



---

## Gestion des réservations d'une agence de voyage (JavaFX)

---

Reda Mdair

2025

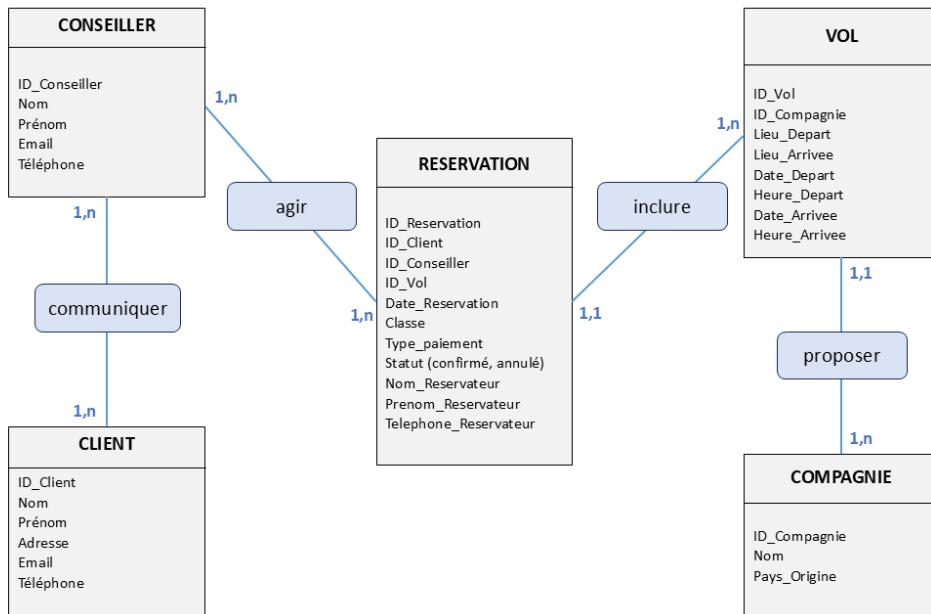
## 1. Introduction

Ce projet vise à développer une application en Java permettant de gérer les réservations d'une agence de voyage. L'objectif est de remplacer une utilisation de fichier Excel par une solution moderne reposant sur une base de données, avec une interface intuitive pour les utilisateurs.

Les informations de réservation incluent notamment le client bénéficiaire, le conseiller associé, l'itinéraire du vol ainsi que la compagnie aérienne l'opérant. C'est ainsi que nous avons défini un modèle conceptuel de données (MCD) à cinq entités :

- Client (Client)
- Conseiller (Travel Agent)
- Réservation (Reservation)
- Vol (Flight)
- Compagnie (Airline)

Les entités sont reliées entre elles par des relations clés primaires et étrangères : par exemple, une réservation est associée à un client, un conseiller, et un vol. Le modèle peut être représenté comme suit :



Les réservations constituent l'unité centrale de notre projet et sont reliées aux autres entités via des clés étrangères. Actuellement, on suppose que l'agence de voyage enregistre toutes les informations liées à une réservation sur une seule ligne dans un fichier Excel. Cette méthode, bien qu'intuitive, est loin d'être optimisée : risque accru d'erreurs humaines (fautes de frappe ou incohérences), manque d'automatisation (absence de relations entre les entités), etc. Notre solution associe un identifiant unique à chaque entité, permettant de référencer efficacement les informations dans les différentes tables. Ainsi, pour une réservation donnée, il suffira d'insérer les identifiants des entités concernées pour afficher tous les attributs associés, grâce à l'intégrité des relations définies dans la base de données.

L'organisation du projet suit une structure modulaire, ce qui facilite la navigation et la compréhension des différentes parties. En voici l'arborescence de ses données source :

```

docker/           contient la configuration du conteneur Docker
├── docker-compose.yml
└── initialization.sql  initialise la base de données directement depuis Docker Compose
src/             dossier principal des sources du projet Java
├── module-info.java
├── config/
│   ├── DatabaseConnection.java  classe gérant la connexion à la base de données
│   └── test.java
├── models/        définit les entités principales du projet
│   ├── Client.java
│   ├── TravelAgent.java
│   ├── Airline.java
│   ├── Flight.java
│   └── Reservation.java
└── dao/          contient les accès aux données pour interagir avec la base
    ├── ClientDAO.java
    ├── TravelAgentDAO.java
    ├── AirlineDAO.java
    ├── FlightDAO.java
    └── ReservationDAO.java
fx/              classes responsables de la gestion des interfaces utilisateur
├── ClientFX.java
├── TravelAgentFX.java
├── AirlineFX.java
├── FlightFX.java
├── ReservationFX.java
├── ExcelViewFX.java
└── Main.java      classe principale de l'interface graphique
executable.jar    fichier permettant l'accès à l'interface depuis un terminal

```

Le fichier `docker-compose.yml` permet de configurer un conteneur MySQL pour le projet Java. Ses principales informations sont les suivantes :

```

environment:
  MYSQL_ROOT_PASSWORD: password
  MYSQL_DATABASE: project_travel_agency
  MYSQL_USER: project_travel_agency
  MYSQL_PASSWORD: password
ports:
  - "3307:3306"
volumes:
  - dbdata:/var/lib/mysql
  - ./initialization.sql:/docker-entrypoint-initdb.d/init.sql

```

Cette configuration simplifie la mise en place d'une base MySQL initialisée avec les données nécessaires au projet. La commande `docker-compose up -d` démarre le conteneur, avec une base de données pré-remplie (par l'intermédiaire du script `initialization.sql` situé dans le même répertoire).

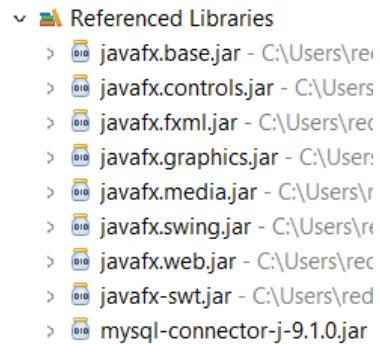
## **2. Gestion du projet via Eclipse**

Le dossier `src` contient l'ensemble des fichiers source du projet, organisés en différents répertoires pour faciliter la lisibilité du code. Ce dossier a été construit depuis l'`IDE Eclipse`, qui a permis de structurer efficacement le projet et de tester les fonctionnalités au fur et à mesure de leur développement. Les fichiers fondamentaux sont les suivants :

- `DatabaseConnection.java` : Responsable de l'établissement et de la gestion des connexions avec la base de données MySQL.
- `Main.java` : Le point d'entrée de l'application, qui lance l'interface principale après l'initialisation de la base de données.

### **Installation de JavaFX**

Pour l'interface graphique, nous avons téléchargé la bibliothèque JavaFX, accessible via [ce lien](#). L'installation a été suivie d'un ajout des librairies associées au SDK JavaFX ainsi que d'un connecteur MySQL ([lien de téléchargement](#)), aux bibliothèques référencées sur Eclipse.



### **Organisation**

Les différents packages collaborent de manière fluide pour produire une application fonctionnelle et modulaire.

- Le package `models` contient les entités principales (par exemple, `Client`, `Flight`, `Reservation`), qui définissent la structure des données manipulées par l'application.
- Le package `dao` implémente les opérations CRUD (Create, Read, Update, Delete) pour chaque entité. Les classes DAO (`ClientDAO`, `FlightDAO`, etc.) interagissent directement avec les instances des classes du package `models`, en utilisant leurs attributs et méthodes pour convertir les données récupérées de la base en objets Java.
- Le package `fx` regroupe les classes responsables de la gestion des interfaces utilisateur. Ces interfaces utilisent les méthodes des classes DAO pour afficher et modifier les données en temps réel.

Le flux de travail commence par une interaction utilisateur dans l'interface graphique (par exemple, ajout d'une réservation), ce qui déclenche un appel à une méthode dans le package `fx`. Cette méthode interagit avec les classes DAO, qui transmettent les requêtes à la base de données via `DatabaseConnection`. Une fois la base mise à jour, les informations sont retournées et affichées

dans l'interface.

L'interface graphique simule l'espace personnel d'un conseiller de l'agence. Suite à une demande d'un client, le conseiller insère les informations du client dans la table **Client**, puis rajoute une réservation à son nom via les vols disponibles dans la table **Flight**.

L'ouverture de l'interface se présente comme suit :

Travel Agent Interface for Booking a Flight for a Client					
Clients	Reservations	Flights	Airlines	Travel Agents	Excel File
Client ID	Last Name	First Name	Address	Email	Phone Number
1	SMITH	James	123 Oxford Street, London, UK	james.smith@example.co.uk	447911123456
2	BROWN	Olivia	45 High Street, Manchester, UK	olivia.brown@example.co.uk	447922234567
3	WILSON	Ethan	78 Baker Street, Birmingham, UK	ethan.wilson@example.co.uk	447933345678
4	JOHNSON	Emily	90 Royal Road, Liverpool, UK	emily.johnson@example.co.uk	447944456789
5	TAYLOR	George	22 Princes Street, Bristol, UK	george.taylor@example.co.uk	447955567890
6	ANDERSON	Sophia	8 Park Lane, Newcastle, UK	sophia.anderson@example.co.uk	447966678901
7	THOMAS	Jacob	11 Piccadilly, Leeds, UK	jacob.thomas@example.co.uk	447977789012
8	ROBERTS	Charlotte	33 Regent Street, Sheffield, UK	charlotte.roberts@example.co.uk	447988890123
9	HARRIS	Daniel	15 Victoria Road, Nottingham, UK	daniel.harris@example.co.uk	447999901234
10	CLARK	Isabella	50 Tower Bridge Road, London, UK	isabella.clark@example.co.uk	447910012345
11	WALKER	Mason	21 Famous Street, Edinburgh, UK	mason.walker@example.co.uk	447911123456
12	LEE	Hannah	17 George Square, Glasgow, UK	hannah.lee@example.co.uk	447912234567
13	EVANS	Matthew	32 Albert Dock, Liverpool, UK	matthew.evans@example.co.uk	447913345678

Figure 1: Accueil de l'interface

Elle permet d'afficher le contenu des cinq entités du projet, ainsi qu'un fichier Excel récapitulatif de toutes les informations détaillées d'une réservation sur une seule ligne.  
Trois boutons sont disponibles :

- **Ajouter** : disponible en permanence.
- **Modifier** : nécessite la sélection préalable d'une ligne.
- **Supprimer** : nécessite la sélection préalable d'une ligne.  
Remarque : pour supprimer un client, il faut d'abord supprimer toutes ses réservations enregistrées.

Les conseillers interagissent uniquement avec les informations des clients et des réservations. Certaines tables, contenant des données statiques ou définies par l'administration, ne sont accessibles qu'en lecture. Les conseillers ne peuvent alors que les consulter :

- **Flight** : Informations sur les vols (lieux, horaires), définies par les compagnies aériennes.
- **Airline** : Détails des compagnies aériennes.
- **Travel Agent** : Informations sur les conseillers, gérées par l'administration de l'agence.

Aucun bouton n'est mis à disposition pour ces tables depuis l'interface graphique. Cette distinction garantit une utilisation optimale de l'application. Les conseillers concentrent leurs efforts sur la

gestion des clients et des réservations. Les données sensibles ou administratives, comme celles des vols et des compagnies, restent protégées contre les modifications accidentelles ou non autorisées.

Voici ci-dessous l'interface affichée pour une modification de réservation. Un formulaire pré-rempli apparaît, permettant de modifier les informations souhaitées.

The screenshot shows a Windows application window titled "Travel Agent Interface for Booking a Flight for a Client". The main window has tabs at the top: Clients, Reservations, Flights, Airlines, Travel Agents, and Excel File. The "Reservations" tab is selected, displaying a grid of flight reservation data. An "Update Reservation" dialog box is open in the foreground, overlaid on the grid. The dialog has fields for "Client ID" (set to 3), "Travel Agent ID" (set to 1), "Flight ID" (set to 5), "Reservation Date" (set to 1/12/2025), "Class" (set to first class), "Payment Type" (set to paypal), and "Status" (set to canceled). Below these are fields for "Booker First Name" (Ethan) and "Booker Phone Number" (447933345678). At the bottom of the dialog are "Save" and "Cancel" buttons. At the very bottom of the main window, there are "Add", "Update", and "Delete" buttons.

Figure 2: Modification d'une réservation

Un conseiller fait notamment recours à cette commande s'il veut annuler la réservation d'un client et la basculer du statut "confirmé" au statut "annulé".

Une fois une réservation ajoutée ou modifiée, elle est instantanément mise à jour dans l'onglet Excel, dont voici la structure :

The screenshot shows the "Excel File" tab selected in the "Travel Agent Interface for Booking a Flight for a Client" window. The main area displays a grid of data with the following columns: Client ID, Client Last Name, Client First Name, Client Address, Client Email, Client Phone, Reservation ID, Reservation Date, Class, Payment Type, Status, Booker Last Name, Booker First Name, and Booked Date. The data rows represent the following information:

Client ID	Client Last Name	Client First Name	Client Address	Client Email	Client Phone	Reservation ID	Reservation Date	Class	Payment Type	Status	Booker Last Name	Booker First Name	Booked Date
1	SMITH	James	123 Oxford Street, London, UK	james.smith@example.co.uk	447911123456	1	2025-01-10	economy	credit_card	confirmed	SMITH	James	447911123456
2	BROWN	Olivia	45 High Street, Manchester, UK	olivia.brown@example.co.uk	447922234567	2	2025-01-11	business	bank_transfer	confirmed	BROWN	Olivia	447922234567
3	WILSON	Ethan	78 Baker Street, Birmingham, UK	ethan.wilson@example.co.uk	447933345678	3	2025-01-12	first class	paypal	canceled	WILSON	Ethan	447933345678
4	JOHNSON	Emily	90 Royal Road, Liverpool, UK	emily.johnson@example.co.uk	447944456789	4	2025-01-13	economy	cash	confirmed	JOHNSON	Emily	447944456789
5	TAYLOR	George	22 Princes Street, Bristol, UK	george.taylor@example.co.uk	447955567890	5	2025-01-14	business	credit_card	confirmed	TAYLOR	George	447955567890

Figure 3: Onglet Excel (coulisser la barre de défilement vers la droite pour les colonnes restantes)

Cet onglet présente une vue globale de toutes les entités liées à une réservation (client, vol, conseiller, etc.), consolidées sur une seule ligne. Il se construit grâce à la classe `ExcelViewFX` qui utilise des jointures entre les différentes entités dans la base de données pour produire une vue complète.

### **3. Gestion du projet via un terminal de commande**

Outre l'interface graphique, l'application permet également de gérer directement la base de données MySQL via un terminal de commande. Cette fonctionnalité offre une flexibilité supplémentaire pour les administrateurs ou les développeurs souhaitant exécuter des commandes SQL manuelles ou automatiser des tâches.

#### **Accès à la base de données**

La base de données est hébergée dans un conteneur Docker, ce qui simplifie son déploiement et son accès. Voici les étapes que nous avons effectuées pour interagir avec elle :

1. Démarrer le conteneur Docker en exécutant la commande suivante dans le terminal :

```
docker-compose up -d
```

2. Accéder à la base MySQL via le conteneur en utilisant une des deux commandes suivantes (choisir selon les droits d'accès nécessaires) :

```
docker exec -it project_travel_agency mysql -u project_travel_agency -p
docker exec -it project_travel_agency mysql -u root -p
```

3. Saisir le mot de passe configuré (donné par [password](#)) lorsque le terminal le demande.

#### **Exécution de commandes SQL**

Une fois connecté, il est possible d'exécuter des commandes SQL pour interagir avec les données :



```
Windows PowerShell

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| performance_schema |
| project_travel_agency |
+-----+
3 rows in set (0.00 sec)

mysql> USE project_travel_agency;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_project_travel_agency |
+-----+
| Advisor          |
| Client           |
| Company          |
| Flight           |
| Reservation      |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Par exemple, pour afficher la liste des vols via la table `Flight`, il faut insérer la commande :

```
SELECT * FROM Flight;
```

```
mysql> SELECT * FROM Flight;
+-----+-----+-----+-----+-----+-----+-----+-----+
| FlightID | AirlineID | DepartureLocation | ArrivalLocation | DepartureDate | DepartureTime | ArrivalDate | ArrivalTime |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 1 | Dubai | Paris | 2025-07-20 | 14:00:00 | 2025-07-20 | 18:00:00 |
| 2 | 2 | Doha | London | 2025-07-21 | 15:30:00 | 2025-07-21 | 19:30:00 |
| 3 | 3 | Zurich | Charleroi | 2025-07-22 | 11:00:00 | 2025-07-22 | 12:40:00 |
| 4 | 4 | Casablanca | Bern | 2025-07-25 | 10:00:00 | 2025-07-25 | 12:55:00 |
| 5 | 5 | Paris | Tokyo | 2025-07-25 | 22:30:00 | 2025-07-26 | 17:00:00 |
| 6 | 6 | London | Los Angeles | 2025-07-26 | 21:00:00 | 2025-07-27 | 05:30:00 |
| 7 | 7 | Frankfurt | Tokyo | 2025-07-28 | 13:00:00 | 2025-07-29 | 01:00:00 |
| 8 | 8 | Atlanta | Toronto | 2025-08-01 | 09:00:00 | 2025-08-01 | 10:30:00 |
| 9 | 9 | Dublin | Brussels | 2025-08-02 | 08:00:00 | 2025-08-02 | 09:45:00 |
| 10 | 10 | Istanbul | Rome | 2025-08-03 | 12:00:00 | 2025-08-03 | 13:30:00 |
| 11 | 11 | Hong Kong | San Francisco | 2025-08-03 | 16:00:00 | 2025-08-04 | 08:00:00 |
| 12 | 12 | Tokyo | Los Angeles | 2025-08-06 | 19:00:00 | 2025-08-07 | 11:00:00 |
| 13 | 13 | Sydney | Auckland | 2025-09-01 | 09:00:00 | 2025-09-01 | 10:30:00 |
| 14 | 14 | Kuala Lumpur | Bali | 2025-09-02 | 11:00:00 | 2025-09-02 | 12:00:00 |
| 15 | 15 | Amsterdam | Bordeaux | 2025-09-03 | 16:30:00 | 2025-09-03 | 18:30:00 |
| 16 | 16 | Madrid | Lisbon | 2025-09-04 | 18:00:00 | 2025-09-04 | 11:00:00 |
| 17 | 17 | Vienna | Warsaw | 2025-10-05 | 14:00:00 | 2025-10-05 | 15:30:00 |
| 18 | 18 | New Delhi | Mumbai | 2025-10-06 | 09:00:00 | 2025-10-06 | 10:30:00 |
| 19 | 19 | Johannesburg | Cape Town | 2025-10-07 | 11:00:00 | 2025-10-07 | 12:00:00 |
| 20 | 20 | Singapore | New York | 2025-10-22 | 23:30:00 | 2025-10-23 | 17:00:00 |
+-----+-----+-----+-----+-----+-----+-----+-----+
20 rows in set (0.00 sec)
```

Cette dernière se génère notamment à partir de clés étrangères pointant vers la table `Airline`, dont voici la structure grâce à la commande :

```
SELECT * FROM Airline;
```

```
mysql> SELECT * FROM Airline;
+-----+-----+-----+
| AirlineID | Name | CountryOfOrigin |
+-----+-----+-----+
| 1 | Emirates | United Arab Emirates |
| 2 | Qatar Airways | Qatar |
| 3 | Swiss International AirLines | Switzerland |
| 4 | Royal Air Maroc | Morocco |
| 5 | Air France | France |
| 6 | British Airways | United Kingdom |
| 7 | Lufthansa | Germany |
| 8 | Delta Airlines | United States |
| 9 | Ryanair | Ireland |
| 10 | Turkish Airlines | Turkey |
| 11 | Cathay Pacific | Hong Kong |
| 12 | All Nippon Airways | Japan |
| 13 | Qantas | Australia |
| 14 | Malaysia Airlines | Malaysia |
| 15 | KLM | Netherlands |
| 16 | Iberia | Spain |
| 17 | Austrian Airlines | Austria |
| 18 | IndiGo | India |
| 19 | South African Airways | South Africa |
| 20 | Singapore Airlines | Singapore |
+-----+-----+-----+
20 rows in set (0.00 sec)
```

## Accès à l'interface graphique

L'interface JavaFX du projet est également accessible depuis un terminal de commande. Nous avons pour cela exporté le projet depuis l'IDE Eclipse sous l'option `Runnable JAR File` et en cochant la gestion de la bibliothèque `Package required libraries into generated JAR`. Cela garantit que toutes les dépendances, y compris JavaFX et les connecteurs MySQL, sont incluses dans le fichier JAR.

Le fichier `executable.jar` généré peut être lancé via la commande :

```
java -p "javafx-sdk-23.0.1/lib" --add-modules
javafx.controls,javafx.base,javafx.fxml,javafx.graphics,javafx.media,javafx.web
--add-opens=javafx.graphics/javafx.scene=ALL-UNNAMED --add-exports
javafx.base/com.sun.javafx.event=ALL-UNNAMED -jar executable.jar
```

## Particularités du terminal de commande

L'accès via le terminal offre une grande flexibilité pour effectuer des opérations avancées qui ne sont pas prises en charge par l'interface graphique. Cependant, un usage inadéquat peut entraîner des erreurs ou des pertes de données si des commandes SQL mal formées sont exécutées.

Cet accès est accordé exclusivement à l'administration et non aux conseillers, qui eux peuvent uniquement consulter l'interface graphique. Cette méthode est essentielle pour diagnostiquer et résoudre rapidement des problèmes liés aux données, ou pour tester des modifications de la structure de la base de données.

De plus, toutes les tâches liées aux privilèges et aux droits d'accès à la base de données, telles que la création de nouveaux utilisateurs, la gestion des permissions, la gestion des tables non modifiables depuis l'interface doivent être effectuées exclusivement depuis le terminal. Cela garantit un contrôle sécurisé et centralisé des autorisations, essentiel pour la robustesse du système.