|  |  |  |
| --- | --- | --- |
| C:\Users\pc dadi\Desktop\logo.jpg | **Université Mohammed Premier**  **Ecole Nationale des Sciences Appliquées**  **Al Hoceima** | http://www.auf.org/media/evenement/umpo.jpg |

***Rapport du Projet :***

***Gestion de vols***

***Réalisés par :***

* *Ayoub Boublil*
* *Ismail Bourhlimi*
* *Taoufik Elfilali*
* *Youssef ait moulay lahbib*

***Encadré par :***

*Mr. Yassir Elmadani Al Alami*

***Plan***

* Introduction
* Etude Préalable :
  + Besoins Fonctionnels
  + Besoins Opérationnels
* Les taches de la méthode scrum
* Conception
  + Les méthodes de conception
  + Les diagrammes UML
    - Diagramme de cas d’utilisation
    - Diagramme de classe
* Implémentation
* Conclusion

***Introduction :***

Dans le cadre de notre formation nous avons été amenés à réaliser une application de gestion.

L’objectif de ce travail est la conception et la réalisation d’une application de gestion de vols en utilisant les différentes taches de la méthode scrum et implémentation des interfaces par SWING et JAVA.

Ce rapport comporte trois parties. La première partie concerne l’étude préalable, la Deuxième la conception et pour finir la partie de la réalisation.

**Chapitre I : étude préalable**

## Introduction :

L’étude préalable constitue une étape préliminaire pour la réalisation d’une application.

En effet, elle permet d’analyser, d’évaluer et de critiquer le fonctionnement habituel, tout en élaborant la liste des solutions possibles.

Dans ce chapitre nous procéderons à l'analyse des besoins fonctionnels et non fonctionnels attendu de l’application à savoir le développement à travers la description des besoins du système qui doivent répondre à l’attente de l’utilisateur.

En effet, l’identification des besoins fonctionnels représente une étape importante du processus de développement, qui est présenté dans l’étude préliminaire.



1. **Les besoins fonctionnels**

Les besoins fonctionnels listent les opérations réalisables de notre application. Ce sont des besoins spécifiant un comportement d'entrée / sortie du système. En fait, le système doit établir les charges préliminaires suivantes :

* Effectuer une réservation du vol.
* Confirmer ou annuler le vol.
* Ajouter un nouvel Aéroport.
* Ajouter une nouvelle Ville.

**2. Les besoins opérationnels**

Les besoins opérationnels représentent les besoins non fonctionnels, qui caractérisent le système comme la performance ainsi que la sécurité et l’ergonomie du système.

Ces besoins peuvent être énoncés suivant des plans de classifications.

**L'ergonomie des interfaces** :

l'interface de l’application, est délicate elle doit être simple et claire :

La manipulation de l'interface ne doit pas nécessiter des connaissances poussées.

Les interfaces de l`application doivent être bien organisée du point de vue graphique, le choix des couleurs, et des styles.

**Robustesse** : assurer une bonne gestion d'erreurs.

**Sécurité** : L'application doit garantir à l'utilisateur connecté l'intégrité et la confidentialité de ses données.

**L'application doit garantir** : la Fiabilité ainsi que la flexibilité, l'évolutivité et la réutilisabilité de ses ressources.

Chapitre II : Conception

## Introduction :

Le succès du projet dépend dès lors de l’adéquation du projet au processus de développement qui est une étape décisive pour l’élaboration d’une application indépendante de toute plateforme d’exécution et de tout langage de programmation. En effet, le processus de développement est constitué d’une succession de phases (spécification, conception et réalisation).

Nous présentons, dans ce chapitre les méthodes de conception les plus citées dans la littérature, et on va choisir une qui sera suivi tout au long de ce projet.

1. **Les méthodes de conception**

On adopte souvent l’une de ces deux méthodologies lors de la conception d’une application quelconque : MERISE comme étant une méthode systémique ou UML pour une méthode orientée objet.

* 1. **MERISE**

MERISE s’appuie sur la séparation des données (la structure des informations que l’application utilise) et des traitements (réaction aux événements externes) en quatre niveaux : conceptuel, organisationnel, logique et physique. Cette séparation va assurer la continuité au modèle. En effet, pour l’ensemble de données comme pour l’ensemble des traitements

MERISE procède d’une manière progressive de l’élément le plus stable à l’élément le plus instable.

* 1. **UML**

UML (Unified Modeling Langage), est un langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un système statique, dynamique, en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.

**Merise ou UML ?**

Les "méthodologues" disent qu'une méthode, pour être opérationnelle, doit avoir 3 composantes:

* + - Une démarche (les étapes, phases et tâches de mise en œuvre).
    - Des formalismes (les modélisations et les techniques de transformation).
    - Une organisation et des moyens de mise en œuvre.

**Merise** est encore tout à fait valable pour:

* La modélisation des données en vue de la construction d'une base de données relationnelle,
* La modélisation des processus métiers d'un SI automatisé en partie par du logiciel.
* la formalisation des besoins utilisateur dans le cadre de cahier des charges utilisateur, en vue de la conception d'un logiciel adapté.

**UML** est idéal pour :

* Concevoir et déployer une architecture logiciel développée dans un langage objet (Java, C++, VB.net). Certes UML, dans sa volonté "unificatrice" a proposé des formalismes.
* Pour modéliser les données (le modèle de classe réduit sans méthodes et stéréotypé en entités), mais avec des lacunes que ne présentait pas l'entité relation de Merise.
* Pour modéliser le fonctionnement métier (le diagramme d'activité et de cas d'utilisation) qui sont des formalismes très anciens.
  1. **La démarche adoptée**

Après cette étude comparative, il est certes que nous adoptons UML comme langage de modélisation puisque nous allons utiliser le concept de l’orienter objet.

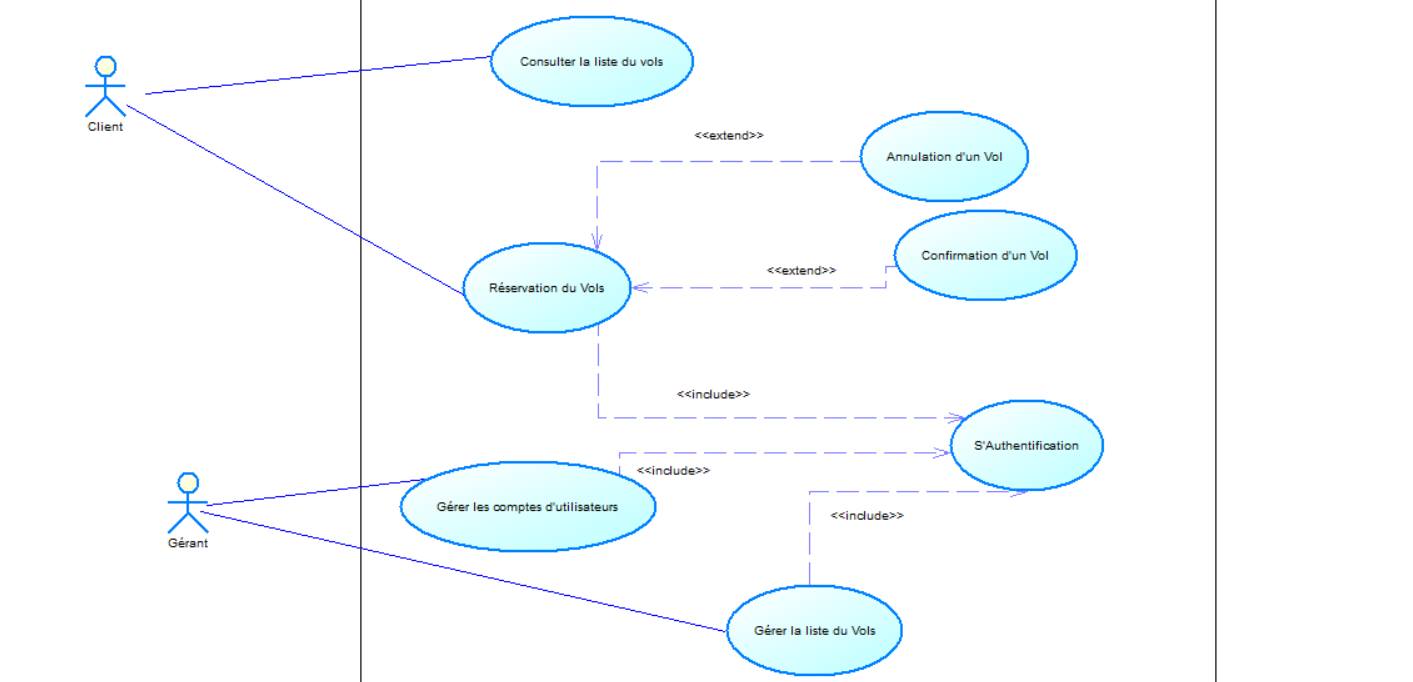
Ainsi, la méthodologie de conception adoptée se base sur le choix de diagrammes UML adéquat. Nous avons utilisé deux diagrammes : diagramme de cas d’utilisation et diagramme de classes.

1. **Les diagrammes UML :**

**2.1** **Diagramme de cas d’utilisation :**

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d’utilisation sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système.   
Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs , ils interagissent avec les cas d'utilisation (use cases).

Voilà le diagramme de cas d’utilisation de notre application :



**Description :**

Ce diagramme présente tous les cas d’utilisations possibles pour cette application. En effet c’est une représentation simple des différentes opérations réalisées par l’utilisateur. Alors dans ce qui suit une description de cas d’utilisations majeures représentés ci-dessus, On trouve donc :

***Client :***

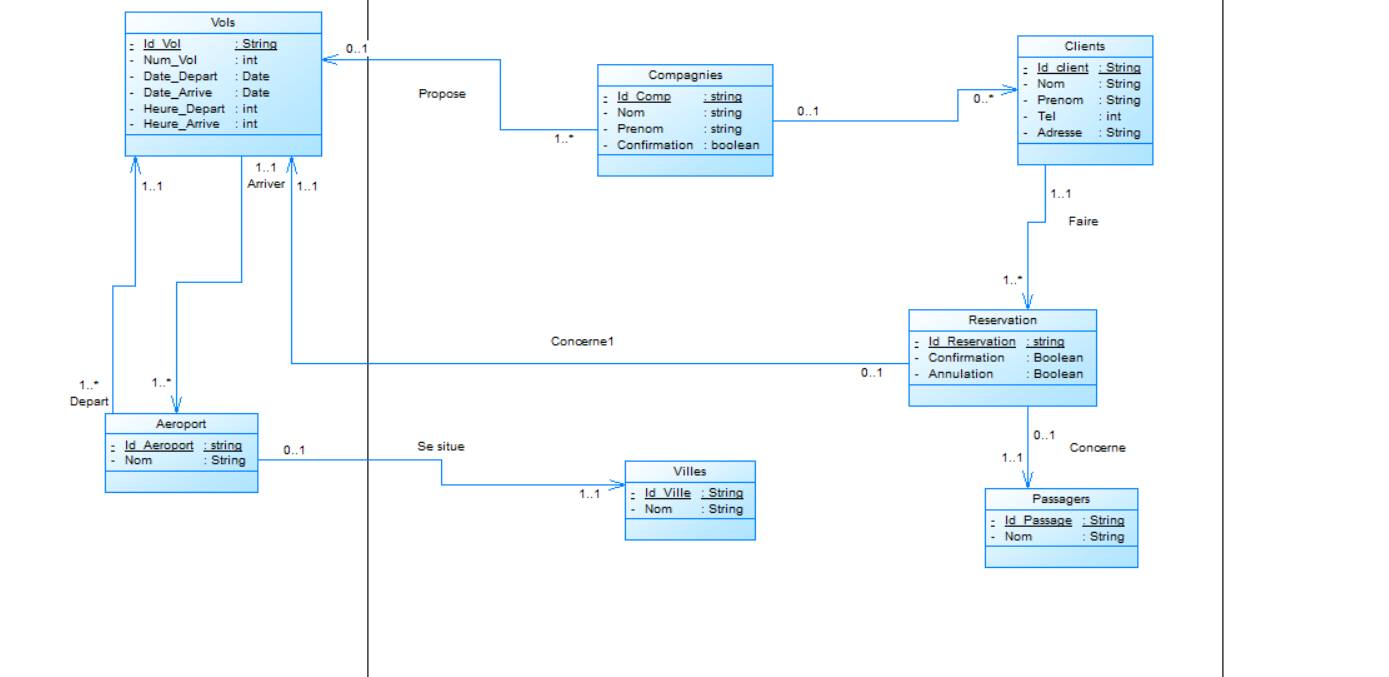
* + Le client peut consulter la liste des vols.
  + Le client peut confirmer ou Annuler une réservation.
  + Le client peut effectuer une réservation d’un vol.

***Gérant :***

* + Le gérant doit s’authentifier.
  + Le gérant peut gérer la liste des utilisateurs.
  + Le gérant peut gérer la liste des vols.
  1. **Diagramme de classe :**

Un diagramme de classes dans le langage de modélisation unifié (UML) est un type de diagramme de structure statique qui décrit la structure d'un système en montrant le système de classes, leurs attributs, les opérations (ou) les méthodes et les relations entre les classes.

Voici notre diagramme de classe globale :



**Chapitre III : Implémentation**

Dans cette partie on va présenter les différentes interfaces qui ont été réalisé par outils NetBeans avec le code source en java :

->Interfaces Login :

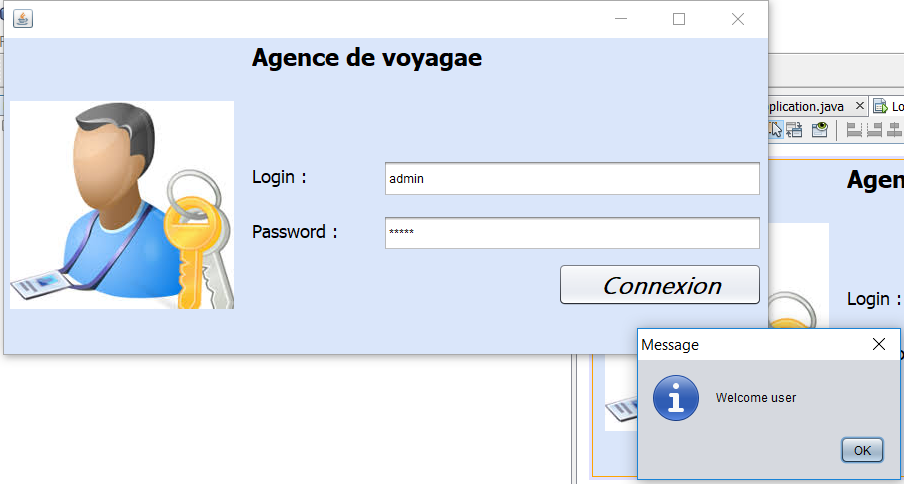


Dans cette interfaces admin s’authentifier pour accéder à l’application.

Si login ou le mot de passe est incorrecte :



Si l’authentification est bien fait :

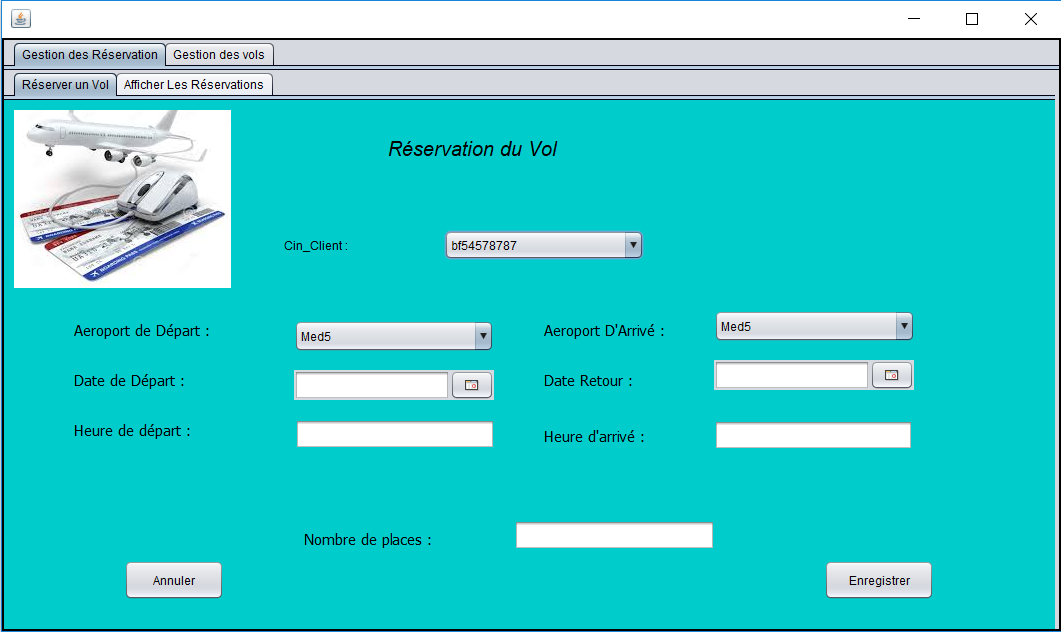


Aprés admin est accédé à l’application :

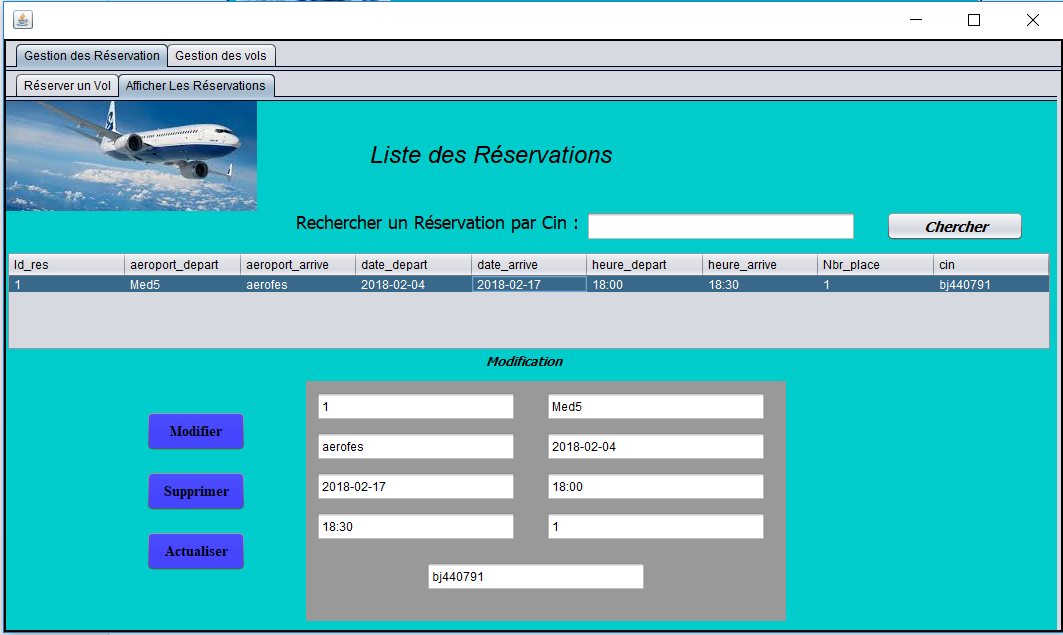
Application contient de partie de gestion :

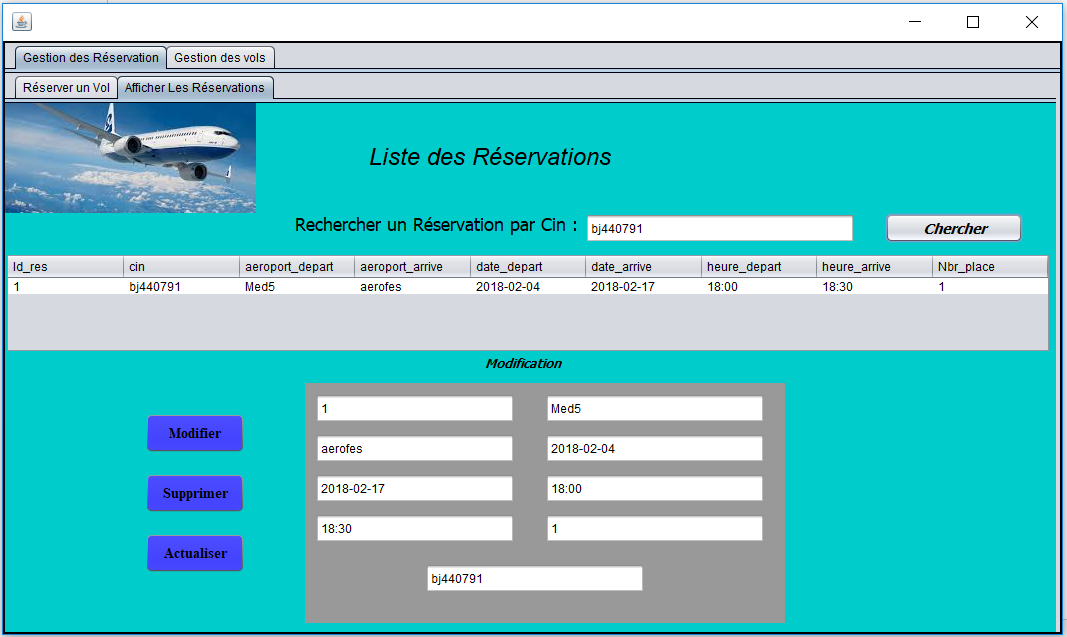
1. ***Gestion de réservation :***

Ajouter une réservation :



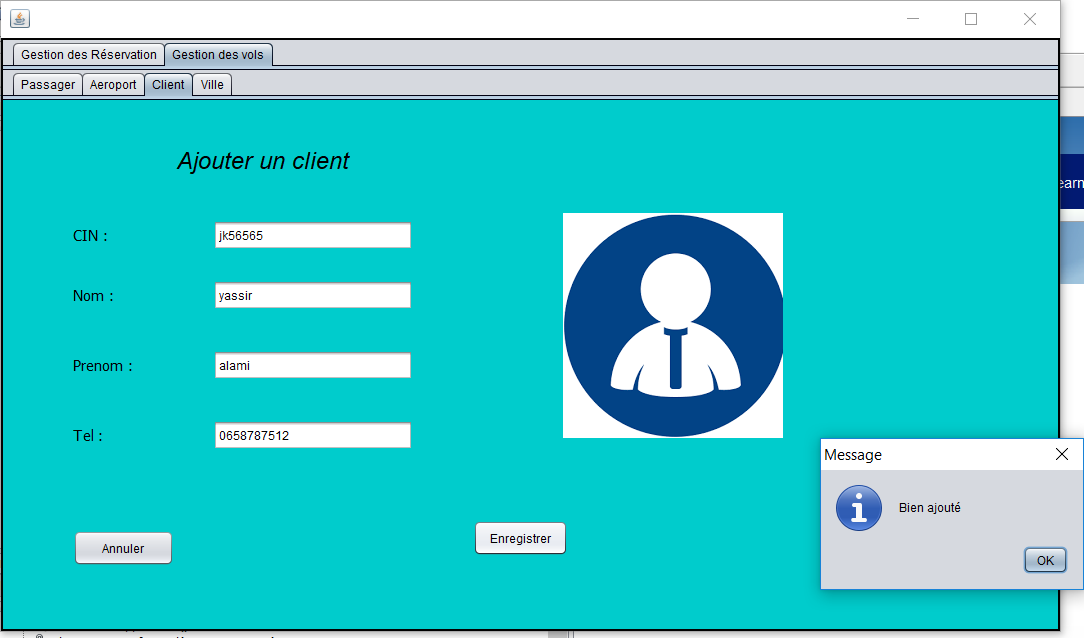
Ou bien lister les réservation, chercher la réservation de chaque client par son cin et la mise à jour de réservation :

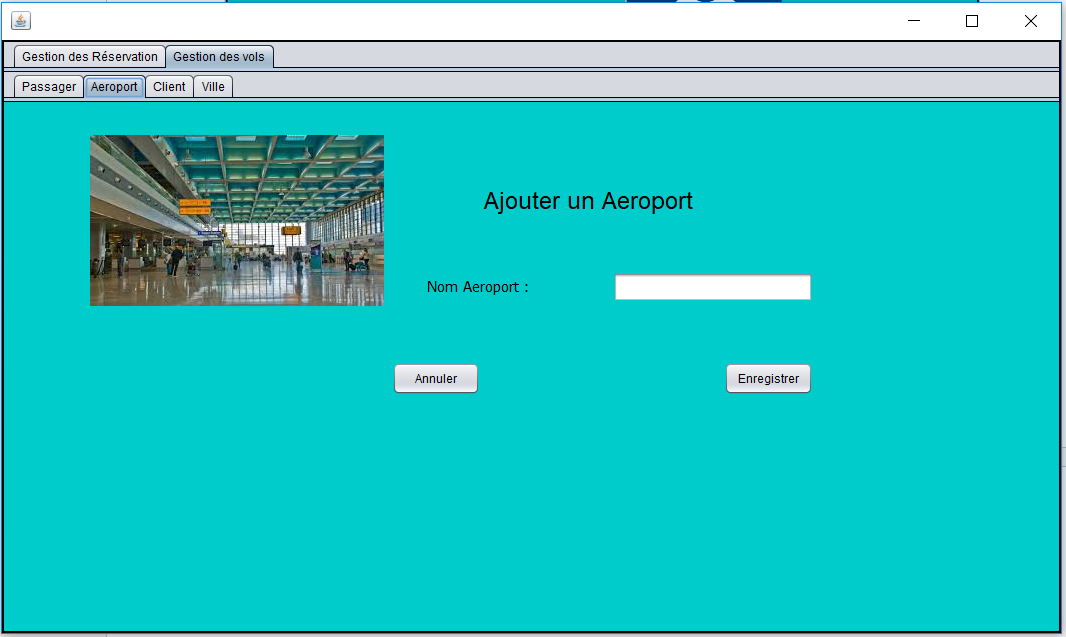


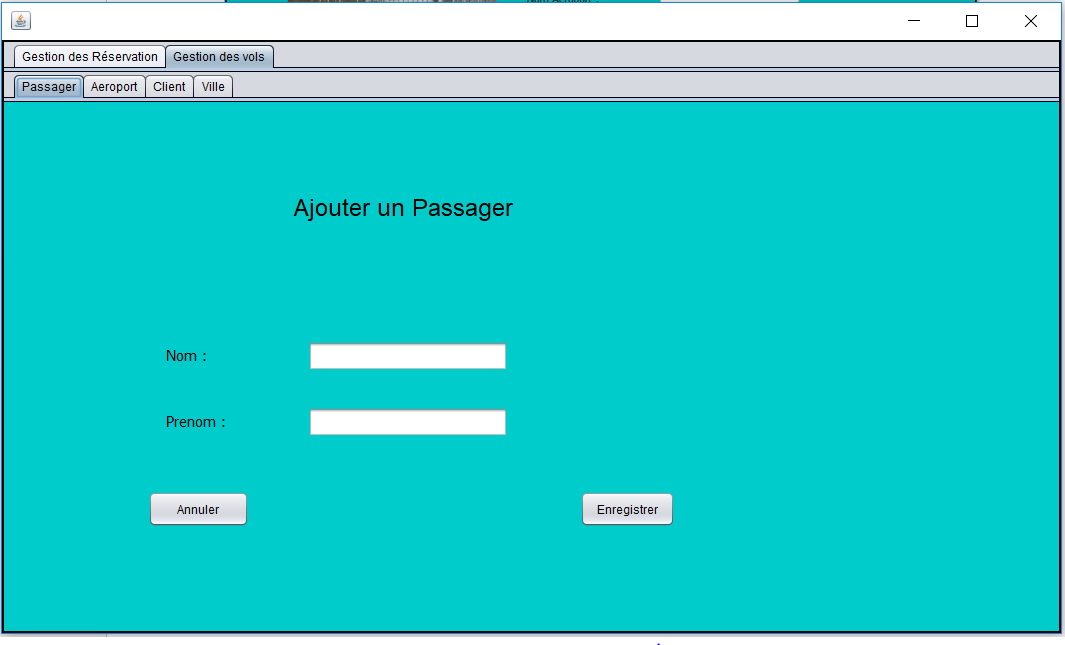


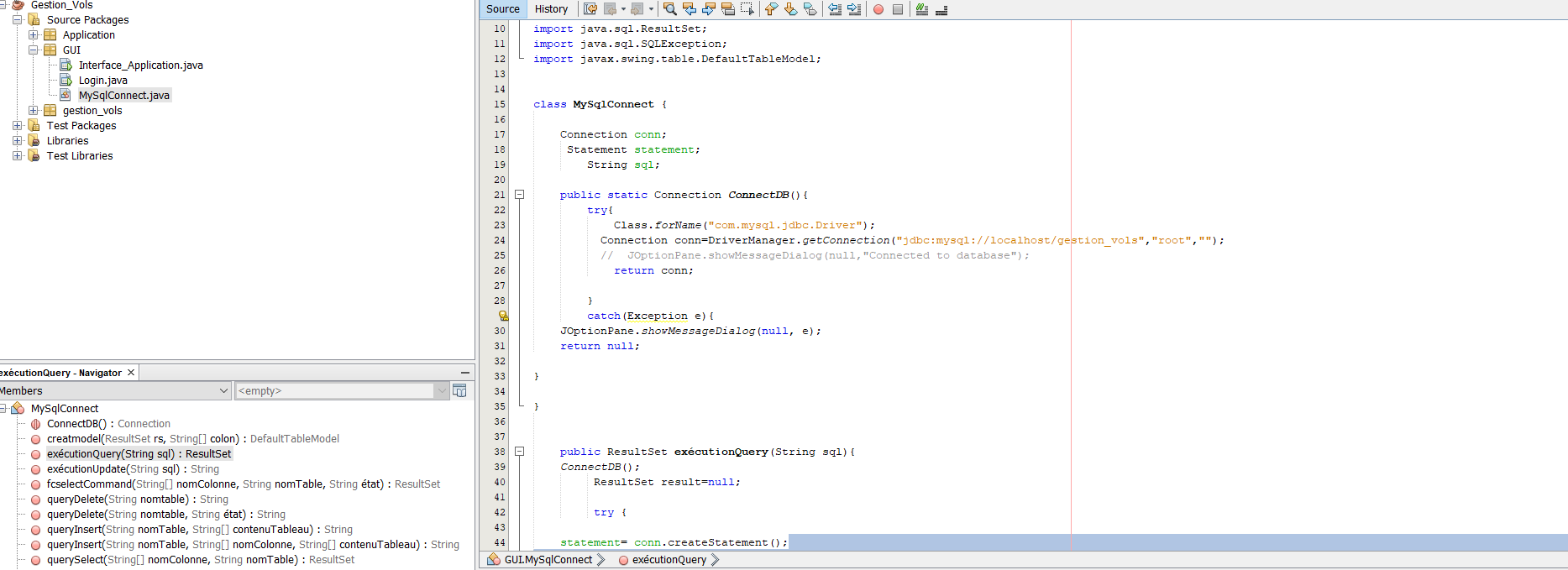
1. ***Gestion de vols :***

Cette partie qui concerne ajout d’un nouveau client, passager, ville et aussi un nouvel aéroport dans notre système :









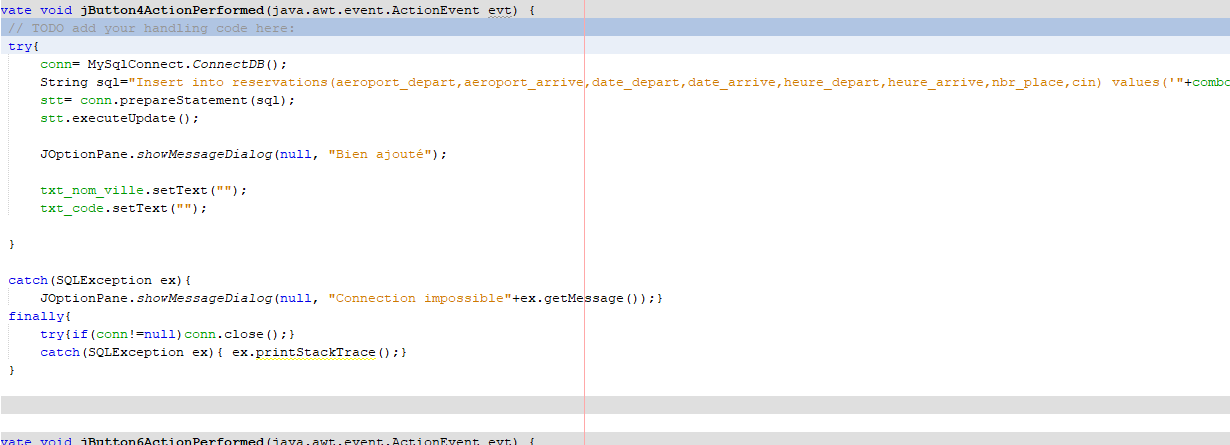
Authentification :



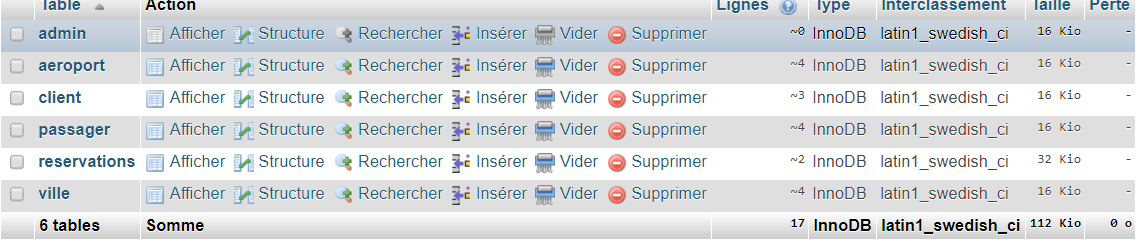
Ajout d’un client :

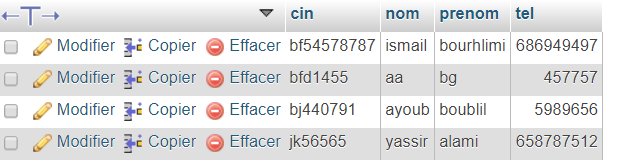


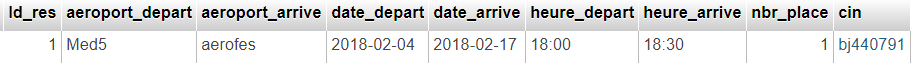
Ajout d’un réservation :



3) ***Pour la partie Base de donnée : MYSQL***







**Conclusion :**

Au cours de ce chapitre, nous avons présenté l’étape d’analyse qui nous a permis de passer d’une structuration fonctionnelle via les cas d’utilisations et les packages à une structuration objet via les classes et les catégories.