COMPTE RENDU — TP8 Partie 2 :

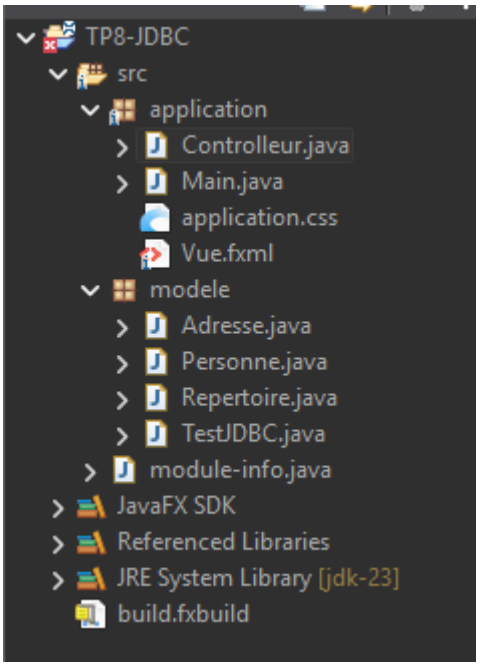
Unal Selahattin SIO2

1. Introduction

Dans cette partie du TP, j'ai intégré une base de données MySQL dans mon application JavaFX du répertoire.
L'objectif était de permettre :

- le chargement des contacts depuis la base lors du lancement de l'application,
- l'affichage des contacts dans l'interface,
- la navigation entre les contacts (**Précédent / Suivant**),
- l'ajout d'un contact dans la base et l'interface.

Ce travail m'a permis de relier JavaFX, JDBC et MySQL dans une même application.



2. Mise en place de la connexion JDBC

Pour interagir avec MySQL, j'ai ajouté le driver `mysql-connector-j` au projet et utilisé `DriverManager`.

Voici le code utilisé dans `Repertoire.java` :

```
private Connection getConnection() throws SQLException {
    return DriverManager.getConnection(
        "jdbc:mysql://localhost/repertoire",
        "root",
        ""
    );
```

3. Chargement des contacts depuis la base

Lors du lancement, le constructeur de la classe `Repertoire` récupère toutes les entrées de la table `contact` :

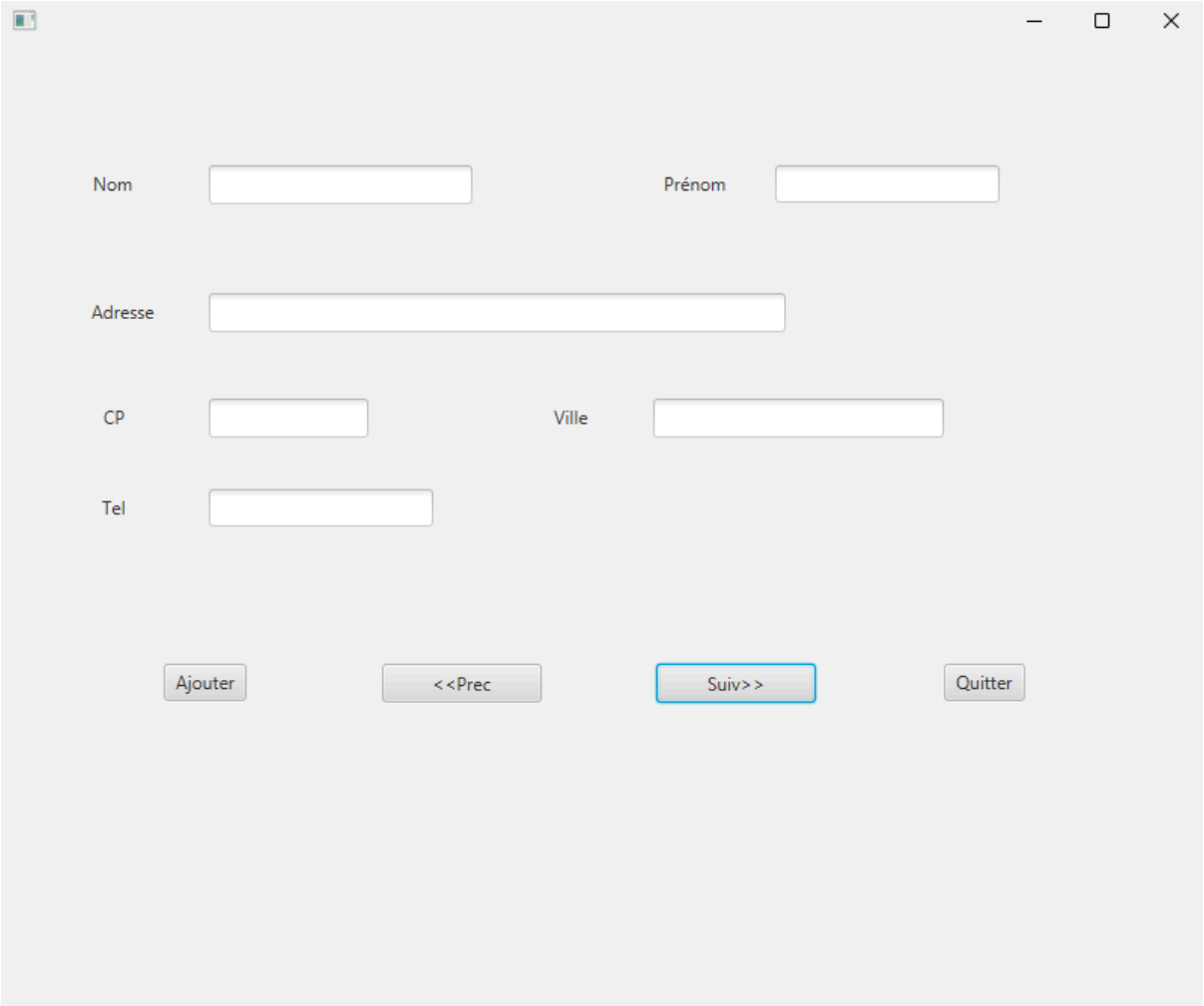
```
            while (res.next()) {
                Adresse adr = new Adresse(
                    res.getString("adresse"),
                    res.getString("cp"),
                    res.getString("ville")
                );

                Personne p = new Personne(
                    res.getString("nom"),
                    res.getString("prenom"),
                    adr,
                    res.getString("tel")
                );

                liste.add(p);
            }
```

4. Affichage dans l’interface JavaFX

J’ai modifié le contrôleur pour que l’interface affiche automatiquement le premier contact :



5. Navigation entre les contacts

Pour la navigation dans les contacts, j’ai mis en place les boutons *Précédent* et *Suivant*.
L’idée est que l’application garde un index du contact actuellement affiché, et selon le bouton sur lequel je clique, cet index augmente ou diminue.

À chaque changement, l’application recharge automatiquement les informations du contact dans les champs (nom, prénom, adresse, etc.).
J’ai aussi prévu des contrôles pour éviter de sortir de la liste : si on est déjà sur le premier contact ou sur le dernier, un message s’affiche pour prévenir l’utilisateur.

Cette partie permet de parcourir facilement tous les contacts récupérés depuis la base de données.

6. Ajout d’un contact dans la base

Pour l'ajout, j’ai connecté le bouton *Ajouter* à un traitement qui récupère tout ce que j’ai écrit dans les champs de texte.
Une fois les données récupérées, l’application crée un nouvel objet contact et fait deux opérations :

- elle l’ajoute dans la liste interne du programme,
- elle l’enregistre aussi dans la base MySQL.

Ensuite, l’interface se met à jour automatiquement et affiche directement le contact que je viens d’ajouter.
L’application garde ainsi tout cohérent entre ce qu’on voit à l'écran et ce qui est vraiment enregistré dans la base.

7. Liaison entre l’interface (SceneBuilder) et le contrôleur

Une étape importante a été de bien relier tous les champs de l’interface avec les variables du contrôleur Java.
Chaque champ de saisie dans SceneBuilder a un `fx:id`, et j’ai utilisé ces identifiants dans mon contrôleur pour pouvoir modifier leur contenu ou récupérer ce que l’utilisateur tape.

C’est aussi comme ça que les boutons déclenchent les bonnes actions : chaque bouton est associé à une méthode du contrôleur.

Cette liaison est essentielle, car sans ça, l’application ne pourrait ni afficher les contacts ni enregistrer ce que l’utilisateur écrit.

8. Conclusion

Avec cette partie du TP, j'ai appris à intégrer une base de données MySQL dans une application JavaFX.
J'ai pu voir comment :

- charger automatiquement les contacts depuis la base,
- les afficher dans l'interface,
- naviguer entre eux,
- ajouter de nouveaux contacts tout en les enregistrant dans MySQL,
- et relier correctement les éléments de SceneBuilder avec le contrôleur Java.

Même si cela demande de bien comprendre plusieurs éléments à la fois (l'interface, le code Java, et la base SQL), cette partie montre comment fonctionne une vraie appli avec une base de données derrière.
C'est quelque chose qu'on retrouve dans beaucoup d'applications professionnelles.