Rapport d'analyse - Projet Java 2019

Application de réservation de billets d'avions

Charles d'Andigné, Benoit Guillot, Florentin Brisebard

Présentation:

Notre projet vise à proposer une application de réservation de billets d'avion destinée à être utilisée par une compagnie aérienne (la compagnie Célibat'Air) pour gérer les réservations des clients, calculer les coûts et rentabiliser les vols
avec un bon rapport avion/passager. C'est à dire qu'un vol devra associer un avion adéquat au nombre de passagers.
Notre compagnie aérienne aura une flotte de 250 avions de 20 modèles différents ainsi que 500 équipages, composés de 2 pilotes et 10 hotesses, le tout desservant 40 villes européennes soit 50 aéroports.
Aucun avion n'est affecté à une liaison particulière et tous peuvent effectuer presque tous les trajets. Chaque avion possède une capacité maximale de passagers appartenant à une des 7 catégories de places maximales (700 places / 400 / 250 / 150 / 80 / 15 / 2), et présente une autonomie maximale (exprimée en km).

Nous attribuerons également des coûts par voyageur pour la compagnie à chaque vol : Le coup du vol dépendra de l'avion utlisé, du nombre de voyageur et de la distance effectuée.
Le prix du billet initial (réservation 1) dépend de la distance effectuée, de l'avion affrété et des bagages emportés, il augmentera ensuite de (n + 0,5) euros pour les n réservations suivantes.

Nous allons donc implémenter nos classes, énumérations avec nos divers attributs et méthodes (qui seront définis dans un diagramme de classe suivant) afin de concevoir cette application, de la rendre opérationnelle et simple d'utilisation, avec des codes détaillés et des explications dans un rapport final qui suivra celui ci.

Notre application va ainsi permettre à la compagnie aérienne

- de localiser ses avions et ses équipages;
- de trouver un trajet reliant deux aéroports;
- d'ajouter/supprimer des voyageurs sur un vol;
- de fournir un récapitulatif d'un voyage;
- de choisir l'avion le plus économique pour un vol donné (avec le meilleur rapport passager/avion)

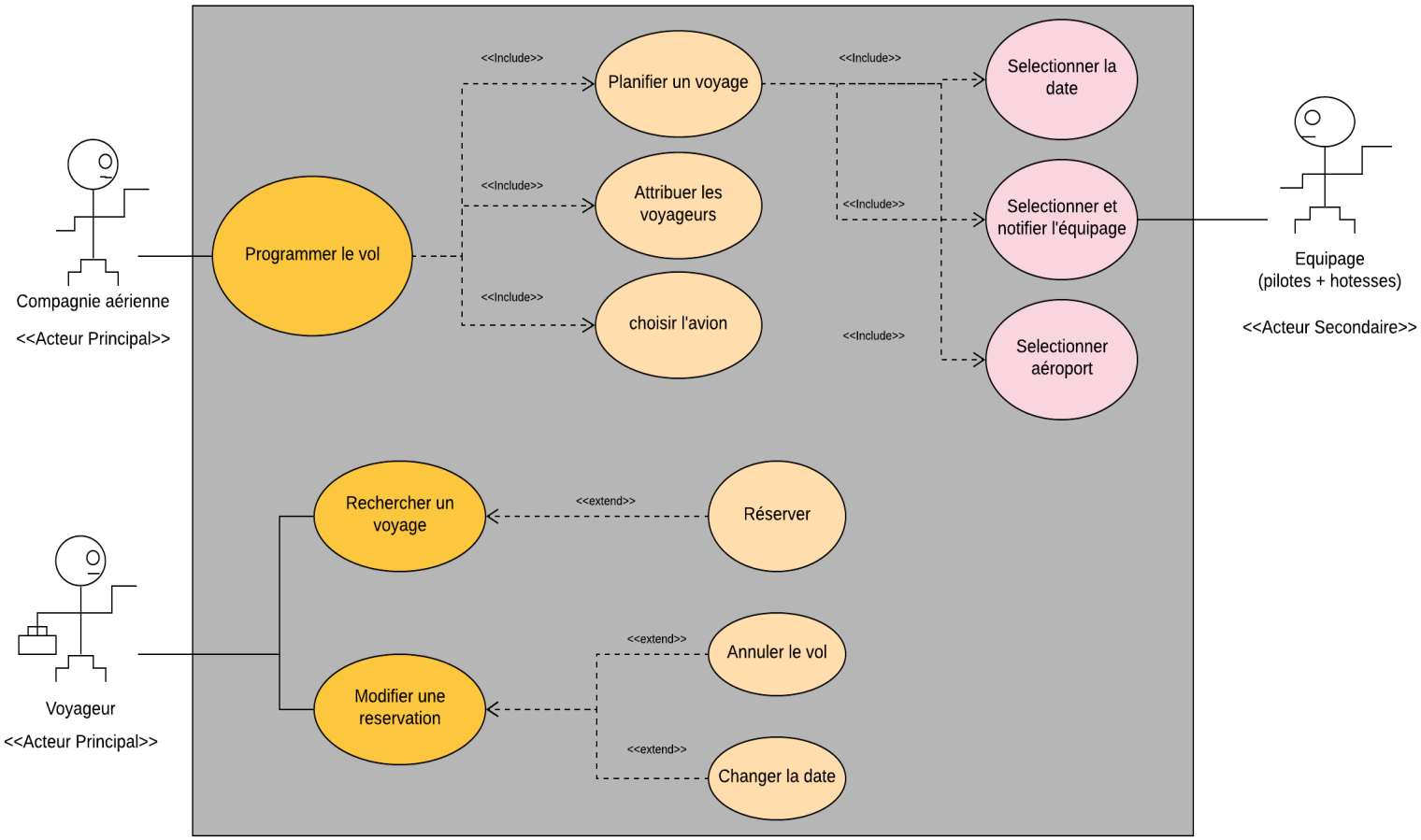
Notre application va aussi permettre aux voyageurs :

- de réserver un vol;
- de le modifier/supprimer. La modification entrainant une majoration du coup.

Plan de lecture du rapport d'analyse:

Les possibilités, capacités et utilisations de notre application sont détaillés dans le diagramme d'utilisation ci dessous. Ensuite, nous expliciterons comment fonctionne concrètement notre application au regard de quelques exemples d'utilisation sur notre diagramme de séquence. Et enfin, nous détaillerons notre plan de programmation via le diagramme de classe, avant de conclure ce rapport d'analyse.

Diagramme d'utilisation



Commentaire du diagramme d'utilisation:

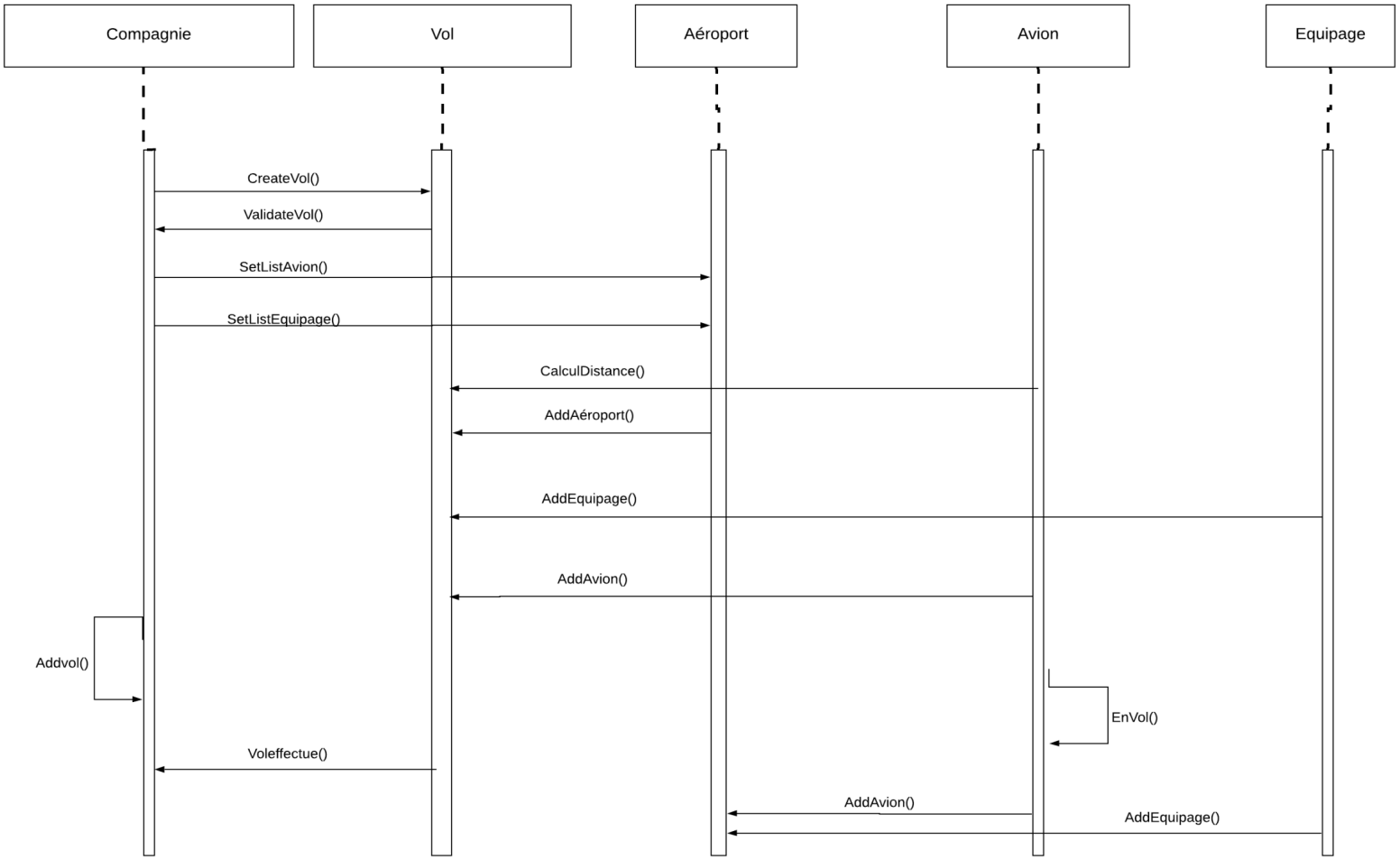
Notre diagramme d'utilisation permet de voir quelles actions peuvent réaliser les différents acteurs de notre application via celle ci.

Ainsi, nos acteurs principaux vont:

- Pour la compagnie aérienne: Programmer un vol, donc planifier un voyage, y attribuer des voyageurs et choisir un avion en fonction du nombre de voyageurs. Plus le nombre de voyageur augmente, plus l'avion prévue aura une capacité de transport importante (sous réserve du nombre d'avions disponibles sur l'aéroport de départ).
  Pour programmer un nouveau vol, donc avant de le proposer à tous les voyageurs et de les y attribuer si réservation, la compagnie va sélectionner la date (en fonction de la demande d'un premier passager), sélectionner un aéroport et sélectionner un équipage en fonction de sa localisation.
- Pour le voyageur, le premier va influer sur la création d'un vol ou non par la compagnie, puis tous les voyageurs pourront rechercher un voyage, le réserver ou bien modifier ou annuler leur réservation.

L'equipage n'a pas accés à l'application. Les conernant, seule la compagnie va utiliser l'application pour les localiser et les affreter à un vol. Il en est de même pour les avions

Diagramme de séquence

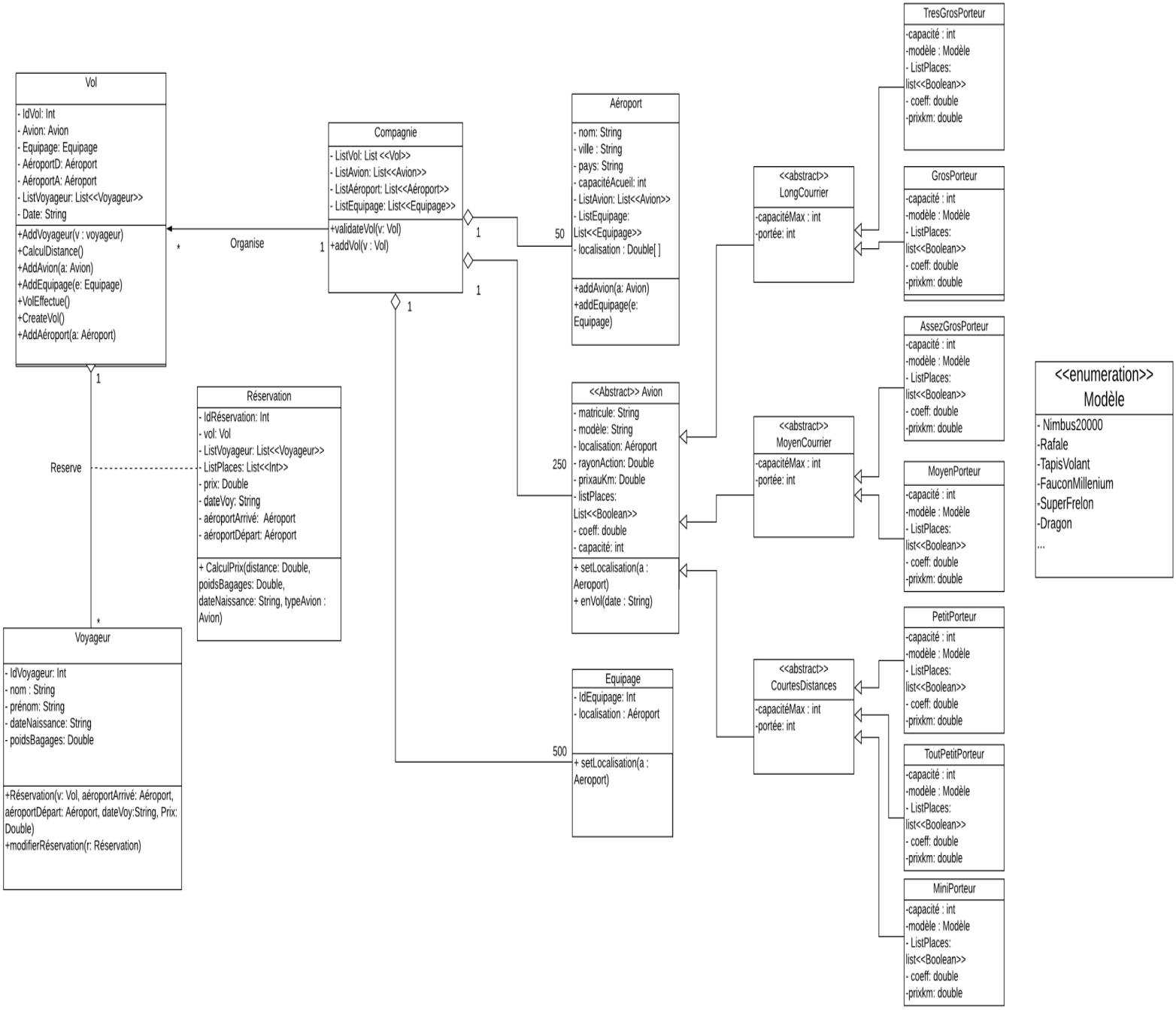


Commentaire du diagramme de séquence:

Notre diagramme de séquence représente ici les actions effectuées par la compagnie aérienne sur l'application afin de programmer un vol, de sa création à son accomplissement. Un diagramme concernant les actions du voyageur de sa recherche du vol à la réalisation de son trajet sera fait ulterieurement pour le rapport final.

Ainsi:
La compagnie appelle la méthode CreateVol() de la classe Vol pour l'initialiser. Un retour de la classe Vol sur la classe compagnie via la méthode ValidateVol() valide cette création.
Puis, avant de pouvoir ajouter notre vol à la réservation, la compagnie regarde la liste des avions et la liste des équipages via les méthodes SetListAvion() et SetListEquipage() afin de localiser lesquels sont présents sur l'aéroport.
Après, la distance de vol est calculée et on peut alors décider de l'avion et de l'equipage qui effectuera le vol en les ajoutant au vol par les méthodes AddEquipage() et AddAvion() de la classe Vol.
Enfin, le vol est ajouté sur la liste de la compagnie par la méthode Addvol() de cette classe, l'avion est considéré Envol() donc plus disponible à la reservation pour ce jour ci, et le vol est effectué. Les équipages et les avions sont ensuite mis à jour sur les listes des aéroports via les méthodes AddAvion() et AddEquipage() de la classe Aéroport afin d'etre disponible sur cet aéroport pour un prochain vol.

Diagramme de Classe



Commentaire du diagramme de classe:

Notre diagramme de classe a pour principal objectif la planification de tout notre code et de l'implementation de nos classes, interfaces, méthodes...etc. Cela permettant de nous organiser pour établir tout cela, de bien nous aligner sur l'utilisation des attributs et des méthodes, et de suivre notre avancé tout en conservant une bonne comprehension du code de notre application et des différentes relations intervenant dans les programmes.
De plus, ce diagramme a pour vocation l'explication générale de notre application, et plus particulièrement sa programmation.

Ainsi, nous codons 17 classes dont 4 abstraites, et une énumération contenant les 20 modèles d'avions utilisés par notre compagnie aérienne. Les 17 classes comportent: - Une classe Compagnie regroupant les données des avions, des vols, des aéroports et des équipages dans des listes, reponsable du bon déroulement et fonctionnement de ceux ci. - Une classe vol dirigeant le vol de sa

création jusqu'a son déroulement. - Une classe Voyageur avec les informations du voyageur et les méthodes permettant d'agir sur la réservation (En créer ou en modifier). - Une classe Réservation permettant la programmation des réservations , de calculer le prix de celle ci, et de faire les liens avec le voyageur, le vol ou la compagnie. - Une classe aéroport en lien avec la compagnie qui permet de mettre à jour les équipages et avions présents sur un aéroport. - Les classes abstraites Avion, LongCourier, MoyenCourier et CourtesDistances pour définir un avion et son rayon d'action, puis les 7 classes porteurs pour définir la capacité d'emport de passagers de chaque modèle d'avion. Ainsi chaque modèle d'avion de l'énumération sera associé à une capacité d'emport (soit 700, soit 15 ou soit 2 passagers..etc), à un rayon d'action (gros courier par exemple) puis défini en temps qu'avion, mis à jour dans un aéroport et/ou dans la compagnie...etc

Voilà pour notre diagramme de Classe.

Organisation du travail:

Désormais, il faut être bien précis sur les taches de chacun pour être efficace et productif.
Ainsi, Florentin et Charles débutent le code principal de l'application qui est la tache la plus exigeante et délicate tandis que Benoit se charge du rapport, de l'interface graphique de l'application et donc de la convergence des programmes pour une execution commune. Nous sommes tous intervenus sur la conception des diagrammes et nous avons tous un regard bienveillant sur le travail de nos camarades afin que le projet soit notre oeuvre global.

La suite dans le prochain rapport.