne fait que réceptionner ou envoyer des données au système.

Diagramme des cas d'utilisation : Use-Case Diagram

Cardinalités sur association acteur et UC

Dans une association UML, la cardinalité représente le nombre d'entités par un intervalle de valeurs entières comprises entre un minimum et un maximum inclus. Exemples:

- "2...*": 2 entités minimum,
- "0...1": une seule entité facultative.
- "1…n" : 1 à n entités (n équivaut à « 1 ou + »)

Lorsque la cardinalité est exprimée du **côté acteur de l'association**, cela indique une contrainte sur le nombre d'entités acteur impliquées pour réaliser une UC.
Ainsi dans l'exemple ci-contre, jouer une partie FPS online requiert au moins 2 joueurs.

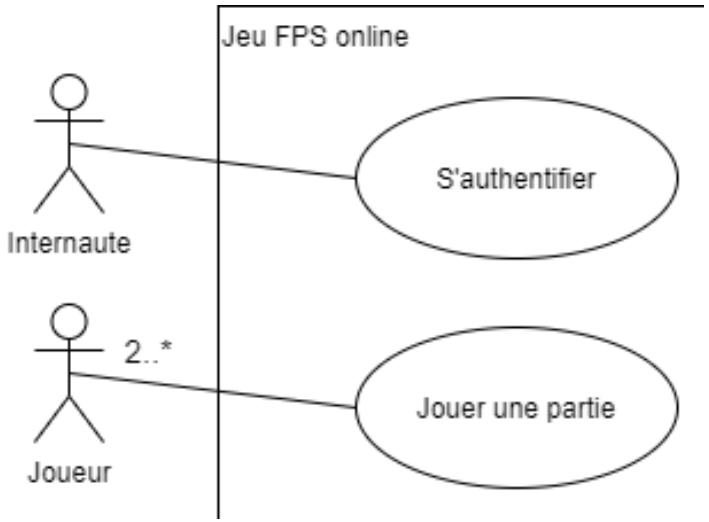


Diagramme des cas d'utilisation : Use-Case Diagram

Cardinalités sur association acteur et UC

Lorsque la cardinalité est exprimée **du côté UC de l'association**, cela indique que une contrainte sur le nombre d'usages en parallèle du UC par une même entité acteur.

Dans l'exemple ci-contre, il faut imaginer une interface qui permet à l'utilisateur de cliquer sur une catégorie pour demander que l'actualisation des news associées. Ensuite durant le temps de rafraichissement de cette catégorie, il a la possibilité de cliquer sur une autre catégorie pour demander son actualisation également.

Si, par contre, le système permettait à l'utilisateur d'actualiser toutes ou plusieurs catégories sélectionnées d'un coup, il ne faut pas écrire de cardinalité dans l'association mais simplement la nommer au pluriel: "Rafraichir catégories"

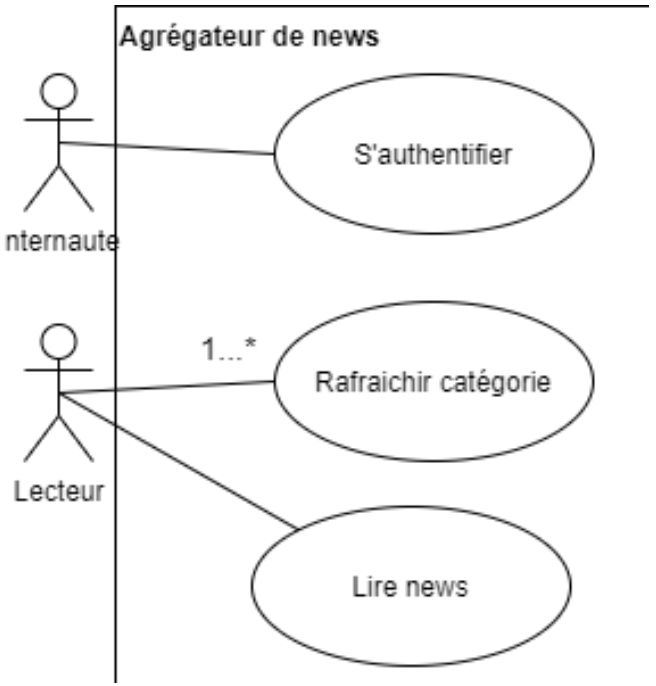


Diagramme des cas d'utilisation : Use-Case Diagram

Généralisation/spécialisation des acteurs

Lorsque qu'un acteur A **bénéficie de tous les privilèges d'un acteur** B en plus des siens propres, on le représente comme une version spécialisée de B (flèche de A vers B) afin de ne pas alourdir le schéma en dédoublant les associations vers les UC communs.

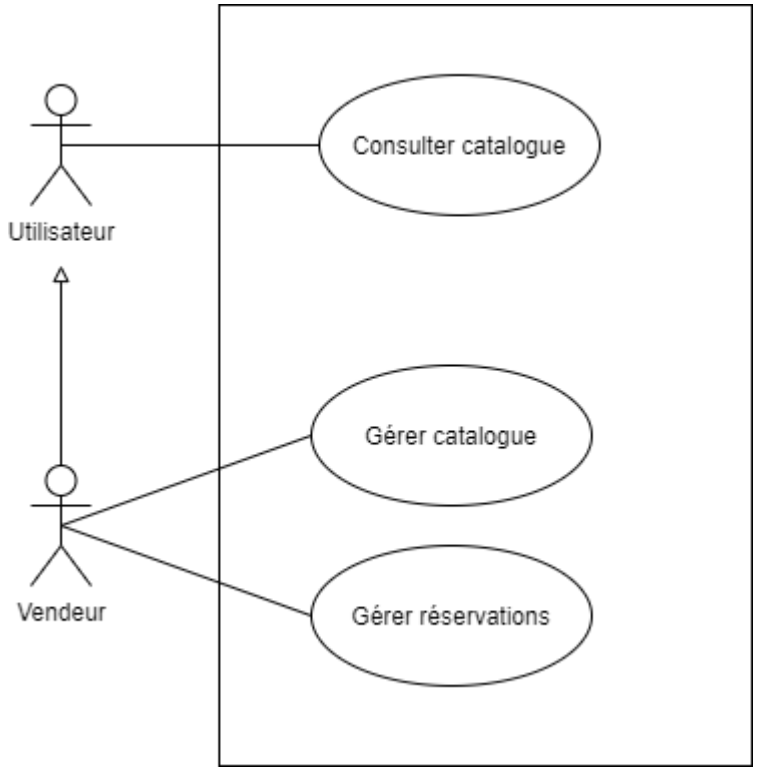
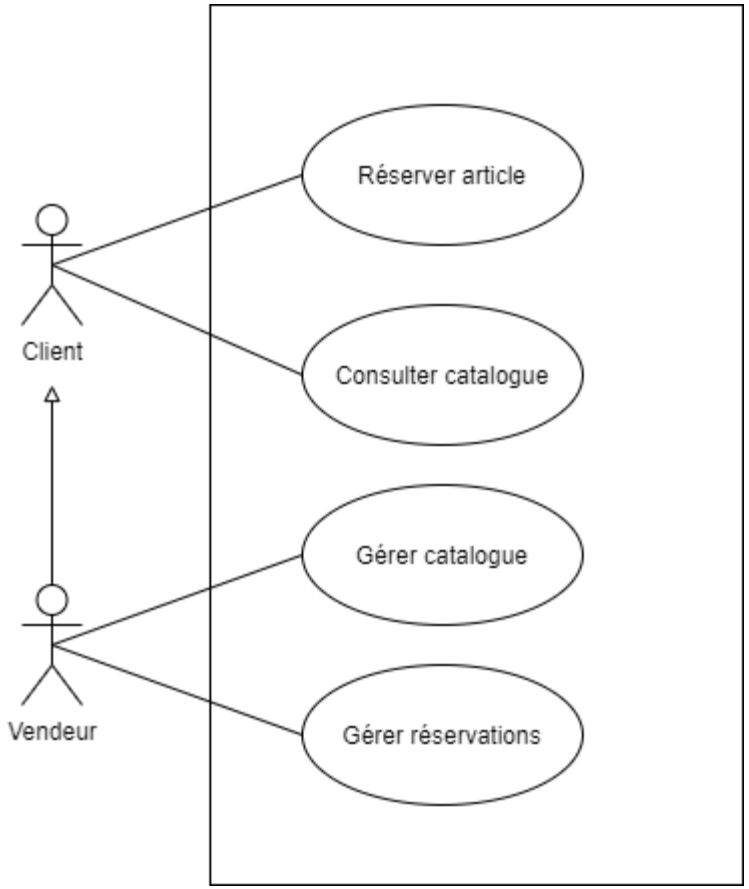


Diagramme des cas d'utilisation : Use-Case Diagram

Spécialisation des acteurs : exemple

Question :

Que pensez-vous de la relation de spécialisation ci-contre ?



Réponse :

C'est erroné.

Bien entendu, un personne qui est vendeur dans le magasin a aussi la possibilité d'être client du magasin; mais dans ce cas, elle endosse un autre rôle (elle devient temporairement un autre acteur.).

Conceptuellement, un vendeur n'est pas un client !

Diagramme des cas d'utilisation : Use-Case Diagram

Spécialisation des acteurs : exemple

Question :

Comment corrigerai-t-on ce diagramme ?

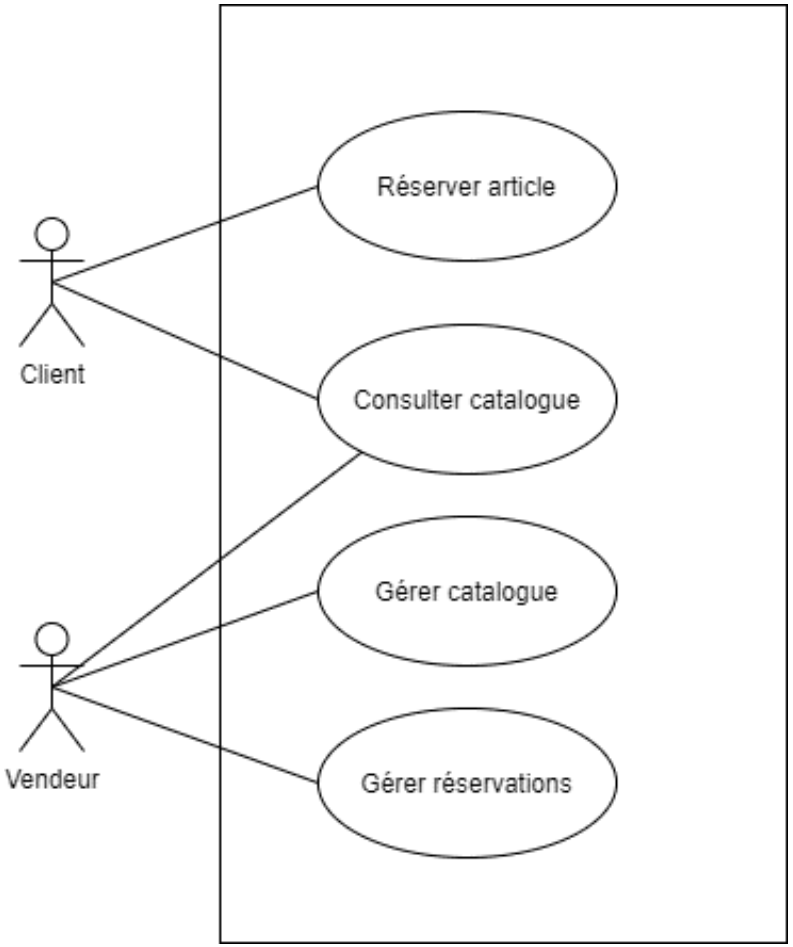


Diagramme des cas d'utilisation : Relations

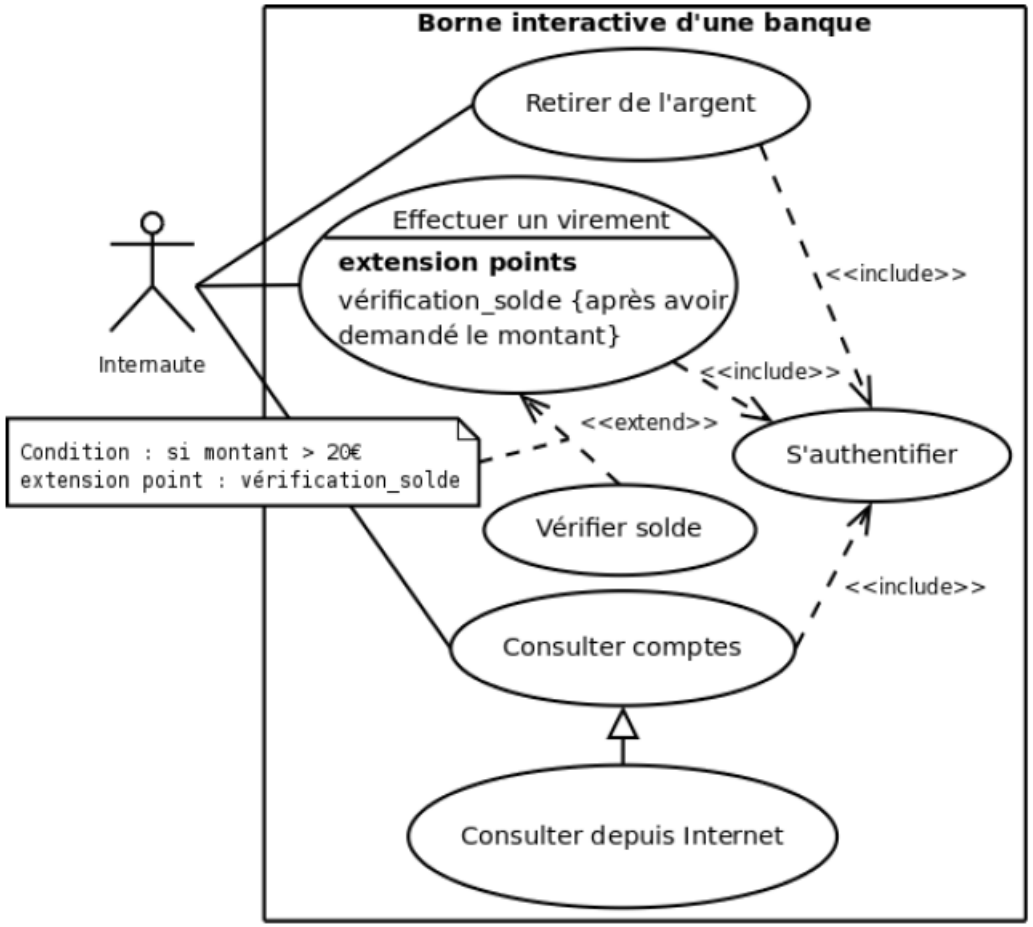


Diagramme des cas d'utilisation : Relations

Dépendance stéréotypée : La relation d'inclusion

La relation d'inclusion est une relation **dirigée entre deux cas d'utilisation** qui est utilisée pour montrer que les opérations du cas d'utilisation inclus (l'ajout) sont insérées dans les opérations du cas d'utilisation d'inclusion (la base).

La relation d'inclusion peut être utilisée :

- pour extraire les parties communes des comportements de deux ou plusieurs cas d'utilisation.

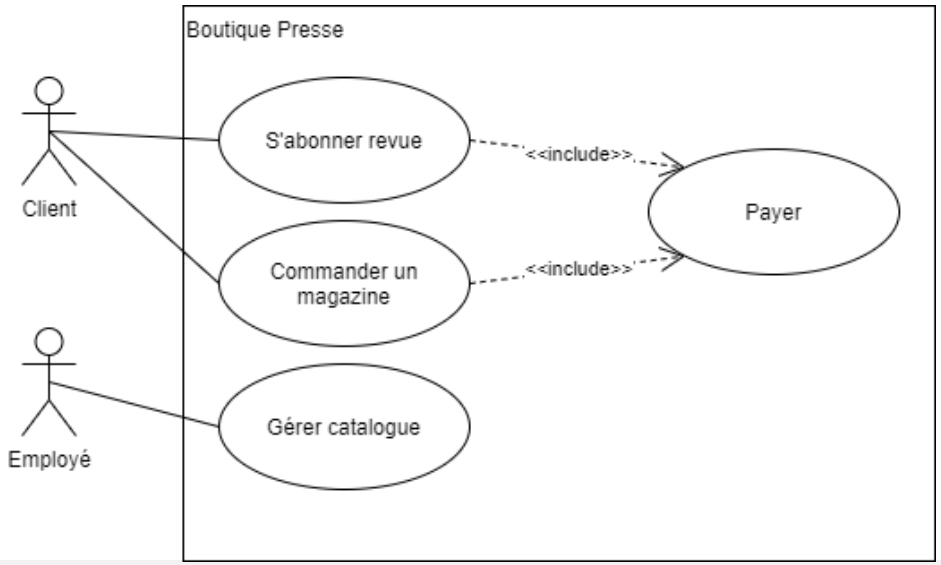
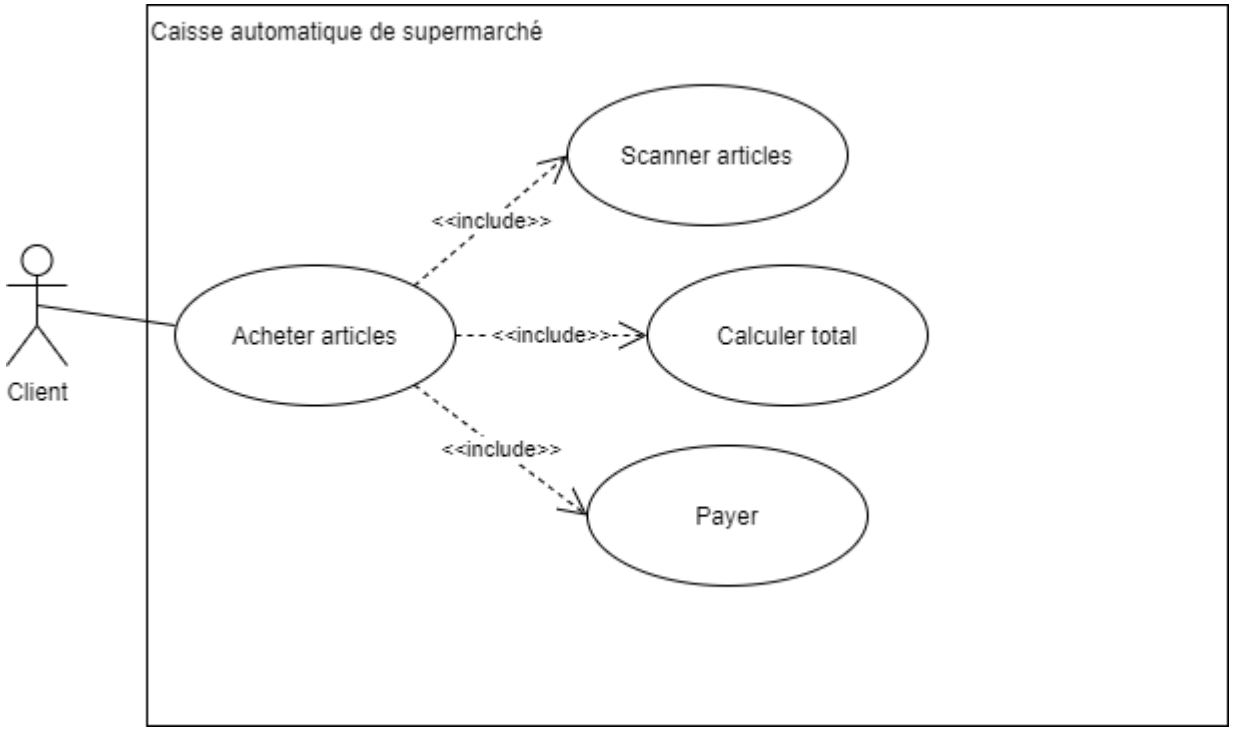


Diagramme des cas d'utilisation : Relations

Dépendance stéréotypée : La relation d'inclusion

- pour simplifier les cas d'utilisation volumineux en les divisant en plusieurs cas d'utilisation



Exemple: <<include>> pour isoler opérations communes entre plusieurs UC

Diagramme des cas d'utilisation : Relations

Dépendance stéréotypée : La relation d'extension

La **relation d'extension** est une relation dirigée qui spécifie comment et des **opérations** définies dans un cas d'utilisation **facultatives** peuvent être insérées dans les opérations définies dans le cas d'utilisation étendu.

Le cas d'utilisation étendu est indépendant du cas d'utilisation d'extension.

Le même cas d'utilisation d'extension peut étendre plus d'un cas d'utilisation, et le cas d'utilisation d'extension peut lui-même être étendu.

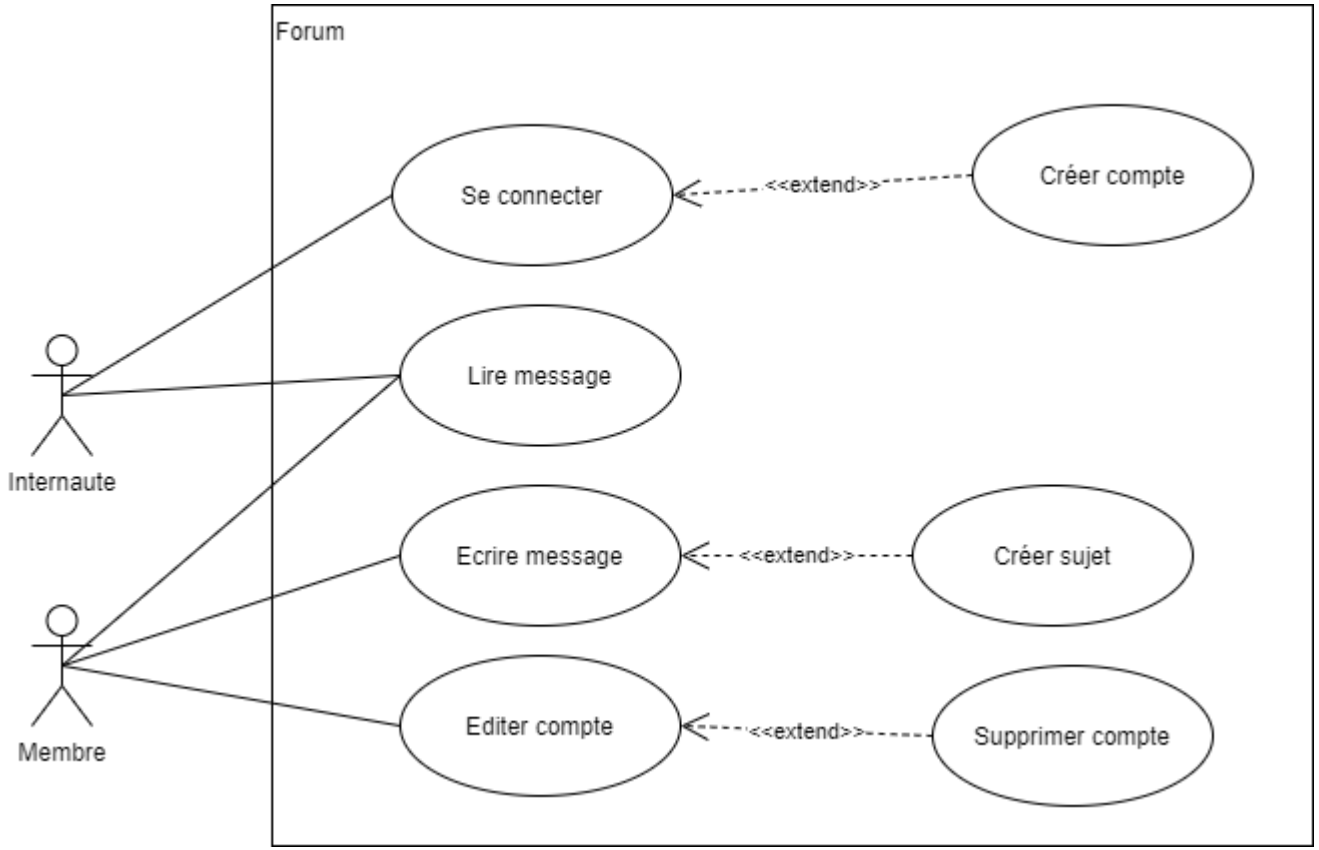
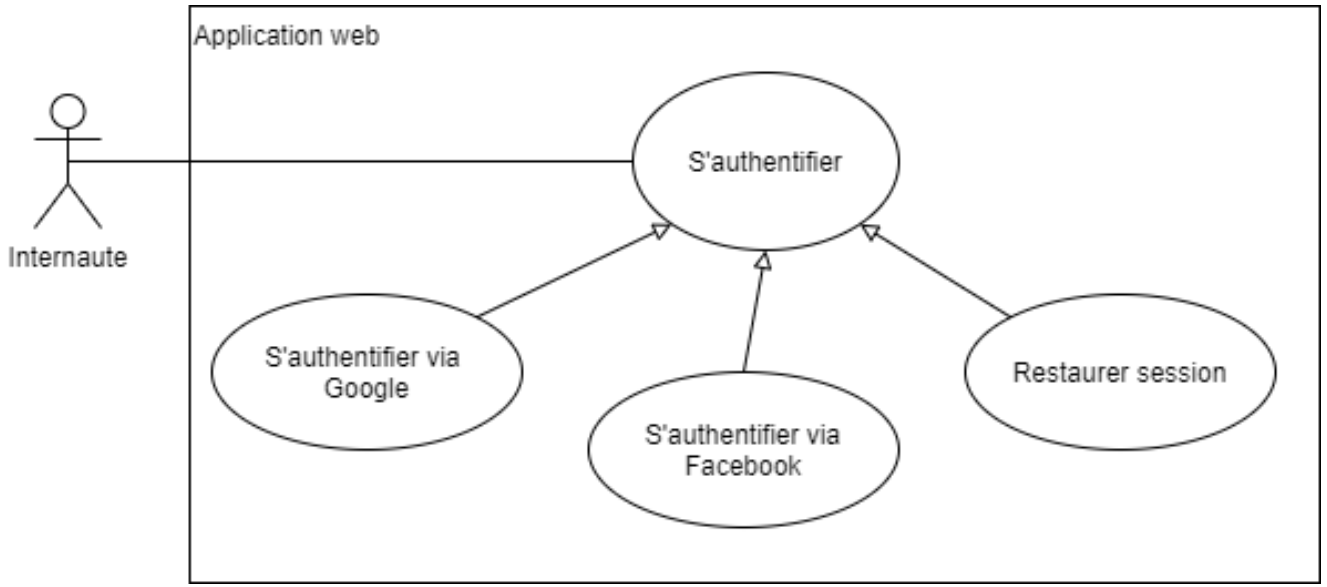


Diagramme des cas d'utilisation : Relations

La relation de spécialisation

La relation de spécialisation (ou de généralisation) spécifie une variante d'opérations (UC "enfant") qui peut être réalisée à la place d'un UC (UC "parent").



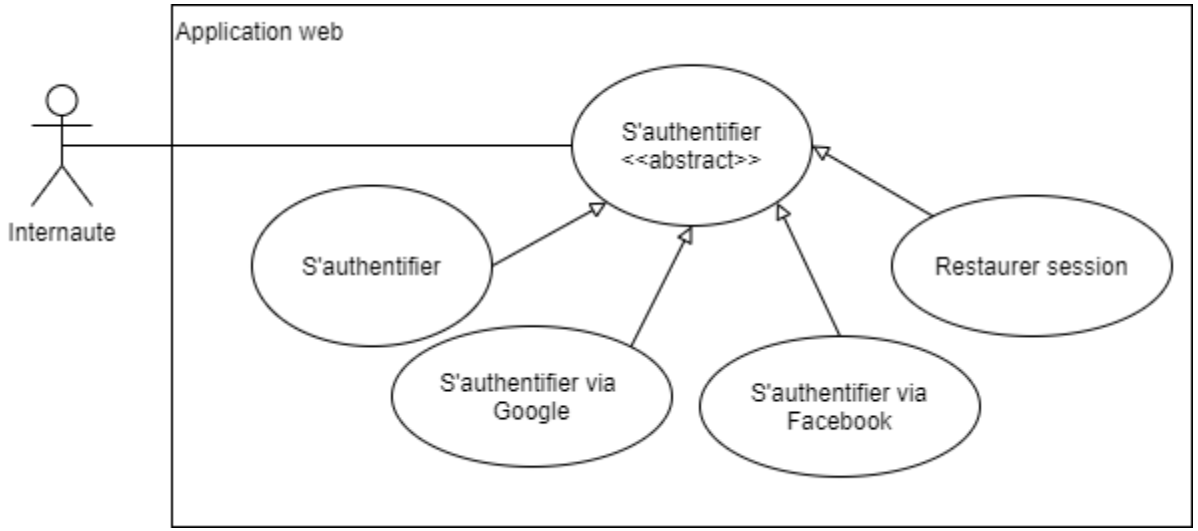
Exemple: authentification locale par défaut

Diagramme des cas d'utilisation : Relations

La relation de spécialisation

Le Cas d'utilisation "parent" peut être déclaré abstrait.

Dans ce cas, le UC ne peut être réalisé en lui-même, seules les variantes spécialisées peuvent être réalisées.

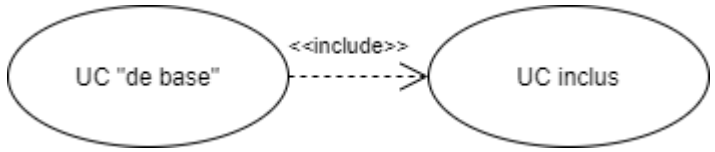


Exemple: pas de technique d'authentification par défaut

Diagramme des cas d'utilisation : Relations

Résumé des différentes relations entre Use Cases

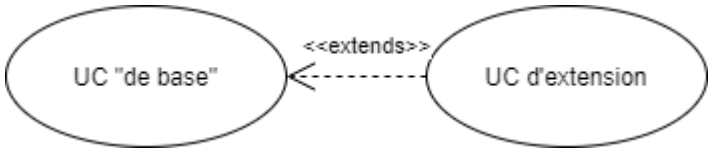
Inclusion <<include>>



Le Use-Case de base ne peut être réalisé seul.

La réalisation de chaque Use-Case "inclus" est obligatoire.

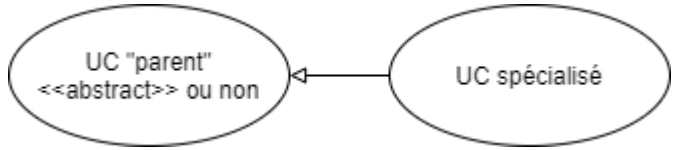
Extension <<extend>>



Le Use-Case de base peut être réalisé seul.

La réalisation d'une Use-Case d'extension est facultative.

Spécialisation



Le Use-Case "parent" peut être abstrait ou non.

Si le Use-Case "parent" est abstrait, une Use-Case spécialisée au moins est obligatoire.

Description des Use-Cases

Le diagramme des UC est utile pour avoir une vision d'ensemble des fonctionnalités offertes par le système.

Cependant, un libellé d'UC ne peut se suffire à lui-même pour exprimer l'ensemble des opérations nécessaires à sa résolution.

Idéalement, chaque UC du diagramme doit être accompagné d'une spécification explicative, indépendante de toute interface.

Description des Use-Cases

Template de spécification d'un UC

Il n'y a pas de formalisme standardisé au sein d'UML pour la description textuelle d'un UC.
Cependant, il est généralement convenu d'adopter un template qui reprend les informations usuelles liées à un UC :

- **Libellé du UC**
- **Acteur(s) / Persona**: noms des acteurs liés au UC,
- **Objectif(s)** : description courte des attentes des acteurs concernés,
- **Précondition(s)** : conditions qui doivent être vraies pour que le UC puisse commencer à se réaliser,
- **Postconditions(s)** : conditions qui doivent être vraies lorsque le UC termine de se réaliser,
- **Scénario nominal** : représentation de la séquence d'étapes qui permet le plus fréquemment de réaliser l'UC,
- **Scénario(s) alternatif(s)** : représentations des variations de séquence d'étapes qui permettent également de réaliser l'UC,
- **Scénario(s) d'erreur** : représentation des variations de séquence d'étapes qui ne permettent pas à l'UC de se réaliser complètement (les postconditions ne peuvent être toutes vraies).

Description des Use-Cases

Template de spécification d'un UC : Scénario nominal

Chaque scénario va être présenté sous forme d'un tableau dont chaque colonne correspond à un acteur primaire, au système proprement dit ou à un acteur secondaire impliqué dans le scénario.

Chaque étape du scénario est numérotée pour indiquer clairement la séquence des étapes.

Le scénario nominal est le premier scénario qui est rédigé et présente soit le scénario le plus court ou le plus fréquent pour parvenir à la réalisation complète du UC (postconditions vraies).

UC-M.4 : Editer chaine

Acteur principal : Membre

Objectifs : Le membre souhaite créer ou modifier une chaine de vidéos

Préconditions : le membre est authentifié

Postconditions : la chaine est créée ou modifiée

Scénario nominal :

1. L'utilisateur sélectionne une chaine et souhaite l'éditer	2. Le système affiche le formulaire d'édition de chaine prérempli (au minimum : nom de la chaine et si elle est publique)
3. L'utilisateur spécifie un nom et si la chaine est publique et valide	4. Le système vérifie qu'un nom est bien spécifié et qu'aucune autre chaine du même utilisateur ne possède le même nom, le système modifie la chaine et affiche un message de confirmation de modification.

Description des Use-Cases

Imbrication de scénarios alternatifs

Tout comme le scénario nominal, un scénario alternatif peut lui-même avoir variantes...

Les scénarios alternatifs d'un scénario alternatif vont être identifiés par une numérotation en plusieurs niveaux.

Dans l'exemple ci-contre, le scénario nominal du UC "S'authentifier" décrit comment un utilisateur possédant un identifiant et un mot de passe peut s'authentifier dans l'application.

Le scénario alternatif 1b permet à l'utilisateur ne possédant pas de compte de s'en créer un afin de pouvoir malgré tout s'authentifier. Ce même scénario possède une alternative 1b.4a qui démarre à la 4ième étape du scénario alternatif 1b si l'utilisateur n'entre pas des données valides.

1b : L'utilisateur souhaite créer un nouveau compte

1. L'utilisateur demande à créer un nouveau compte	2. Le système affiche le formulaire de création de compte
3. L'utilisateur fournit un login, un mot de passe, une répétition du mot de passe, une adresse courriel, son nom et son prénom et valide	4. Le système vérifie si l'adresse courriel est valide et unique, si le login est unique, si le mot de passe est identique à sa répétition et que tous les champs obligatoires sont remplis.
	5. Le système crée un compte pour l'utilisateur. Le scénario nominal reprend à l'étape 3.

1b.4a : Une ou plusieurs données fournies par l'utilisateur sont invalides

	1. Le système détecte une ou plusieurs données invalides (format courriel invalide, mot de passe diffère de sa répétition, un des champs obligatoires est vide, login ou courriel déjà lié à un autre utilisateur)
	2. Le système affiche un message d'erreur informant l'utilisateur des erreurs détectées. Le formulaire de création de compte est affiché avec les données précédemment encodées par l'utilisateur. Le scénario alternatif 1b reprend à l'étape 3.

Description des Use-Cases

Scénarios d'erreur

Un scénario d'erreur est un scénario qui va clôturer celui-ci sans pouvoir satisfaire les postconditions.

Ces scénarios peuvent soit être présentés sous forme de tableaux (comme les autres scénarios), soit sous forme d'une simple phrase si ces scénarios ne comportent qu'une étape.

UC-M.5 Supprimer chaine

Acteur principal : Membre

Objectifs : Le membre souhaite supprimer une de ses chaines

Préconditions : le membre est authentifié et la chaine ne contient aucune vidéo

Postconditions : la chaine est supprimée

Scénario nominal :

	1. Le système vérifie qu'aucune vidéo n'est liée à la chaine, affiche un message d'avertissement et demande la confirmation de suppression
2. L'utilisateur confirme sa demande de suppression de sa chaine	3. Le système supprime la chaine et affiche un message de confirmation de suppression.

Scénarios d'erreur :

1a : le système détecte qu'une vidéo au moins est liée à la chaîne et avertit l'utilisateur que la suppression est refusée.

2a : l'utilisateur ne confirme pas sa demande de suppression.

Bibliographie / Ressources

Bibliographie :

- UML 2 : De l'apprentissage à la pratique (Laurent AUDIBERT) : https://laurent-audibert.developpez.com/Cours-UML/
- Tutoriel UML Haute Ecole Libre de Mosane https://dartagnan.cg.helmo.be/~p150107/tutoriels/uml-uc/

Ressources :

- Playlist de courtes vidéos expliquant de façon claire les différents diagrammes (Delphine Longuet) https://www.youtube.com/watch?v=GC5BdRve38A&list=PLSVDd2z0rl6PlOscYvneTFvU8qqPFzFkl
- Spécification https://www.omg.org/spec/UML/2.5.1/PDF