UML : USE CASE (Cas d'Utilisation)

Introduction

Ce module a pour objectif la Conception d'un Système d'Information.
Nous allons découvrir au travers de ce module, la modélisation UML (Unified Modeling Language).

UML est un langage unifié de modélisation.
Il permet de décrire sous forme de diagrammes lisible les expressions du besoin orientées métiers.

Il est composé de 14 diagrammes :

- 7 diagrammes de structure (comme le diagramme de classe)
- 7 diagrammes comportementaux (comme le diagramme de cas d'utilisation, diagramme d'activité, diagramme de séquence)

Objectif

Le but est de découvrir le diagramme de cas d'utilisation.

Auteur : Yoann DEPRIESTER
Relu, validé & visé par :
☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création : 01-09-2023
Date révision : 01-09-2023

UML : USE CASE (Cas d'Utilisation)

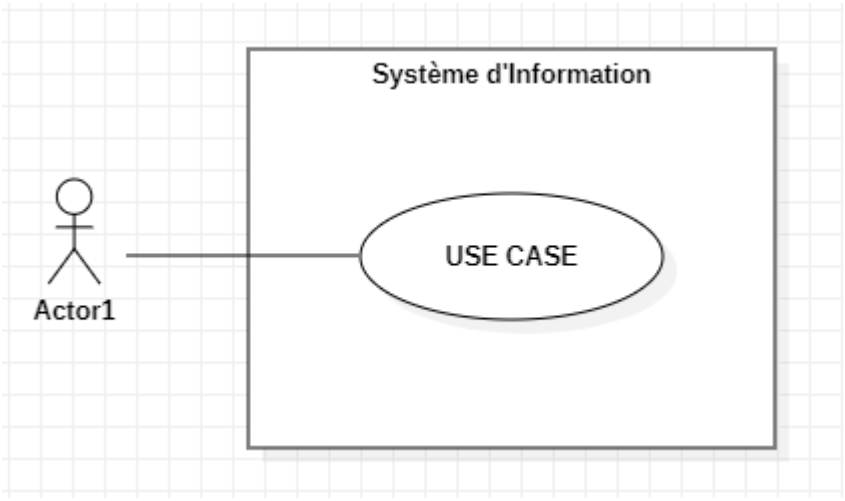
Diagramme : Cas d'Utilisation = Use Case

Le Diagramme de Cas d'Utilisation a pour rôle de modéliser les fonctionnalités qu'un Système offre à ses utilisateurs.

Il permet de visualiser de manière graphique Qui a accès à Quoi, et Quelles sont les relations entre les Cas d'Utilisation.

Représentation Graphique d'un Use Case

Dans sa forme la plus élémentaire, un Diagramme de Cas d'Utilisation représente un **Cas d'Utilisation** au sein d'un **Système d'Information,** et relié à un **Acteur**.



Use Case (Cas d'Utilisation) : une fonctionnalité offerte par le Use Case Subject. Il est relié à un Acteur, et commence toujours pas un verbe à l'infinitif inscrit dans un ovale. Un Use Case est constitué d'un ensemble d'étape en vue d'atteindre un objectif bien défini.

Exemple de Use Case : Afficher un Article, Créer un Compte Client, Importer une image d'Avatar, Supprimer un Commentaire, Modifier son Profil, Payer une Commande, …

Exemple de ce que n'est pas un Use Case :

- *Cliquer sur un lien (il s'agit d'une étape pour accomplir un Use Case)*
- *Remplir un formulaire (pareil, il s'agit d'une étape pour accomplir un Use Case)*
- *Gérer quelque chose (trop vague, on ne sait pas précisément de quel type d'action de gestion on parle : Création, Suppression, Modification, … Chacun de ses objectifs demandant des étapes différentes)*

Use Case Subject : le Système d'Information qui offre des fonctionnalités. Il est représenté par un rectangle portant son nom, et contenant l'ensemble des Uses Case.

UML : USE CASE (Cas d'Utilisation)

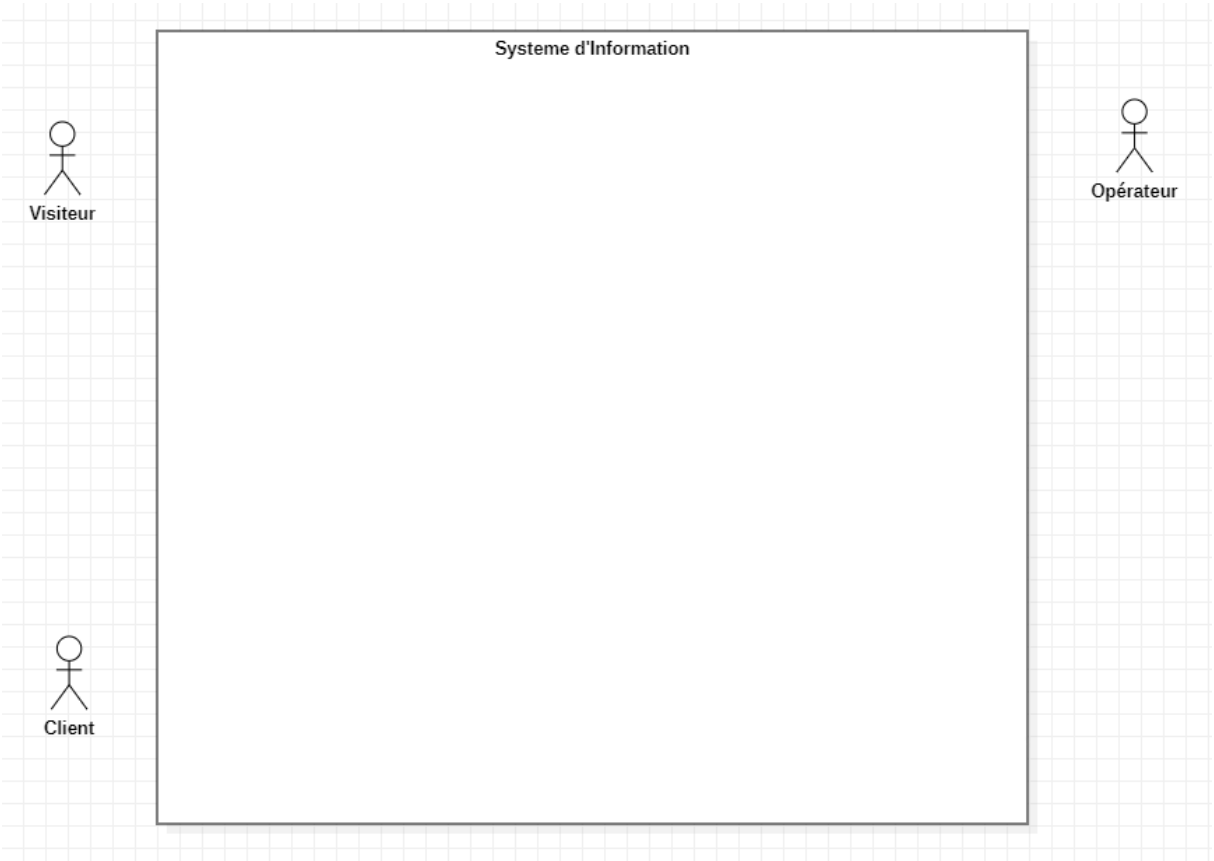
Acteur : Désigne un Rôle occupé par un utilisateur externe (humain ou non) au Use Case Subject. Un utilisateur peut occuper différents Rôle tour à tour. Par exemple, un Acteur Visiteur qui se connecte, après avoir créer son compte administrateur, occupe le rôle de l'Acteur Administrateur.

Exemple de Use Case : La Compagnie Aérienne BonVol

Voici un petit exemple d'illustration.

BonVol, une compagnie aérienne, nous demande de concevoir un système permettant à des internautes de se créer un compte en ligne pour pouvoir acheter un billet d'avion. Le système doit aussi permettre à des opérateurs de la compagnie de modifier le catalogue des vols.

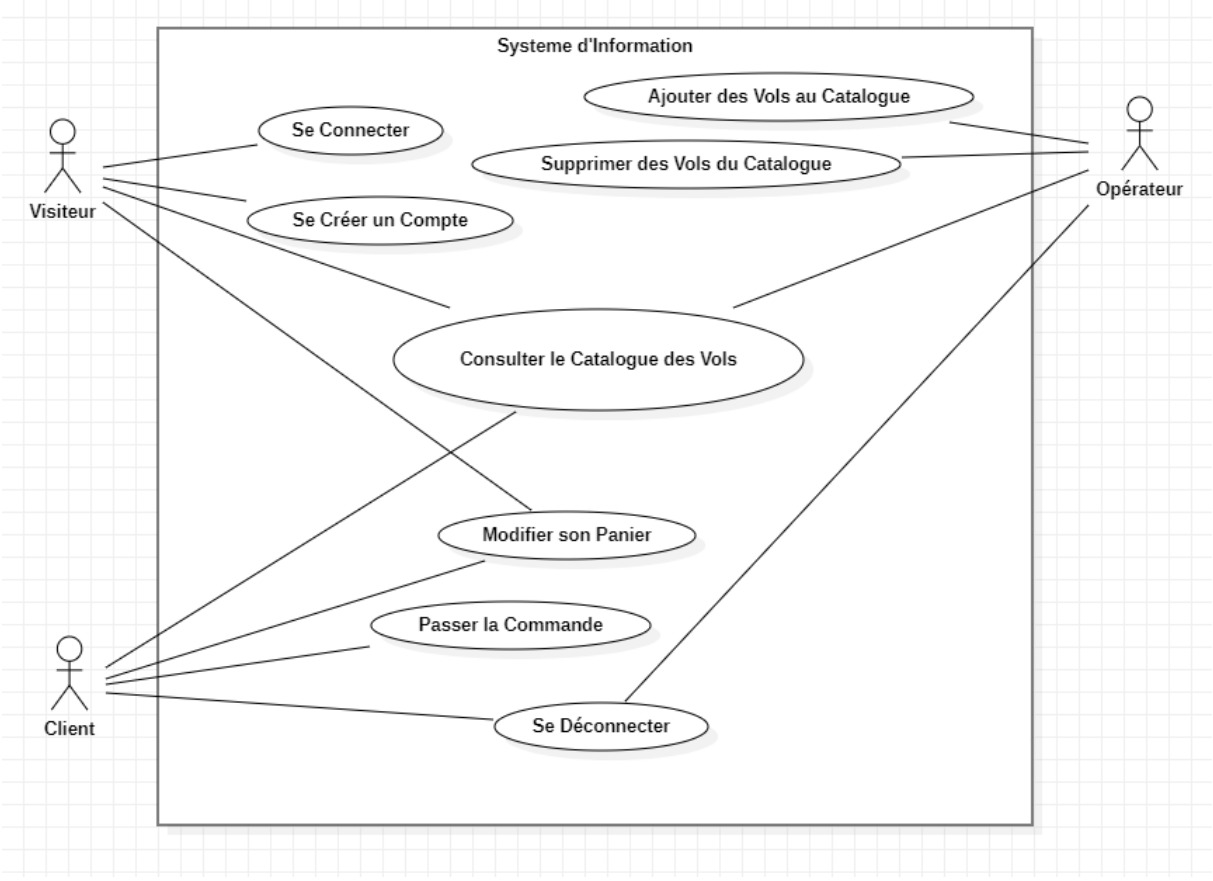
En analysant la demande, on peut distinguer plusieurs **Acteurs** : les Internautes, que l'on va appeler ici Visiteur, et les Opérateurs. Cependant, il existe un troisième **Acteur** implicite : les Internautes Connectés que l'on va appeler ici Client. Pourquoi ? Tout simplement parce que lorsqu'un visiteur se connecte à son compte sur un site, il change de rôle pour accéder à des **Cas d'Utilisations** différents d'un visiteur non connecté. Ainsi, selon le type de compte qu'il possède sur le site de la compagnie, il deviendra soit Client, soit Opérateur.



UML : USE CASE (Cas d'Utilisation)

Maintenant que l'on a identifié nos **Acteurs**, nous allons identifier les **Use Cases** qui leur sont attachés.

- Visiteur : on a vu qu'il pouvait déjà Se Connecter. Pour cela, il doit posséder un compte, donc, il doit pouvoir s'en créer un. Il possède donc la fonctionnalité Se Créer un Compte. La compagnie souhaitant lui vendre des billets d'avion, il doit pouvoir Consulter le Catalogue des Vols. Enfin, pour des raisons pratiques, on va lui permettre d'ajouter des billets dans son panier, et aussi d'en supprimer s'il souhaite (pour corriger une erreur par exemple). Ainsi il est relié à la fonctionnalité Modifier son Panier. Par contre, s'il souhaite finaliser la commande, il devra être se connecter, et donc changer de rôle pour celui de Client.
- Client : puisqu'il occupe ce rôle parce qu'il s'est connecté, il doit pouvoir Se Déconnecter. Le but étant toujours de lui vendre des billets d'avion, il doit aussi, tout comme le Visiteur, pouvoir Consulter le Catalogue des Vols, Modifier son Panier, et enfin Passer la Commande.
- Opérateur : puisqu'il occupe ce rôle parce qu'il s'est connecté, il doit pouvoir Se Déconnecter. Le but de l'Opérateur est de pouvoir Ajouter un Vol au Catalogue ou de Supprimer un Vol du Catalogue (contrairement à Modifier son Panier, on préfère séparer les deux, parce qu'il s'agit de process métiers qui utilisent différemment la Base de Donnée). Enfin, pour pouvoir Modifier le Catalogue de Vol, il faut voir le catalogue. L'Opérateur pourra donc Consulter le Catalogue des Vols.



UML : USE CASE (Cas d'Utilisation)

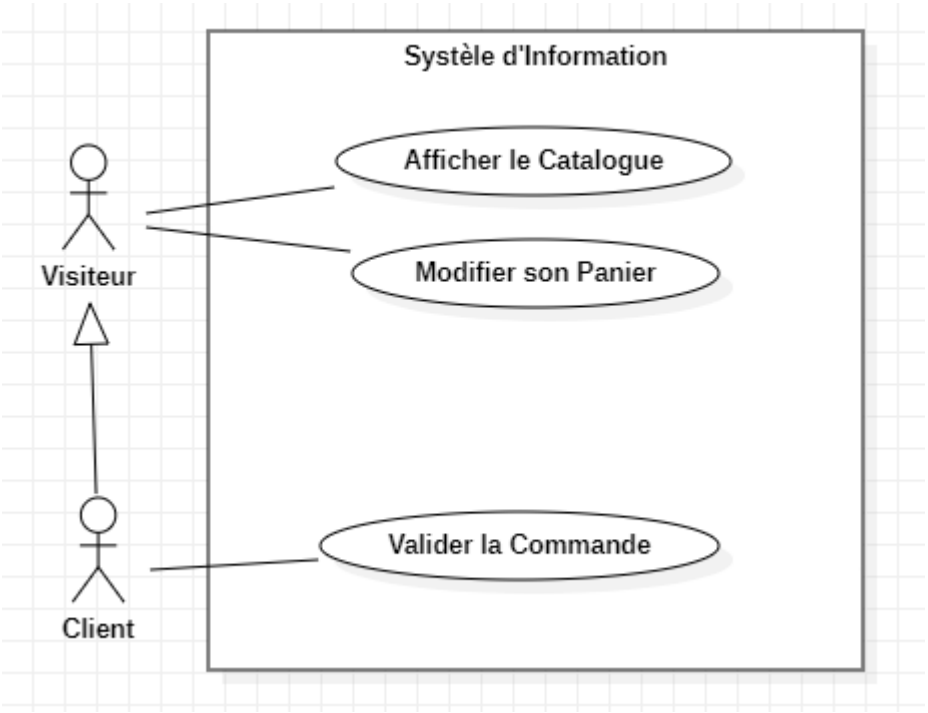
L'Héritage (ou Généralisation)

Lorsqu'on regarde le **Diagramme de Cas d'Utilisation** de l'exemple précédent, on constate que des **Acteurs** sont reliés aux même **Use Cases**. Cela reste lisible ici, car nous avons un diagramme très simple et basique, mais ce n'est généralement pas le cas. Et lorsqu'on a des diagrammes avec plusieurs dizaines de **Use Case** et des associations qui se croisent, l'ensemble devient vite illisible.

C'est là qu'intervient la notion d'**Héritage**, aussi appelé **Généralisation**.

Il s'agit d'une relation entre plusieurs **Acteurs** qui signifie qu'un **Acteur A** hérite de TOUS les **Use Cases** d'un **Acteur B**. Autrement dit, l'**Acteur A** possède ses propres fonctionnalités et ceux de l'**Acteur B**, mais l'**Acteur B** possède uniquement ses propres fonctionnalités (et non ceux de l'**Acteur A**)

On représente la relation d'**Héritage** par une flèche à tête blanche allant de l'**Acteur** qui hérite vers celui dont il hérite.

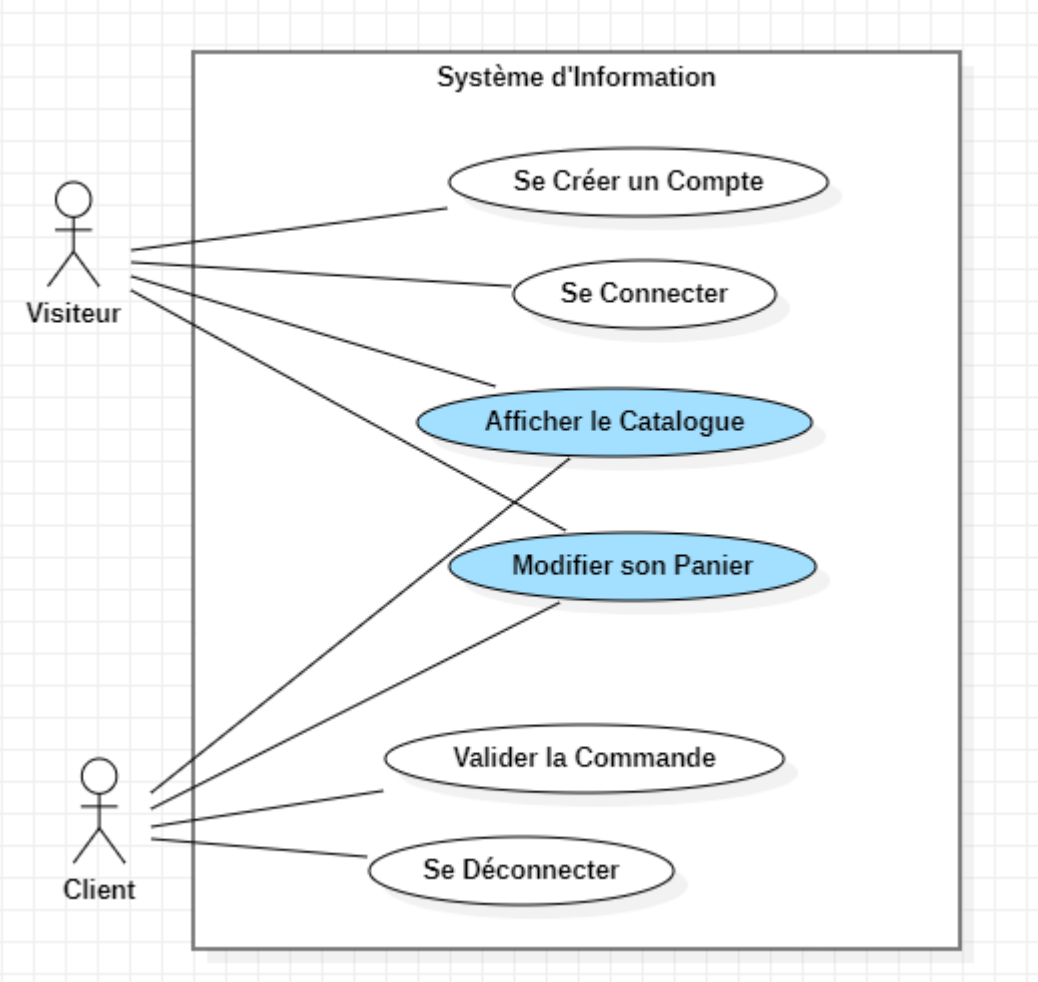


Ici, le **Client** hérite de **Visiteur**. Le **Client** peut donc Valider une Commande, mais il peut aussi Afficher le Catalogue et Modifier son Panier. Cependant, le **Visiteur** ne peut pas Valider une Commande, car il n'hérite pas de **Client**.

UML : USE CASE (Cas d'Utilisation)

Petit Conseil

Il arrive régulièrement que plusieurs **Acteurs** possèdent des **Use Cases** en commun, et des **Use Cases** uniques à eux. Dans ce cas, on ne peut pas directement faire hériter un des **Acteurs** de l'autre. Pour résoudre ce problème, on va devoir tout simplement créer un nouvel **Acteur** qui rassemblera les **Use Cases** en commun, et dont les autres **Acteurs** hériteront.

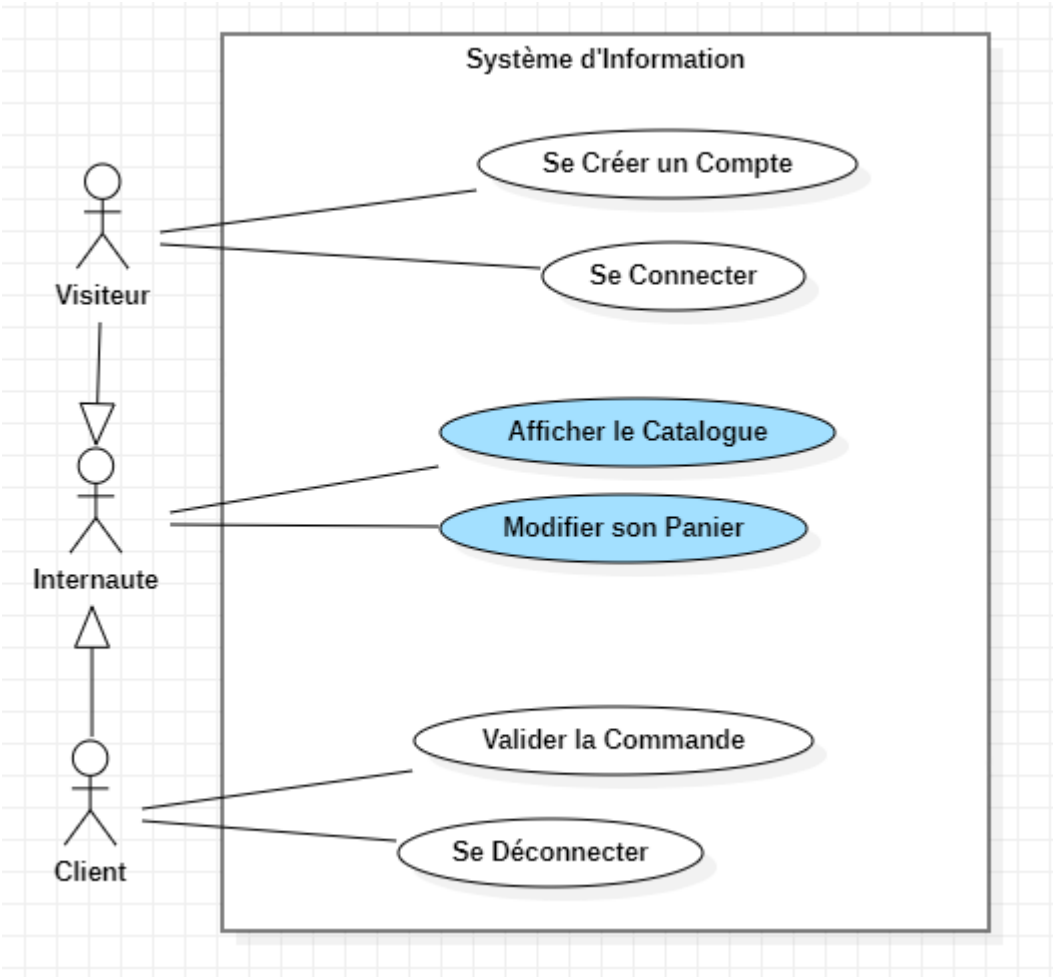


Ici, **Visiteur** et **Client** ont Afficher le Catalogue et Modifier son Panier en commun. Mais ils ont aussi chacun leur **Use Cases** propres (Se Créer un Compte et Se Connecter pour le **Visiteur**, et Valider la Commande et Se Déconnecter pour le **Client**).

Visiteur ne peut donc pas hériter de **Client**, sinon cela voudrait dire qu'il possède aussi Valider la Commande et Se Déconnecter, ce qu'on ne veut pas. Et **Client** ne peut pas hériter de **Visiteur**, sinon il possèderait aussi Se Créer un Compte et Se Connecter, ce qui n'on ne veut pas non plus.

Il faut donc créer un nouvel **Acteur** (que l'on va appeler **Internaute**) qui rassemblera les **Use Cases** communs (Afficher le Catalogue et Modifier son Panier), dont **Visiteur** et **Client** hériteront.

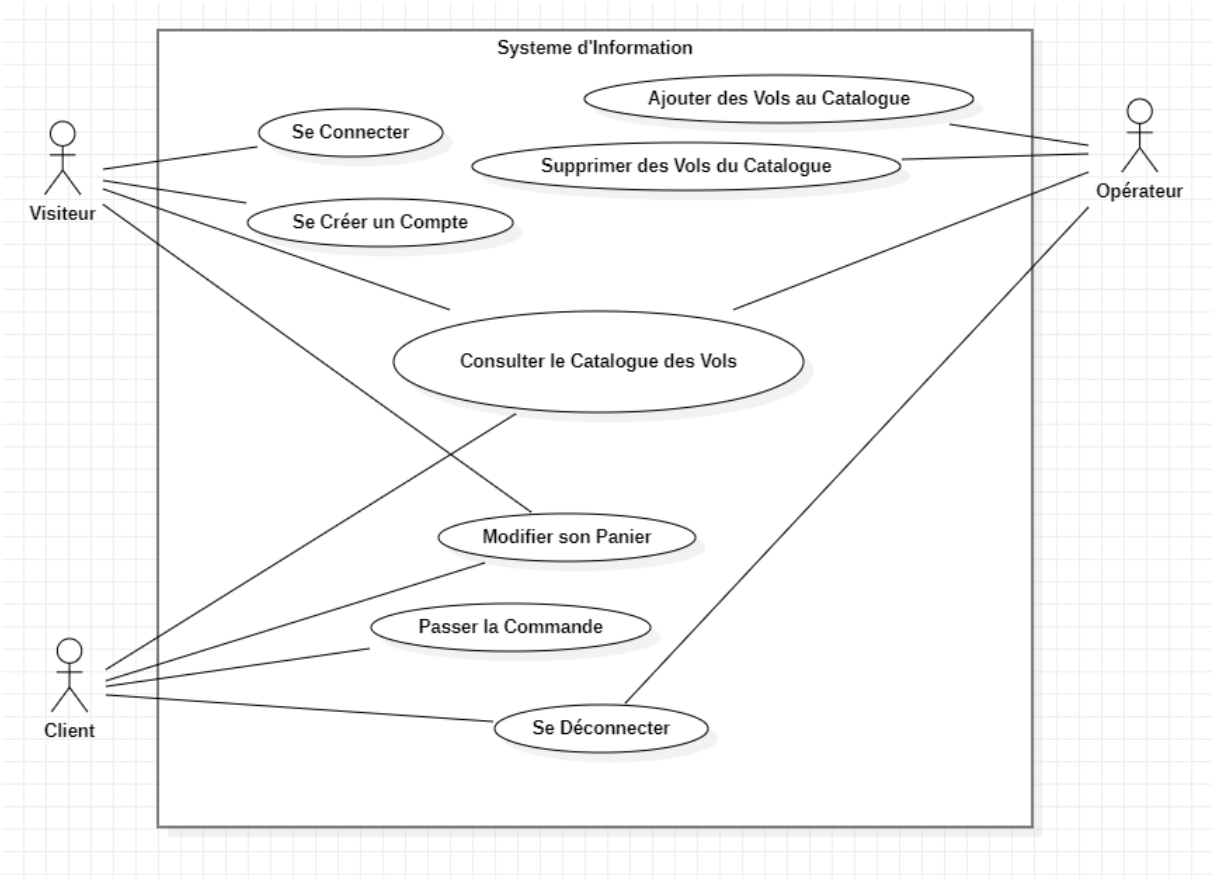
UML : USE CASE (Cas d'Utilisation)



UML : USE CASE (Cas d'Utilisation)

Application de l'Héritage à BonVol

Reprenons le **Diagramme de Cas d'Utilisation** de la compagnie BonVol

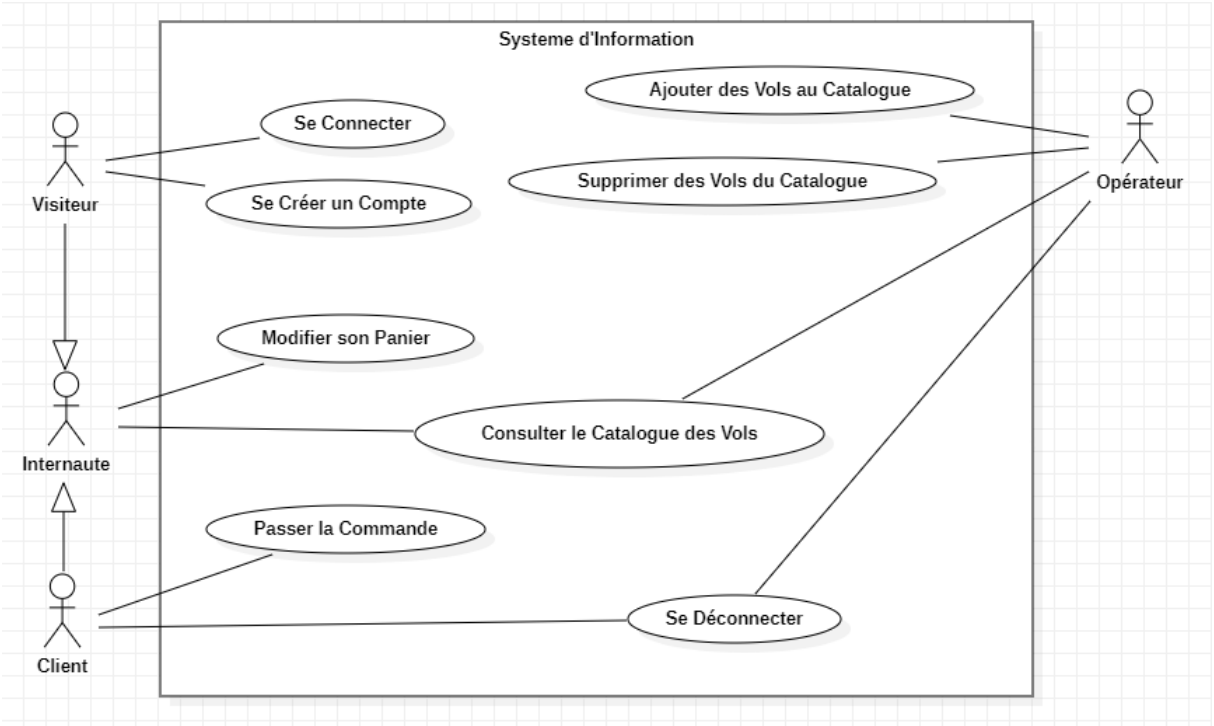


Les **Use Cases** possédés par plusieurs **Acteurs** sont Consulter le Catalogue des Vols, Modifier son Panier, et Se Déconnecter.

Commençons par appliquer le principe d'**Héritage** pour les **Use Cases** communs au **Visiteur** et au **Client**. Ne pouvant pas hériter directement l'un de l'autre, car chacun possède des **Uses Cases** que l'autre ne possède pas, nous devons créer un nouvel **Acteur** qu'on appellera **Internaute**.

On obtient le diagramme suivant :

UML : USE CASE (Cas d'Utilisation)



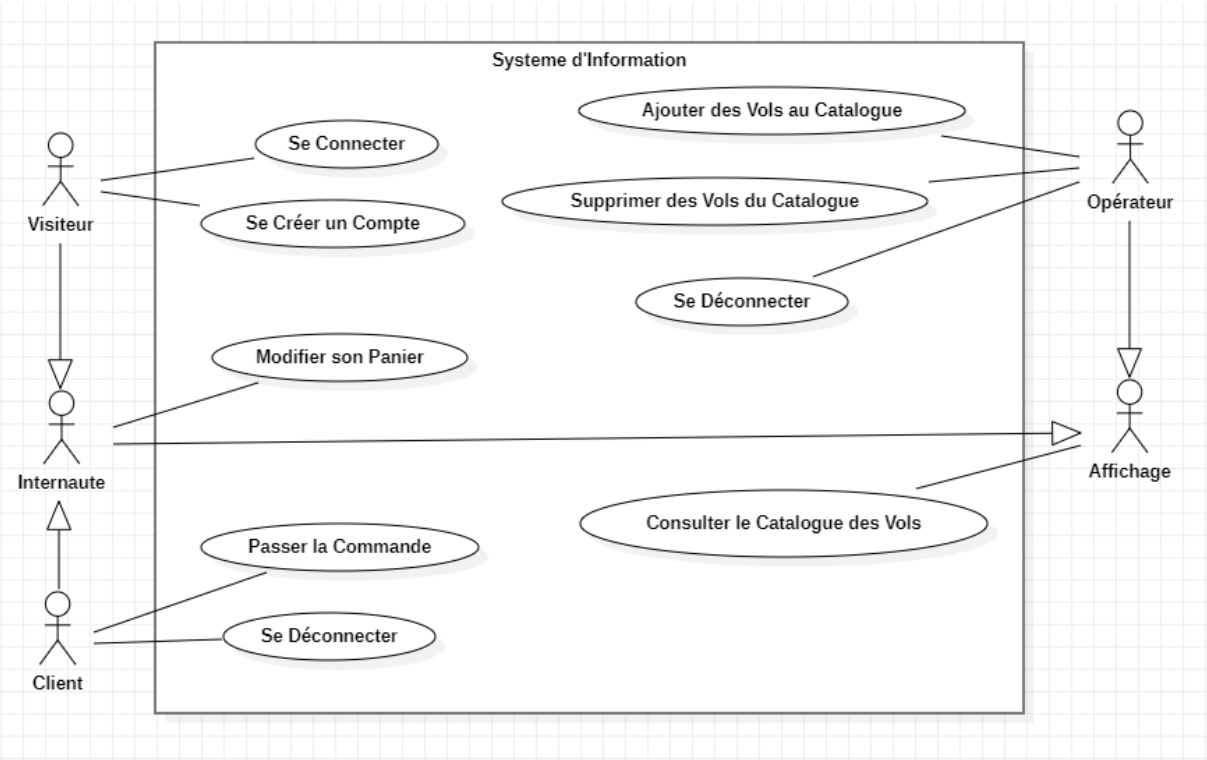
Nous remarquons maintenant que Consulter le Catalogue des Vols est commun à **Internaute** et à **Opérateur**. Mais chacun possède des **Use Cases** que l'autre ne possède pas. On va donc de nouveau créer un nouvel **Acteur** qu'on va nommer **Affichage**.

Concernant la fonctionnalité Se Déconnecter, exceptionnellement, on va le dédoubler pour ne pas avoir besoin de créer un nouvel **Acteur**. C'est OK si cela concerne peu de **Use Cases** et que cela favorise la lisibilité.

En effet, la lisibilité du diagramme est un critère majeur qui justifie l'utilisation de l'**Héritage**. Mais si créer des **Acteurs** pour appliquer l'**Héritage** rend le diagramme illisible, alors il peut être judicieux de s'en passer.

Dans notre exemple de BonVol,, créer un nouvel **Acteur** pour Se Déconnecter rendrait le diagramme plus chargé et complexe à lire.

UML : USE CASE (Cas d'Utilisation)

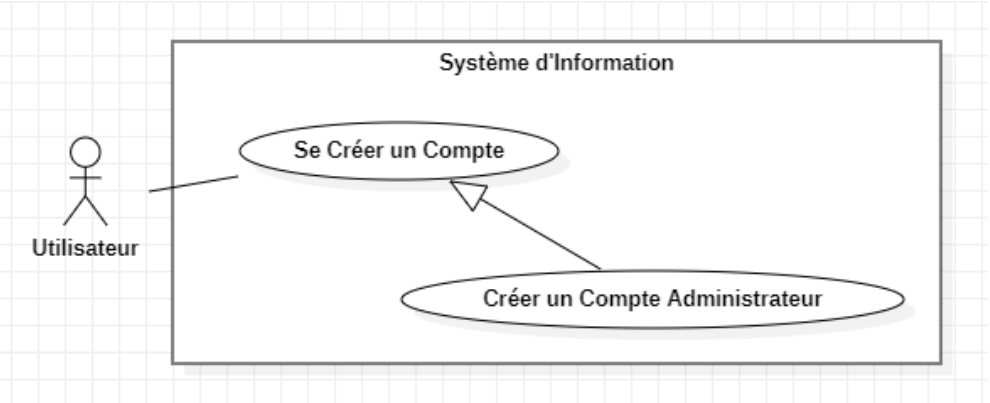


La Généralisation entre Use Cases

Tout comme les **Acteurs**, les **Use Cases** peuvent bénéficier d'une relation de **Généralisation**.

Dans le cas de **Use Cases**, une **Généralisation** est une relation où un **Use Case** est un cas particulier d'un autre. Par exemple : Créer un Compte Administrateur est un cas particulier de Créer un Compte Utilisateur. Il y a donc une relation de **Généralisation** entre les 2 **Use Cases**. Ici on dit que Créer un Compte Utilisateur est une **Généralisation** de Créer un Compte Administrateur.

La représentation graphique d'une **Généralisation** est une flèche à tête blanche (comme pour l'**Héritage**) allant du cas particulier vers le cas général.



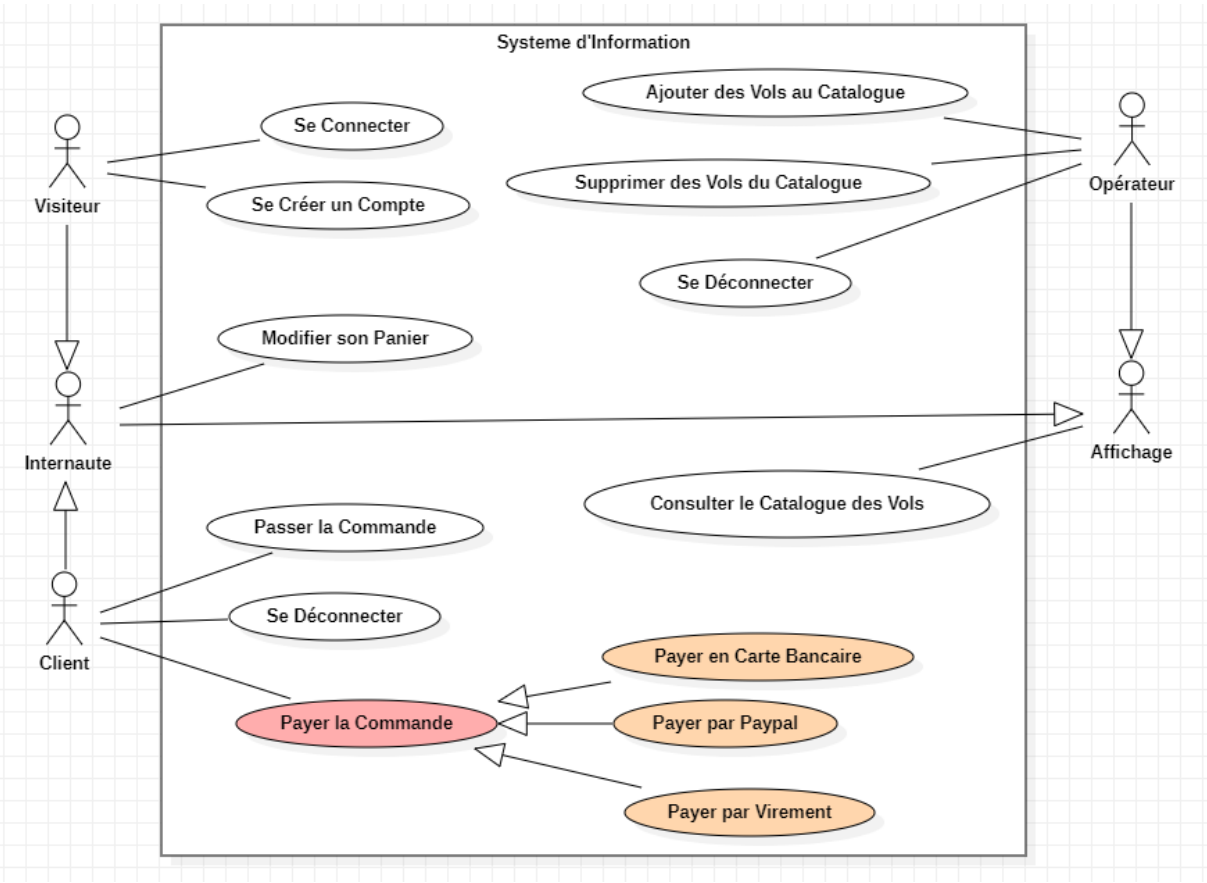
UML : USE CASE (Cas d'Utilisation)

La Généralisation au sein de BonVol

Reprenons le **Diagramme de Cas d'Utilisation** de la compagnie BonVol.

La compagnie souhaite que ceux qui commande un billet puissent payer au choix par carte bancaire, paypal, ou virement. Il s'agit là de 3 **Use Cases** particuliers du **Use Case** général Payer la Commande.

Le diagramme devient alors celui-ci :



UML : USE CASE (Cas d'Utilisation)

La relation d'Inclusion (Include)

Une relation d'**Include** est une relation entre 2 **Use Cases**.

Dans une telle relation, un **Use Case A** est une étape obligatoire pour faire aboutir un **Use Case B**. Le **Use Case A** intervient alors durant le déroulement du **Use Case B**, mais ni avant, ni après, le lancement de ce dernier. Si le **Use Case A** n'aboutit pas, alors le **Use Case B** ne pourra pas aboutir.

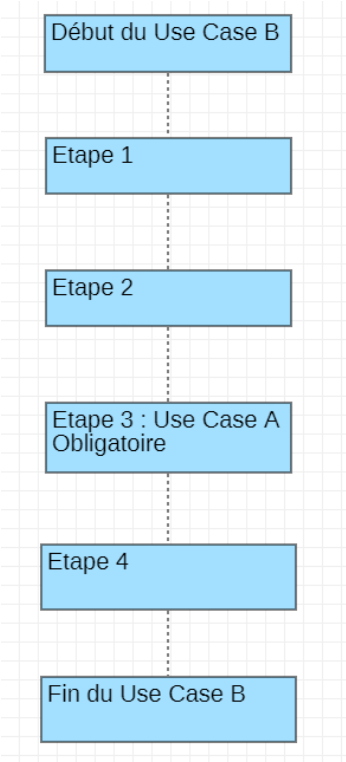
On dit que le **Use Case B** inclut le **Use Case A**.

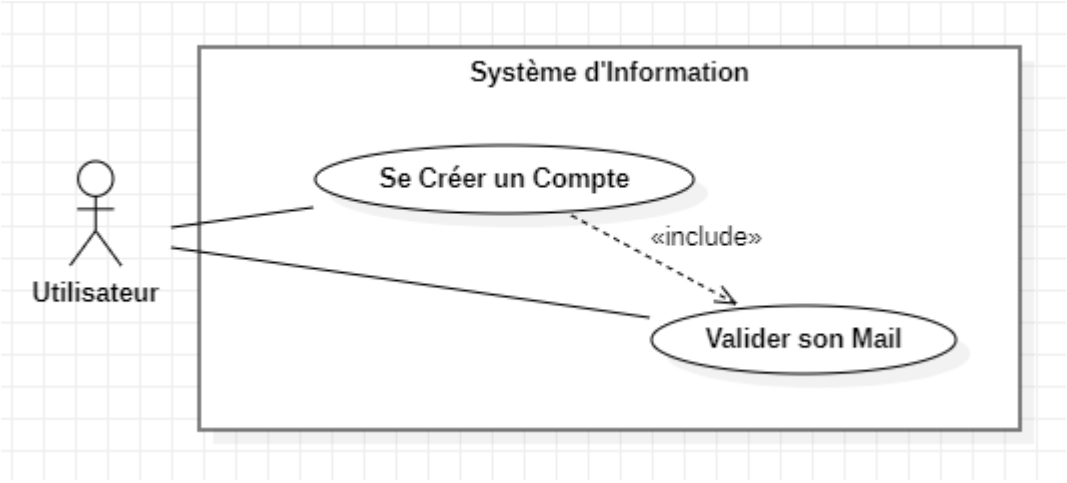
Par exemple, dans le cas où un utilisateur doit valider son mail pour finir de créer son compte, on a une relation d'**Include** entre Se Créer un Compte et Valider son Mail.

La représentation graphique d'un **Include** est une flèche en pointillé allant du **Use Case** que l'on veut faire aboutir, vers le **Use Case** qui en est une étape obligatoire, avec la mention « include » noté à côté.

Ici, la flèche part du **Use Case B** vers le **Use Case A**.

Concernant l'exemple du compte utilisateur, elle va de Se Créer un Compte vers Valider son Mail.

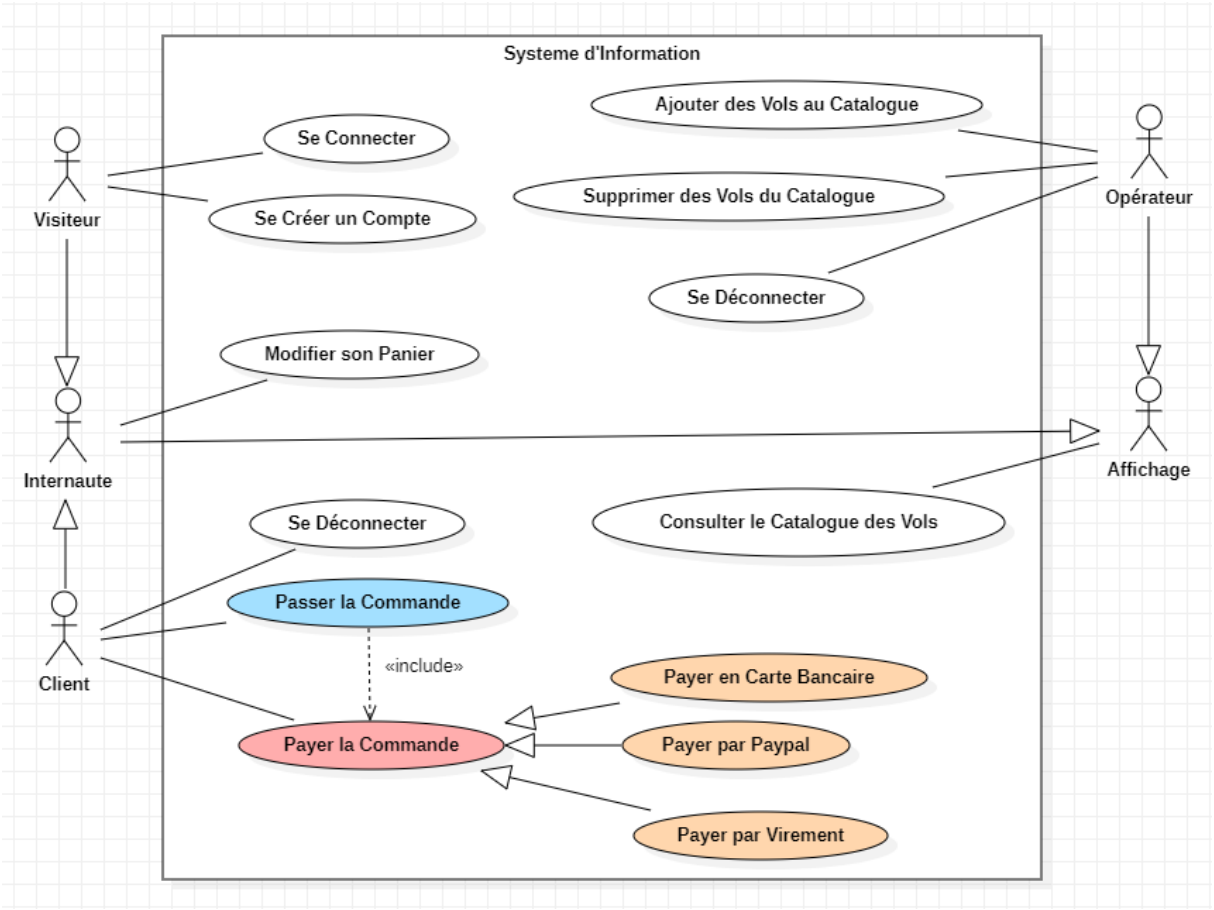




UML : USE CASE (Cas d'Utilisation)

L'Include chez BonVol

En reprenant le diagramme de la compagnie aérienne, on peut se dire que pour finaliser le fait de Passer la Commande, le **Client** est obligé de Payer la Commande. Il y a donc une relation d'**Include** entre ces 2 **Use Cases**.



UML : USE CASE (Cas d'Utilisation)

La Relation d'Extension (Extend)

Une relation d'**Extend** est une relation entre 2 **Use Cases**.

Dans une telle relation, un **Use Case A** est une étape accessible, mais optionnelle, à partir d'un **Use Case B**. Le **Use Case A** intervient alors durant le déroulement du **Use Case B**, mais n'a pas besoin d'aboutir pour mener le **Use Case B** à bien.

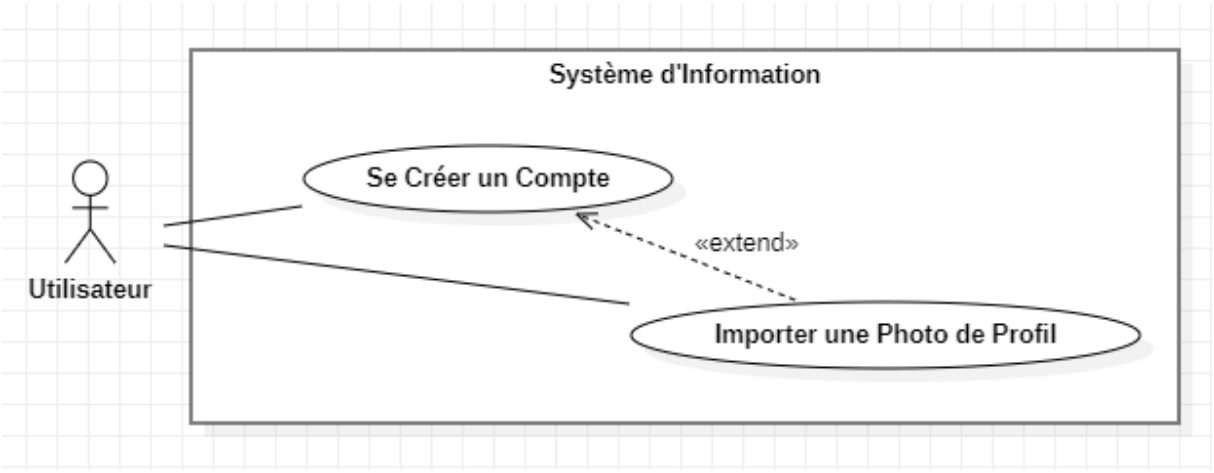
On dit que le **Use Case A** étend le **Use Case B**.

Par exemple, dans le cas où un utilisateur crée son compte, il a la possibilité d'importer une photo de Profil. On a alors une relation d'**Extend** entre Se Créer un Compte et Importer une Photo de Profil.

La représentation graphique d'un **Extend** est une flèche en pointillé allant du **Use Case** optionnel, vers le **Use Case** qui est en train de se dérouler, avec la mention « extend » noté à côté.

Ici, la flèche part du **Use Case A** vers le **Use Case B**.

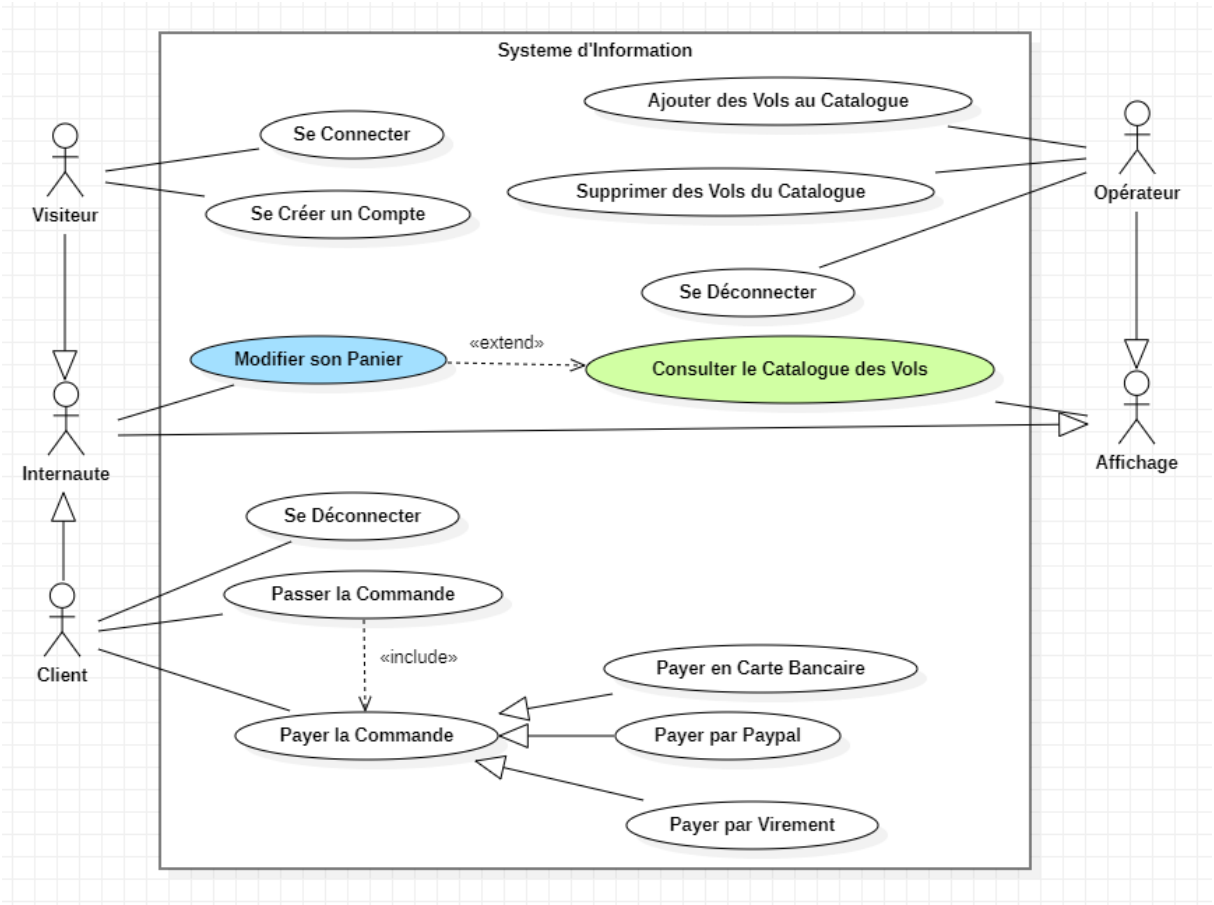
Concernant l'exemple du compte utilisateur, elle va de Importer une Photo de Profil vers Se Créer un Compte.



UML : USE CASE (Cas d'Utilisation)

L'Extend chez BonVol

En reprenant le diagramme de la compagnie aérienne, on peut vouloir faire en sorte qu'un visiteur puisse Modifier son Panier lorsqu'il est en train de Consulter le Catalogue des Vols. Dans ce cas, on a une relation d'**Extend** entre ces 2 **Use Cases**.



UML : USE CASE (Cas d'Utilisation)

Récapitulatif