Développement d’une appli bancaire en Java/Javafx

1 Présentation de l'application

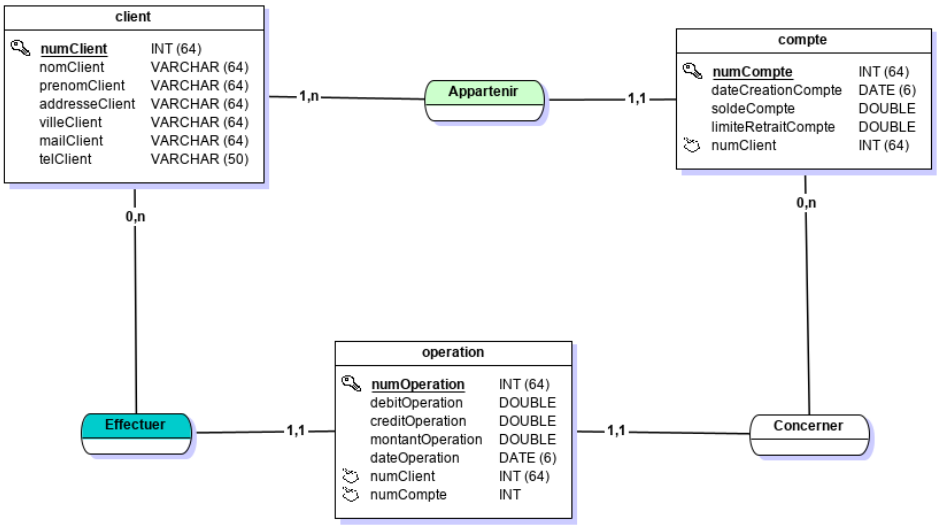
Contexte : Bankappli est une application de gestion des comptes clients, qui sera utilisé par les agents à l'accueil.

Outils & environnement de développement :

- Langages : Java
- SGBDR : MySQL, connecteur JDBC MySQL pour Java
- Framework : JavaFx et SceneBuilder pour l'interface graphique
- IDE : Eclipse

1.1 Le mcd

La base de données : MCD et MLD



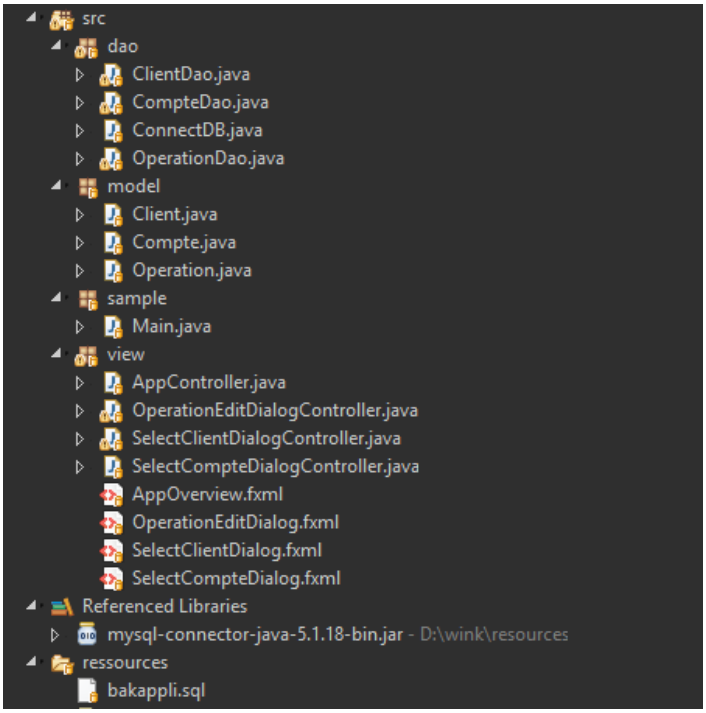
CLIENT (idClient, nomClient, prenomClient, adresseClient, villeClient, mailClient, telClient)

COMPTE (idCompte, dateCreation, soldeCompte, limiteRetrait, #idClient)

OPERATION (idOperation, typeOperation, montantOperation, dateOperation, #idClient, #idCompte)

1.2 L'architecture de l'application.

L'application est organisée en plusieurs packages, en fonction de leur utilité :



- Le package **DAO** contient tout ce qui est nécessaire pour faire le lien entre la base de données et Java. C'est dans ce package que l'on va faire le lien et interagir avec la base de données (DAO = Data Access Object).

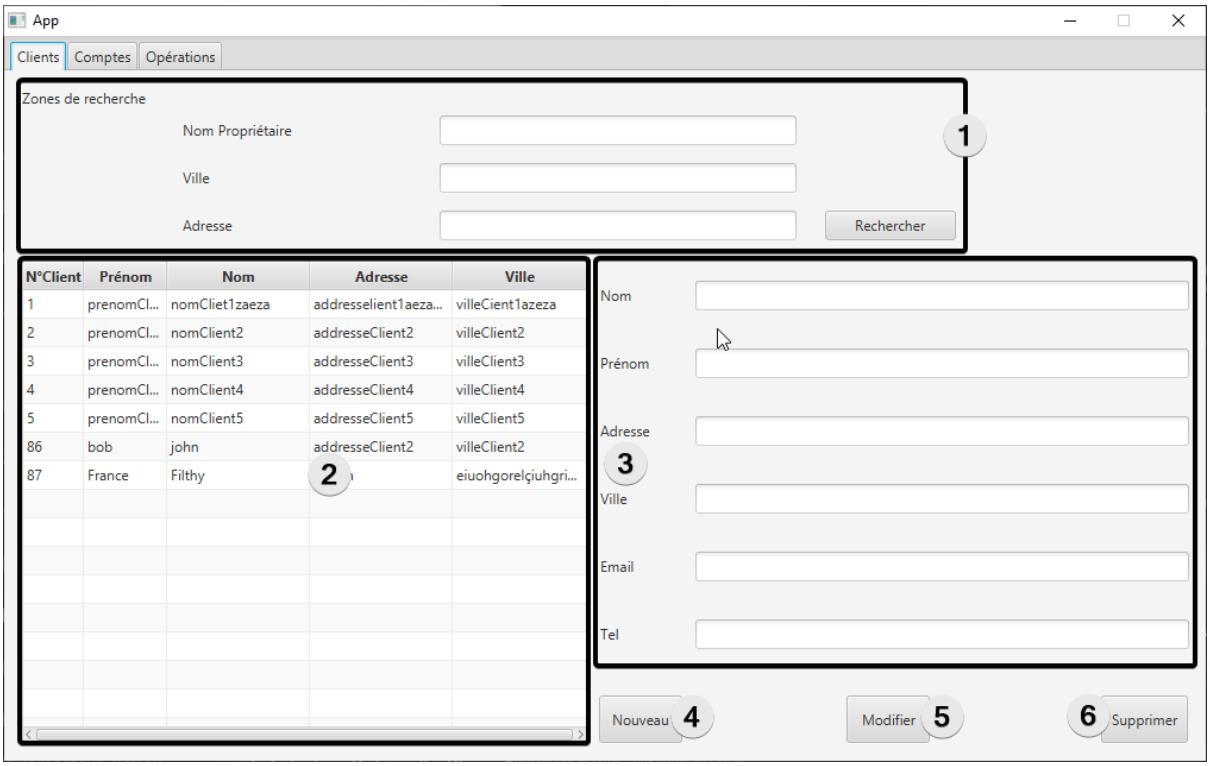
  ConnectDB est la classe faisant le lien avec la base de données. Cette classe est à éditer pour changer l'utilisateur de la base de données affilié à l'application. Un mot de passe, un nom d'utilisateur et un lien vers la base de données sont stockés dans cette classe.

- Le package **model** contient la définition des objets Java que le code utilise. Ils ont été créés directement à partir du MCD (indépendant de JavaFx)
- **Sample** contient le Main. Il s'agit de la classe principale de l'application.
- Le package **View** contient l'interface graphique ainsi que les Controller, qui permettre l'interaction avec l'interface graphique. Chaque fenêtre à son Controller.

1.3 Quelques informations supplémentaires

Nous allons beaucoup parler de **tableView,** de labels ou encore de **listenner**. Il est nécessaire de bien comprendre à quoi se réfèrent ces noms avant d'utiliser la documentation technique.

2 L'onglet client



Cette fenêtre est liée à l'AppController, qui est la classe centrale i de l'application. La classe est dans le package View.

Il s'agit du controller principal, qui se charge de faire marcher la fenêtre principale. Ce controller est divisé en trois parties, Client, Comptes et Operations. On détaillera ici l'interface de l'onglet Client.

2.1 Zone de recherche

La zone de recherche contient trois **textFields** et un bouton. Les trois **textFields** sont utilisés quand l'on appuie sur le bouton rechercher. Le bouton rechercher va appeler la fonction **handleClientRechercher**, qui utilise **ClientDao**, le **Main**, et le model Client.

2.2 TableView principal

Ceci est un **tableView,** qui s'initialise au lancement de l'application. Chaque colonne, ainsi que le **tableView** est déclarée séparément dans l'**AppController**. Le nom du **tableView** est **clientTable**.

Un petit extrait du Controller ce serait le bienvenu ici.

Le **tableView** est déclaré initialisé dans la méthode **initialize**. On le remplit ensuite en utilisant une liste observable. Notre liste observable étant déclarée dans le Main, on utilise un getter que l'on va mettre dans la fonction **setMain** afin de pouvoir récupérer et utiliser les données de l'**observableList** du Main. Impossible à comprendre par quelqu'un d'autre qui n'a pas ton projet en tête !

Enfin, le tableView est sujet à un **listenner**, déclaré dans la méthode **initialize**. Le **listenner** consiste à, quand on clique sur le tableView, envoyer l'objet java sélectionné dans les textFields du 3.

2.3 Détails du client (TextField ne mérite pas un titre)

Les textFields du côté droit, initialement vides, se remplissent automatiquement si l'utilisateur clique sur un objet du **tableView**. Les boutons 'modifier', **'supprimer'** et 'nouveau' vont dépendre de ces textFields. La méthode de remplissage automatique est **showClientDetails**, qui actualisé à chaque clique grâce à un **listenner** déclaré dans la méthode **initialize**. Remarque R1

2.4 Bouton 'Nouveau'

Le bouton 'Nouveau' ajoute un client en récupérant les informations des textFields du 3. Ce bouton va ajouter un nouveau client dans la base de données. Il se verra attribuer un nouveau numéro de client, que l'on ne pourra pas changer.

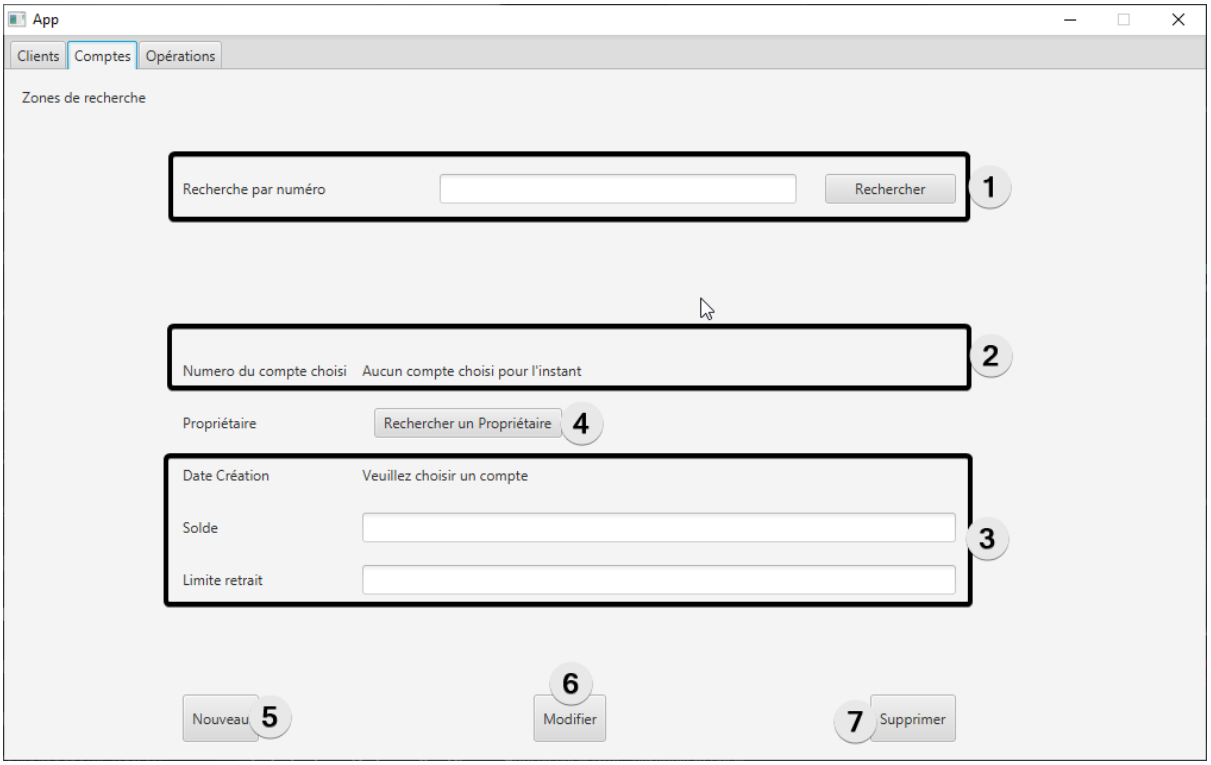
2.5 Bouton 'Modifier'

Le bouton modifier va actualiser la base de données en utilisant les données des textFields. On ne peut pas changer le numéro de client. **HandleClientModifier** est appelé à chaque fois que le bouton est pressé. Cette méthode est en lien avec la base de données, à travers **clientDao.update**.

2.6 Bouton 'Supprimer'

Le bouton supprimer va actualiser la base de données en supprimant le compte sélectionné dans le **tableview**. La méthode **clientDao.delete** est appelée à travers la méthode **handleClientSupprimer**.

3 L'ONGLET COMPTE



L'onglet compte est toujours sur la même fenêtre que l'onglet client ; les deux onglets partagent donc le même **controller**, à savoir l'AppController du package View.

3.1 Zone de recherche

Le bouton rechercher va récupérer le numéro dans le **textField** à sa gauche. Il va ensuite questionner la base de données à travers **handleCompteRechercherParNumero** ainsi que **compteDao.speceficFind**, qui va faire une recherche par numéro de compte dans la base de données.

3.2 Label de sélection de compte

Ce label nous indique si un compte a été choisi. On ne pourra pas modifier un compte si un compte n'a pas été sélectionné. Ce label n'est rempli que quand on sélectionne un compte en faisant une recherche par numéro. Il utilise la méthode **showCompteClientDetails**. Un compte ne peut être modifié si ce label indique qu'aucun compte n'a été sélectionné.

3.3 Les textFields Solde, Limite retrait et label date création compte

Les textFields permettent de modifier la solde du comte, la limite de retrait du compte, ou de créer un compte avec les données des textFields. Le label est mis à jour avec la méthode **showCompteClientDetails.**

3.4 Le bouton 'Rechercher un Propriétaire'

Le bouton 'Rechercher un Propriétaire' va ouvrir une boite de dialogue. La méthode ouvrant la boite de dialogue est **handleCompteRechercherPropriétaire**. Les boites de dialogue sont détaillées dans une autre partie de la documentation.

3.5 Le bouton 'Nouveau'

Le bouton 'Nouveau' exécute la méthode **handleCompteAjouter**, qui va ajouter le compte dans la base de données, et sélectionner automatiquement le nouveau compte. L'interaction avec la base de données est faite avec la méthode **clientDao.add.**

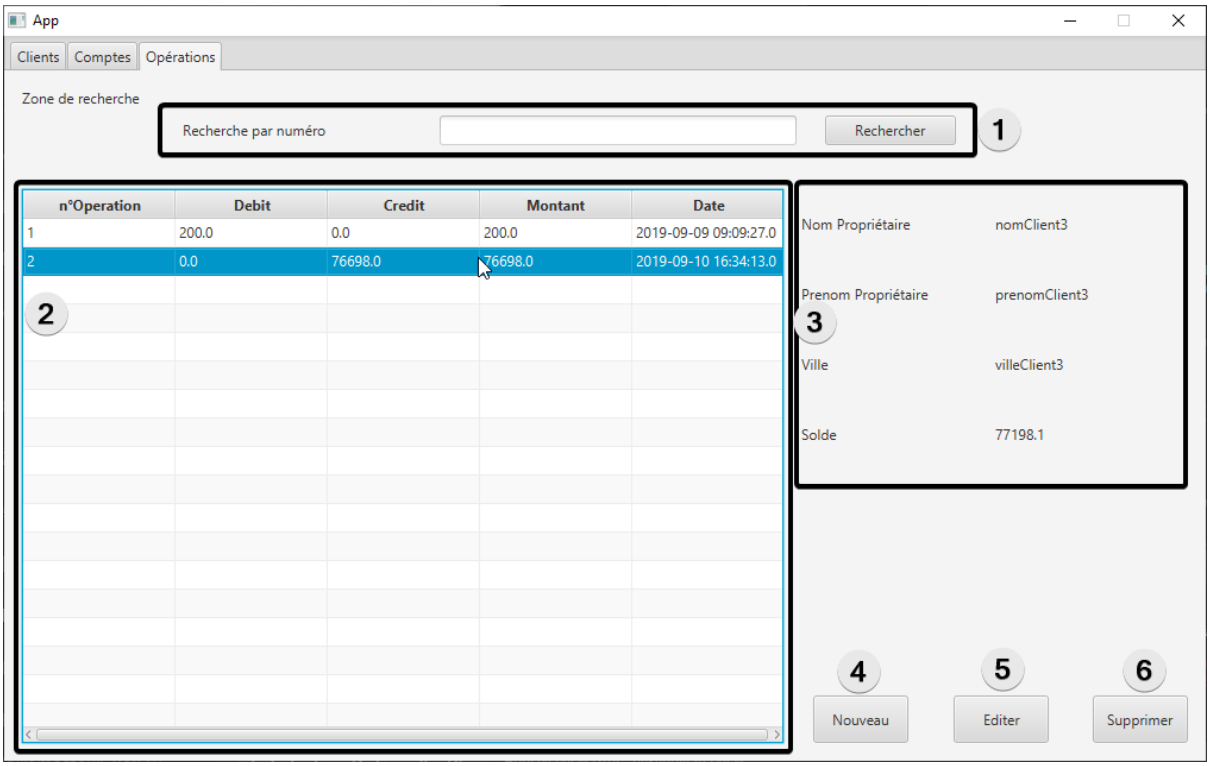
3.6 Le bouton 'Modifier'

Le bouton modifier va exécuter la méthode **handleCompteModifier,** qui va mettre à jour la base de données avec les nouvelles informations. L'interaction avec la base de données est faite avec la méthode clientDao.update.

3.7 Le bouton 'Supprimer'

Le bouton supprimer va exécuter la méthode **handleCompteSupprimer,** qui va supprimer le compte sélectionné de la base de données et de Java. L'interaction avec la base de données est faite avec la méthode **clientDao.remove**.

4 L'ONGLET OPERATION



4.1 La zone de recherche

La zone de recherche est composée d'un textField et d'un bouton 'rechercher'. Le bouton rechercher exécute la méthode **handleOperationRechercherParNumero,** qui va questionner la base de données à travers **OperationDao.specificFind.**

4.2 Le tableView principal

Le tableView de la table opération est initialisé dans la méthode initialize, et récupère ses données dans la méthode **setMain**. **ShowOperationDetails** est appelée à chaque click sur un objet du TableView.

4.3 Les labels du côté droit

Les labels du côté droit de l'écran sont directement liés à la méthode **showOperationDetails.** Ils vont se remplir à chaque click sur un objet du tableView, en affichant les informations relatives à l'**Operation**. Une requête est nécessaire à l'affichage de ces informations, et cette méthode est liée à la base de données à travers **ClientDao.specificFind**.

4.4 Le bouton 'Nouveau'

Le bouton 'Nouveau' va ouvrir une boite de dialogue qui va nous permettre de créer l'opération. Les boites de dialogues sont détaillées à la fin de cette section. La méthode appelée est **handleOperationAjouter**, est liée à la base de données à travers **OperationDao.check** et **OperationDao.add**. **OperationDao.check** va vérifier que l'opération est légale vis à vis de la limite de retrait du compte, et **OperationDao.add** va ajouter l'opération dans la base de données.

4.5 Le bouton ‘Editer’

Le bouton ‘Editer’ va ouvrir une boite de dialogue qui va se remplir automatiquement se remplir avec les détails de l’opération que l’on édite. La méthode appelée est **handleEditOperation**, qui est liée à la base de données à travers OperationDao.check et **OperationDao.edit**. La boite de dialogue est appelée avec **main. showOperationEditDialog**.

4.6 Le bouton ‘Supprimer’

Le bouton ‘Supprimer’ va supprimer l’objet sélectionné dans la base de données, en utilisant **OperationDao.remove**.

5 LES BOITES DE DIALOGUES

Les boites de dialogues ont leurs propres contrôleurs. La fenêtre intitulée ‘Select Client’ a pour controller **SelectClientDialogController**, et la fenêtre **‘SelectCompte’** a pour controller **SelectCompteDialogController**.

5.1 Select Client / Select Compte

Les deux boites de dialogues partagent presque exactement le même code, et donc presque les mêmes fonctions.