

# Table des matières

<b>I.</b>	<b>Introduction</b>	2
<b>II.</b>	<b>Etude de faisabilité</b>	3
A.	Entrée	3
B.	Sortie	3
<b>III.</b>	<b>Cahier des charges</b>	3
A.	Introduction	3
B.	Cadre du projet	4
1.	Résumé du projet	4
2.	Les besoins fonctionnels	4
3.	Les livrable du projet	4
4.	Planning prévisionnel	5
<b>C.</b>	<b>Spécifications fonctionnelles</b>	7
1.	Périmètre fonctionnel	7
2.	Arborescence des pages	10
3.	Aperçu des pages	10
D.	Spécifications techniques	19
<b>IV.</b>	<b>Diagramme des cas d'utilisation</b>	20
A.	Service trafic aérien	21
1.	Gestion des pistes	21
2.	Gestions des vols	22
3.	Gestion des demandes de décollages et atterrissages	27
4.	Gestion des décollages et atterrissages	28
B.	Service d'informations	33
1.	Gestion de la liste noire	33
2.	Gestion aéroports	34
3.	Gestion compagnie aériennes	34
4.	Gestion des équipages	35
5.	Gestion des aéronefs	36
6.	Gestion type de vols	36
C.	Service d'alerte	37
D.	Service d'administration	38
1.	Gestion des redevances	38
2.	Gestion des statistique	39
3.	Gestion des comptes	39
<b>V.</b>	<b>Diagramme de séquence</b>	39

A.	Ajouter .....	40
B.	Supprimer.....	40
C.	Modifier .....	41
<b>VI.</b>	<b>Diagramme de classe.....</b>	<b>42</b>
A.	Statistique .....	43
1.	Statistique .....	43
2.	L'interface StatistiqueGlob.....	43
B.	Utilisateur.....	44
1.	Utilisateur.....	44
2.	Poste .....	45
3.	Connection .....	45
C.	Problèmes et incident .....	45
D.	Vu détailler de chaque diagramme.....	46
1.	Aéronef .....	46
2.	Compagnie .....	48
3.	Vol en cours.....	50
4.	Vol .....	52
5.	Equipage.....	54
<b>VII.</b>	<b>Conception architecturale.....</b>	<b>57</b>
<b>VIII.</b>	<b>Conception détailler .....</b>	<b>58</b>

## **I. Introduction**

Les services de la circulations aériens ATS (Air traffic service) sont un ensemble de services assurer par un organisme qui ont pour but d'assurer les vols et leur sécurité.

Ces services sont divisés en trois (le service de contrôle, le service d'information et le service d'alerte).

- Le service de contrôle aérien ATC (Air trafic contrôle) ont pour objectif d'exécuter les vols en toute sécurité et rapidité, ce service est assuré par des contrôleurs aériens
- Le service d'information informe les pilotes de toutes les informations indispensables pour le bon déroulement des vols
- Le service d'alerte intervient en action quand un aéronef a besoin d'aide, il alerte les organismes de recherche et de sauvetage.

## II. Etude de faisabilité

### A. Entrée

C'est quoi le logiciel	Pour qui ?	Pour quoi	Quels moyens	Budget
Le logiciel va servir les multiples services de la circulation aérienne, elle satisfait les besoins de chaque service et assure la coordination des services et fonctionnalité en temps réel.	Pour un service de circulation aérienne (aéroport, zone de contrôle)	<ul style="list-style-type: none"><li>- Afin de faciliter le travail et la coordination entre les multiples services.</li><li>- A diminuer le nombre de collisions entre les aéronefs</li></ul>	<ul style="list-style-type: none"><li>- <b><u>Personnel</u></b> : Syphax BARACHE Et Massinissa TIGHILIT</li><li>- <b><u>Matériel</u></b> : Eclipse, SQL, ArgoUML</li><li>- <b><u>Temps</u></b> : 2 mois</li></ul>	Gratuit !

### B. Sortie

Décision finale	Cycle de vie	Qualité	
Oui, on développe le logiciel	Par incrémentation	Haute	

## III. Cahier des charges

### A. Introduction

Les services de la circulation aérienne ont pour objectif :

- Renforcer la sécurité du trafic aérien assuré par le service du trafic aérien ATC (Air Traffic control)
- Fournir aux aéronefs tous les renseignements utiles à l'exécution sûre et efficace des vols assurés par le service d'informations
- Alerter les organismes appropriés lorsque des aéronefs ont besoin de l'aide, par le service d'alerte qui fait le travail

Afin de faciliter le travail et la coordination entre les trois services (service du trafic aérien, services d'informations et le service d'alerte) on a pour but de réaliser un logiciel qui répond au besoin des contrôleurs aériens

## B. Cadre du projet

### 1. Résumé du projet

L'application va servir les trois services, elle satisfait les besoins de chaque service et assure la coordination des services et leur fonctionnalité en temps réel.

### 2. Les besoins fonctionnels

Les besoins fonctionnels du projet sont résumés dans le tableau suivant regroupés par service :

<u>Services de trafic aérien</u>	<u>Services d'informations</u>	<u>Services d'alerte</u>
Gestion des vols	Gestion des compagnies aériennes	Gestion des problèmes et incidents
Gestion des demandes de décollage et atterrissage	Gestion des aéronefs	
Gestion des pistes	Gestion des catégories des aéronefs	
Gestion des décollage et atterrissage	Gestion des aéroports (de destination et départ)	
	Gestion de la liste noire	
	Gestion des types de vol	
	Gestion des équipages	
<u><b>Autre fonctionnalité</b></u>		
Gestion des redevances (automatiquement)		
Gestion des statistiques		
Gestion des comptes		

### 3. Les livrables du projet

Pour bien mener le projet notre groupe va réaliser les livrables suivants :

Cahier des charges	Jeudi 16 Avril 2020
Diagramme des cas d'utilisation	Jeudi 23 Avril 2020
Diagramme d'activité et diagramme de séquence	Lundi 27 Avril 2020
Diagramme de classe et diagramme d'objet	Jeudi 30 Avril 2020
Conception architecturale	Jeudi 7 Mai 2020
Conception détailler	Jeudi 14 Mai 2020
Code source	Samedi 30 Mai 2020

#### 4. Planning prévisionnel

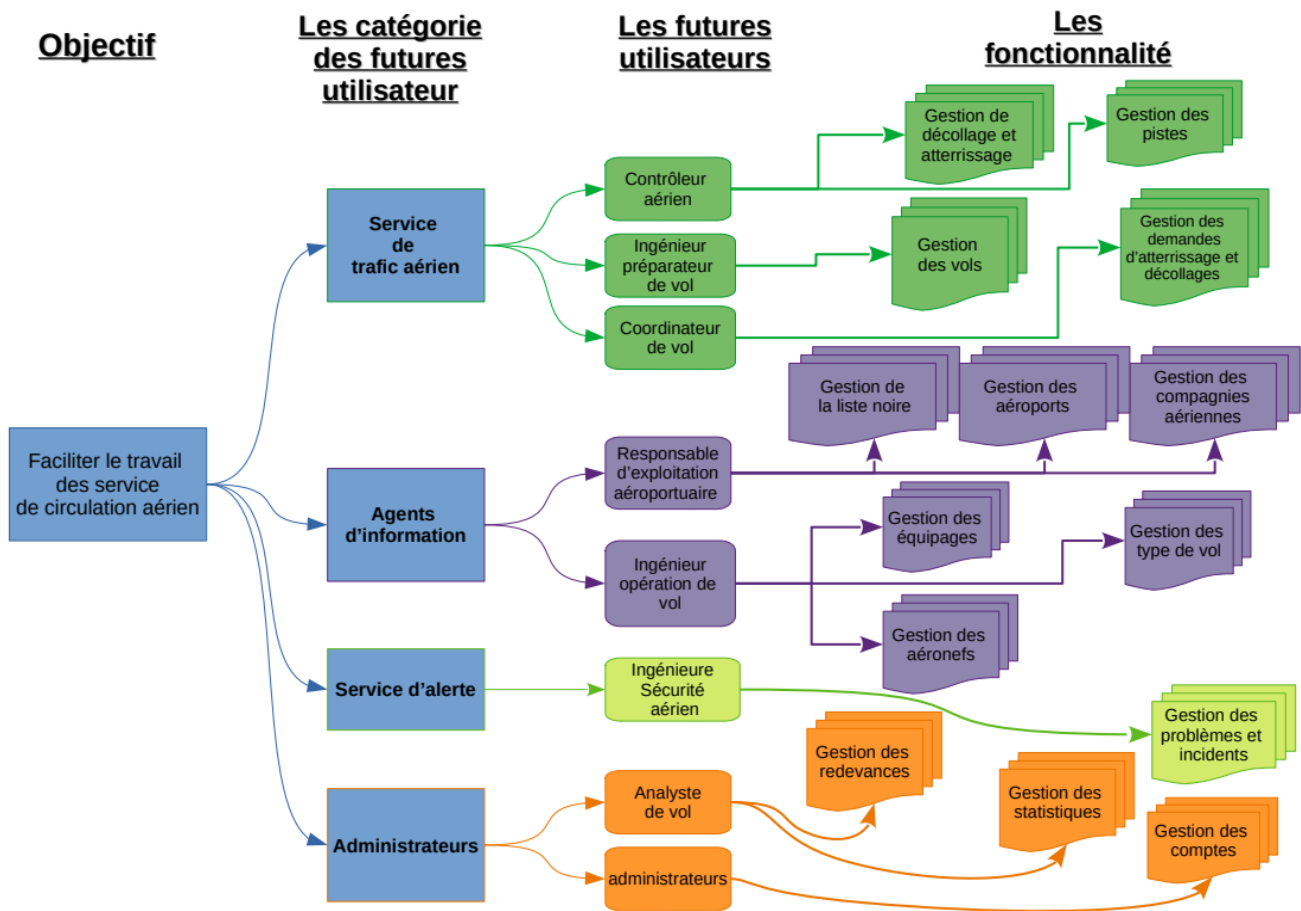
Projet	Tache	Date début	Date fin	Progression	Réalisateur
Cahier des charge		11 avril 2020	16 avril 2020	66%	Massinissa && syphax
	Recueil des besoin	11 avril 2020	11 avril 2020	100%	massinissa
	Cadre du projet	11 avril 2020	12 avril 2020	100%	massinissa
	Arborescence des pages	11 avril 2020	11 avril 2020	100%	massinissa
	Périmètre fonctionnelle	13 avril 2020	15 avril 2020	20%	Syphax && massinissa
	Aperçu des pages	12 avril 2020	15 avril 2020	10%	Syphax
Spécification des besoin		17 avril 2020	14 mai 2020	0%	Massinissa && syphax
	Diagramme des cas d'utilisation	17 avril 2020	23 avril 2020	0%	Massinissa && syphax
	Diagramme d'activité	23 avril 2020	27 avril 2020	0%	massinissa
	Diagramme de séquence	23 avril 2020	27 avril 2020	0%	syphax

	Diagramme de classe	27 avril 2020	30 avril 2020	0%	massinissa
	Diagramme d'objet	27 avril 2020	30 avril 2020	0%	syphax
Conception architectura		30 avril 2020	7 mai 2020	0%	
Conception détailler		7 mai 2020	14 mai 2020	0%	
Programmation		14 mai 2020	30 mai 2020	0%	
<b><u>Générale</u></b>		11 avril 2020	30 mai 2020	16,50 %	Massinissa && syphax

## C. Spécifications fonctionnelles

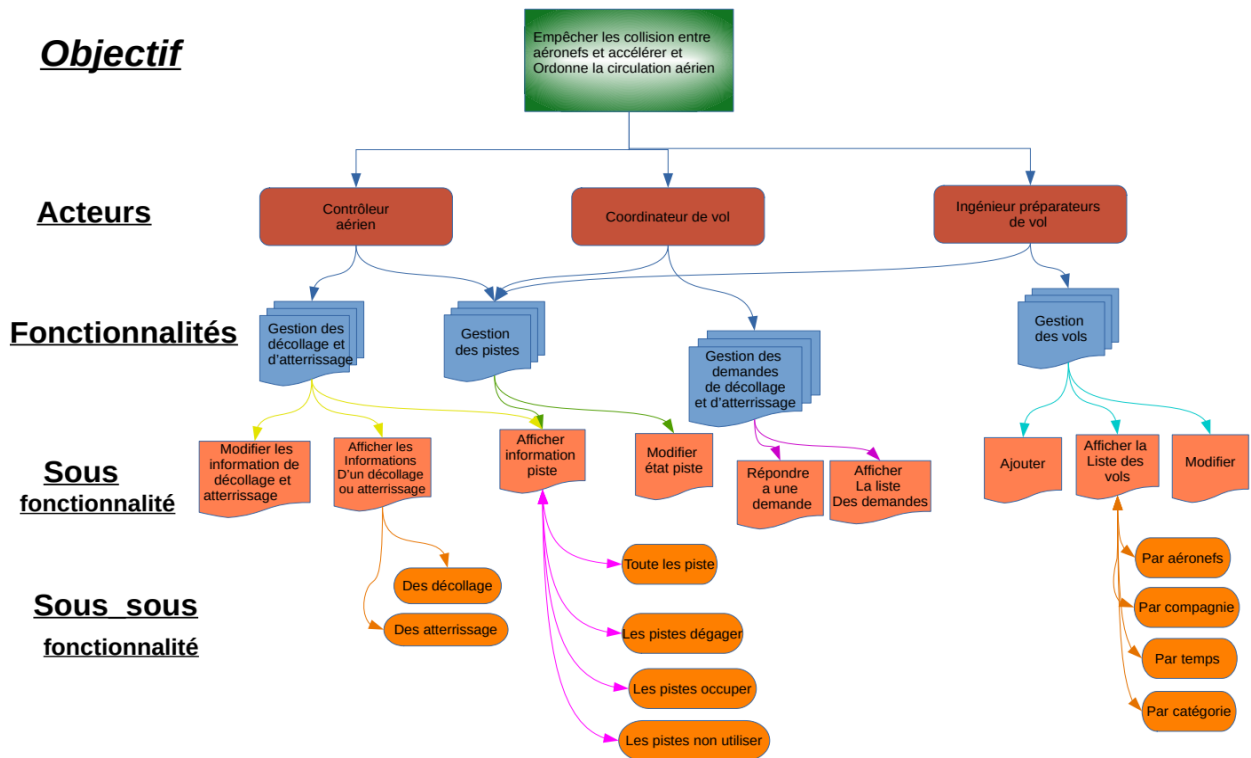
### 1. Périmètre fonctionnel

a) *Vu globale*

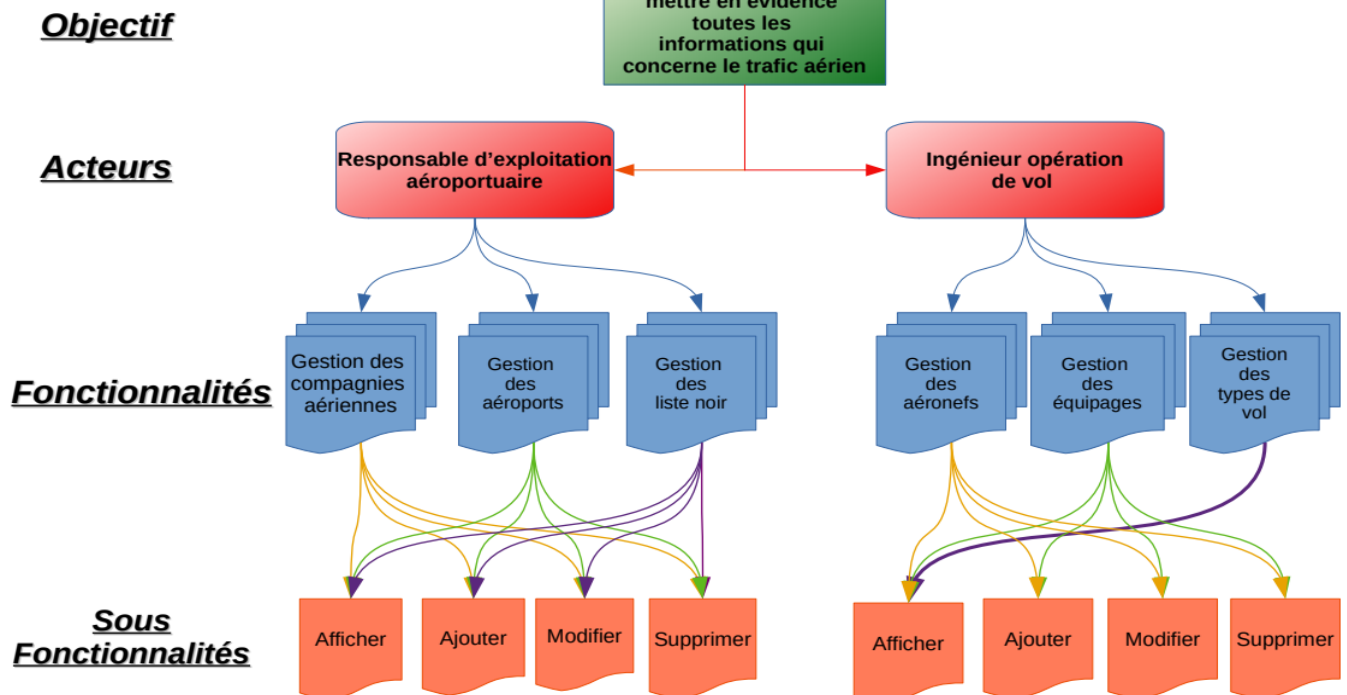


## b) Vue détailler

### (1) Service trafic aérien

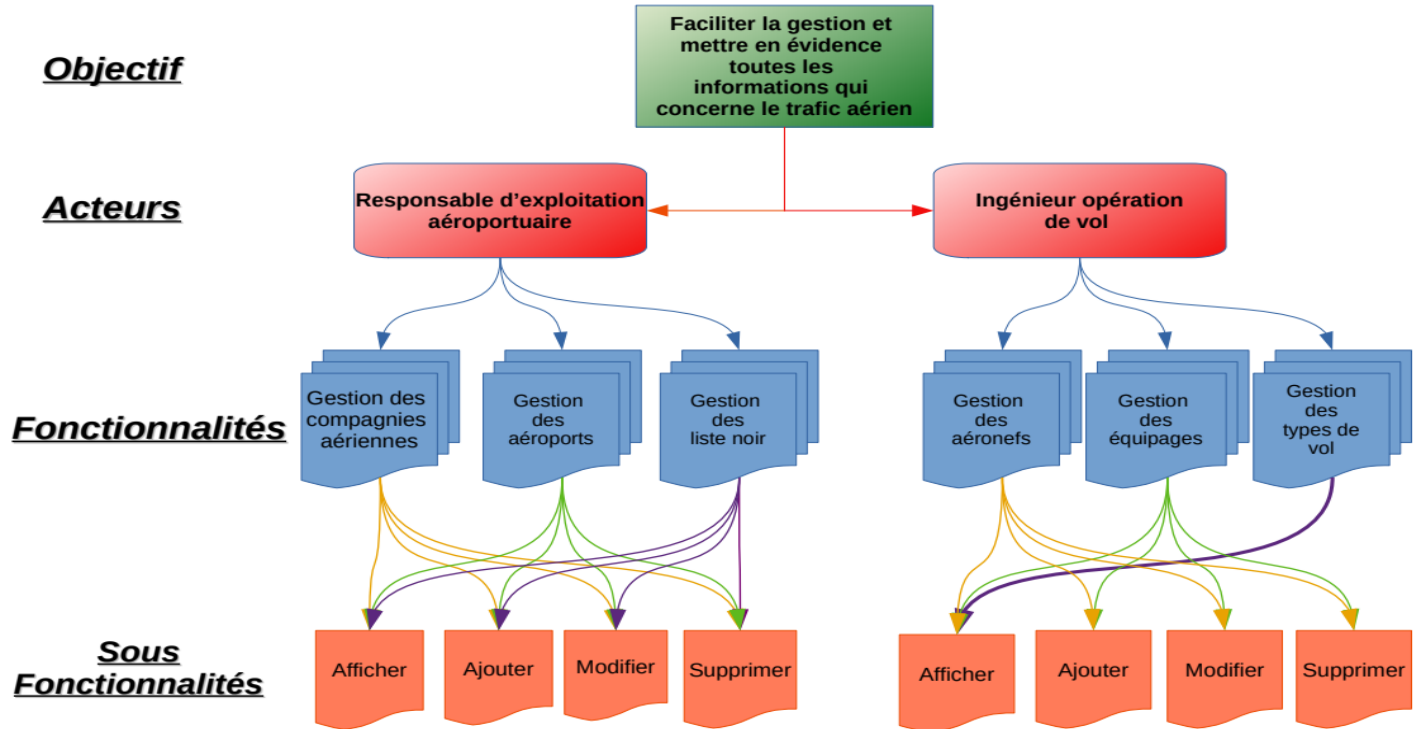


### (2) Service d'information

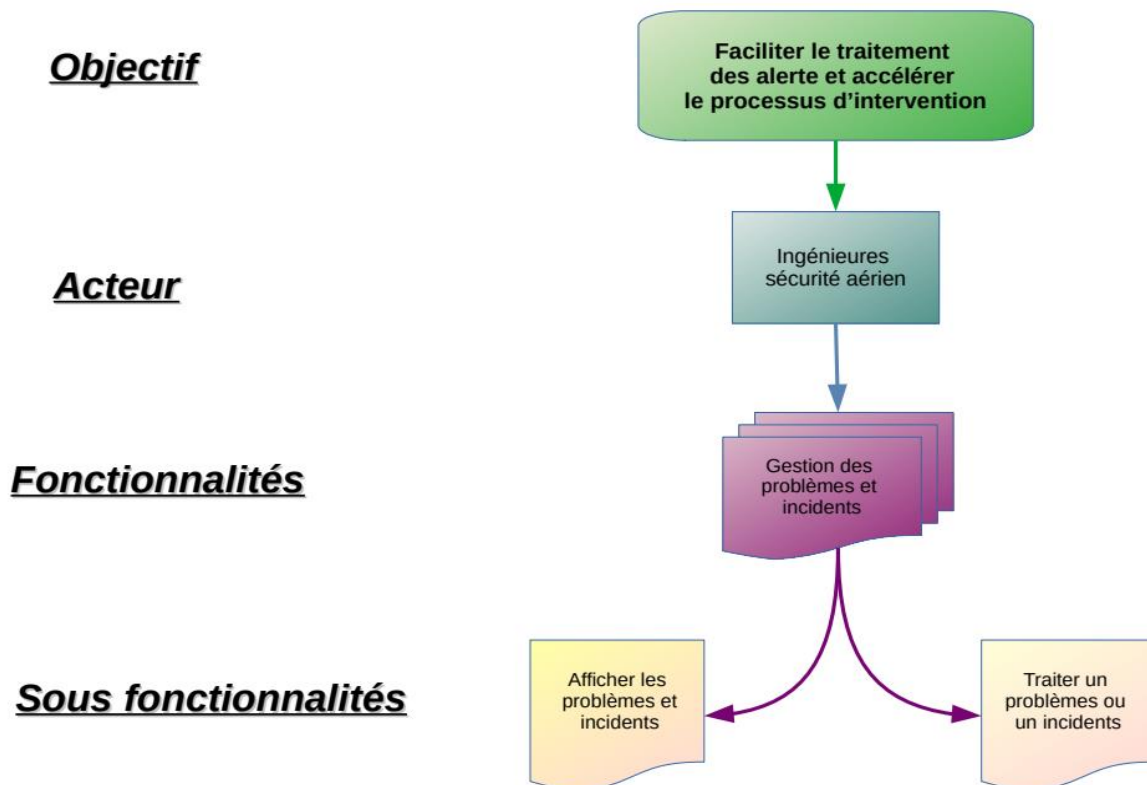




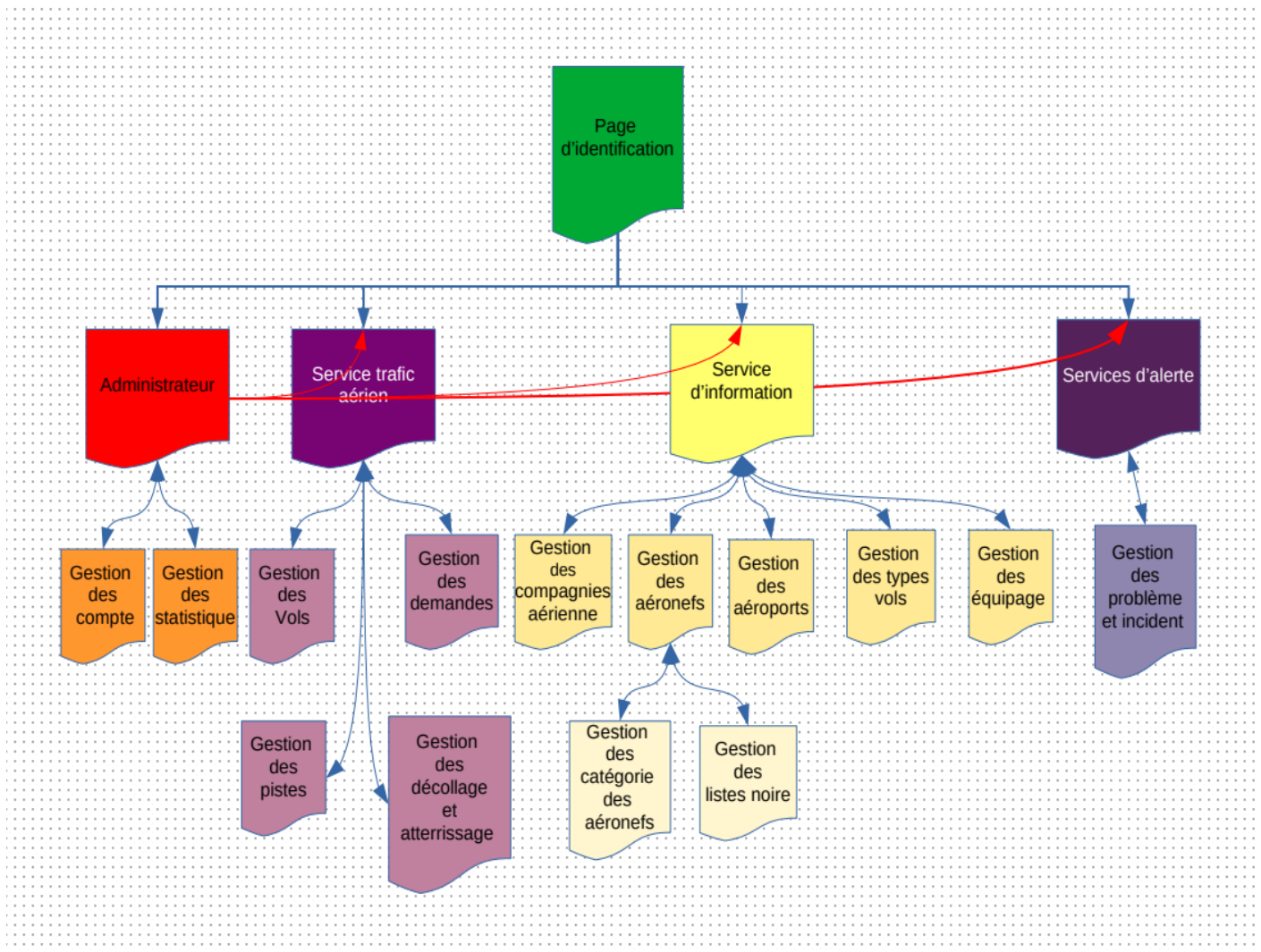
(3) Service d'administration



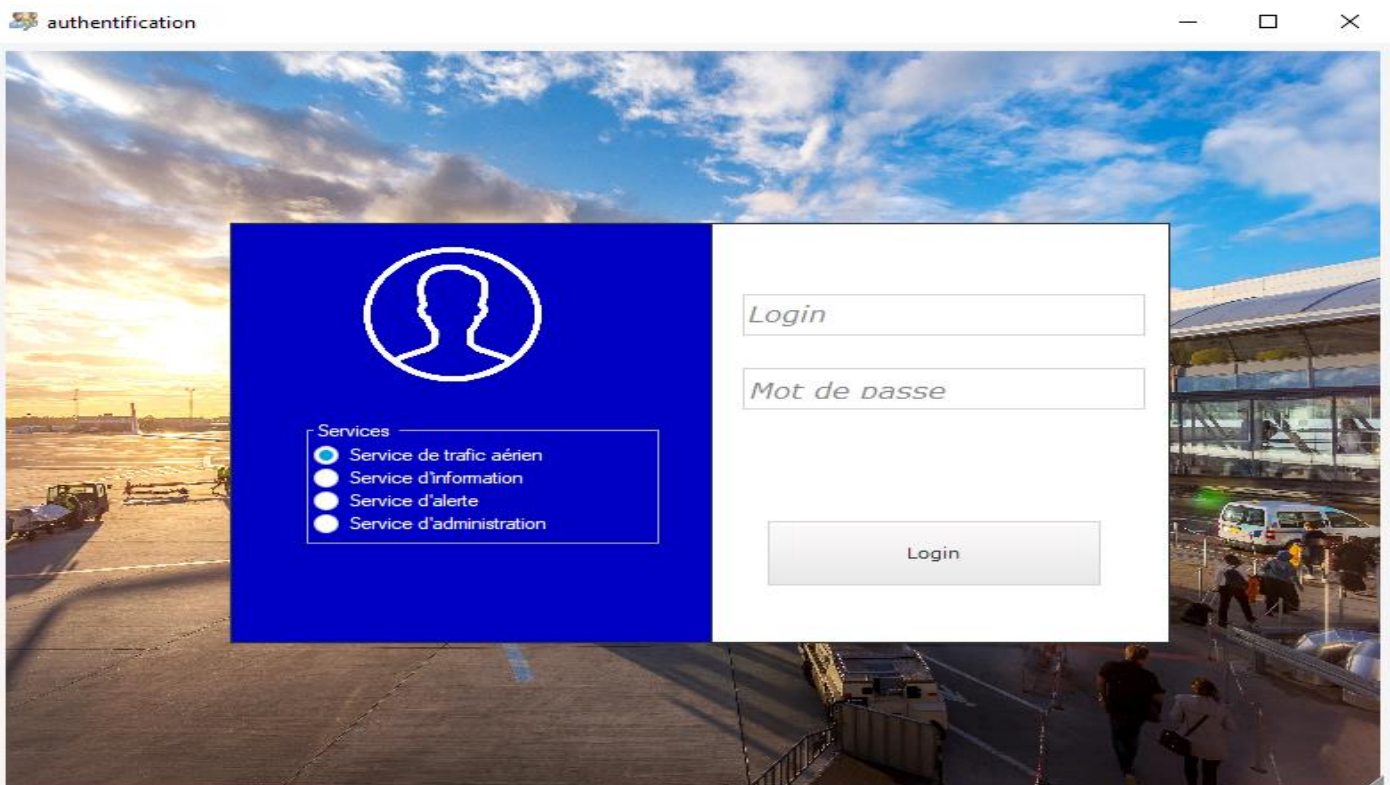
(4) Service d'alerte



## 2. Arborescence des pages



## 3. Aperçu des pages



[illegible][illegible]

**! Service d'alerte**

### Paramètres



## Répondre a une demande



**Afficher La**  
**liste Des**  
**demande**

### Données

### Paramètres



**Modifier information Décollage et atterrissage**



**Afficher les Informations nécessaire**



Annuler un  
Décollage ou un  
atterrissag

### Données

—      □      ×



### Données

—      □      ×



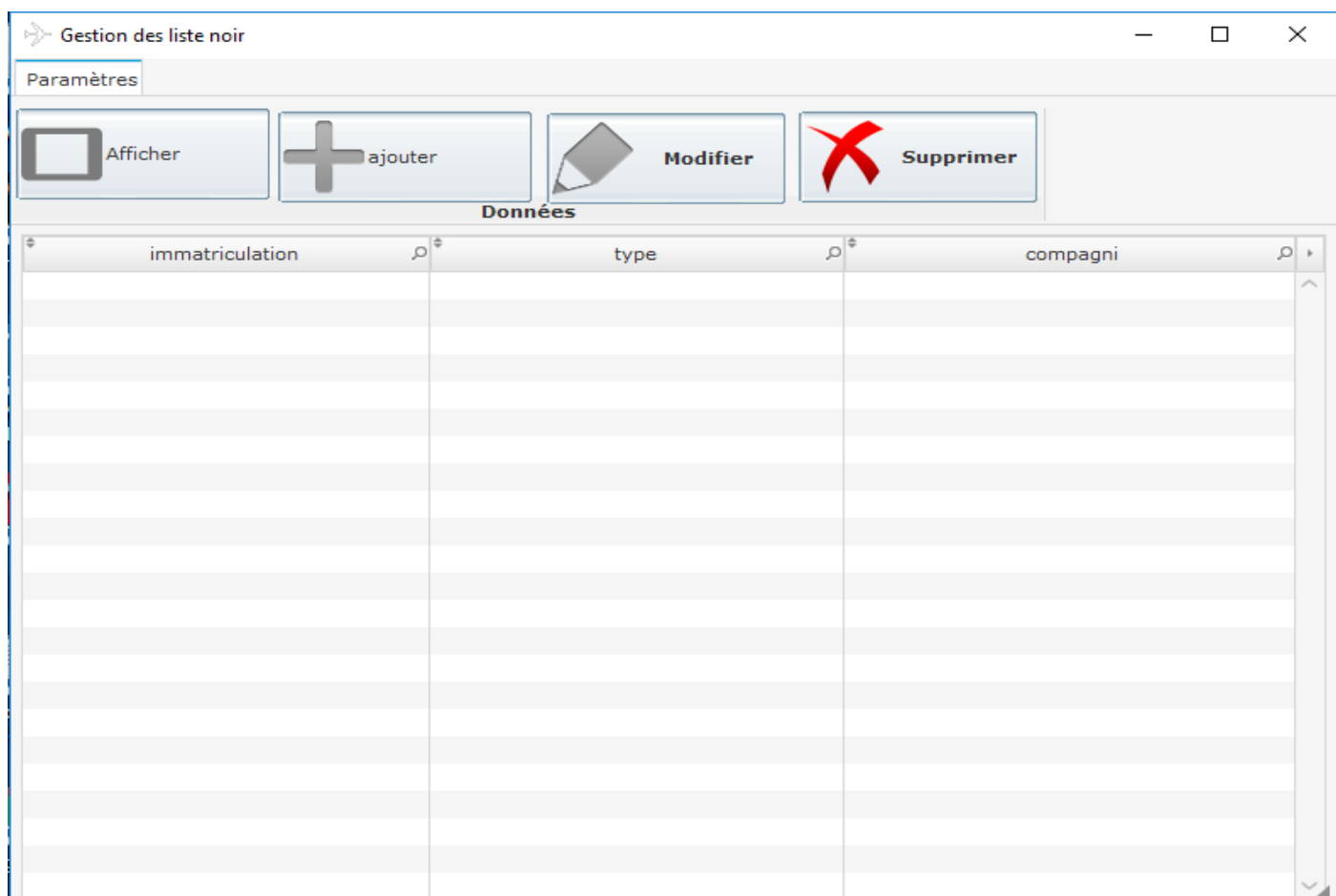
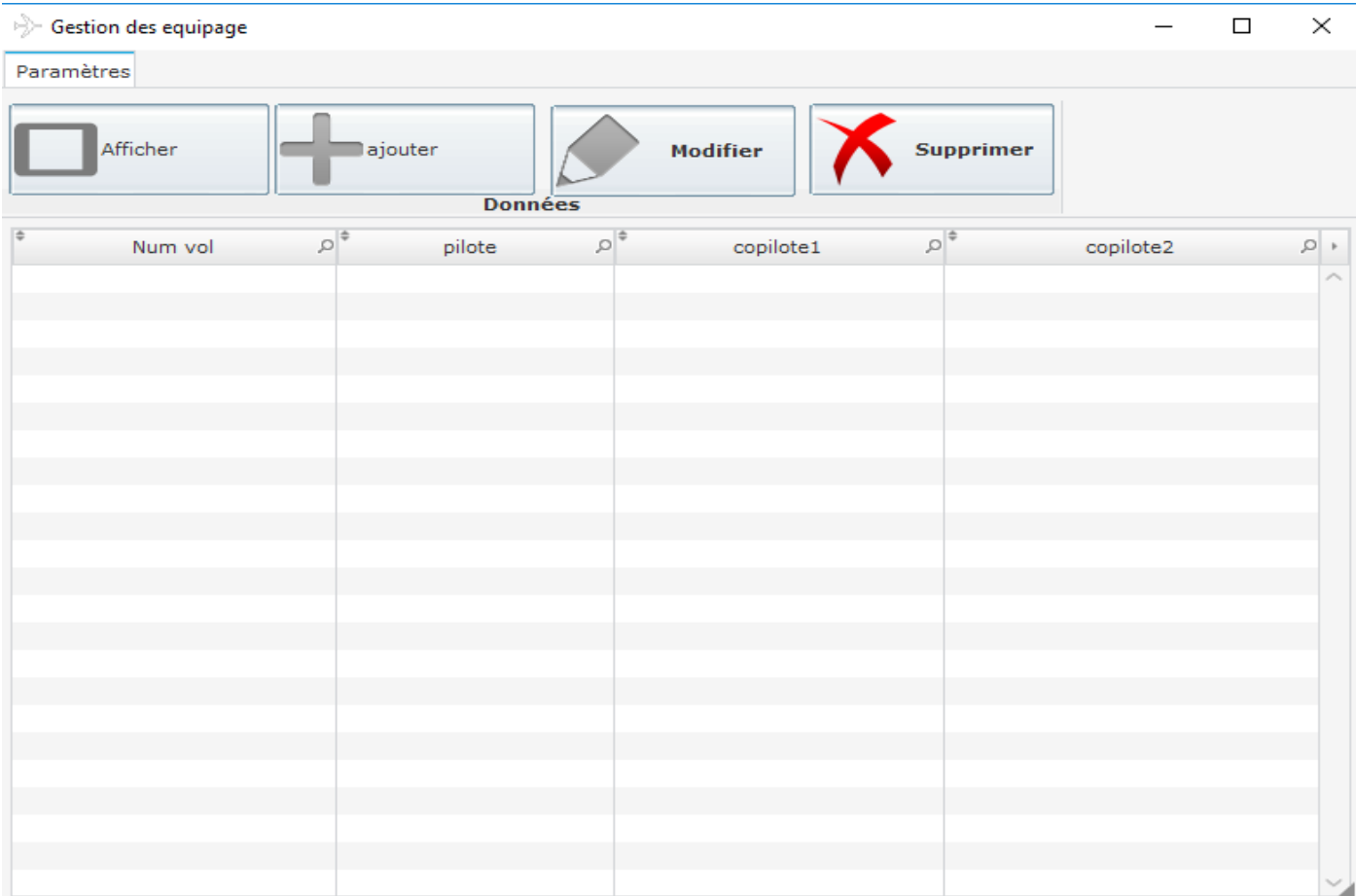
### Données

—      □      ×

 Afficher
  ajouter
  Modifier
  Supprimer

—      □      ×

 Afficher
  ajouter
  Modifier
  Supprimer








—      □      ×

 Afficher
  ajouter
  Modifier
  Supprimer

—      □      ×

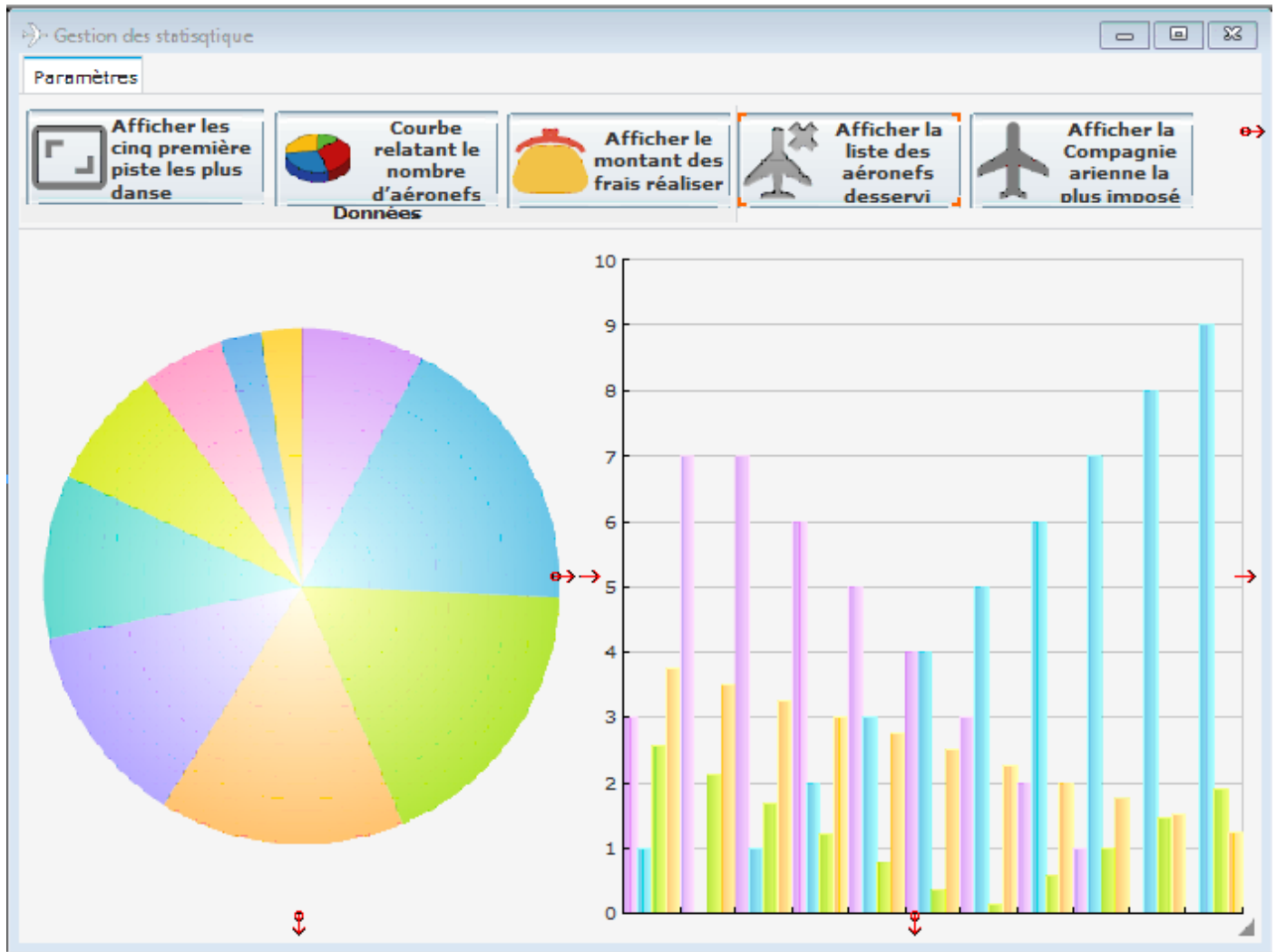
-  Traiter un problème ou un incident
-  Afficher les problèmes et incidents

—      □      ×

 Ajouter
  Modifier
  Supprimer

—      □      ×

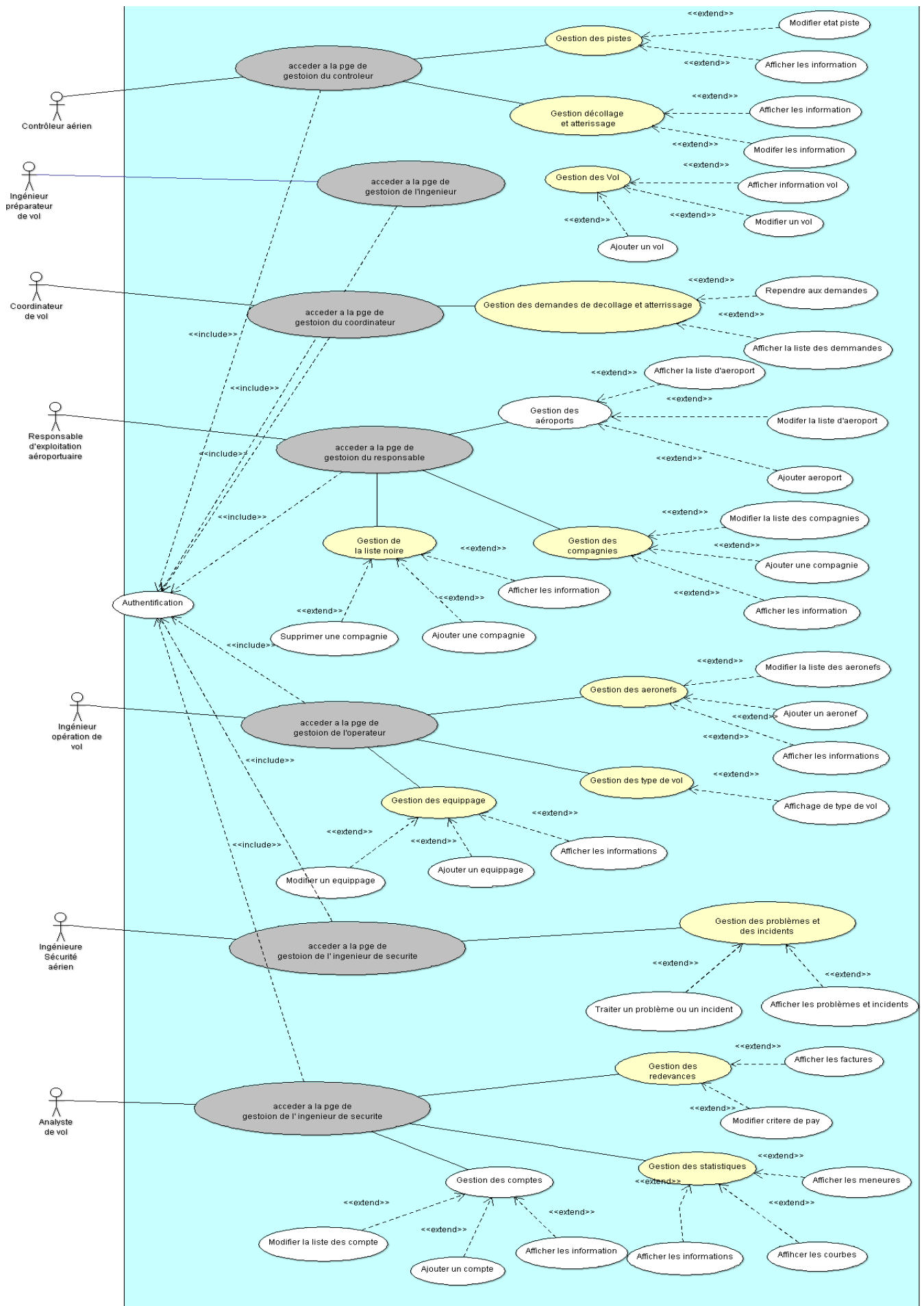
**Aficher les  
facture**



## D. Spécifications techniques

- Programmation avec JAVA
- Utilisation de la bibliothèque JAVA FX

## IV. Diagramme des cas d'utilisation

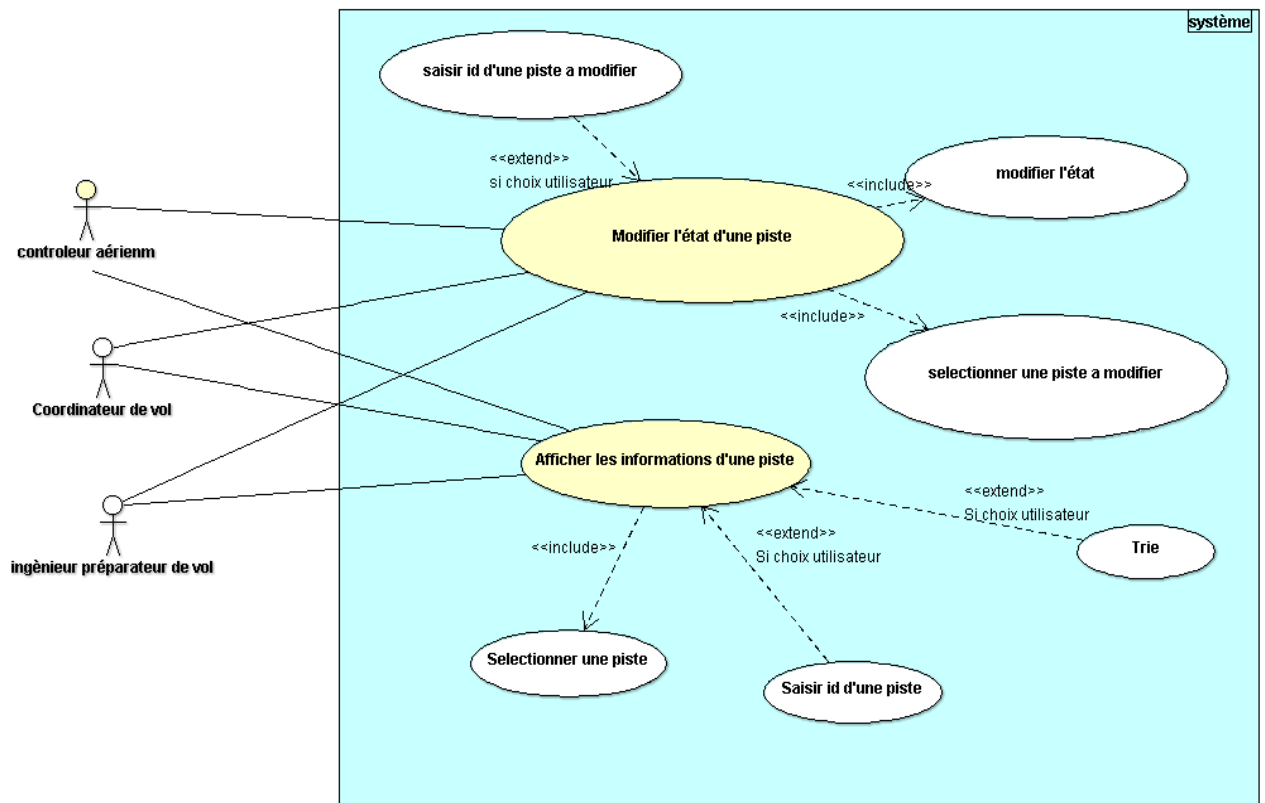


Les diagrammes de cas d'utilisation sont divisés en 4 packages

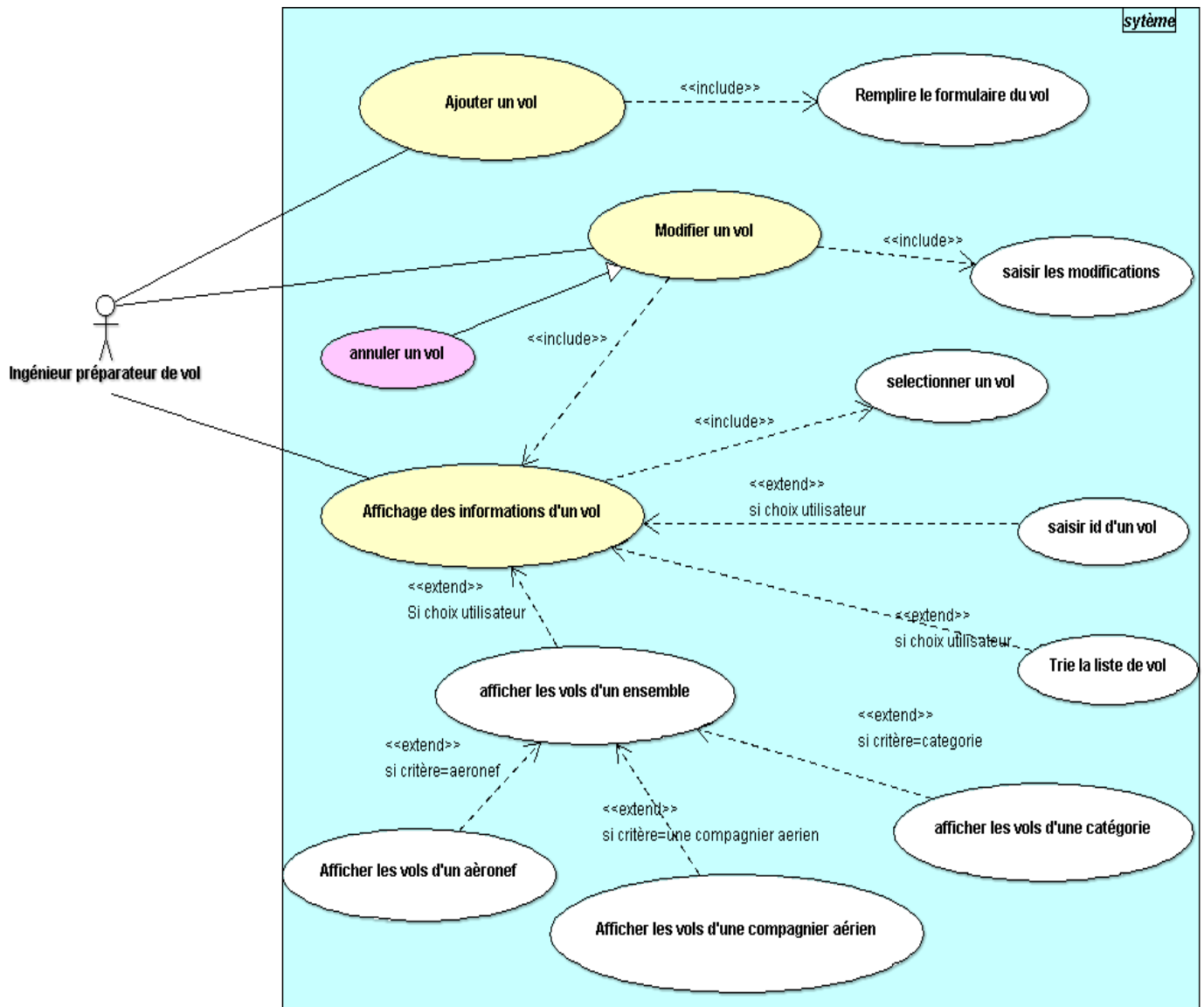
- Service trafic aérien
- Service d'informations
- Services d'alerte
- Service d'administration

## A. Service trafic aérien

### 1. Gestion des pistes

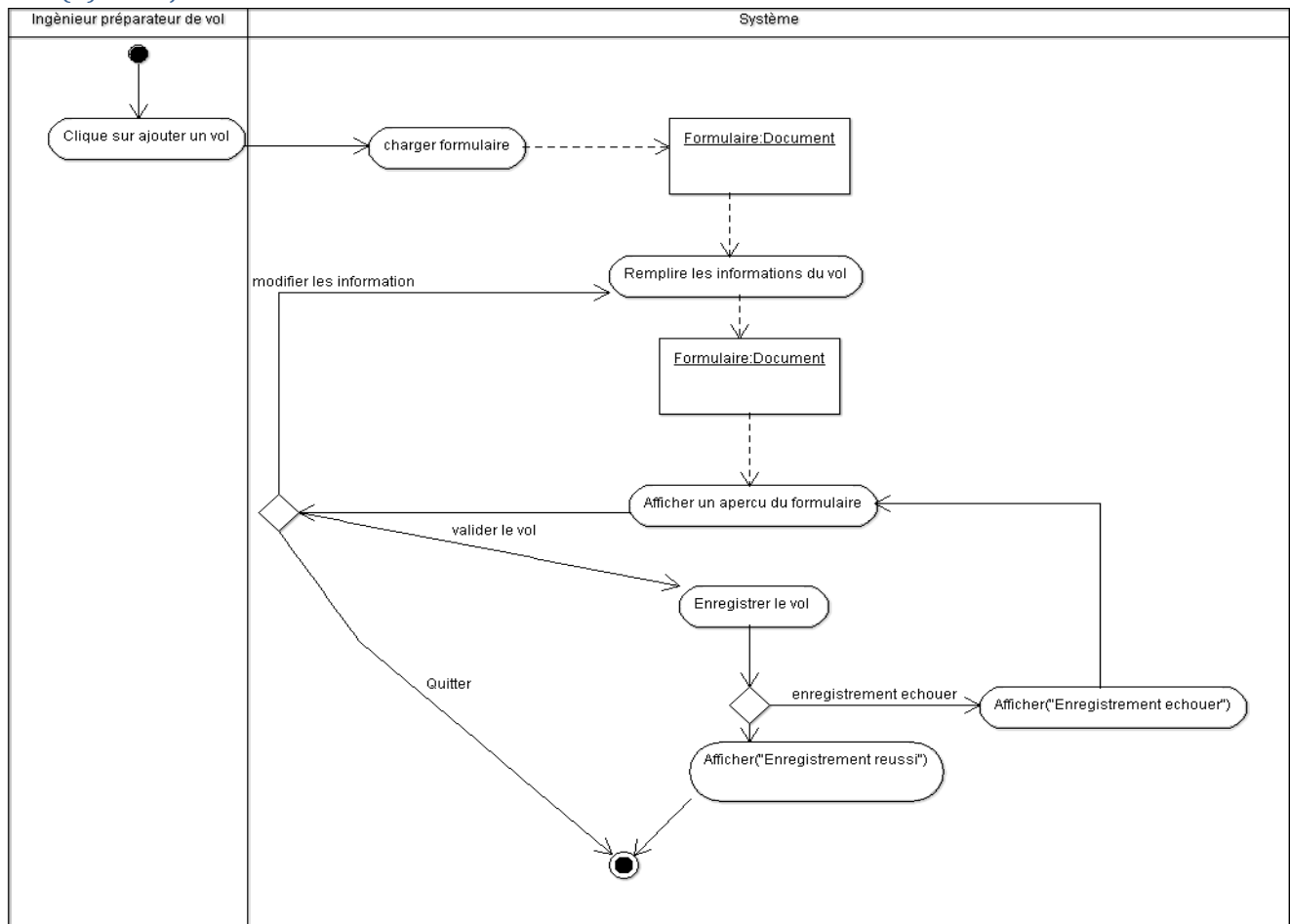


## 2. Gestions des vols

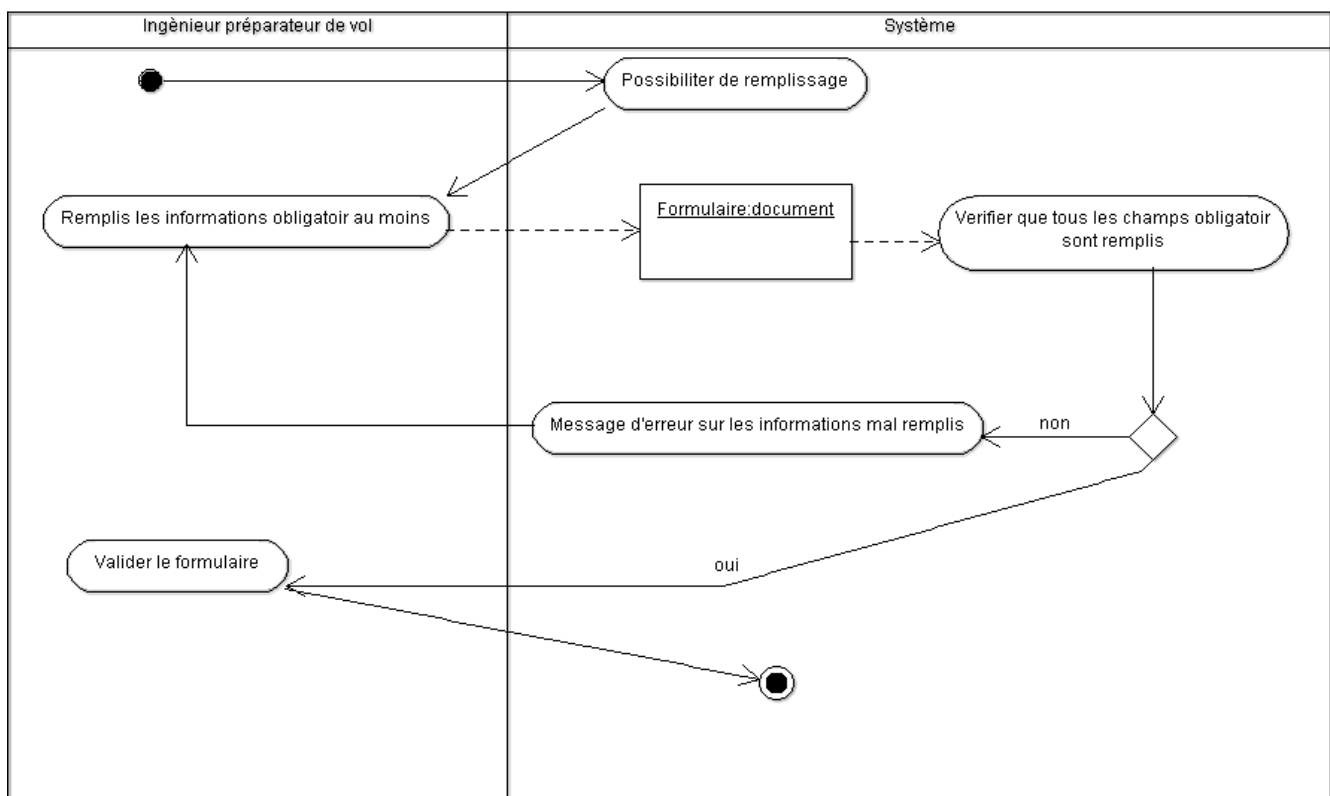


a) Description avec diagramme d'activité

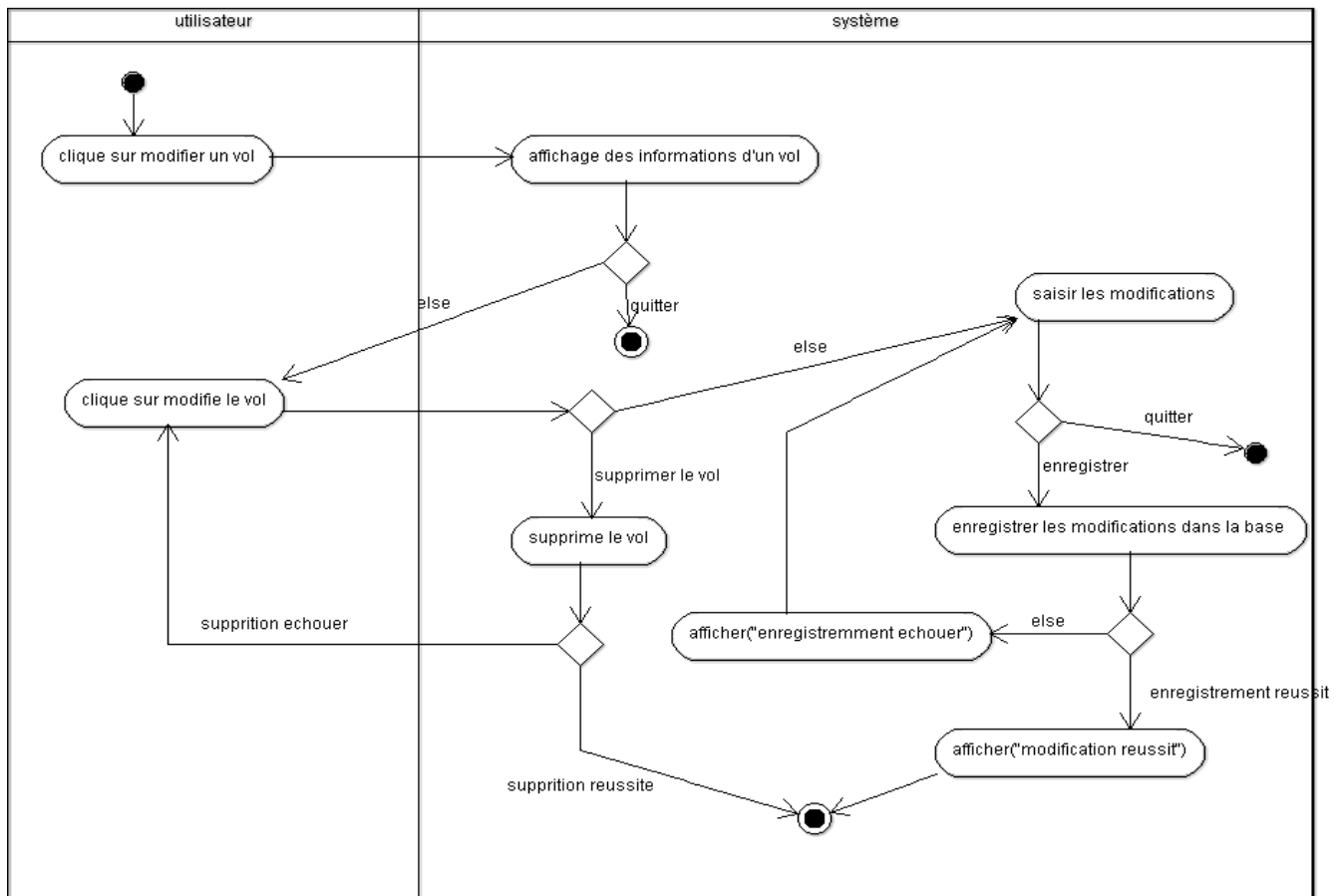
(1) Ajouter un vol



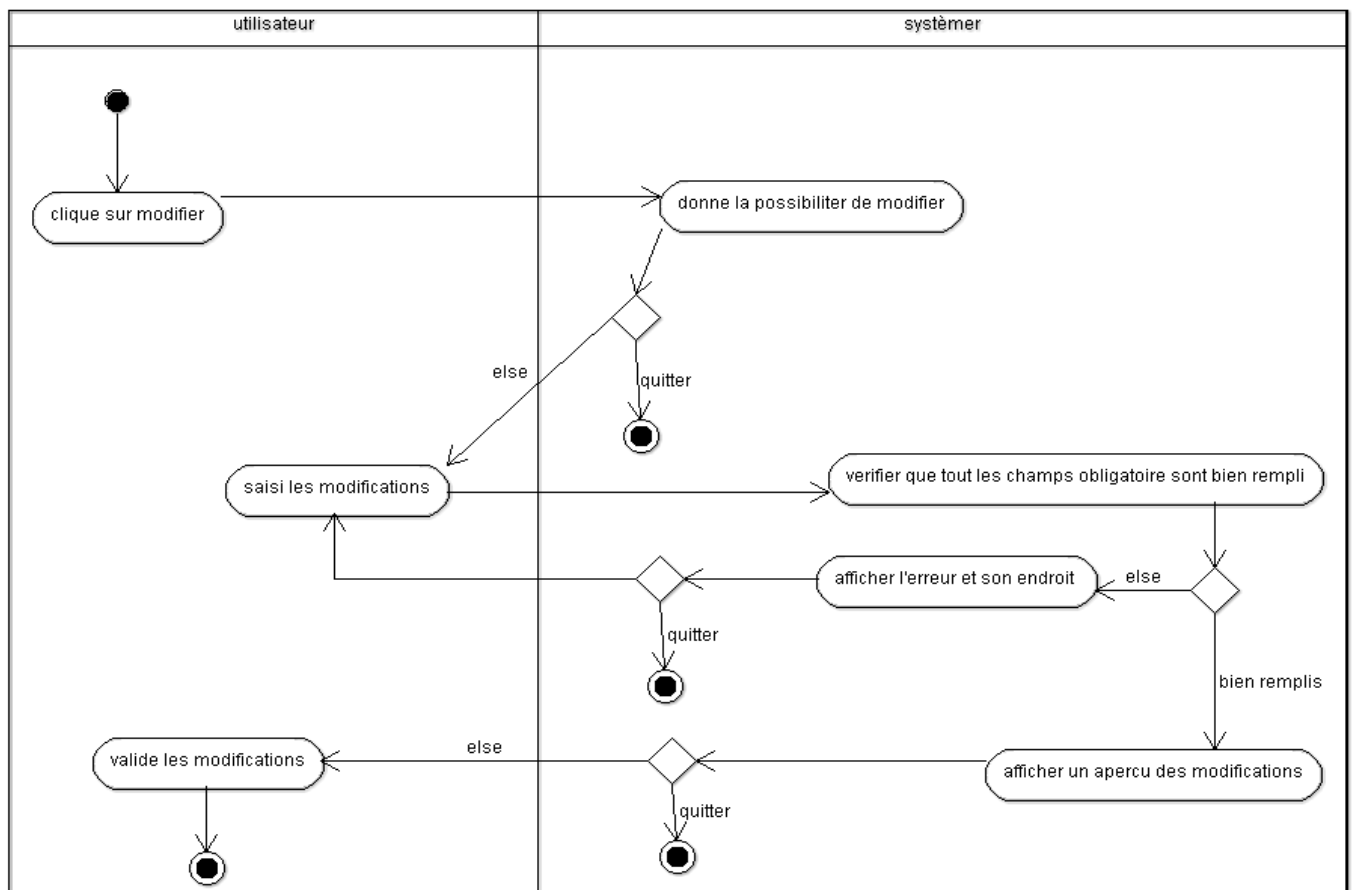
(a) Remplire le formulaire



## (2) Modifier un vol



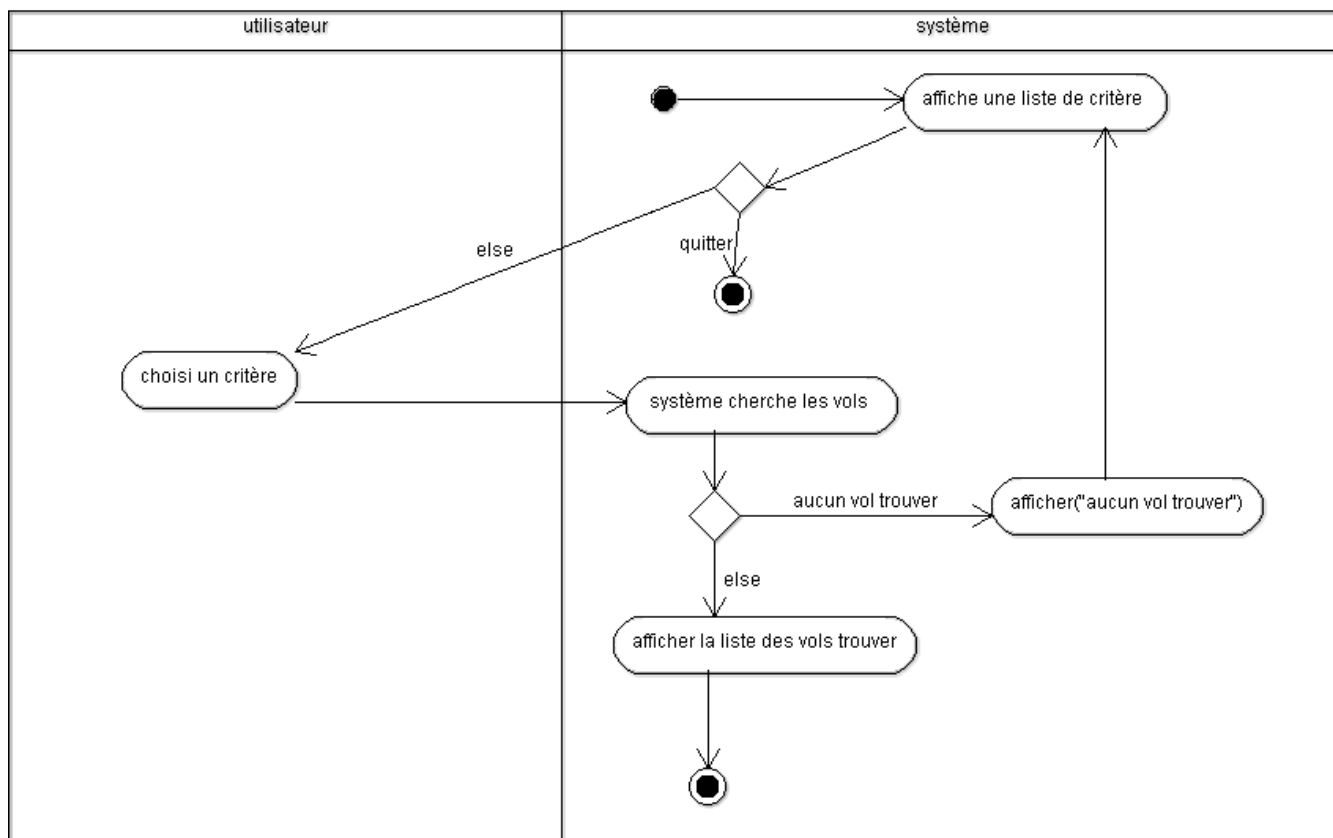
(a) Saisir les modifications



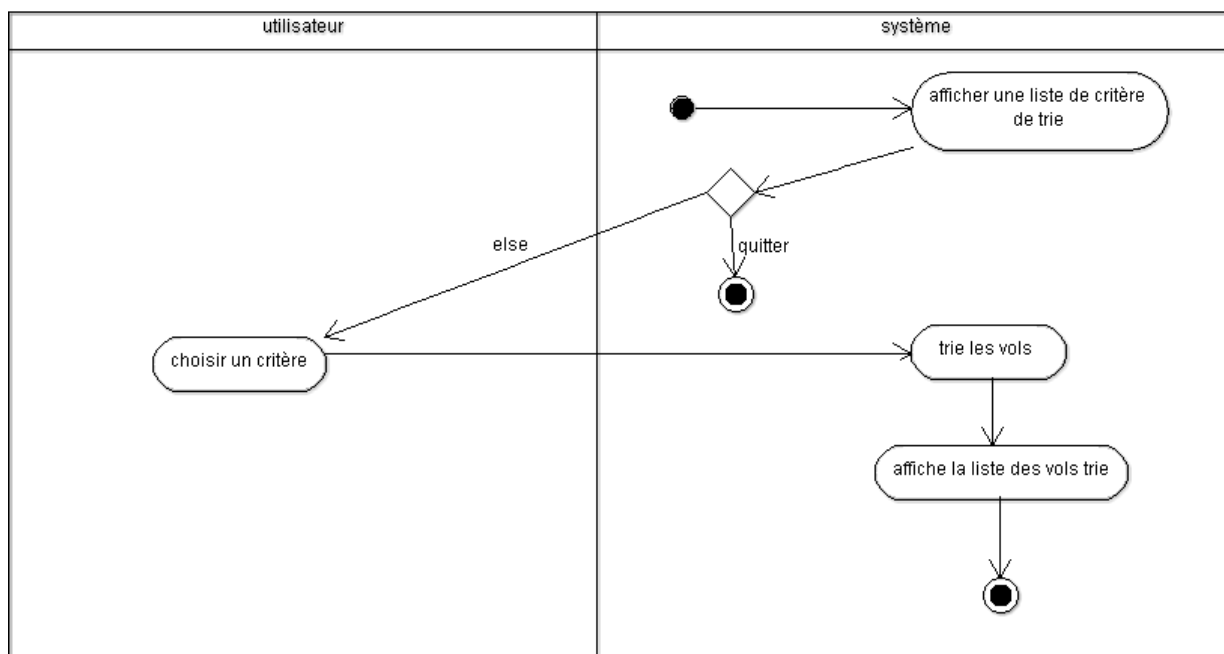




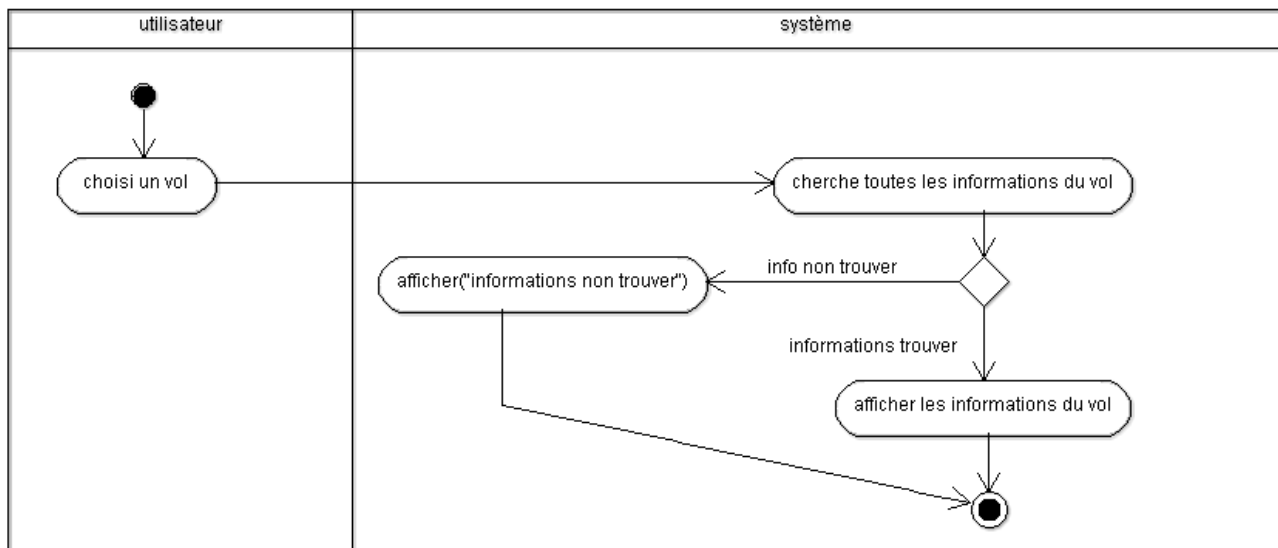
(a)      Afficher les vols d'un ensemble



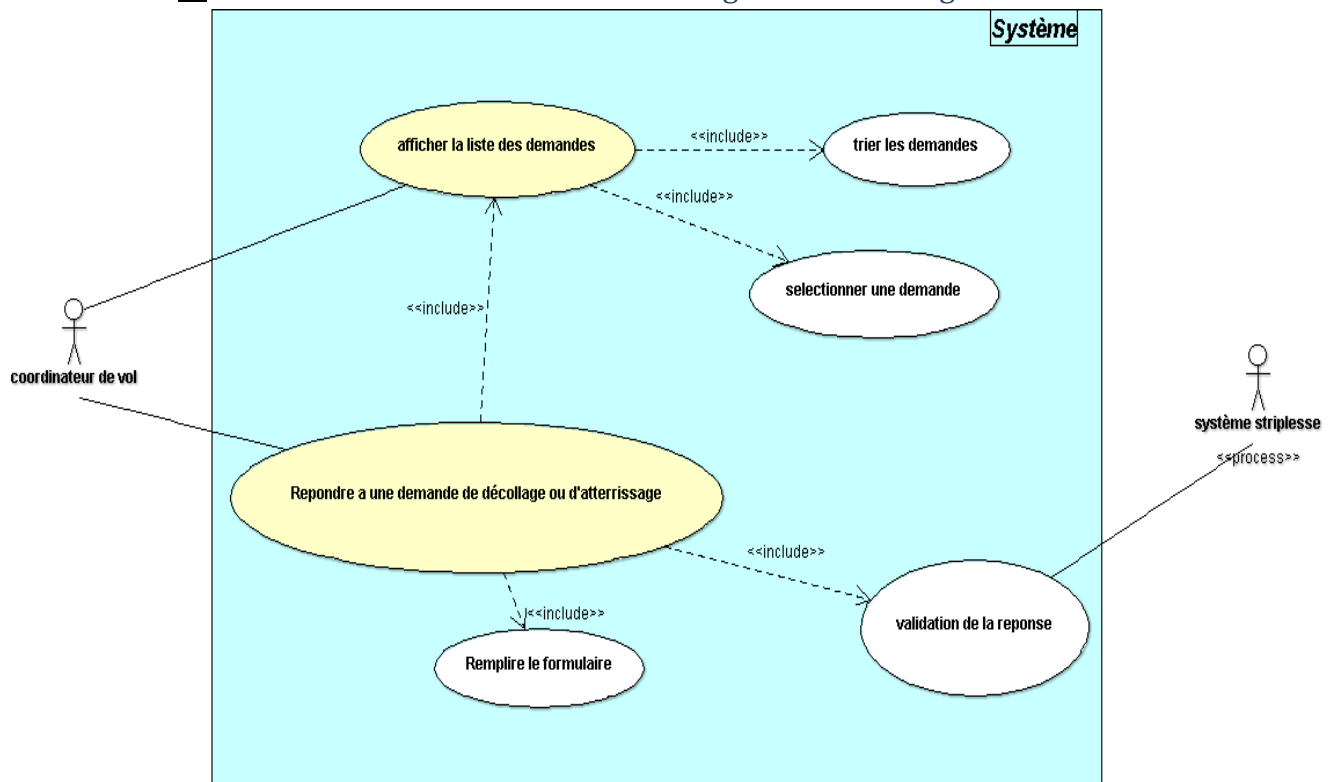
(b)      Trie la liste des vols



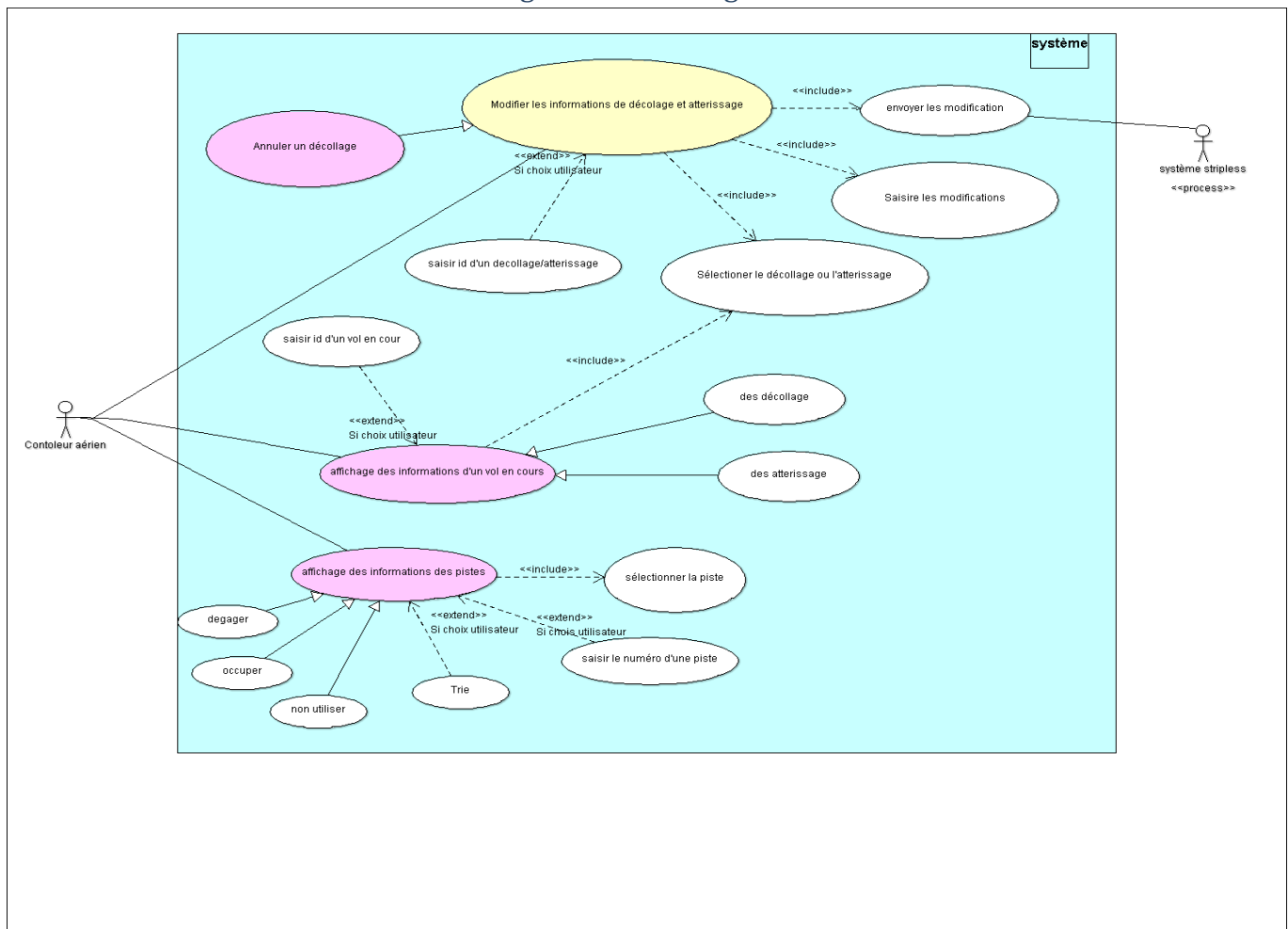
(c) Sélectionner un vol



3. Gestion des demandes de décollages et atterrissages



## 4. Gestion des décollages et atterrissages



### a) La description textuel du cas d'utilisation

#### (1) Modifier les informations du décollage ou atterrissage

Cas n°1	
<b>Nom :</b>	Modifier les informations de décollage et atterrissage (package gestion des décollages && atterrissages)
<b>Acteur :</b>	Contrôleur aérien
<b>Description :</b>	La modification doit être possible que pour les contrôleur aérien
<b>Auteur</b>	TIGHILT Massinissa
<b>Date</b>	23/04/2020
<b>Préconditions</b>	L'utilisateur doit être identifier en tant que contrôleur aérien
<b>Démarrage</b>	L'utilisateur demande la fonctionnalité « modifier un décollage ou un atterrissage »
<b>Description</b>	
<b>Le scenario nominal</b>	

1.Le système affiche une liste des décollages et atterrissages
2.Si L'utilisateur choisi d'insérer id du décollage ou d'atterrissage alors système fait appel au cas d'utilisation interne « saisir id du décollage ou atterrissage »
3.Le système fait appel au cas d'utilisation interne « sélectionner le décollage ou l'atterrissage »
4.Le système fait appel au cas d'utilisation interne « saisir les modification »
5.Le système fait appel au cas d'utilisation interne « envoyer les modification»
<b>Fin :</b>
<b>Scenario nominal :</b> sur décision de l'utilisateur après point 5
<b>Post-conditions</b>

(a) Sélectionner un décollage/atterrissage

<b>Cas n°1.1</b>	
<b>Nom :</b>	Sélectionner un décollage/atterrissage (package gestion des décollage && atterrissage
<b>Acteur :</b>	Modifier un décollage ou un atterrissage
<b>Description :</b>	
<b>Auteur</b>	TIGHILT Massinissa
<b>Date</b>	23/04/2020
<b>Préconditions</b>	
<b>Démarrage</b>	L'utilisateur veux sélectionner un décollage    atterrissage
<b><u>Description</u></b>	
<b>Le scenario nominal</b>	
1.L'utilisateur sélectionne un décollage/atterrissage	
2.Le système cherche toutes les informations du décollage/atterrissage	
3.Le système affiche toutes les informations du décollage/atterrissage	
<b>Le scénarios alternatifs</b>	
1.a L'utilisateur choisi de quitter la sélection d'un décollage/atterrissage	
1.b L'utilisateur choisi de quitter la modification des informations d'un décollage/atterrissage	

**Fin :**

**Scenario nominal :** a l'étape 3

(b) Saisir les modifications

## Cas n°1.2

<b>Nom :</b>	Saisir les modifications (package gestion des décollage && atterrissage
<b>Acteur :</b>	Modifier un décollage ou un atterrissage
<b>Description :</b>	
<b>Auteur</b>	TIGHILT Massinissa
<b>Date</b>	23/04/2020
<b>Pré_condition</b>	Toutes les informations sont affichées
<b>Démarrage</b>	L'utilisateur veut modification les informations

## Description

### Le scenario nominal

- 1.Le système donne la possibilité de modifier les informations
- 2.L'utilisateur modifie les informations du décollage/atterrissage
- 3.Le système enregistre les informations temporairement
- 4.Le système affiche un aperçu de toutes les informations

### Le scénarios alternatifs

- 2.a L'utilisateur choisi de quitter la modification d'un décollage/atterrissage
- 2.b L'utilisateur choisi de quitter la modification des informations d'un décollage/atterrissage

**Fin :**

**Scenario nominal :** a l'étape 4

**Postconditions**

**Scénario nominal** : les informations sont enregistrées dans un fichier ou un endroit temporaire

(c) Envoyer les modifications

### Cas n°1.3

<b>Nom :</b>	Envoyer les modifications (package gestion des décollage && atterrissage)
<b>Acteur :</b>	Modifier un décollage ou un atterrissage
<b>Description :</b>	Envoie des modifications au système qui s'occupe de les envoyer au commandant de bord des vols
<b>Auteur</b>	TIGHILT Massinissa
<b>Date</b>	23/04/2020
<b>Préconditions</b>	Toutes les modifications sont enregistrées
<b>Démarrage</b>	L'utilisateur demande l'envoi

### Description

#### Le scenario nominal

1. **Le système** récupère les informations dans le format idéale a l'envoi
2. **Le système** donne la possibilité d'envoyer
3. **L'utilisateur** choisi d'envoyer les informations
4. **Le système** envoie les informations au système striplesse
5. **Le système** modifie les informations dans la base de données
6. **Le système** affiche que les informations sont bien envoyées

#### Les scénarios d'exception

- 1.a les informations ne sont pas récupérer au format idéal, **le système** affiche un message « récupération des informations échouer » et fait appel au cas d'utilisations « saisir les modifications »
- 3.a **L'utilisateur** choisi de quitter, le **système** supprime les informations enregistrer temporairement (Toutes les modifications sont annulées)
- 4.a envoie des informations échouer, retourner à l'étape 2
- 5.a la modification des informations échoue, **le système** affichage un message qui indique l'erreur

#### Fin :

**Scénario nominal** : à l'étape 5

**Scénario d'exception** : après l'étape 1 si les informations ne sont pas récupérées

Après l'étape 3 si l'utilisateur choisi de quitter  
Après l'étape 5 si la modification des informations dans la base échoue

## Postconditions

**Scénario nominal :** les informations sont envoyées au système striplesse

**Scenario d'exception :** affichage des erreurs

(d) Saisir id d'un décollage/atterrissage

## Cas n°1.4

<b>Nom :</b>	Saisir id d'un décollage/atterrissage (package gestion des décollage && atterrissage)
<b>Acteur :</b>	Modifier un décollage ou un atterrissage
<b>Description :</b>	La saisi de l'id doit être déclencher par le contrôleur aérien dans un champs dédié
<b>Auteur</b>	TIGHILT Massinissa
<b>Date</b>	23/04/2020
<b>Précondition</b>	
<b>Démarrage</b>	L'utilisateur demande la saisi d'un id

## Description

### Le scenario nominal

1.Le système donne la possibilité de la saisi

2.L'utilisateur saisi les l' id

3.Le système récupère l' id

4.Le système recherche le vol (décollage ou l'atterrissage) correspond a l' id entrer

5.Le système affiche un aperçu du vol (une description)

### Les scénarios d'alternatifs

2.a L'utilisateur choisi de quitter la saisi

2.b L'utilisateur choisi de quitter la modification

### Le scénario d'exception

5.a aucun vol ne correspond a l' id, le système affiche un message «aucun vol n'est trouver»

**Fin :**

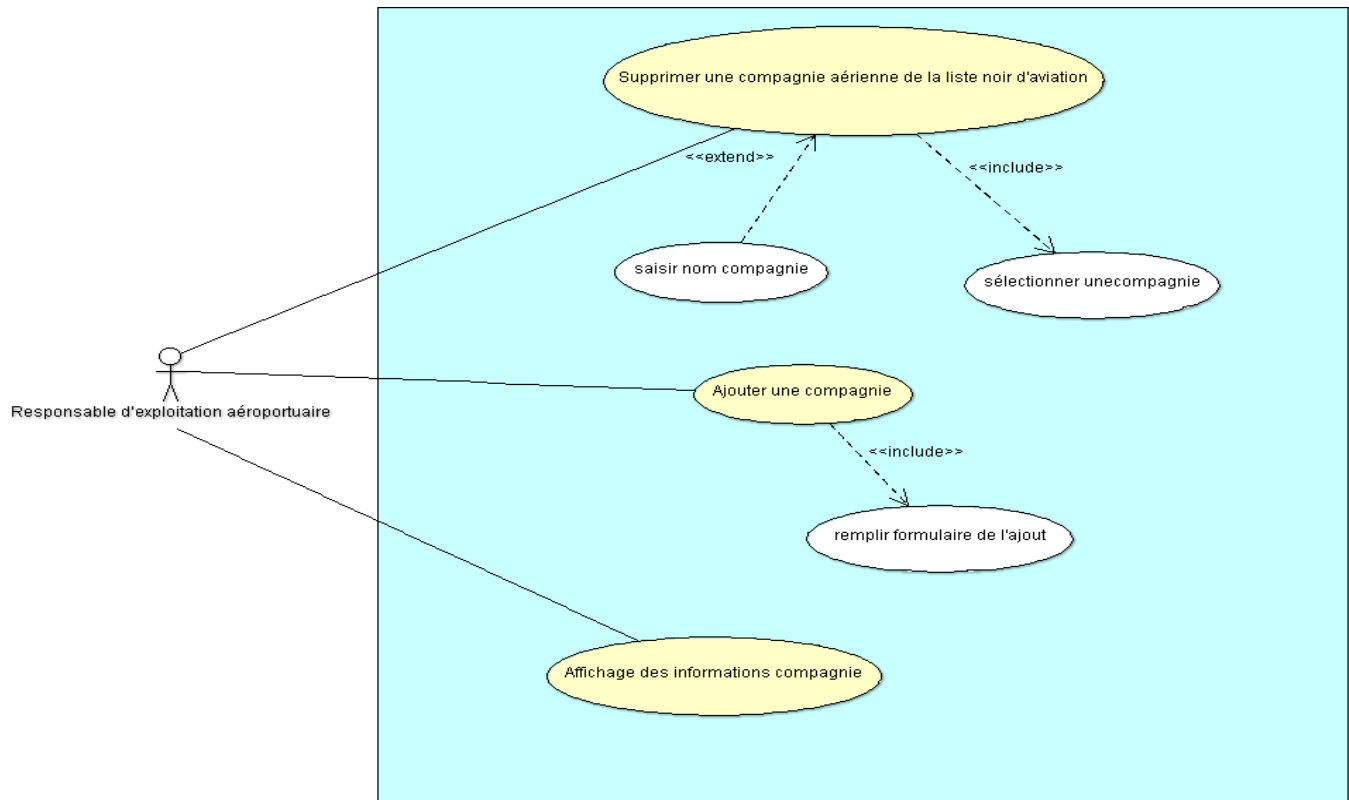


a l'étape 5

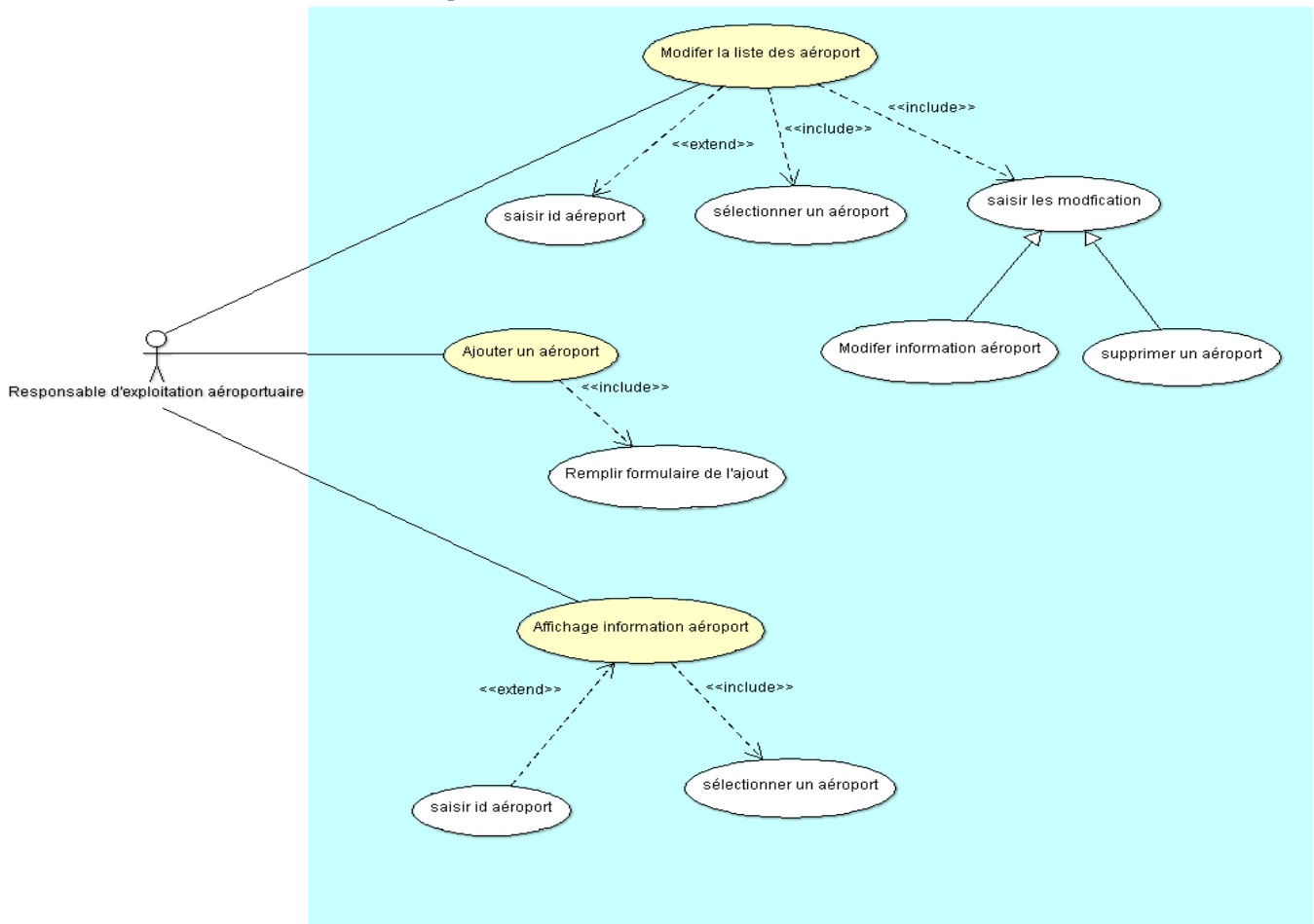
## Postconditions

### B. Service d'informations

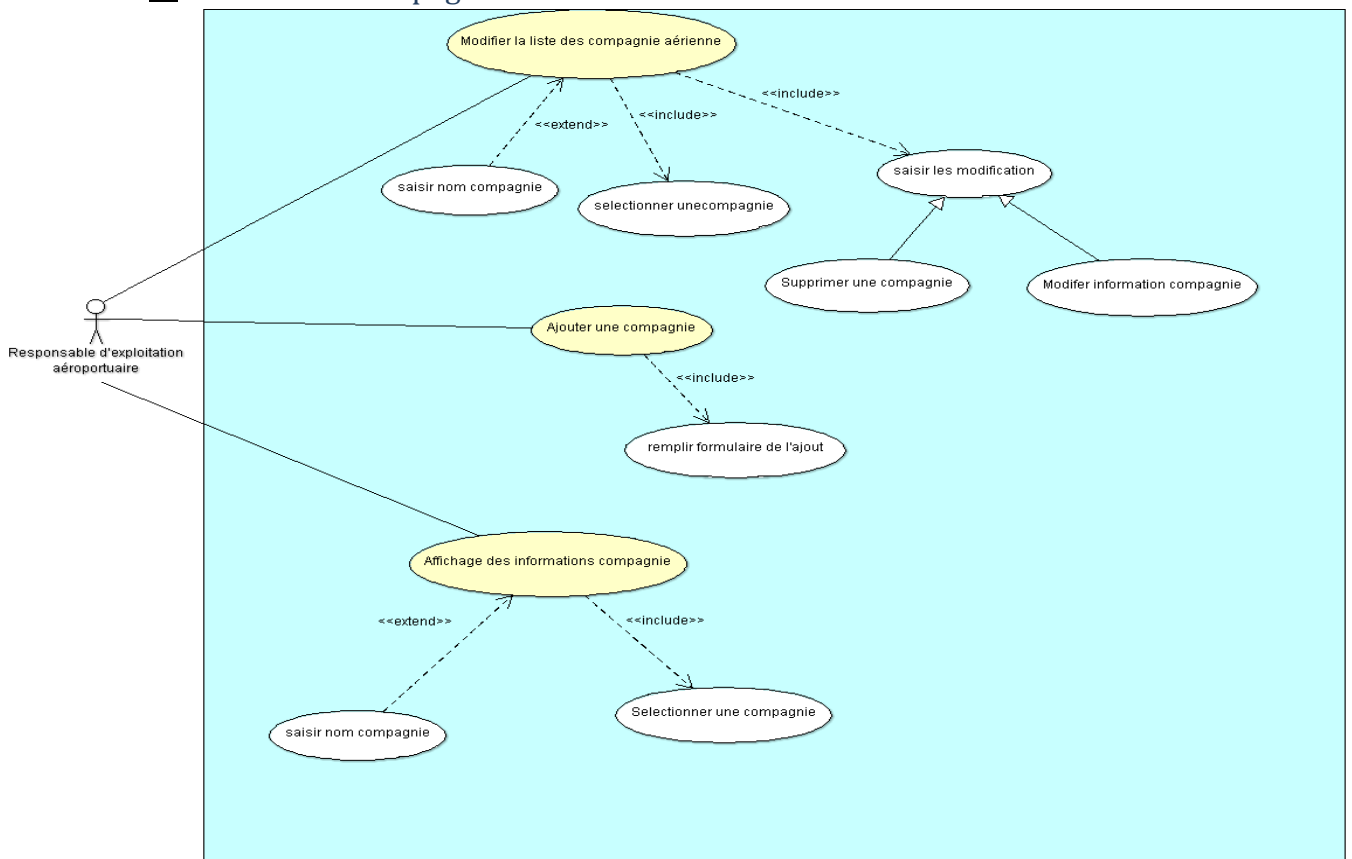
#### 1. Gestion de la liste noire



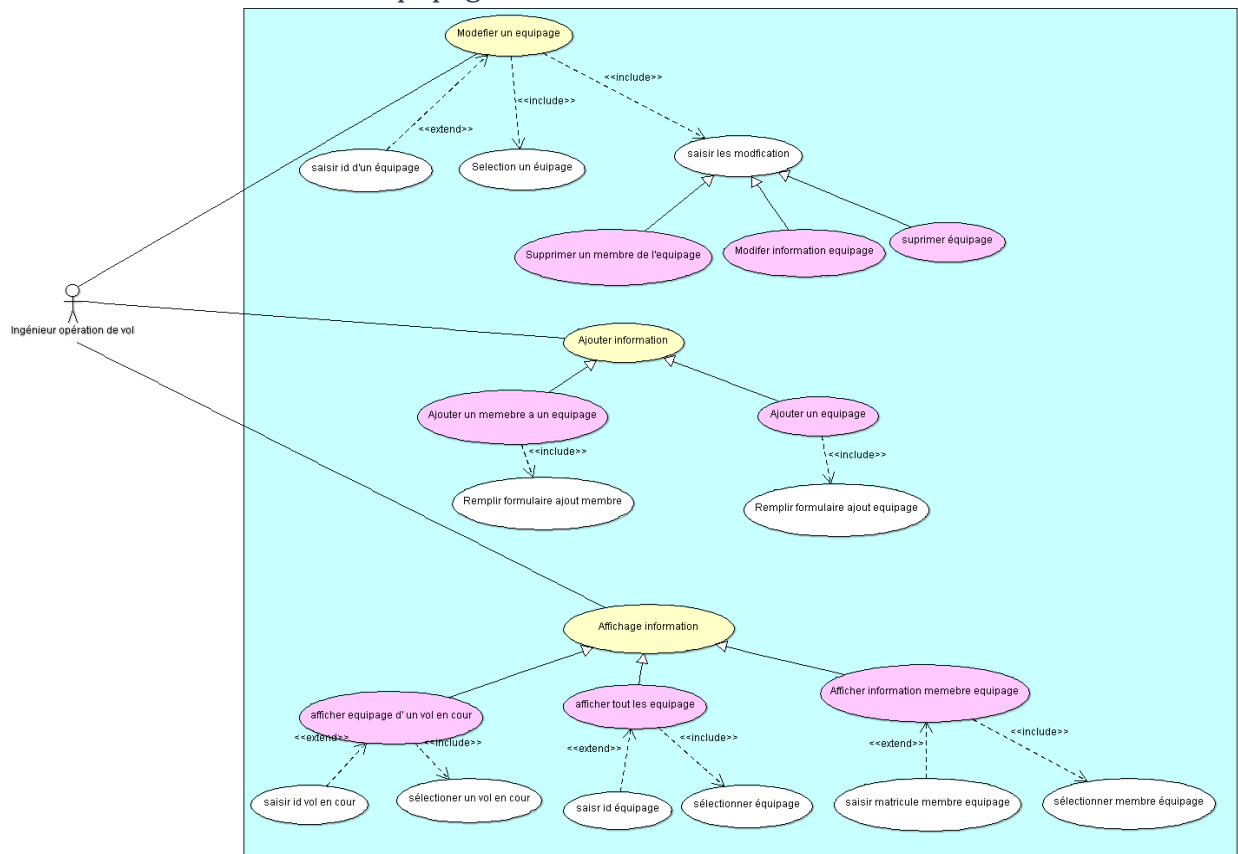
## 2. Gestion aéroports



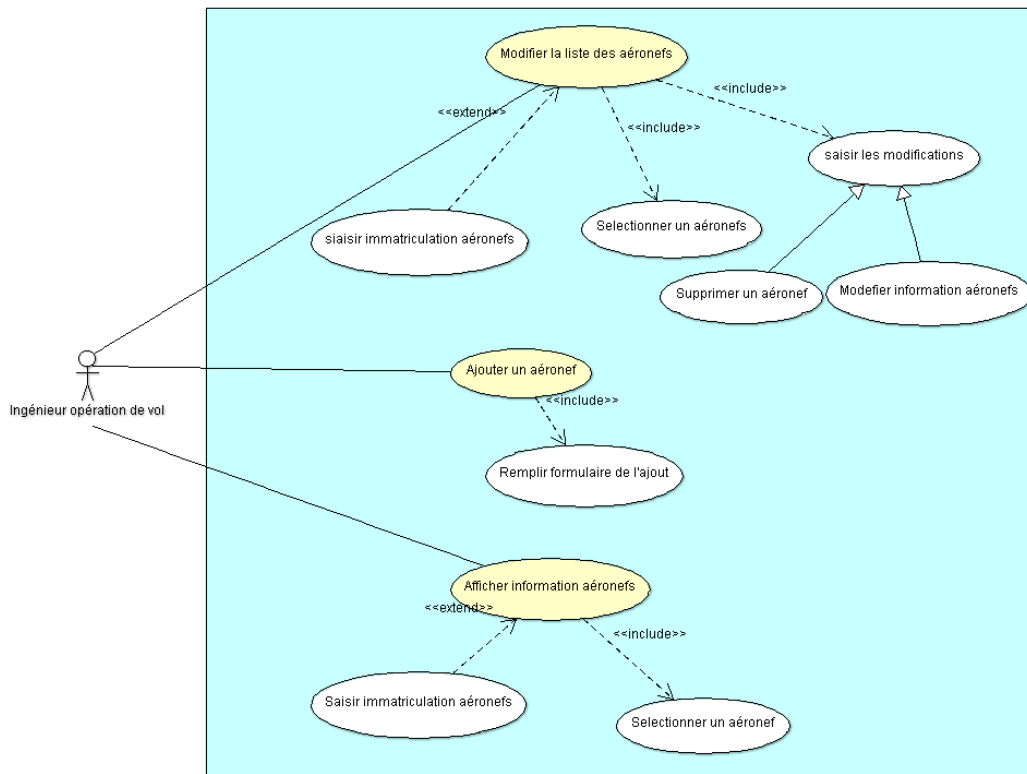
## 3. Gestion compagnie aériennes



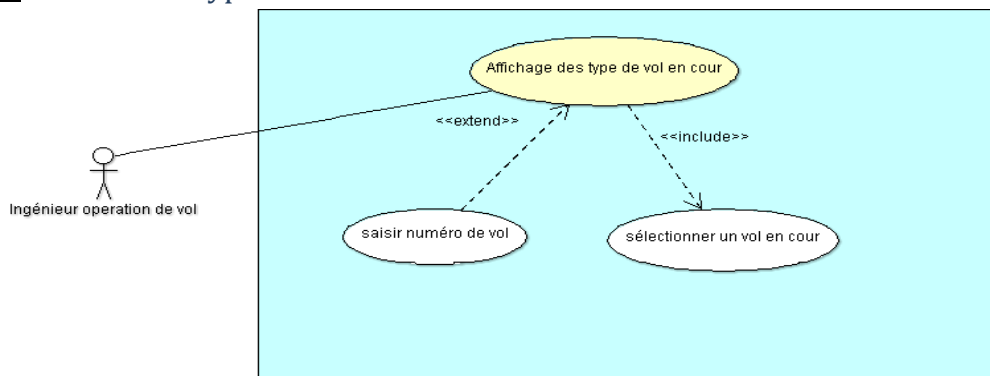
## 4. Gestion des équipages



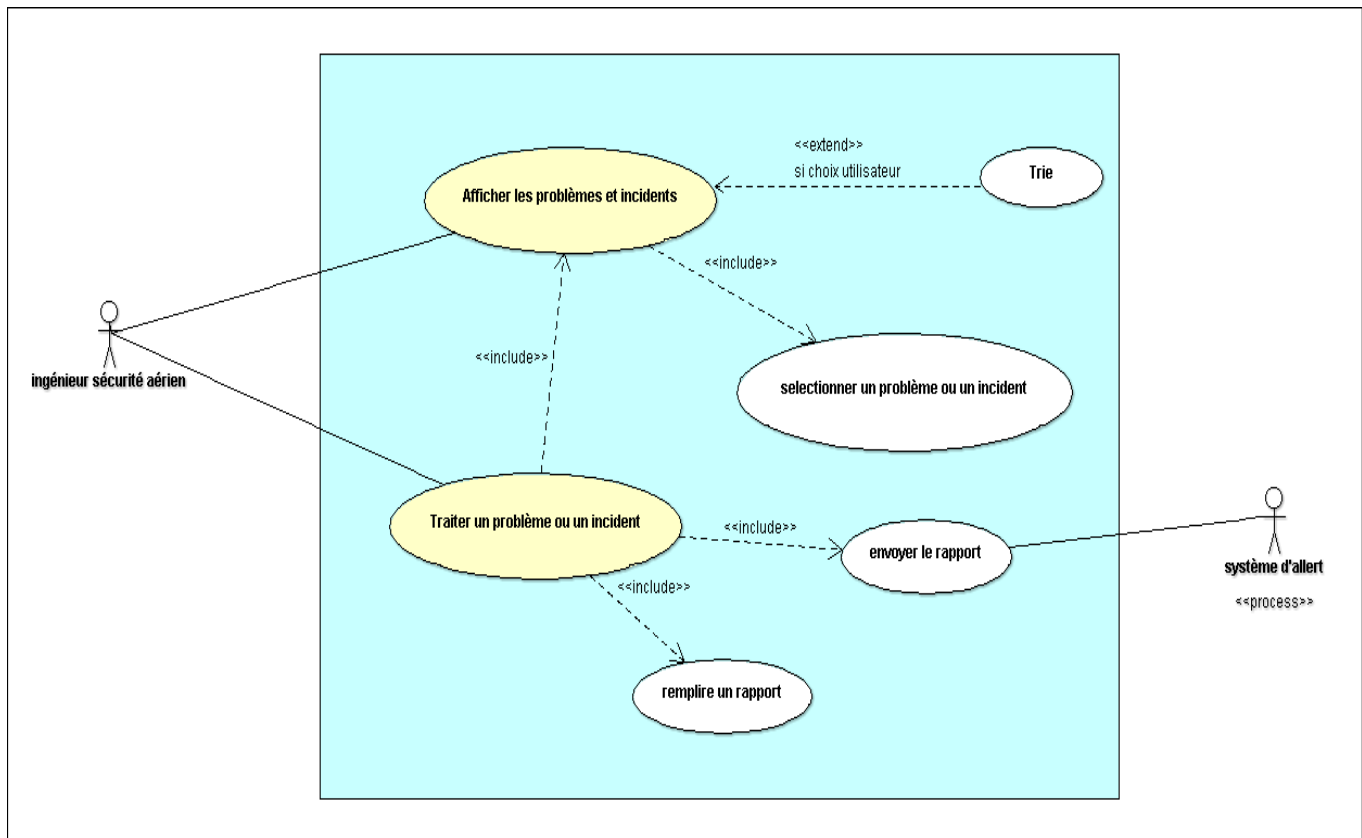
## 5. Gestion des aéronefs



## 6. Gestion type de vols

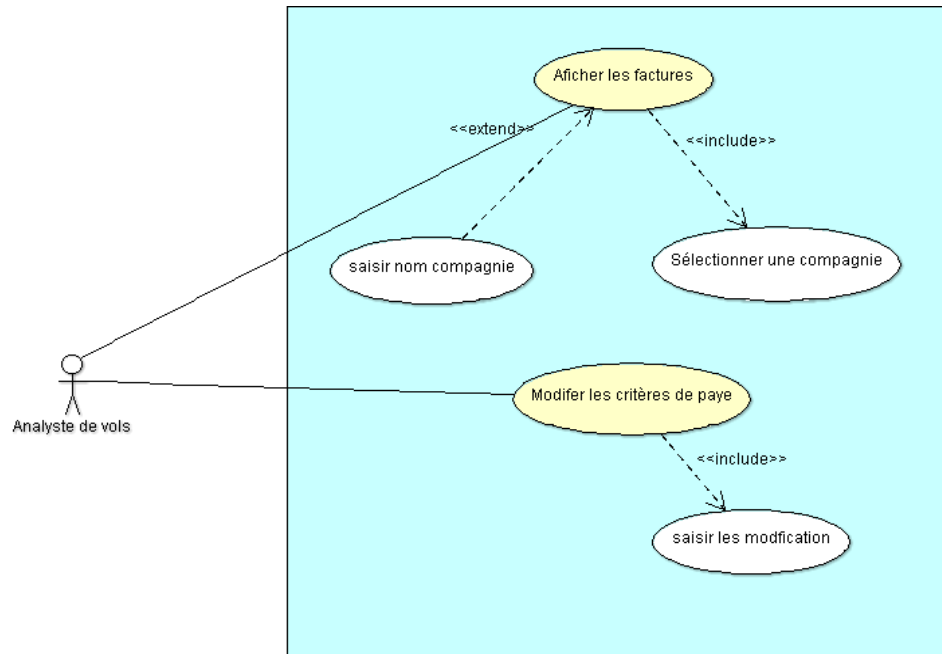


## C. Service d'alerte

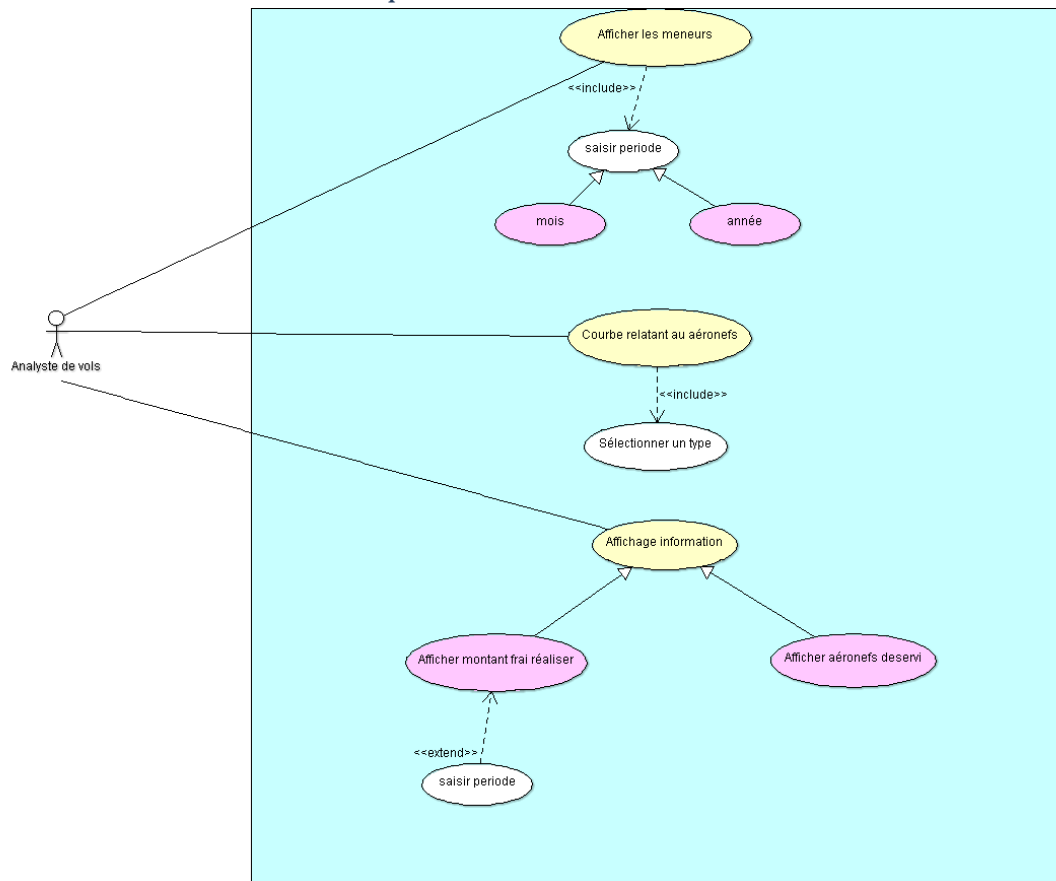


## D. Service d'administration

### 1. Gestion des redevances



## 2. Gestion des statistique

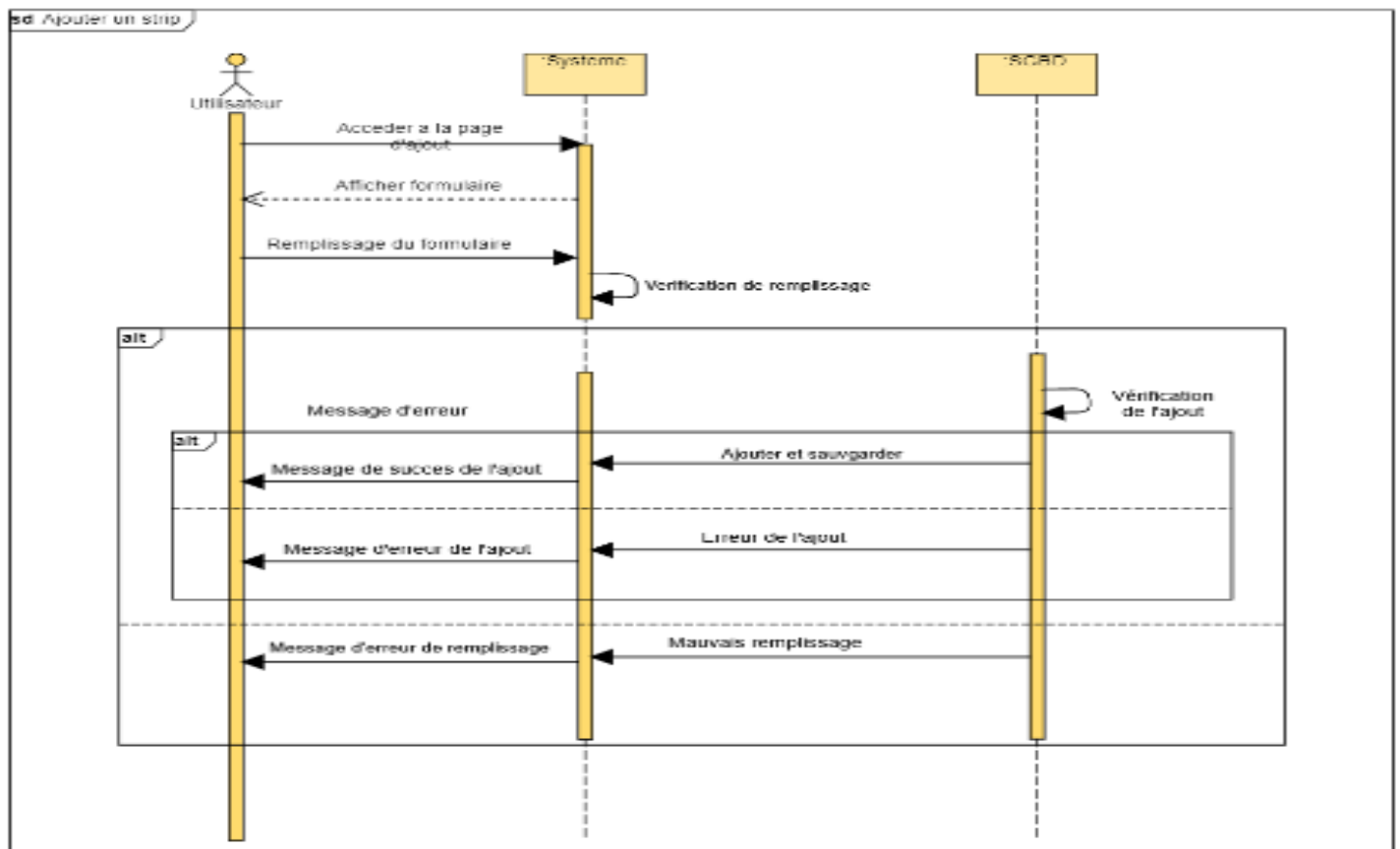


## 3. Gestion des comptes

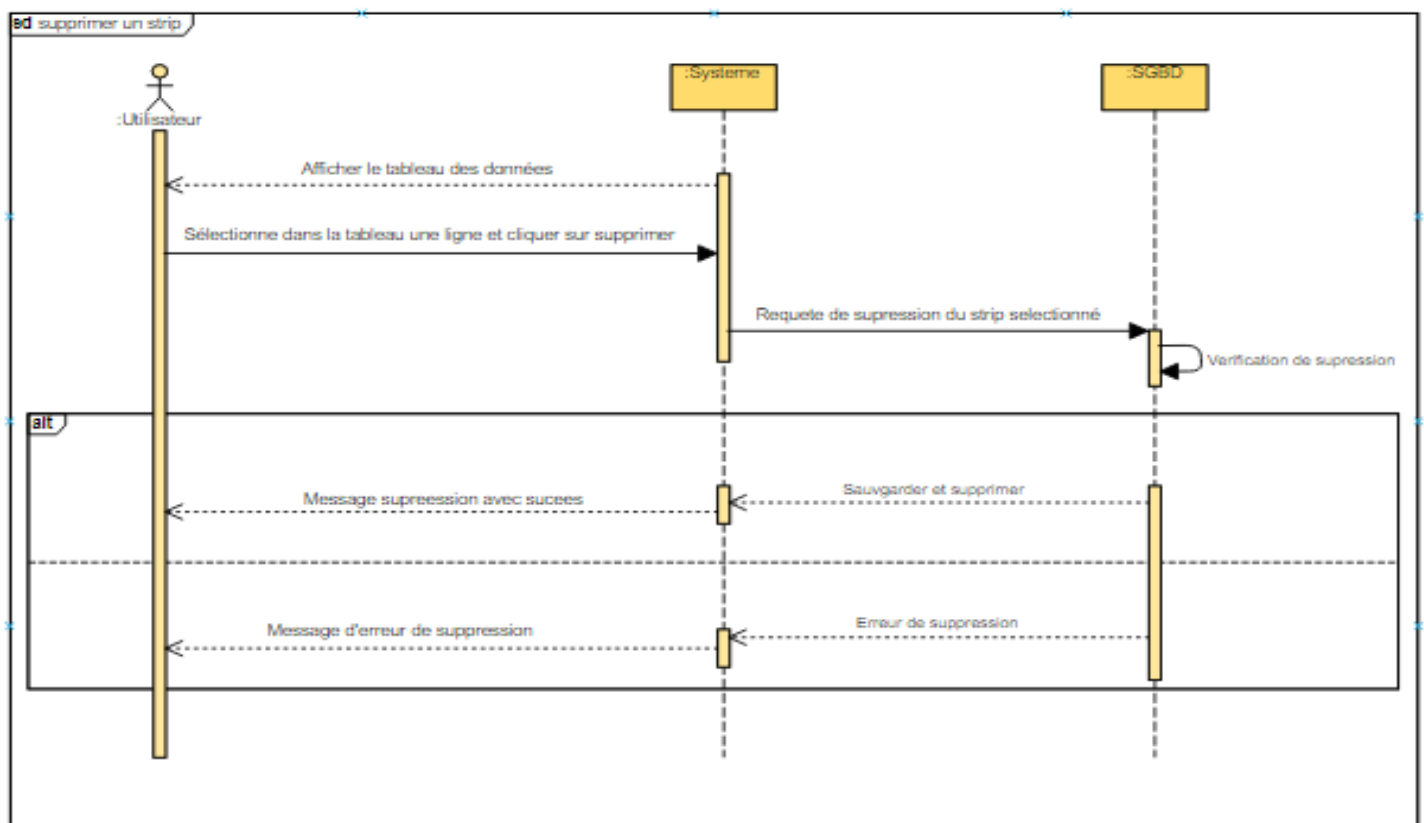
# V. Diagramme de séquence

Les Diagramme de séquences des principales fonctions du logiciel ajouter, modifier et supprimer

## A. Ajouter



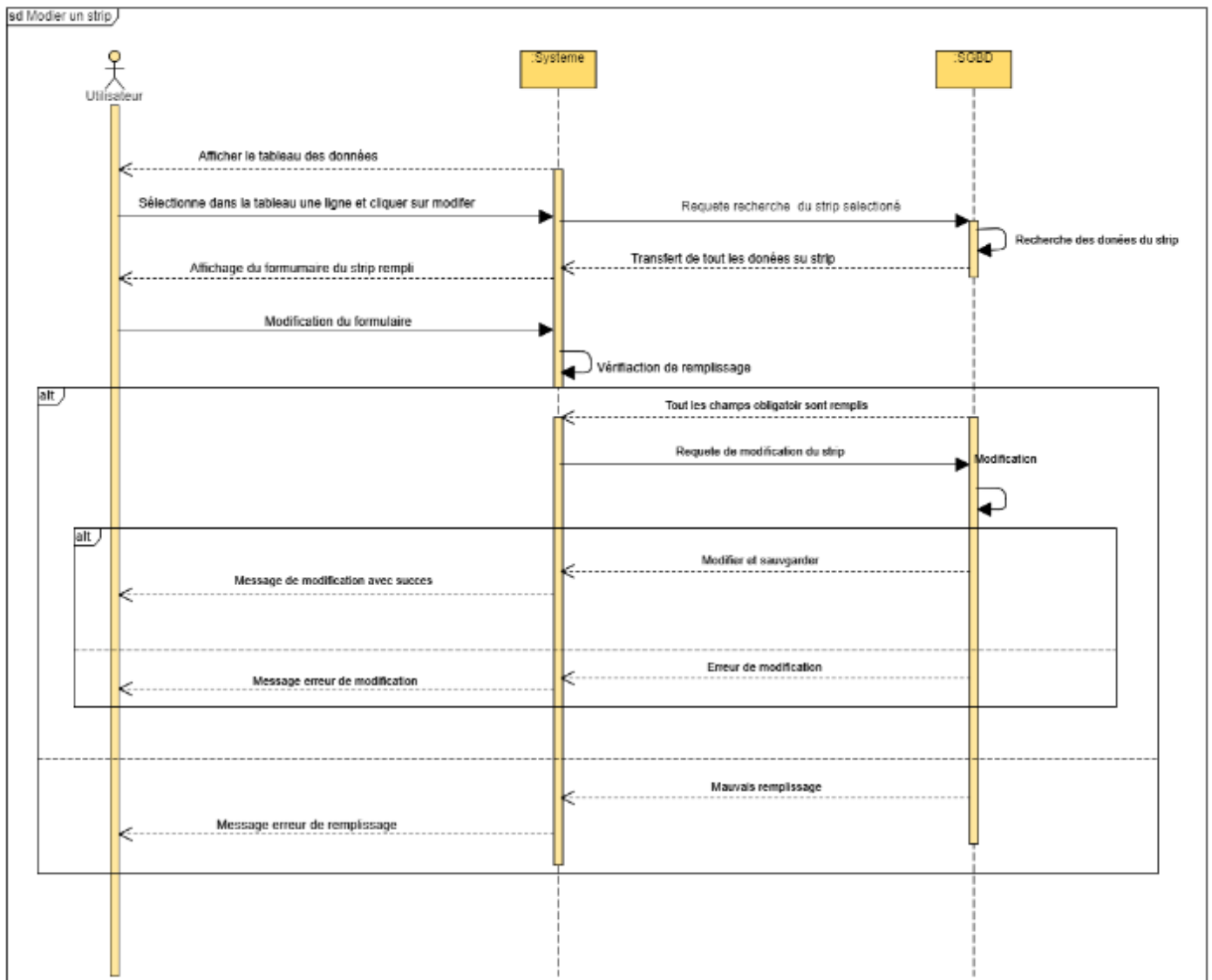
## B. Supprimer



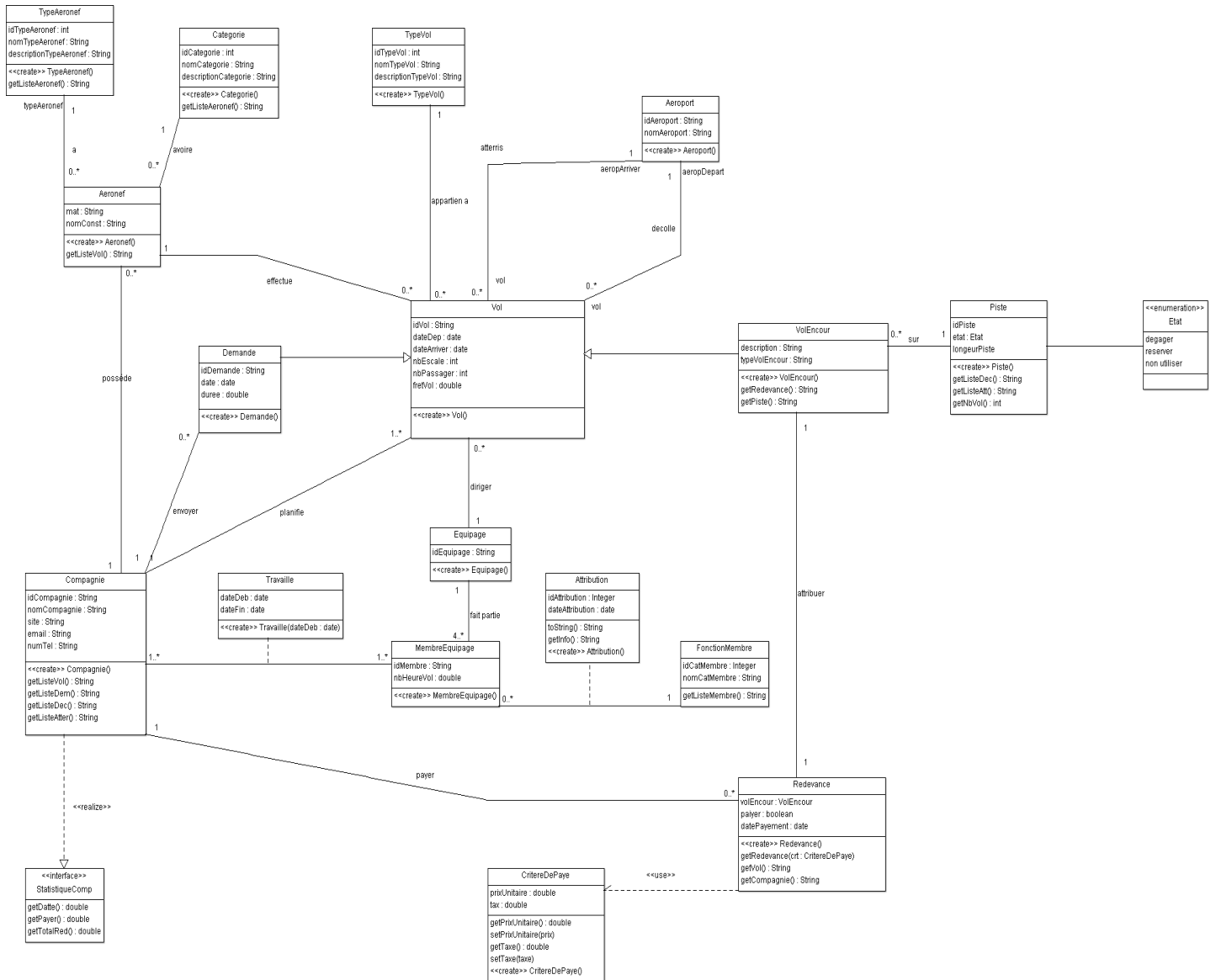


## C. Modifier

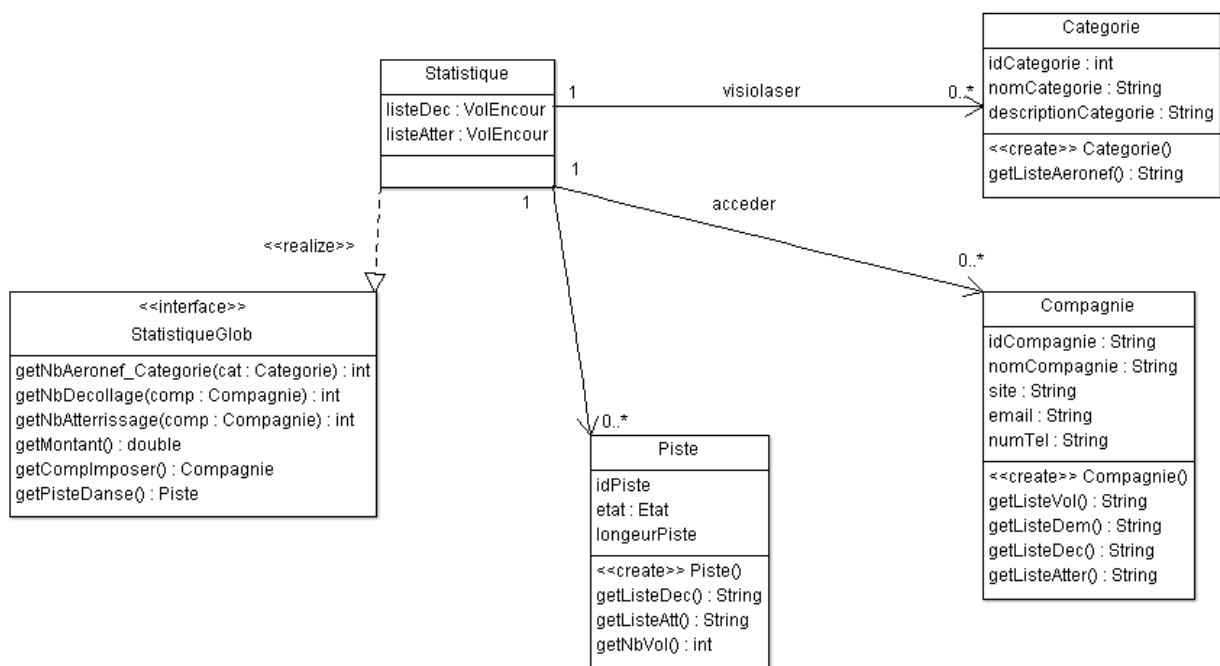
sd Modifier un strip



# VI. Diagramme de classe



## A. Statistique



### 1. Statistique

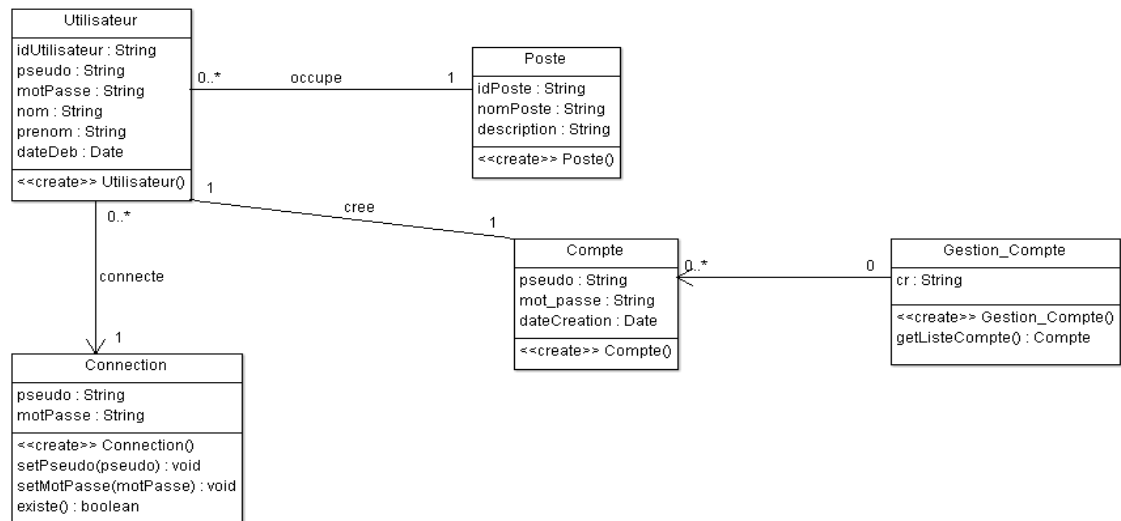
Implémente l'interface StatistiqueGlob

Les attributs	Type	Val initiale	Description
listeCategorie	Vector<Categorie>		
listeDecollage	Vector<VolEncour>		
listeAtterrissage	Vector<VolEncour>		
listeUtilisateur	Vector<Utilisateur>		La liste des utilisateurs qui peuvent avoir accès au fonctionnalité
listePiste	Vector<Piste>		La liste des piste
listePisteDanse	Piste[5]		La liste des 5 piste les plus danse
Les méthodes			Descriptions
<u>Statistique()</u>			
Les getters de chaque attribut			

### 2. L'interface StatistiqueGlob

Les méthodes	Descriptions
<u>getNbAeronef_categorie(Categorie cat) : int</u>	Calculer le nombre d'aéronef d'une catégorie
<u>getNbAtterrissage(Compagnie comp) : int</u>	Calcule liste des atterrissages d'une compagnie
<u>getNbDec(Compagnie comp) : int</u>	Calcule liste des décollages d'une compagnie
<u>getMontant() : double</u>	Le montant total des revenu
<u>getPisteDense() :Piste[5]</u>	Retourne les 5 piste les plus dense

## B. Utilisateur



### 1. Utilisateur

- La classe qui définit les utilisateurs du logiciel

Les attributs	Type	Val initiale	Description
idUtilisateur	String		
pseudo	String		
motDePasse	String		
nom	String		
prenom	String		
poste	Poste		
Les méthodes			Descriptions
<u>Utilisateur()</u>			
Les getters et setters de chaque attribut (sauf motDePasse)			
setMotsDePasse() :void			Une méthode qui permet a un utilisateur de modifier son mot de passe

## 2. Poste

Les attributs	Type	Val initiale	Description
idPoste	String		,
nomPoste	String		
description	String		
listeUtilisateur	Vector<Utilisateur>		La liste des employeur qui occupe le poste
Les méthodes			Descriptions
<u>Poste(String nomPoste)</u>			
Les getters et setters de chaque attribut			

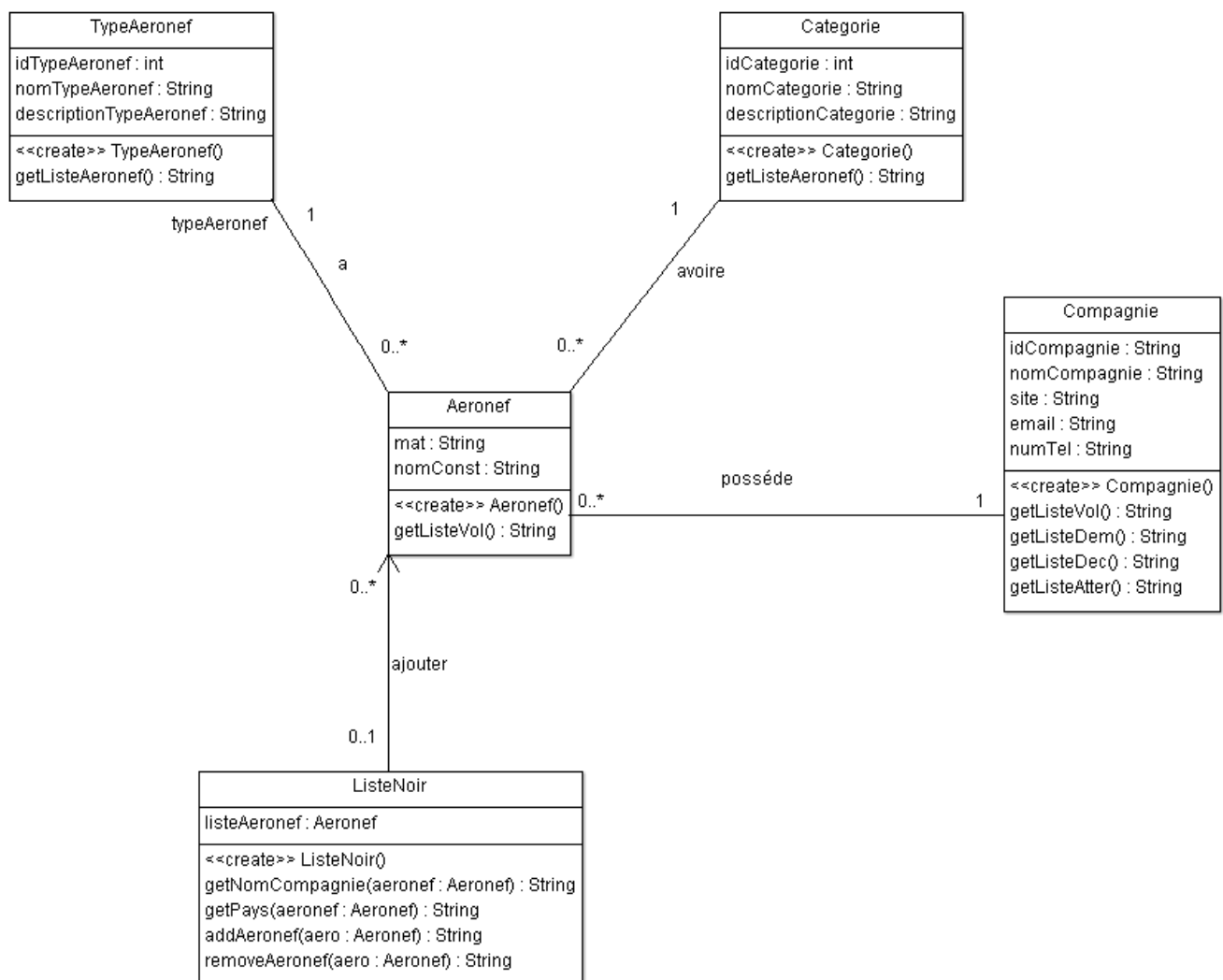
## 3. Connection

Les attributs	Type	Val initiale	Description
pseudo	String		
motPasse	String		
poste	String		
listeUtilisateur	Vector<Utilisateur>		La liste des utilisateur qui peuvent avoir accès au fonctionnalité
Les méthodes			Descriptions
<u>Connection()</u>			
setPseudo(String pseudo) : void			
setMotPasse(String motPasse) : void			
existe() : boolean			Vérifier si l'utilisateur existe

## C. Problèmes et incident

## D. Vu détailler de chaque diagramme

### 1. Aéronef



#### a) La classe aeronef

Les attributs	Type	Val initiale	Description
mat	String		
nomConst	String		
compagnier	Compagnier		
typeAeronef	TypeAeronef		
catégorie	Catégorie		
poids	double		Le poids de l'aéronef
capaciter	Int		Le nombre de place maximal
fret	double		Le poids maximum qu'il peut transporter
Listevol	ArrayList<Vol>		La liste des vols qu'il a effectuée
Les méthodes		Descriptions	
<u>Aeronef()</u>			
<u>addVol(Vol vol) : void</u>		La méthode qui permet d'ajouter un vol a la liste des vols de l'aéronef	
Les getters et setters de chaque attribut			

b) *TypeAeronef*

Les attributs	Type	Val initiale	Description
idTypeAeronef	String		Automatiquement et auto-incrémente (statique)
nomTypeAeronef	String		
description	String		
ListeAeronef	ArrayList<Aeronef>		La liste des aéronefs de ce type
Les méthodes		Descriptions	
<u>TypeAeronef(String nomTypeAeronef)</u>			
Les getters et setters de chaque attribut			

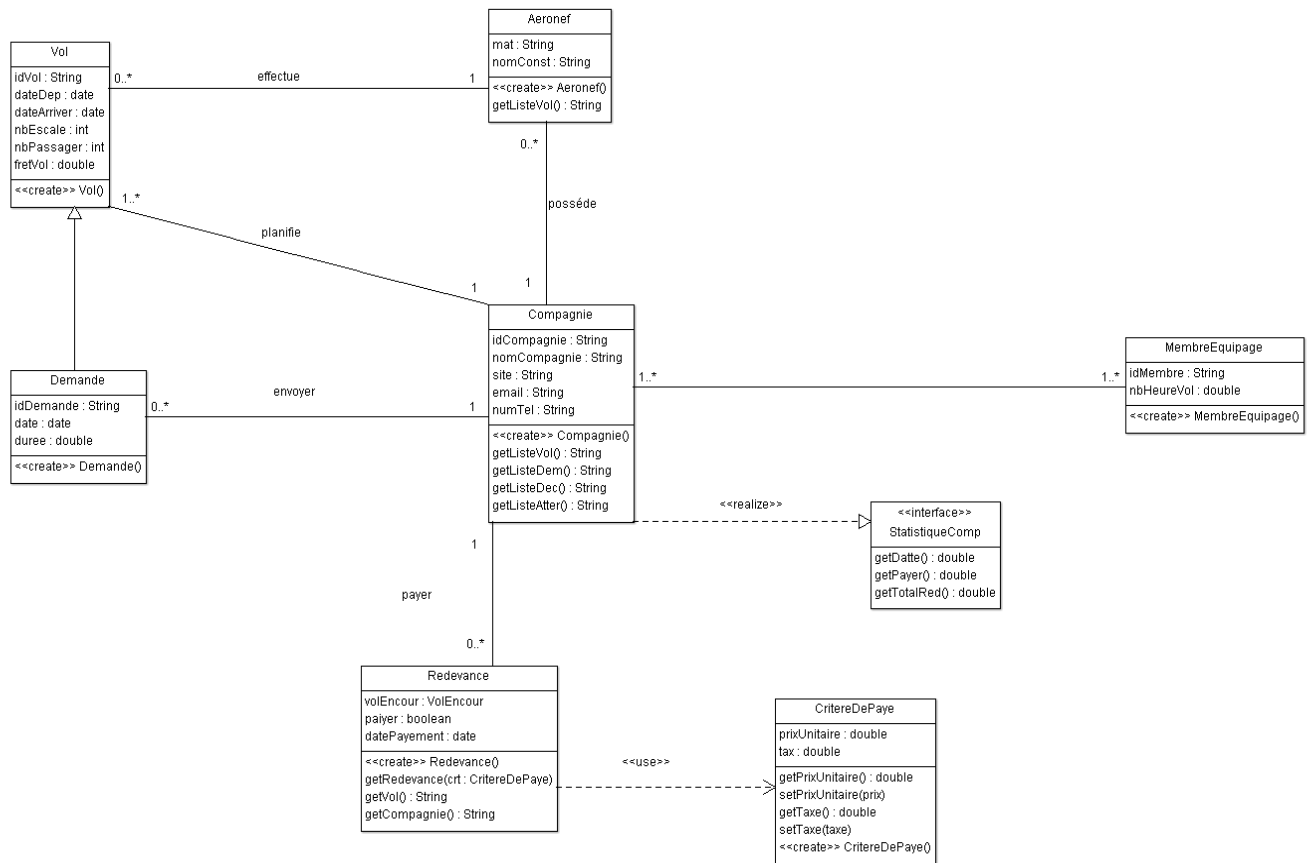
c) *Categorie*

Les attributs	Type	Val initiale	Description
idCategorie	String		Automatiquement et auto-incrémente (statique)
nomCategorie	String		
description	String		
ListeAeronef	ArrayList<Aeronef>		La liste des aéronefs de cette catégorie
Les méthodes		Descriptions	
<u>Categorie(String nomCategorie)</u>			
Les getters et setters de chaque attribut			

d) *listeNoire*

Les attributs	Type	Val initiale	Description
listeAeronef	Vector<Aeronef>		
Les méthodes			Descriptions
<u>ListeNoire()</u>			
getNomComp(Aeronef aero) : String			
getPays(Aeronef aero) : String			
addAeronef(Aeronef aero) : void			
removeAeronef(Aeronef aero) : void			

## 2. Compagnie



### a) Compagnie

Elle implémente l'interface statistiqueCompagnie

Les attributs	Type	Val initiale	Description
idCompagnie	String		L'id est codifier selon le code OACI (sur 3 lettres) -Les <b>codes OACI des compagnies aériennes</b> sont des codes à trois lettres, attribués par l' <a href="#">Organisation de l'aviation civile internationale</a> (OACI, « ICAO » en anglais) aux compagnies aériennes du monde entier. « Wikipédia »
nomCompagnie	String		
site	String		
email	String		
numTel	String[10]		Un tableau de type String de 10 champs (on peut avoir jusqu'à 10 numéros de tel)
listeAeronef	Vector<Aeronef>		Une liste chaîner des aeronef que possède la compagnie
listeVol	Vector<Vol>		La liste des vols qu'elle a planifiée (ordonner par ordre croissant de la date)

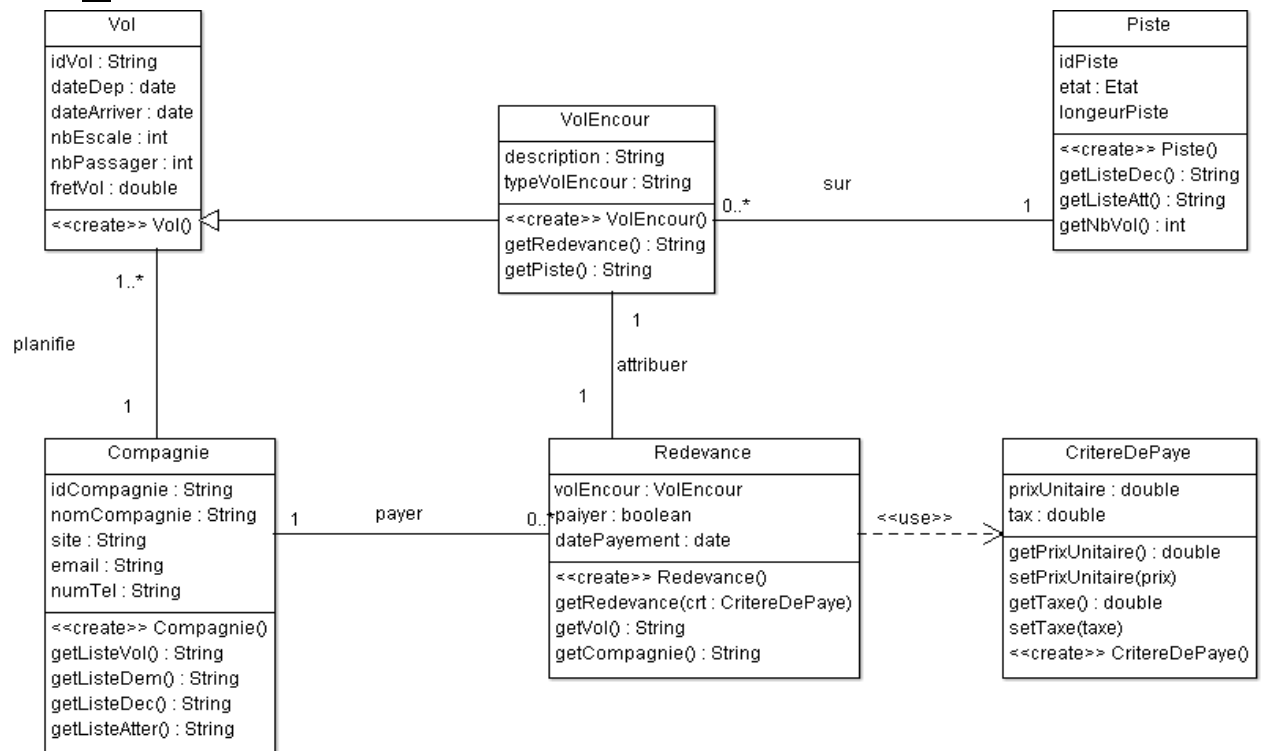


listeDemande	Vector<DemandeVol>		La liste des demandes de décollage et atterrissage qu'elle a envoyé (ordonner par ordre croissant de la date)
listeDesMembre	Vector<MembreEquip>		La liste des travailleurs (pilote, co-pilote, hôtesse, steward)
listeTravaille	Vector<Travaille>		La liste des contrat de travail qui il ont effectuer
listeRedevance	Vector<Redevance>		La liste des redevances qu'elle doit payer (trie par ordre croissant du prix)
Les méthodes			Descriptions
<u>Compagnie (String nom, String nomCompagnie)</u>			
Compagnie (String nom, String nomCompagnie, String site, String email, String [] numTel)			
Compagnie ()			
Les getters et setters de chaque attribut (sauf id pas de setter juste les getters)			
Implémenter les méthodes de l'interface statistique			
Les methde addXXX et removeXXXX des methode qui ont une cardinaliter > 1			

*b) Interface statistiqueCom*

Les méthodes	Descriptions
<u>getDette() : double</u>	La méthode qui permettra de calculer les dettes d'une compagnie
<u>getRedevanceNon() : Vector&lt;Redevance&gt;</u>	Retourner les redevances non payer
<u>getTotalRed() : double</u>	Calcule le montant total des redevance

### 3. Vol en cours



#### a) VolEncour

Les attributs	Type	Val initiale	Description
description	String		
piste	Piste		
redevance	Redevance		
typeVolEncour	String		Soit décollage ou atterissage
Les méthodes			Descriptions
VolEncour(Aeroport aeropDep, Aeroport aeropArriver, int nbPassager, double fretVol, Equipage equipage, String description)			
VolEncour()			
getRedevance() : double			Méthode qui retourne le cout du vol
Les getters et setters de chaque attribut			

#### b) Piste

Les attributs	Type	Val initiale	Description
IdPiste	Int	nbPiste	L'id de la piste
Etat	Etat	degager	
Longueur	double		

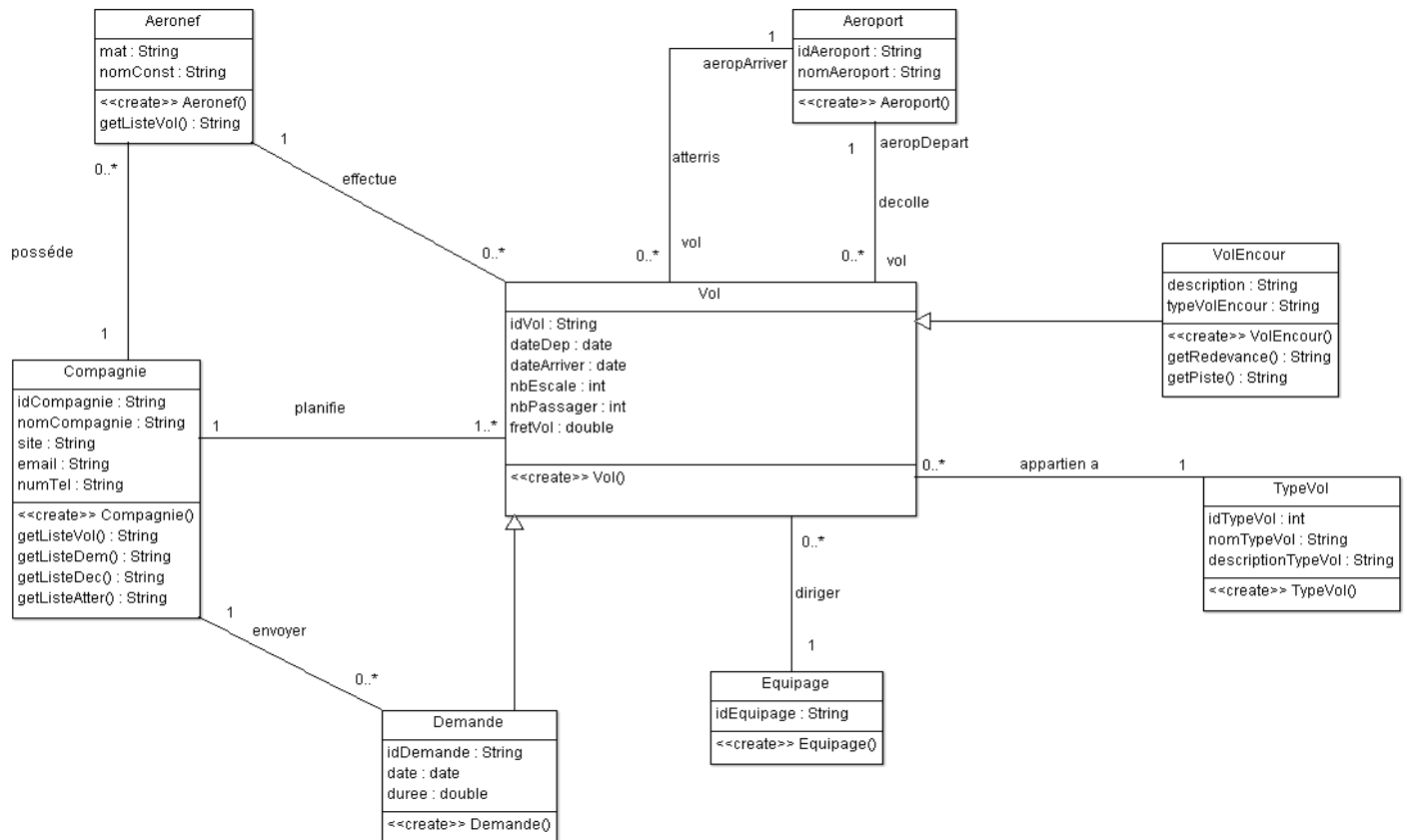
c) *Redevance*

Les attributs	Type	Val initiale	Description
vol	VolEncour		Le vol qui est facturer
prix	double	0	Le prix qui doit être payer
payer	boolean	false	Un booléen qui vérifier si la redevance est payer ou non
datePayement	LocalDate		
compagnie	Compagnier	VolEncour.Compagnie	La compagnie qui vas payer la redevance
Les méthodes			Descriptions
<u>Redevance(VolEncour vol)</u>			
<u>Redevance(VolEncour vol, boolean payer, LocalDate datePayement)</u>			
<u>Redevance()</u>			Méthode qui retourne le cout du vol
getPrix(Critere crt) : double			La méthode qui calcule le prix de la redevance et le retourne
reduction() :boolean			Une méthode qui teste si une réduction sera appliquer retourne true
gratuit() :boolean			La méthode qui teste si le vol est gratuit retourne true
Les getters et setters de chaque attribut(compagnie juste un getter)			

d) *CritereDePaye*

Les attributs	Type	Val initiale	Description
prixUnitaire	double		
taxe	double	17	
Les méthodes			Descriptions
CritereDePaye(double prixUnitaire, double taxe)			
CritereDePaye()			
Les getters et setters de chaque attribut			

## 4. Vol



### a) Vol

Les attributs	Type	Val initiale	Description
idVol	String		L'id est générer automatiquement, et qui sera codifier sur 9 positions tel que (3 pour id de la compagnie, 2 pour le jour, 2 pour le mois, 2 pour l'année, 2 un code séquentiel auto-incrémente {aura comme signification le nombre de vol de la compagnie dans tel date) La date de départ qui est prise en considération
dateDep	LocalDate	Aujourd'hui(now())	Attribut de type date qui prend comme valeur initial la date d'aujourd'hui
dateArriver	LocalDate	Aujourd'hui(now())	Attribut de type date qui prend comme valeur initial la date d'aujourd'hui
aeropDep	Aeroport		Associer à l'aéroport de départ
aeropArriver	Aeroport		Associer à l'aéroport d'arriver
nbEscale	int	0	
nbPassager	int	100	
fretVol	double		Attribut qui correspond a la charge max des bagages du vol, l'unité le tonne)
compagnie	Compagnie		Attribut qui correspond à la compagnie qui demande se vol
typeVol	TypeVol	Commercial	
aeronef	Aeronef		
equipage	Equipage		
Les méthodes		Descriptions	
<u>Vol(Aeroport aeropDep, Aeroport aeropArriver, int nbPassager, double fretVol, Equipage equipage)</u>			
Vol()			

Les getters et setters de chaque attribut (sauf idVol pas de setter juste getter)	
---	--

#### b) TypeVol

Les attributs	Type	Val initiale	Description
idTypeVol	String		Automatiquement et auto-incrémente (statique)
nomTypeVol	String		
description	String		
ListeVol	ArrayList<Aeronef>		La liste des vol de ce type
Les méthodes		Descriptions	
<u>TypeVol(String nomTypeVol)</u>			
Les getters et setters de chaque attribut			

#### c) DemandeVol

Hérite de la classe Vol

-Quand une demande est t'envoyer, quel qu'attribut du type vol peuvent être modifier automatiquement

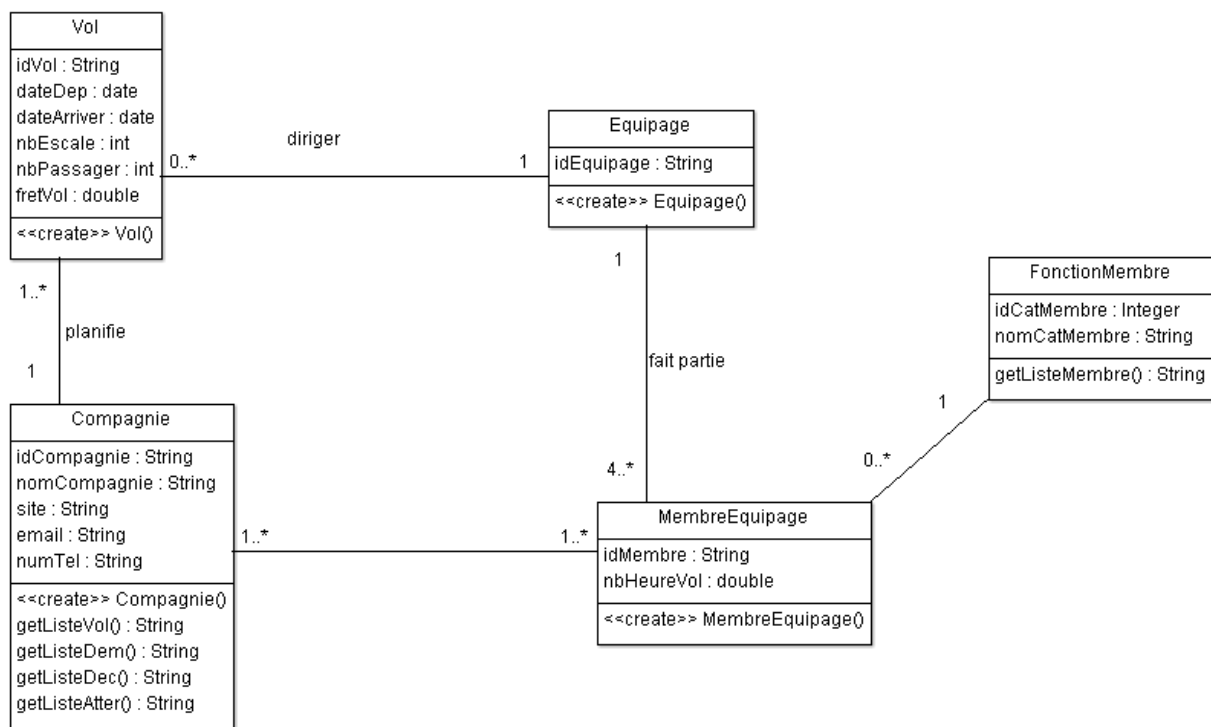
Les attributs	Type	Val initiale	Description
idDemande	String		Il est composé de l'id du vol et une position (idVol + codSed (qui s'auto incrémente si on a plusieurs demande pour un seul vol))
date	LocalDateTime	Aujourd'hui()	Ses la date d'envoi de la demande
durree	double		Attribut qui correspond à la durée du décollage ou de l'atterrissage
accorder	boolean	true	Attribut qui désigne si le décollage ou l'atterrissage est accorder
type	String		Décollage ou atterrissage
Les méthodes			Descriptions
<u>DemandeVol(Aeroport aeropDep, Aeroport aeropArriver, int nbPassager, double fretVol, Equipage equipage, double duree)</u>			
DemandeVol()			
Les getters et setters de chaque attribut			

#### d) Aeroport

Les attributs	Type	Val initiale	Description
idAeroport	String		L'id est codifier selon le code OACI (sur 4 lettres) Le <b>code OACI des aéroports</b> est un <u>code</u> de classement géographique à quatre lettres attribué à chaque <u>aéroport</u> à travers le monde par l' <u>Organisation</u>

			<a href="#">de l'aviation civile internationale</a> (OACI, soit ICAO en anglais) « Wikipédia »
nomAeroport	String		
pays	String		
ListeDep	ArrayList<Vol>		La liste des vol qui ont cet aéroport comme départ
ListArriver	ArrayList<Vol>		La liste des vols qui ont cet aéroport comme destination
<b>Les méthodes</b>		<b>Descriptions</b>	
<u>Categorie(String nomCategorie)</u>			
Les getters et setters de chaque attribut			

## 5. Equipage



### a) Equipage

Dans ce logiciel on prend en considération qu'un même équipage peut effectuer plusieurs vols

Un membre peut appartenir à plusieurs équipages

Les attributs	Type	Val initiale	Description
idEquipage	String		Automatiquement (tel il est constitué de la concaténation de l'id du pilote + id des copilotes + code séquentiels(auto-incrémente))
pilote	MembreEquip		
copilote	MembreEquip[2]		
hotesse	MembreEquip[]		La liste des hôtesse de cette équipages au moins un
steward	MembreEquip[]		La liste des stewards de cette équipage au moins un

Les méthodes	Descriptions
<u>Equippage(MembreEquip pilote, MembreEquip copilote, MembreEquip hotesse, MembreEquip steward )</u>	
Les getters et setters de chaque attribut Les méthodes add et remove des attributs qui ont une cardinalité > 1	

*b) MembreEquip*

- un membre peut avoir plusieurs fonction a un moment donner
- un membre peut travailler à une seule compagnie a un moment donner

Les attributs	Type	Val initiale	Description
idMembre	String		Automatiquement et auto-incrémente (statique)
nbHeure	double	0	Nombre d'heure de vol
fonction	FonctionMembre[]		
attribution	ArrayList<Attribution>		L'attribut qui permet d'accéder à l'historique des fonctions qu'il a occupé ou qu'il occupe
listEquip	ArrayList<Equippage>	ListeTravalle.getFirst().compagnie	La liste des équipages à qui il fait partit
compagnie			La compagnie là qu'elle il travaille actuellement
listeTravalle	ArrayList<Travailler>		La liste des compta qu'il as fais avec la compagnie ou les compagnie
Les méthodes			Descriptions
<u>MembreEquip()</u>			
Les getters et setters de chaque attribut			
Les méthodes addXXX et removeXXX des attributs qui ont une cardinalité > 1			

*c) FonctionMembre*

Les attributs	Type	Val initiale	Description
idFonMembre	String		Automatique (auto-incrémente)
nomFonMembre	String		
description	String		
listeMembre	Vector<MembreEquip>		La liste des employer qui occupe cette fonctionnalité
listeAttribution	Vector<Attribution>		La liste des attributions qui ont été faite
Les méthodes			Descriptions
<u>FonctionMembre (String nomFonction)</u>			
Les getters et setters de chaque attribut			

*d) Attribution*

Les attributs	Type	Val initiale	Description
idAttribution	String		Concaténation de la date d'attribution de la fonctionnalité, id du membre de l'équipage et id de la fonction

membre	MembreEquip		
fonction	FonctoinMembre		
date	Date		La date d'attribution de la fonctionnalité au membre
<b>Les méthodes</b>			Descriptions
<u>Attribution (MembreEquip membre, Fonction fonction, Date date)</u>			
Les getters et setters de chaque attribut			

e) *Travail*

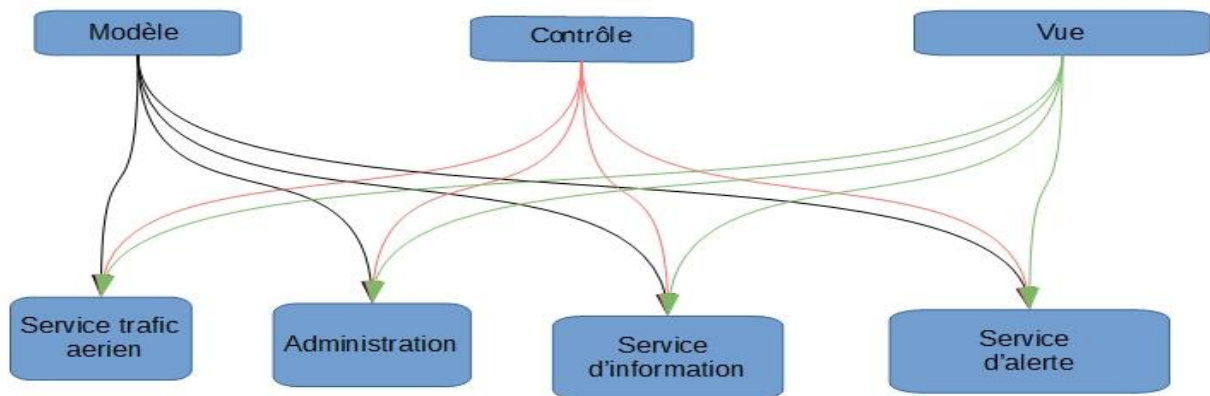
<b><u>Les attributs</u></b>	Type	Val initiale	Description
idTravail	String		Concaténation de la date de début, idCompagnie et id de l'employeur
employeur	MembreEquip		
compagnie	Compagnie		
dateDeb	Date		La date de début du travail de l'employeur dans la compagnie
dateFin	Date		La date de fin du travail de l'employeur dans la compagnie
<b>Les méthodes</b>			Descriptions
<u>Travail (Compagnie compagnie, MembreEquip employeur, Date date)</u>			
Les getters et setters de chaque attribut			



## VII. Conception architecturale

On va utiliser l'architecture MVC (model vue et contrôle), tel que dans chaque package on aura 4 package comme montre le schéma suivant

- Le model auras les différentes composant illustrer dans le diagramme de composant dans le chapitre conception détailler
- La vue aura les différentes fenêtre
- Le contrôle va avoir les contrôles des interactions des utilisateur tel que clic sur un bouton, fermer la fenêtre, entrer des donner .....



## VIII. Conception détailler

