Rapport de conception du projet de programmation orientée objet

Licence d'informatique – 2ème année
Faculté des sciences et techniques de Nantes

Gestion de Transports Aérien de Passagers

présenté par
ASHOK Thanish, BTEICH Elio, GAUTIER Mélody, LAZRI Lynda et LE GALLIC Malo. Groupe 385K
le 08/11/2022

Lien Projet Gitlab

`https://gitlab.univ-nantes.fr/E19C167P/POO22_385K_A`

encadré par
DUFOUR Richard et GALLINA Ygor

Nantes Université

1 Cahier des charges

Notre projet de POO se place dans le contexte des transports aériens de passagers. C'est-à-dire que des clients vont pouvoir utiliser des vols pour aller d'une ville de départ à une ville d'arrivée. D'un autre côté, nous avons étendu le contexte des transports aériens pour ne pas le limiter aux clients mais pour avoir aussi le point de vue de l'équipage. Ainsi que de l'aéroport avec les avions disponibles, et de la compagnie aérienne pour les différents vols mis à disposition.

Les objectifs de l'application et de l'interface graphique seront, d'une part, de permettre au client de sélectionner un vol en rentrant dans une barre de recherches les lieux d'arrivée et de départ pour les vols correspondants ; puis, de faire son enregistrement en montrant aux clients les places disponibles dans l'avion au moment de la réservation du siège. D'autre part, de permettre à la compagnie aérienne de pouvoir afficher et créer des vols. Nous avons également trouvé intéressant que le pilote puisse voir son historique de vols passés, avec un espace dédié à l'équipage.

Pour faire cela, le projet aura plusieurs fonctionnalités applicatives. La principale, sera la simulation d'une réservation de vol par un ou des client(s). Elle comprendra des fonctionnalités au moment de la réservation, telles que : le choix de la date, le choix de la classe qui fera varier le prix du vol, la possibilité d'enregistrer un ou plusieurs bagages de tailles différentes, etc. Une fois la réservation faite, le passager pourra alors soit annuler son vol, faire une nouvelle réservation ou effectuer son enregistrement. Une fois la réservation confirmée, le passager aura la possibilité de choisir son siège dans l'avion parmi ceux disponibles. De plus, l'application fournira la possibilité à l'administrateur de créer des vols (ici, on se placera donc du point de vue de la compagnie aérienne), en rentrant les attributs spécifiques au vol. Enfin on retrouvera des fonctionnalités dédiées aux pilotes d'avions et membres de l'équipage comme la consultation du nombre de vols effectués.

L'application va se positionner dans plusieurs cas d'utilisations. Ainsi, pour gérer les différents types d'utilisateurs de l'application (passager, administrateur, équipage), nous utiliserons un système d'authentification qui permettra d'afficher différentes options sur une même interface en fonction du statut de l'utilisateur. Le premier cas de figure étant dans le cadre d'une réservation de vol jusqu'à l'enregistrement du siège par un passager. La deuxième situation concernera l'utilisation de l'application par la compagnie aérienne qui va gérer les vols en les créant et en vérifiant lesquels sont complets. Enfin, la troisième possibilité intéressera le pilote et son équipage avec la consultation de l'historique de vols effectués ainsi que ceux à venir.

2 Architecture

2.1 Description générale

Nous allons ici développer et décrire nos classes pour notre projet.

- Company : elle représente une compagnie aérienne qui va avoir ses propres clients, son équipage de vol, ses avions, ses vols et ses classes de vol ;
- Person : super-classe qui définit une personne et des informations importantes comme un numéro de téléphone ou une adresse mail. On retrouvera dans cette classe la méthode getTotalFlightDuration(), qui permettra de retrouver le temps de vol des Clients pour d'éventuelles réductions, ou des Crew Members à titre indicatif ;

— Crew Member : sous-classe qui hérite de Person, donne quelques attributs supplémentaires pour définir un membre de l'équipage comme la description précise de leur métier à bord et leur date d'embauche ;
— Client : sous-classe qui hérite de Person, donne quelques attributs et surtout plusieurs méthodes pour définir un client de la compagnie aérienne. On retrouvera dans Client notamment les méthodes liées aux réservations comme reserveFlight() ainsi que d'autres méthodes pour éventuellement modifier ou annuler une réservation effectuée.
— Luggage : super-classe abstraite qui définit tous les attributs d'un bagage ainsi que les signatures des sous-classes sans les implémenter ;
— HandLuggage : sous-classe qui hérite de Luggage, représente les bagages cabine en implémentant les méthodes de Luggage ;
— CheckedLuggage : sous-classe qui hérite de Luggage, représente les sacs et valises en soute et implémentera les méthodes de Luggage.
— FlightClass : classe qui définit les différentes classes des sièges dans le vol (1ère, business, éco...) ;
— Seat : classe qui définit un siège d'un avion par son numéro et sa FlightClass. Cette classe aura notamment une méthode isAvailable() pour savoir si le siège est libre pour un vol passé en paramètre ;
— Airplane : classe qui définit les propriétés d'un avion comme son nom ou son nombre de sièges ;
— Airport : classe qui définit un aéroport avec son nom et son code d'immatriculation. Il y aura également les méthodes getFutureArrivals() et getFutureDepartures() qui donneront la liste des vols sur le point d'atterrir ou de décoller ;
— Address : classe qui définit l'adresse d'un aéroport, contient la ville, le code postal et le pays. On sépare cette classe d'Airport car il est possible d'avoir deux Airport pour la même Address.
— Flight : classe qui réunit toutes les informations pour un vol (ex : date du départ, les Address de départ et d'arrivée, prix de base du vol, l'Airplane utilisé). Pour les méthodes, on aura getAvailableSeats() qui rendra la liste des sièges encore disponibles pour le vol, getDuration() pour connaître la durée du vol, addCrewMember() pour définir tout le personnel d'équipage pour ce vol, en vérifiant qu'ils ne soient pas déjà occupés ;
— Reservation : classe qui réunit tous les éléments nécessaires à une réservation (le Client qui l'effectue, son Luggage, son Flight réservé). On aura les méthodes calculateTicketPrice() pour connaître le prix du billet en fonction du prix de base du vol et de la classe de siège réservée et addLuggage() pour définir lors de la réservation les bagages prévus pour le vol ;
— Check-in : classe qui lie Reservation avec un Seat quand le Client fera son check-in avant le vol.

2.2 Diagramme de classes

La classe main est composée de plusieurs CrewMember, plusieurs Address, plusieurs FlightClass, plusieurs Airport, plusieurs Airplane qui seront déjà présents et qui pourront varier dans le futur. Elle sera composée aussi de plusieurs Client et plusieurs Flight qui seront ajoutés au fur et à mesure.

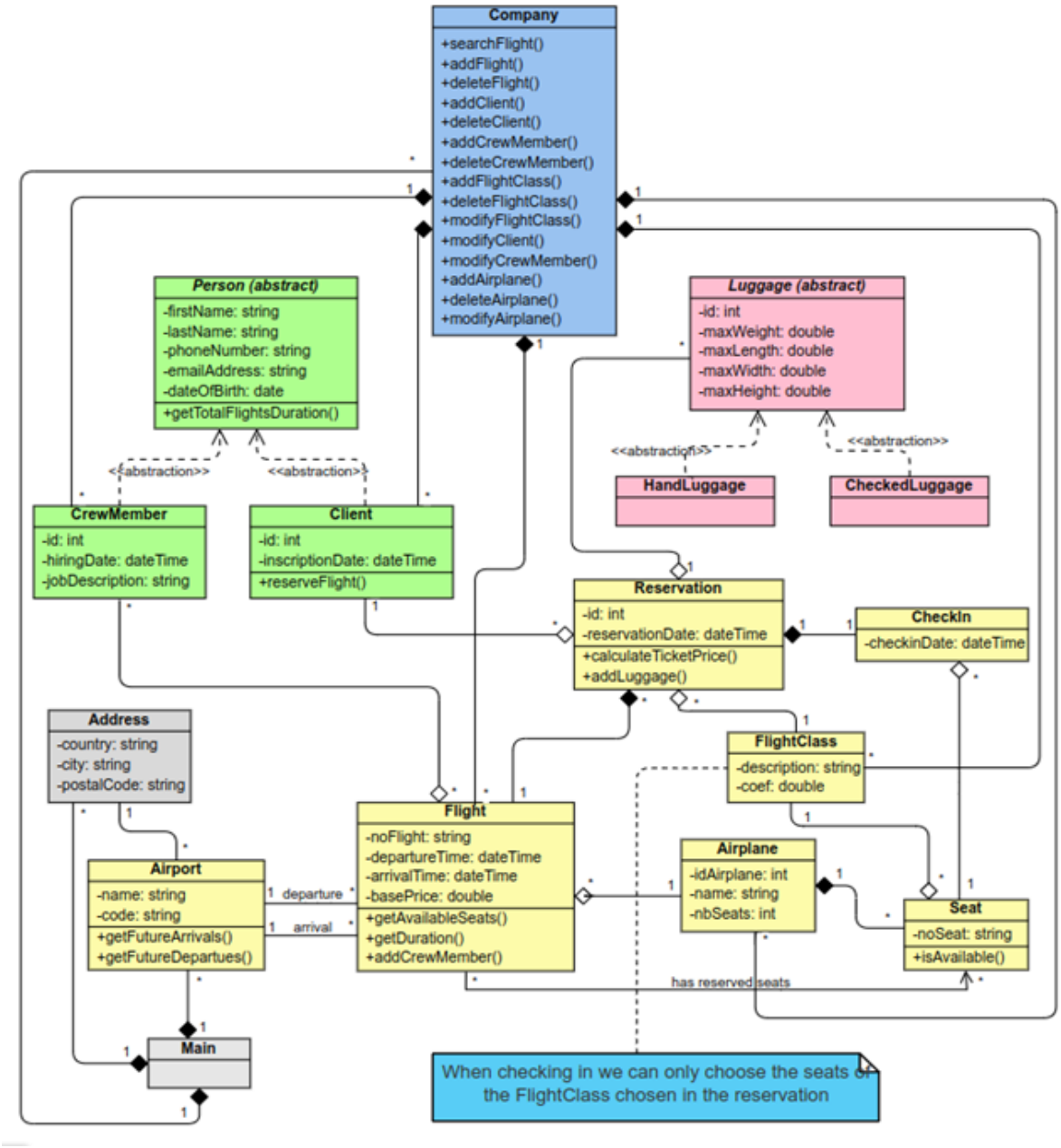

Diagramme de classe

2.3 Interfaces

Nous allons décrire dans cette partie l'interface graphique que nous souhaitons réaliser. Notre projet consiste à mettre en marche un programme gérant les vols au sein d'une compagnie aérienne. Pour illustrer cela de façon visuelle, nous allons utiliser plusieurs méthodes.

Tout d'abord, l'une de nos interfaces graphiques consistera à montrer la partie administrative de la compagnie, dans cette interface cette dernière pourra créer des vols, ajouter, modifier et supprimer des clients, des membres d'équipage, des avions, des classes de vols, etc.

Par la suite le client pourra s'identifier sur le système et visualiser les vols disponible et réserver, et par la suite s'enregister, le membre d'équipage pourra de sa part s'identifier aussi et visualiser

les futures vols associés à lui.

Finalement, la compagnie aérienne pourra gérer toute les réservations de vol et même les enregistrements.

3 Regard critique

Pour rappel, l'objectif du projet est de développer une application de Gestion de Transports Aérien de passagers. Pour la mise en œuvre de cette application, plusieurs concepts de la programmation orientée objet ont été utilisés en commençant par l'utilisation du concept central de classe qui permettra l'instanciation de plusieurs objets. Ainsi le projet défini plusieurs classes comme Person, Luggages, Reservation, etc.

Nous avons également utilisé la notion d'héritage qui est appliquée sur les classes Person et Luggages. La classe mère Person possède deux classes filles Client et CrewMember. La classe mère Luggages possède deux classes filles HandLuggages et checkedLuggages. Ces classes ont été définies comme classes abstraites pour permettre de définir les méthodes dans les classes filles

Pour la réalisation de cette application, la durée totale de réalisation du projet telle que définie dans le cahier des charges est de 13 semaines. A ce jour, la majeure partie du temps de travail a été consacrée à la conception de l'application en définissant son architecture. Plusieurs réunions d'arbitrage ont été nécessaires pour arrêter les différents choix de conception.

En plus des séances de TP réservées au projet, l'utilisation des outils de travail collaboratif comme Gitalb, Discord et Overleaf ont été de grands facilitateurs pour assurer la synchronisation de l'avancement de l'ensemble de l'équipe sur les différentes tâches attribuées à chacun.

Nous avons commencé par répartir les tâches entre les différents membres du groupe de façon équilibrée tout en veillant à communiquer pour nous entraider en cas de difficulté .Le fait de travailler dans une bonne ambiance nous a aidé à créer une cohésion d'équipes qui nous a permis de rester motivé tout au long du projet.

Au début du projet chacun des membres possédait des compétences théoriques en programmation orientée objet.

Ce projet a permis à chacun des membres la mise en pratique des concepts fondamentaux vus en cours sur une application complexe qui peut-être utilisée dans un cas réel. Pour cela nous avons dû développer des compétences techniques et de gestion. Les compétences techniques incluent la conception d'une application qui possède plusieurs classes ainsi que sa mise en œuvre en java. Nous avons également acquis des compétences de gestion comme la répartition des tâches et des échéances, la tenue des délais de réalisations ainsi que la communication au sein de l'équipe.

L'application a été conçue de sorte à faciliter son extension à travers les différents concepts de l'orientée objet. L'une des perspectives d'extension que nous avons envisagée est d'inclure la gestion des marchandises. C'est l'un des objective que nous avons définis au début du projet mais que nous avions écarté pour faute de temps.

Les instructions précises du projet et des questions associées ainsi que le suivi régulier des encadrants nous ont permis de poser des questions au fur et à mesure, ce qui nous a permis d'avancer sereinement.