

AUZANNEAU Maxime

CAFFY Cédric

G32

Création d’un logiciel de gestion d’aéroclub



*Année 2015-2016*

Sommaire

# Introduction

Dans le cadre de notre projet de première année de cycle d’ingénieur en informatique et modélisation, nous avons souhaité faire un logiciel de gestion d’aéroclub.

Un aéroclub est une association à but non lucratif qui a pour but de permettre la pratique du pilotage d’avions légers.

Notre logiciel doit alors permettre de gérer tout ce qui est en rapport avec les avions et les membres qui composent l’aéroclub.

En première partie, nous présenterons la partie analyse de notre projet (analyse UML, outils utilisés…). En deuxième partie nous expliquerons comment nous avons développé notre application. Enfin, nous aborderons les difficultés rencontrées au sein du projet.

# La partie « analyse » du projet

L’analyse est une phase très importante au sein d’un projet. Elle permet de faire un lien entre les développeurs et l’utilisateur final, mais permet également d’avoir une vue d’ensemble du projet afin que les développeurs puissent avoir un même socle de départ.

## Cahier des charges

### La gestion des membres de l’aéroclub

Le logiciel doit permettre à l’utilisateur de saisir et sauvegarder toutes les informations nécessaires à l’identification d’un membre de l’aéroclub. Chaque membre possède :

* Un nom
* Un prénom
* Une adresse
* Un email
* Un numéro de téléphone
* Une date de naissance
* Une date de péremption de sa visite médicale classe B
* 0,1 ou plusieurs brevets
* Un numéro d’instructeur

#### Le brevet

Un membre n’a pas le droit de voler seul à bord sans avoir un brevet. Le brevet est en réalité un diplôme délivré par un instructeur habilité suite à la réussite du vol de contrôle.

Il existe plusieurs types de brevet :

* Le brevet de base (BB) qui permet au pilote de piloter seul dans un rayon de 30 km autour de l’aérodrome de départ. Le pilote peut aller plus loin si l’instructeur lui en donne l’autorisation.
* Le PPL (Private Pilote Licence) qui permet au pilote de piloter seul ou accompagné de passagers n’importe où dans le monde entier.
* Le LAPL (Light Aircraft Pilot Licence) qui est un PPL allégé dans le sens où le pilote ne peut voyager qu’en Europe.
* CPL (Commercial Pilot Licence) qui est le brevet obligatoire pour pouvoir se faire rémunérer sur chaque vol (Les instructeurs ont tous un CPL).

#### La visite médicale

Chaque membre doit passer une visite médicale d’aptitude au pilotage afin de pouvoir piloter les avions du club.

#### Instructeur

Un instructeur est un membre du club ayant reçu une autorisation (par les autorités compétentes) pour pouvoir donner des leçons au sol et en vol à un élève.

L’instructeur possède un numéro d’instructeur sous la forme FI-XXXXXXXXXXXX.

### La gestion des avions de l’aéroclub

Les membres de l’aéroclub peuvent louer des avions.

Chaque avion possède :

* Un nom (Ex : Robin DR-400, Cessna 172)
* Un type (Voyage, voltige, remorquage)
* Une immatriculation (Sous la forme X-XXXX avec X une lettre de A à Z)
* Une autonomie (en régime de croisière)
* Une capacité de réservoir
* Un nombre de places
* Une masse maximale à ne pas dépasser
* Un centrage (Le centre de gravité de l’avion varie en fonction du poids et de la répartition du chargement). Le calcul du centrage doit être fait avant chaque vol, si le centre de gravité dépasse une limite avant ou une limite arrière, l’avion devient dangereux.
* Un coût à l’heure de vol
* Une disponibilité (Loué, non loué, en réparation)
* Une vitesse de croisière

Le logiciel doit pouvoir stocker toutes les informations de chaque avions de l’aéroclub. Ces informations devront être utilisées pour l’aide à la préparation d’un vol ou pour la réservation d’un avion.

### La gestion du retour d’un vol

Après chaque vol, le pilote doit rentrer les informations de son vol dans le logiciel. La saisie du temps de vol est le plus important car c’est lui qui va déterminer le prix que devra payer le pilote pour son vol.

Le logiciel doit pouvoir enregistrer :

* La date et heure du vol
* Les différentes étapes du vol (Aérodrome de départ, aérodromes d’escale, aérodrome de destination)
* Le temps de vol entre chaque étape
* L’instructeur avec lequel le pilote a volé
* Le nombre de passagers présents

Après la saisie de ces informations, le logiciel doit calculer combien le pilote doit payer à l’aéroclub pour son vol.

### La gestion des paiements

Le pilote ne doit pas être en négatif, la survie du club en dépend.

L’utilisateur doit pouvoir saisir un paiement. Plusieurs types de paiement sont possibles :

* Chèque bancaire
* Espèces

La date du paiement, le nom et le prénom du pilote et le montant payé doivent être enregistrés.

#### Chèque bancaire

Le logiciel doit enregistrer les informations suivantes pour le dépôt d’un chèque :

* La banque à débiter
* Le numéro de chèque

#### Espèces

Le pilote peut payer directement en espèce (après sa saisie, il ira mettre ses espèces dans la caisse de l’aéroclub prévue à cet effet).

### L’aide à la préparation d’un vol (optionnel)

Chaque pilote « consciencieux » doit préparer son vol avant de partir en navigation.

Après préparation, le pilote doit avoir avec lui un log de navigation (Figure 1).

Notre logiciel doit alors permettre d’aider le pilote à préparer son vol en générant ce log de navigation en fonction des informations saisies.

Notre logiciel doit aussi être capable de calculer le centrage de l’avion (en fonction de son chargement) afin que le pilote puisse savoir s’il peut voyager comme prévu en toute sécurité ou s’il va devoir adapter son chargement…

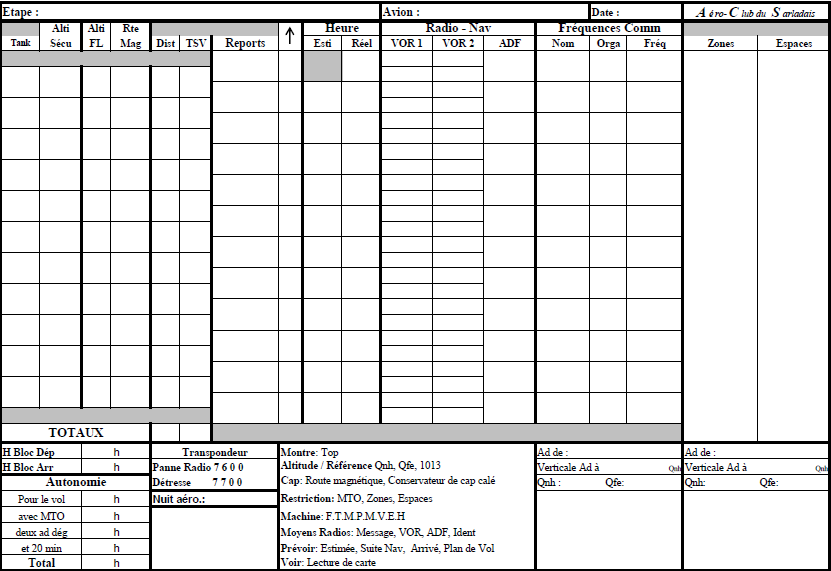


Figure : Un log de navigation

## L’analyse UML

L’analyse UML (Unified Modeling Language) permet de fournir une méthode normalisée afin de visualiser la conception d’un système. Nous avons opté pour ce type d’analyse car nous connaissons son efficacité de par notre expérience antérieure.

### Le diagramme de Use Case

Ce diagramme permet de décrire les cas d’utilisations du logiciel. Il permet de répondre de manière synthétique aux questions « Que peut faire le logiciel ? »,  « Qui peut faire telle ou telle action ? », « Qu’est-il nécessaire de faire pour effectuer une action ? ».

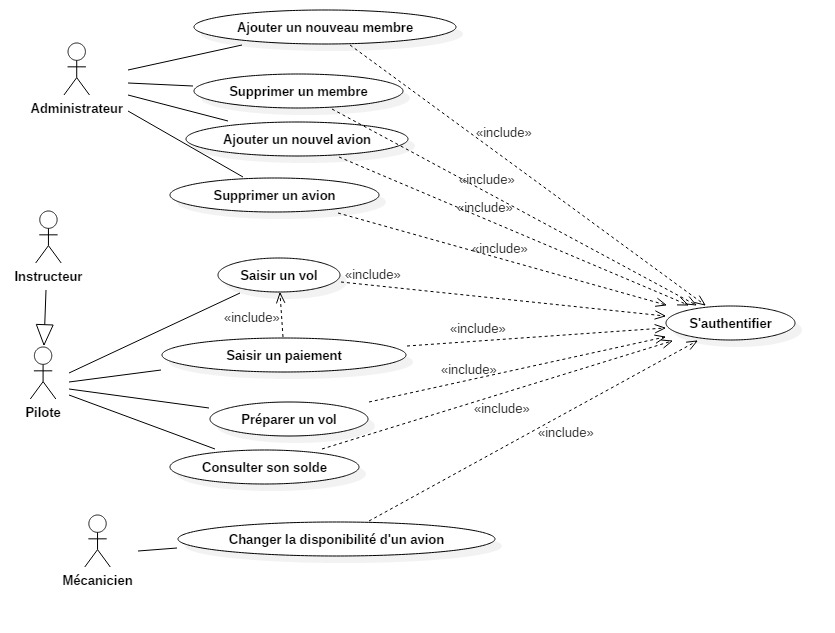
Voici le diagramme de Use Case que nous avons réalisé :

Figure :Le diagramme de Use Case de notre logiciel

### Description détaillée de chaque Use Case

##### Use case « Ajouter un nouveau membre »

Objectif : Inscription d’un nouveau membre dans le logiciel

Acteur principal : l’administrateur

Préconditions : L’administrateur doit être authentifié

Post conditions : Un nouveau membre sera ajouté

##### Use case « ajouter un nouvel avion »

Objectif : Ajouter un nouvel avion dans le logiciel

Acteur principal : l’administrateur

Préconditions : L’administrateur doit être authentifié

Post conditions : Un nouvel avion sera ajouté à la flotte de l’aéroclub

##### Use case « Saisir un vol »

Objectif : Saisie d’un vol lors du retour d’un pilote

Acteur principal : le pilote

Préconditions : le pilote doit être authentifié

Post conditions : Le compte pilote sera débité du montant de l’heure de vol

##### Use case « Saisir un paiement

Objectif : Saisie d’un paiement pour réapprovisionner le compte pilote du membre

Acteur principal : le pilote

Préconditions : le pilote doit être authentifié

Post conditions : le compte pilote sera crédité du montant de l’heure de vol

##### Use case « Supprimer un membre »

Objectif : Supprimer un membre de la base de données

Acteur principal : l’administrateur

Préconditions : l’administrateur doit être authentifié et le membre doit exister

Post conditions : Toutes les informations relatives au membre seront supprimées.

##### Use case « Supprimer un avion »

Objectif : Suppression d’un avion de la base de données

Acteur principal : l’administrateur

Préconditions : l’administrateur doit être authentifié et l’avion doit exister

Post conditions : Toutes les informations relatives à l’avion seront supprimées

##### Use case « Consulter un solde »

Objectif : Consulter le solde d’un membre

Acteur principal : le membre

Préconditions : Le membre doit être authentifié

Post conditions : aucune

##### Use case « Préparer un vol »

Objectif : Aider le pilote à préparer son futur vol

Acteur principal : le pilote

Préconditions : Le pilote doit être authentifié

Post conditions : aucune

##### Use case « Changer la disponibilité d’un avion

Objectif : Permet au mécanicien de changer l’état d’un avion (en état de vol, en réparation, en révision)

Acteur principal : le mécanicien

Préconditions : le mécanicien doit être authentifié

Post conditions : aucune

### Le diagramme de classe

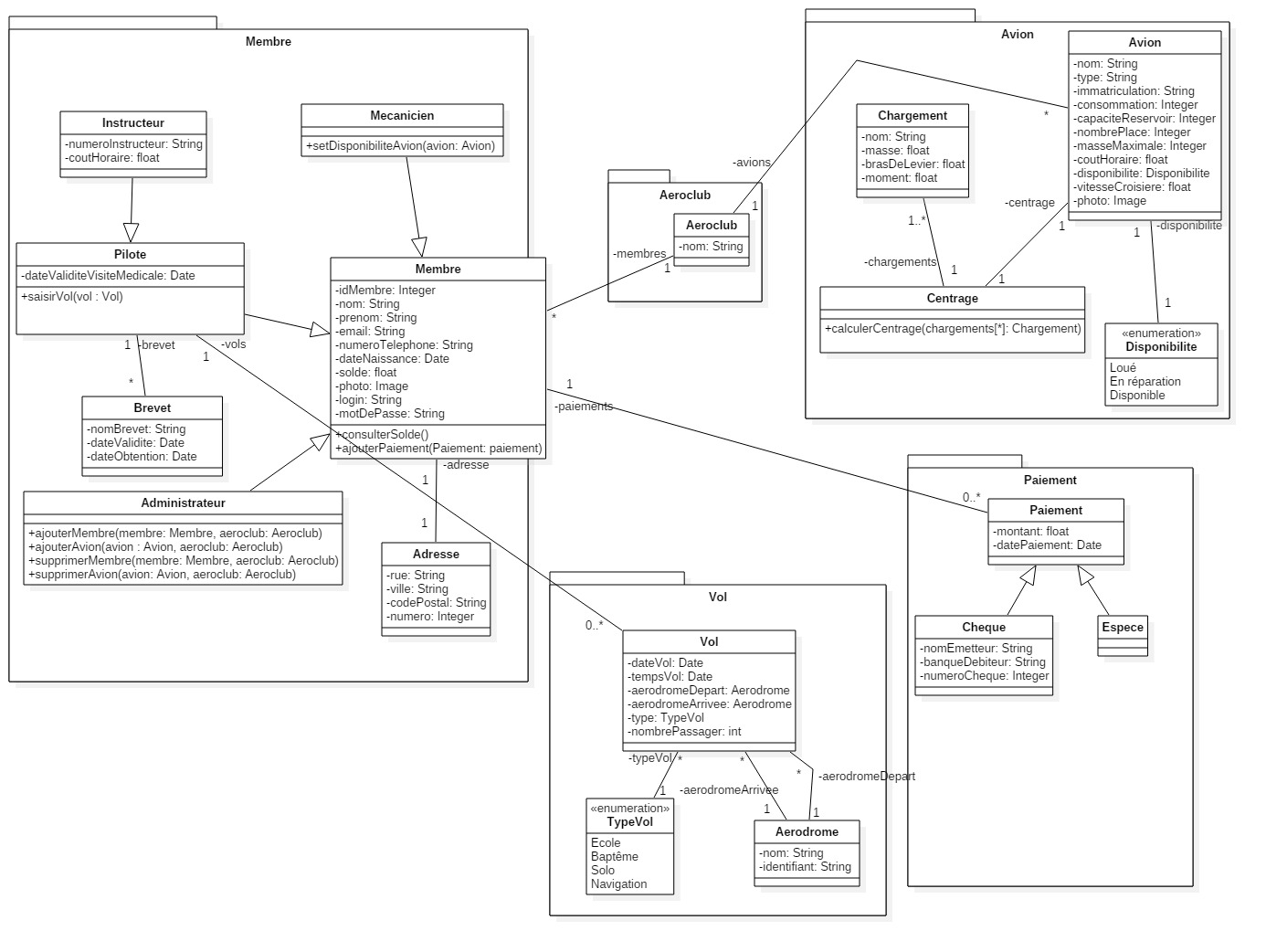
Le diagramme de classe permet d’avoir une vue d’ensemble des différentes classes métiers de notre application. Il permet également de faciliter l’implémentation de ces classes car les liaisons entre chaque classe sont visibles d’un seul coup d’œil.

Figure :Le diagramme de classe de notre application

## La base de données

Nous avons réalisé la base de données à l’aide du diagramme de classe UML. Bien entendu, nous avons dû adapter certaines choses car le diagramme de classe UML n’est pas adapté pour faire de la base de données.

Voici le Modèle Physique de Données (MPD) que nous avons créé :

LORSQUE LA BDD SERA FINIE, il faut mettre le schéma correspondant.

## Les maquettes

Notre application disposant d’une interface graphique, nous avons dû élaborer les maquettes de ces interfaces.

Ces maquettes sont données en annexe.

## Les outils utilisés

### L’analyse UML

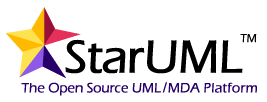
Pour l’analyse UML, nous avons choisi d’utiliser le logiciel StarUML qui permet de réaliser des diagrammes par « Drag’n’Drop ».

Figure :StarUML

### Le langage de programmation

Pour le développement de notre application, nous avons choisi le langage JAVA qui est orienté objet, et que l’on a appris lors de notre DUT.

Figure : Java

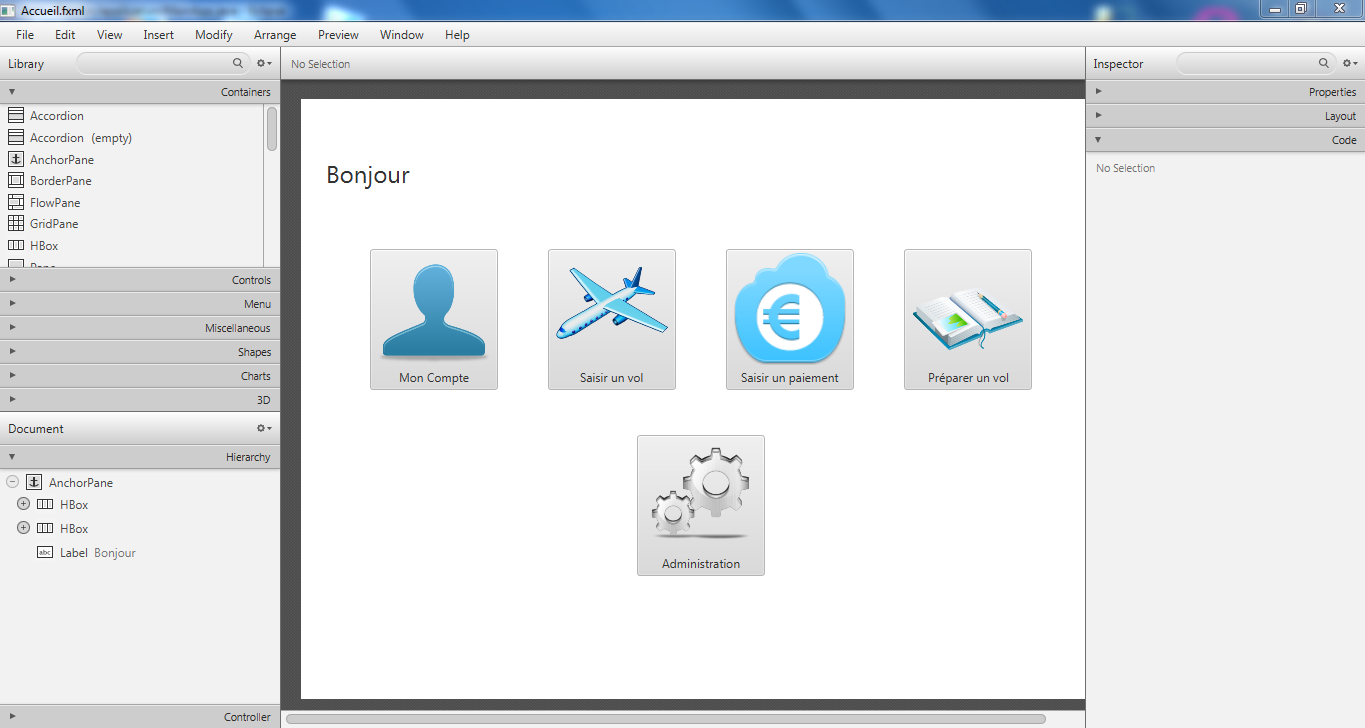
Pour la réalisation des interfaces graphiques, nous avons utilisé le framework JavaFX et Scene Builder qui permet de réaliser les interfaces par « Drag’n’Drop ». 

Figure :L'interface de SceneBuilder

### La base de données

Nous avons choisi d’utiliser le SGBD (Système de Gestion de Base de Données) MySQL pour notre base de données.

Figure :MySQL

C’est un SGBD que nous avons déjà utilisé lors de précédents projets ce qui a facilité sa mise en œuvre.

Afin de consulter le contenu de notre base de données, nous avons utilisé l’application web PhpMyAdmin qui permet de gérer l’administration de la base de données de manière graphique et intuitive.

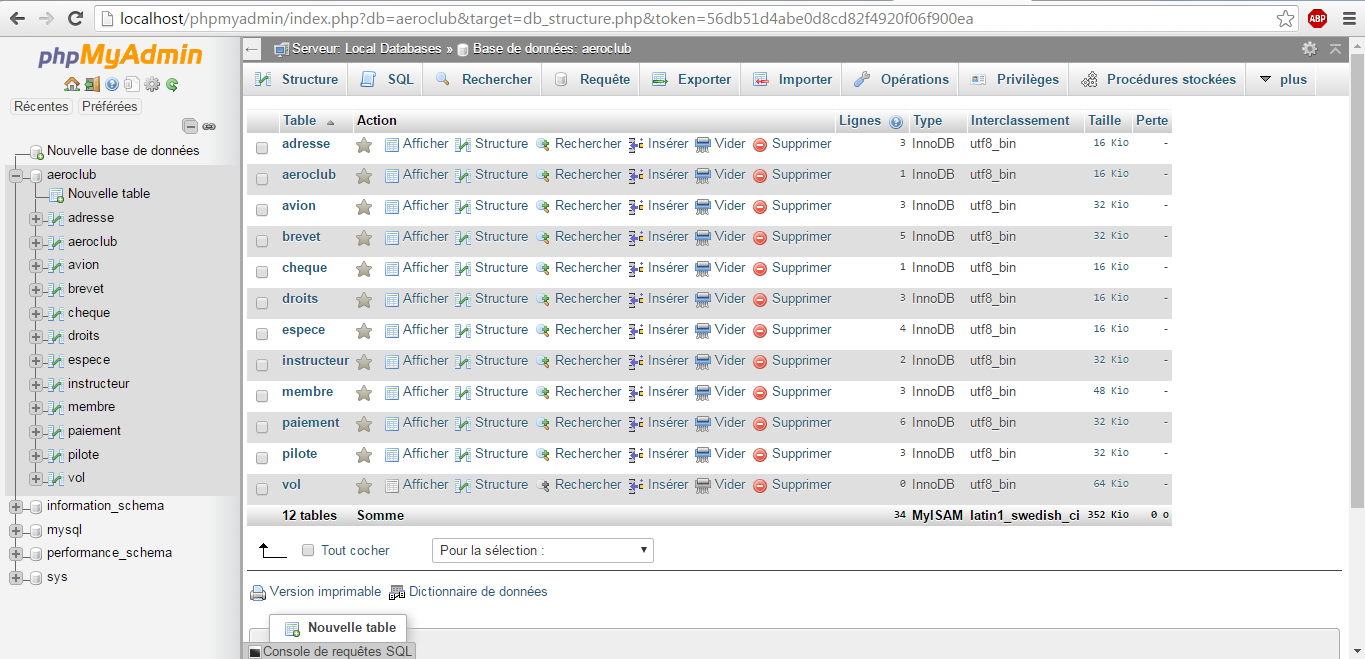


Figure : L'interface graphique de PhpMyAdmin

### L’IDE (Integrated Development Environment)

L’IDE que nous avons choisi d’utiliser est Eclipse qui est un IDE très complet permettant de développer en Java rapidement (auto-complétion, debugger…).

Figure :Eclipse

### Le gestionnaire de version

Il est arrivé que avions à travailler simultanément sur le même projet, mais sur des machines différentes. Nous avons donc dû utiliser un logiciel qui permet de gérer facilement les versions de notre application.

Ce logiciel est GitHub.

Figure :GitHub

# Le développement de notre application

Une fois la partie analyse terminée, nous avons pu commencer le développement de notre logiciel.

## Le design pattern MVC

La bonne pratique pour le développement d’une application graphique est d’utiliser le design pattern MVC.

Ce design pattern permet de séparer de manière distincte les différentes couches de l’application : Model (Couche métier), View (Couche graphique), Controller (Fait le lien entre la couche graphique et la couche métier).



Figure : Le design pattern MVC

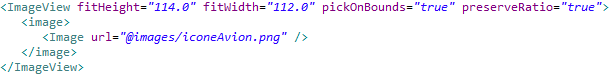
Après avoir réalisé l’interface graphique par « Drag’n’Drop » sur SceneBuilder, un fichier .fxml est généré. Il contient du code xml qui représente l’interface graphique réalisée. Un fichier fxml est créé pour chaque interface graphique. Ce fichier représente la couche « Vue » du design pattern MVC.

Figure :Exemple de code généré par SceneBuilder pour l'affichage d'une image

La couche « Controller » est réalisée par des classes java qui sont liées au fichier .fxml grâce à SceneBuilder.



Figure : Le lien entre la vue et le controller est fait via Scene Builder

La couche « Model » est simplement l’implémentation du diagramme de classe en java.

## La gestion de la base de données en java

Afin de connecter la base de données à notre application java, il a fallu tout d’abord installer le driver JDBC adapté à MySQL.

Ce driver contient les bibliothèques qui permettent de réaliser toutes les interactions possibles avec la base de données.

### Le design pattern Singleton

Nous voulons être sûr qu’au cours de l’utilisation de l’application, une seule instance de la connexion à la base de données soit présente.

Ceci ne peut être fait qu’à l’aide du design pattern Singleton.

Le principe de ce design pattern est le suivant :

C’est une classe contenant une méthode qui permet d’instancier la classe uniquement si elle n’est pas déjà instanciée.