

Esalaf

**Aya Laadaili
Groupe 1**

03/04/2023

—
**Programmation Orientée Objet en
JAVA**
—

**Pr. El Mokhtar EN-NAIMI & Pr.
Elachaak**

Table des matières

Page de garde	1
Table des matières.....	2
Introduction.....	3
Plan de developpement.....	4
Base DAO	4
1.Interface Authentification.....	5
<i>Image de l'interface</i>	<i>8</i>
<i>Classe Admin</i>	<i>9</i>
<i>Classe AdminDAO.....</i>	<i>9</i>
<i>Classe AdminController.....</i>	<i>9</i>
<i>Fichier XML admin-view</i>	<i>10</i>
2.Interface Client.....	8
<i>Image de l'interface</i>	<i>8</i>
<i>Classe Client</i>	<i>9</i>
<i>Classe ClientDAO</i>	<i>9</i>
<i>Classe HelloController.....</i>	<i>9</i>
<i>Fichier XML hello-view</i>	<i>10</i>
3.Interface Produit	12
<i>Image de l'interface</i>	<i>12</i>
<i>Classe Produit.....</i>	<i>13</i>
<i>Classe ProduitDAO</i>	<i>13</i>
<i>Classe ProduitController</i>	<i>13</i>
<i>Fichier XML produit-view</i>	<i>14</i>
4.Interface Credit	15
<i>Image de l'interface</i>	<i>15</i>
<i>Classe Credit</i>	<i>16</i>
<i>Classe CreditDAO</i>	<i>16</i>
<i>Classe CreditController</i>	<i>16</i>
<i>Fichier XML credit-view.....</i>	<i>17</i>
5.Interface Commandes	18
<i>Image de l'interface</i>	<i>18</i>
<i>Classe Commandes</i>	<i>19</i>
<i>Classe CommandesDAO</i>	<i>19</i>
<i>Classe CommandesController</i>	<i>19</i>
<i>Fichier XML commande-view</i>	<i>20</i>
6.HelloApplication	22
Conclusion	23

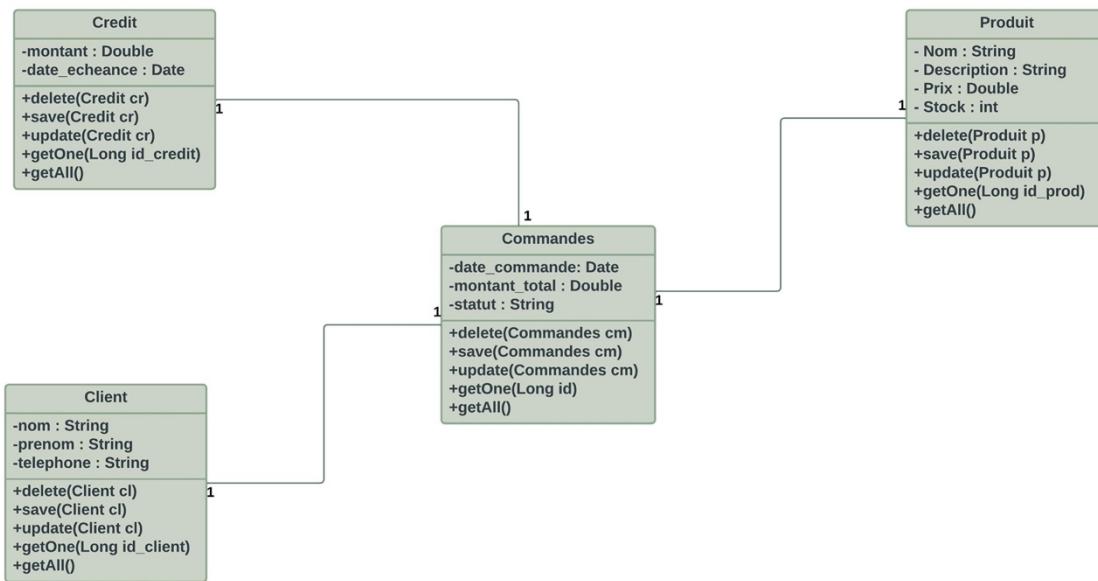
Introduction

Le développement d'applications de bureau est une tâche complexe qui nécessite l'utilisation des bons outils pour réussir. Dans ce projet, nous avons choisi d'utiliser JavaFX et JDBC pour développer une application qui permet la gestion des crédits, des commandes, des produits et des clients. La base de données MySQL sera utilisée pour stocker les données de l'application.

JavaFX est une technologie graphique moderne qui permet aux développeurs de créer des interfaces graphiques riches et esthétiquement attrayantes pour leurs applications de bureau. JDBC est une API Java standard qui permet aux applications Java d'accéder aux bases de données relationnelles. Ensemble, JavaFX et JDBC fournissent une plate-forme de développement puissante et efficace pour créer des applications de bureau. Dans ce rapport, je détaillerais les étapes nécessaires pour développer cette application à l'aide de ces outils. J'expliquerai également les différentes fonctionnalités de l'application et comment elles sont implémentées à l'aide de JavaFX, JDBC .

Plan de développement

Diagramme de classe:



Les clés des classes Credit , Client , Produit vont immigrer vers la classes commandes

Processus de développement

BASEDAO :

Pour travailler avec les tables de base de données, nous aurons besoin de BaseDAO .

Cette classe contient des interfaces JDBC telles que Connect, Statement, Prepare et ResultSet pour interagir avec la base de données. Dans le constructeur, une instance de connexion est créée à l'aide du pilote MySQL JDBC pour se connecter à la base de données "esalaf" avec l'URL, le nom d'utilisateur et le mot de passe spécifiés. Les méthodes abstraites sont utilisées pour effectuer des opérations sur la base de données, y compris l'ajout, la mise à jour ou la suppression d'enregistrements, l'obtention d'un seul enregistrement ou l'obtention de tous les enregistrements de la table respective. La méthode getOne renvoie un objet T avec l'ID spécifié, tandis que la méthode getAll renvoie une liste d'objets T

1.Interface Login

Avant d'accéder au interface il faut tous d'abord ce connecter autant qu'admin : donc la première interface est l'interface d'authentification l'admin a comme valeur {id=1,user=root,pwd=0}



Processus de développement

Classe Admin :

Classe appelée "Admin" qui a trois attributs privés : "id", "user" et "pwd". Les deux premiers sont de type String et le dernier est un entier. La classe dispose d'un constructeur par défaut et d'un autre constructeur qui prend en paramètres l'id, le nom d'utilisateur et le mot de passe. Elle possède également des méthodes pour accéder aux attributs "id", "user" et "pwd", et une méthode `toString()` pour afficher les valeurs des attributs.

Classe AdminDAO :

Classe "AdminDAO" qui hérite de "BaseDAO<Admin>" et qui est utilisée pour la communication avec la base de données. Elle a un constructeur par défaut qui appelle le constructeur de la classe parente. Cette classe implémente les méthodes de l'interface "BaseDAO" pour effectuer les opérations de base de données CRUD (Create, Read, Update, Delete). Cependant, ces méthodes ne sont pas implémentées car elles sont inutiles pour la classe. La méthode "connect" est utilisée pour vérifier si un utilisateur est un admin en vérifiant s'il existe dans la table "Admin" de la base de données. Elle prend en entrée un identifiant, un nom d'utilisateur et un mot de passe, et renvoie true si l'utilisateur est un admin, sinon false.

Classe AdminController :

Le contrôleur est responsable de la manipulation des événements déclenchés par l'utilisateur dans l'interface graphique de l'application, tels que la saisie de données ou le clic sur des boutons. Elle a trois champs texte "id", "user" et "pwd" qui permettent de saisir l'identifiant, le nom d'utilisateur et le mot de passe de l'administrateur. La méthode "onConnectButtonClick" est appelée lorsqu'un utilisateur clique sur le bouton de connexion. Elle crée une instance de la classe "AdminDAO" pour vérifier si l'utilisateur est connecté en appelant la méthode "connect" de cette classe. Si l'utilisateur est l'admin, la méthode charge une autre vue en utilisant

le fichier FXML "hello-view.fxml"(l'interface client). Sinon, elle affiche un message d'erreur

Fichier XML admin-view :

fichier FXML qui définit l'apparence de l' interface utilisateur pour l'application de d'authentification. L'objectif de cette interface utilisateur est de fournir une zone de connexion à un utilisateur, où il peut entrer son identifiant, son nom d'utilisateur et son mot de passe, puis appuyer sur le bouton "Connexion" pour se connecter à un système ou une application. Cette interface utilisateur est définie en utilisant des conteneurs VBox et des éléments d'interface utilisateur tels que des étiquettes, des champs de texte et des boutons

2.Interface client

Client

Nom:

Prenom:

Telephone:

Enregistrer **Mettre à jour** **Supprimer** **Obtenir** **Clear**

Liste des clients

ID	Nom	Prénom	Téléphone
1	Laadaili	Aya	0608824150
2	Dahiry	Badr	0608824150
3	Aissatene	Nouhaila	0618345678
4	Ibrahimi	Nour	0662774539
5	zert		345

Voir les produits **Voir les crédits** **Voir les commandes**

Processus de développement

Classe Client :

Utilisée pour stocker des informations sur un client, notamment son identifiant, son nom, son prénom et son numéro de téléphone, et pour fournir des méthodes pour accéder et modifier ces informations(getters, setters) qui seront utiliser par la classe ClientDAO

Classe ClientDAO :

Classe ClientDAO étend de la classe de base BaseDAO et fournit les méthodes pour interagir avec la base de données pour la table client. Le constructeur de la classe ClientDAO établit une connexion à la base de données via l'appel à la méthode super() qui appelle le constructeur de la classe parent BaseDAO qui configure la connexion à la base de données.

La méthode save insère un nouvel objet Client dans la table client de la base de données. La méthode update met à jour un objet Client existant dans la table client. La méthode delete supprime un objet Client existant de la table client.

Les méthodes getOne et getAll récupèrent les objets Client de la base de données. La méthode getOne retourne un seul objet Client avec un identifiant spécifié. La méthode getAll retourne une liste de tous les objets Client de la base de données.

Classe HelloController :

Le contrôleur est responsable de la manipulation des événements déclenchés par l'utilisateur dans l'interface graphique de l'application, tels que la saisie de données ou le clic sur des boutons.

Le contrôleur implémente l'interface Initializable, ce qui signifie que la méthode initialize () est appelée automatiquement après que la vue a été chargée.

Les annotations @FXML sont utilisées pour injecter les éléments de la vue dans le contrôleur. Les éléments de vue sont les champs texte nom, prenom, tele, mytab (TableView) et les colonnes de la table (TableColumn).

La méthode onSaveButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Sauvegarder" pour ajouter un nouveau client à la base de données. Elle crée un nouvel objet Client avec les valeurs entrées par l'utilisateur dans les champs de texte, puis l'enregistre dans la base de données en utilisant la méthode save() de la classe ClientDAO.

La méthode onUpdateButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Mettre à jour" pour modifier un client existant dans la base de données. Elle récupère l'objet Client sélectionné dans le tableau, met à jour ses propriétés avec les nouvelles valeurs entrées par l'utilisateur dans les champs de texte, puis utilise la méthode update() de la classe ClientDAO pour mettre à jour l'enregistrement correspondant dans la base de données.

La méthode onGetOneButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Obtenir". Elle récupère l'objet Client sélectionné dans le tableau et affiche ses propriétés dans les champs de texte correspondants.

La méthode onClearInput() est appelée lorsque l'utilisateur clique sur le bouton "Effacer". Elle efface les champs de texte pour permettre à l'utilisateur d'entrer de nouvelles données.

La méthode onDeleteButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Supprimer" pour supprimer un client existant dans la base de données. Elle récupère l'objet Client sélectionné dans le tableau et utilise la méthode delete() de la classe ClientDAO pour supprimer l'enregistrement correspondant de la base de données.

La méthode UpdateTable() est appelée pour mettre à jour le contenu de la table des clients.

La méthode getDataClients() récupère tous les clients dans la base de données en utilisant la classe ClientDAO et les stocke dans une liste ObservableList. Cette liste est retournée et utilisée pour remplir la table des clients.

Enfin, la méthode initialize () appelle la méthode UpdateTable () pour mettre à jour la table des clients lors du chargement de la vue.

Fichier XML hello-view :

Définit l' interface utilisateur graphique

La première section contient des champs de saisie pour le nom, le prénom et le téléphone du client. Les champs de saisie sont inclus dans une boîte de mise en page VBox. Chaque champ de saisie a un identifiant unique qui permet à la logique de l'application de récupérer la valeur de la saisie lorsque les boutons sont utilisés.

La deuxième section contient une série de boutons pour enregistrer, mettre à jour, supprimer, obtenir et effacer les données des clients. Les boutons sont inclus dans une autre boîte de mise en page VBox, et sont alignés horizontalement dans une boîte de mise en page HBox. Chaque bouton a un identifiant unique et est associé à une méthode définie dans le contrôleur qui est appelée lorsqu'un utilisateur clique sur le bouton.

La troisième section contient une zone de liste de clients qui est affichée dans un TableView. Les colonnes ont des identifiants uniques qui permettent à la logique de l'application de récupérer les données client pour les afficher et les modifier.

La dernière section contient trois boutons qui redirigent vers trois autres pages : une page pour visualiser les produits , une page pour visualiser les crédits et une page pour visualiser les commandes . Ces boutons ont également des identifiants uniques et sont associés à des méthodes définies dans le contrôleur qui sont appelées lorsqu'un utilisateur clique sur le bouton.

3.Interface Produit

Produits

Nom:	<input type="text"/>
Description:	<input type="text"/>
Prix:	<input type="text"/>
Stock:	<input type="text"/>

Enregistrer **Mettre à jour** **Supprimer** **Obtenir** **Clear**

Liste des produits

ID	Nom	Description	Prix	Stock
1	Produit 1	Description du produit 1	10.99	50
2	Produit 2	Description du produit 2	24.5	25
3	produit3	description du produit 3	34.0	5667
4	Produit 4	Description du produit 4	8.75	75
5	Produit 5	Description du produit 5	15.0	30
8	Produitx	Description du produit x	15.6	234

Voir les crédits **Voir les clients** **Voir les commandes**

Processus de développement

Classe Produit :

Utilisée pour stocker les informations sur des produits , notamment son identifiant, son nom, sa description ,son prix et son stock, et pour fournir les méthodes pour accéder et modifier ces informations(getters, setters) qui seront utiliser par la classe ProduitDAO

Classe ProduitDAO :

La classe ProduitDAO est une sous-classe de la classe générique BaseDAO qui gère la persistance des objets de la classe Produit. Cette classe implémente les méthodes pour effectuer des opérations CRUD (Create, Read, Update, Delete) sur la table produit dans la base de données essalaf.

La méthode save permet d'ajouter un nouveau produit dans la base de données, la méthode update permet de mettre à jour les informations d'un produit existant, et la méthode delete permet de supprimer un produit de la base de données. La méthode getOne permet de récupérer un objet Produit à partir de son identifiant (id_prod).

La méthode getAll permet de récupérer la liste de tous les produits dans la base de données. Cette méthode utilise une requête SQL pour récupérer les informations de chaque produit et les stocke dans une liste de type List<Produit>.

Classe ProduitContrôleur :

Le contrôleur utilise la classe ProduitDAO pour communiquer avec la base de données et effectuer des opérations CRUD (créer, lire, mettre à jour, supprimer) sur les produits.

A l'aide de la classe PropertyValueFactory on définit des éléments de l'interface utilisateur, tels que des champs de texte et un tableau de produits, qui sont liés à des propriétés de la classe Produit .

Egalement des méthodes qui sont appelées lorsque l'utilisateur interagit avec l'interface, telles que la méthode onSaveButtonClick() qui crée un nouveau produit à partir des données saisies dans les

champs de texte et l'ajoute à la base de données à l'aide de ProduitDAO.

La méthode UpdateTable() met à jour le tableau de produits avec les données de la base de données à l'aide de la méthode getDataProduits(), qui renvoie une liste observable d'objets Produit. Les méthodes onViewCreditsButtonClick(), onViewClientsButtonClick(), et onViewCommandesButtonClick() sont utilisées pour ouvrir des fenêtres supplémentaires qui affichent des informations sur les crédits, les clients et les commandes. Les annotations @FXML permettent à JavaFX de lier les éléments de l'interface utilisateur définis dans le fichier FXML à des champs et des méthodes de la classe de contrôleur.

Fichier XML produit-view :

Fichier FXML utilisé pour créer l'interface utilisateur graphique. Le fichier FXML contient une VBox principale qui définit la fenêtre principale. Cette VBox contient deux éléments: une VBox pour saisir les informations du produit et une HBox pour les boutons d'action. La VBox pour saisir les informations du produit contient plusieurs Label et TextField pour chaque champ d'informations.

La HBox pour les boutons d'action contient plusieurs boutons qui sont liés à des méthodes spécifiques de la classe de contrôleur ProduitController via l'attribut fx:onAction.

Une ScrollView qui contient une VBox contenant un TableView. Le TableView est rempli dynamiquement en fonction des données de la base de données. Les colonnes de la table sont également définies dans le fichier FXML en utilisant des TableColumn.

Le fichier FXML se termine par une Hbox qui contient trois boutons qui redirigent vers trois autres pages : une page pour visualiser les clients , une page pour visualiser les crédits et une page pour visualiser les commandes . Ces boutons ont également des identifiants uniques et sont associés à des méthodes définies dans le contrôleur qui sont appelées lorsqu'un utilisateur clique sur le bouton.

4.Interface crédit

Credits

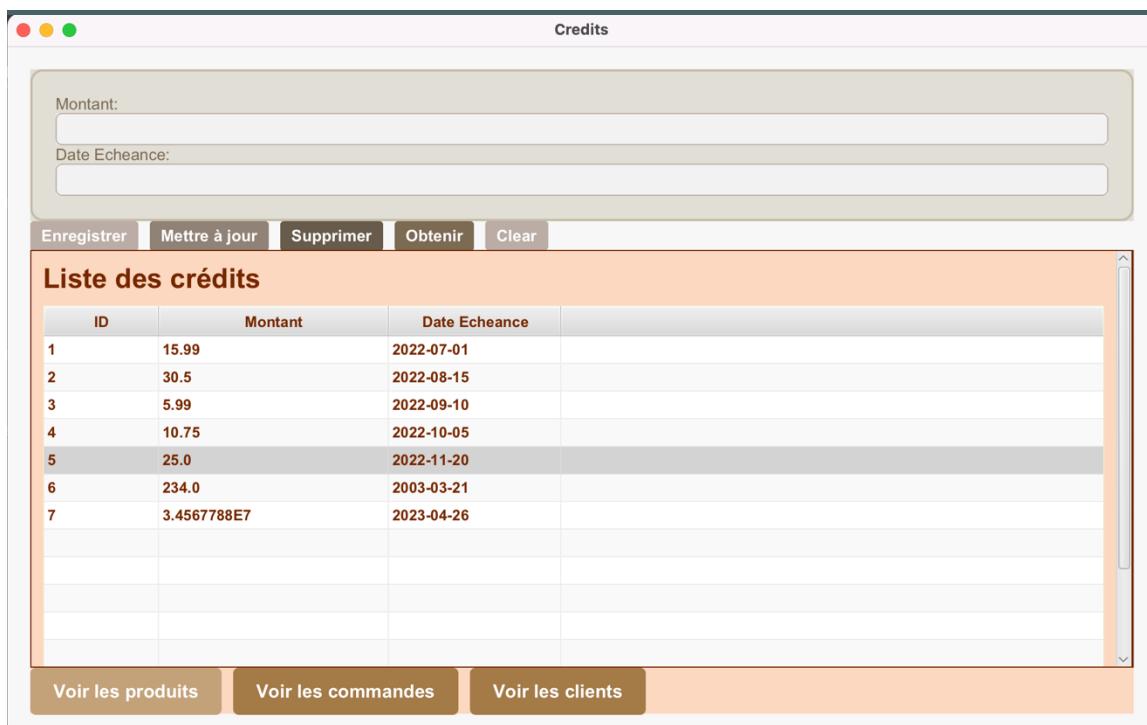
ID	Montant	Date Echeance
1	15.99	2022-07-01
2	30.5	2022-08-15
3	5.99	2022-09-10
4	10.75	2022-10-05
5	25.0	2022-11-20
6	234.0	2003-03-21
7	3.4567788E7	2023-04-26

Montant:
Date Echeance:

Enregistrer Mettre à jour Supprimer Obtenir Clear

Liste des crédits

Voir les produits Voir les commandes Voir les clients



Processus de développement

Classe Crédit :

Utilisée pour stocker les informations sur des crédits , notamment son identifiant, son montant, sa date d'échéance et pour fournir les méthodes pour accéder et modifier ces informations(getters, setters) qui seront utiliser par la classe CreditDAO

Classe CreditDAO :

La classe "CreditDAO" hérite de la classe abstraite "BaseDAO" qui fournit une connexion à la base de données et des méthodes pour l'interaction avec la base de données telles que "save", "update", "delete", "getOne" et "getAll" qui seront utilisée dans le contrôleur. La méthode "save" est utilisée pour insérer un nouvel objet "Credit" dans la base de données. La méthode "update" est utilisée pour mettre à jour un objet "Credit" existant dans la base de données. La méthode "delete" est utilisée pour supprimer un objet "Credit" de la base de données. La méthode "getOne" est utilisée pour récupérer un objet "Credit" en fonction de son identifiant. La méthode "getAll" est utilisée pour récupérer tous les objets "Credit" de la base de données. Chaque méthode utilise des requêtes SQL pour effectuer des opérations sur la base de données, telles que la sélection, l'insertion, la mise à jour et la suppression de données.

Classe CreditContrôleur :

Contrôleur de vue pour une interface utilisateur graphique (GUI) qui permet de gérer les crédits. comporte plusieurs méthodes qui sont appelées lorsque l'utilisateur interagit avec les différents éléments de la GUI.

La méthode UpdateTable() met à jour le tableau des crédits avec les données récupérées de la base de données en utilisant la classe CreditDAO. La méthode getDataCredits() récupère les données de tous les crédits en utilisant la même classe CreditDAO.

La méthode onSaveButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Sauvegarder" pour ajouter un nouveau crédit à la base de données. Elle crée un nouvel objet Credit avec les valeurs entrées par l'utilisateur dans les champs de texte, puis l'enregistre

dans la base de données en utilisant la méthode save() de la classe CreditDAO.

La méthode onUpdateButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Mettre à jour" pour modifier un crédit existant dans la base de données. Elle récupère l'objet Credit sélectionné dans le tableau, met à jour ses propriétés avec les nouvelles valeurs entrées par l'utilisateur dans les champs de texte, puis utilise la méthode update() de la classe CreditDAO pour mettre à jour l'enregistrement correspondant dans la base de données.

La méthode onGetOneButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Obtenir". Elle récupère l'objet Credit sélectionné dans le tableau et affiche ses propriétés dans les champs de texte correspondants.

La méthode onClearInput() est appelée lorsque l'utilisateur clique sur le bouton "Effacer". Elle efface les champs de texte pour permettre à l'utilisateur d'entrer de nouvelles données.

La méthode onDeleteButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Supprimer" pour supprimer un crédit existant dans la base de données. Elle récupère l'objet Credit sélectionné dans le tableau et utilise la méthode delete() de la classe CreditDAO pour supprimer l'enregistrement correspondant de la base de données.

Enfin, les méthodes onViewProductsButtonClick(),
onViewClientsButtonClick() et onViewCommandesButtonClick() sont appelées lorsque l'utilisateur clique sur les boutons "Produits",
"Clients" et "Commandes" respectivement pour afficher des interfaces utilisateur graphiques différentes correspondant à chaque bouton.

Fichier XML credit-view :

Fichier FXML utilisé pour créer l'interface graphique de l'application. Il définit un formulaire de crédit avec des champs de saisie pour le montant et la date d'échéance, ainsi que des boutons pour enregistrer, mettre à jour, supprimer, obtenir et effacer les données de crédit. Le formulaire est affiché dans une boîte de dialogue avec une bordure de style et une couleur de fond. Le code comprend également un tableau pour afficher une liste de crédits existants, ainsi que des boutons pour afficher des produits et des commandes..

5.Interface commandes

Commandes

Id_client:	
Id_prod:	
Id_credit:	
Date commande:	
Montant:	
Statut:	

Enregistrer **Mettre à jour** **Supprimer** **Obtenir** **Clear**

Liste des commandes

ID	ID_CLIENT	ID_PROD	ID_CREDIT	DATE COMMANDE	MONTANT TOTALE	STATUT
2	2	3	5	2022-02-15	60.5	Expédiée
3	3	3	3	2022-03-20	345.89	Payée
4	4	5	2	2022-04-05	22.75	En cours
11	25	105	105	2022-03-20	345.89	Payée
12	1	1	1	2023-06-19	345.87	Payée
13	5	2	4	2021-04-30	495.05	Expédiée

Voir les produits **Voir les crédits** **Voir les clients**

Processus de développement

Classe Commandes :

Utilisée pour stocker les informations sur les commandes, notamment son identifiant, identifiant client, identifiant produit, identifiant crédit , sa date , son montant et son statut et pour fournir les méthodes pour accéder et modifier ces informations(getters, setters) qui seront utiliser par la classe CommandesDAO

Classe CommandesDAO :

La classe "CommandesDAO" hérite de la classe abstraite "BaseDAO" qui fournit une connexion à la base de données et des méthodes pour l'interaction avec la base de données telles que "save", "update", "delete", "getOne" et "getAll" qui seront utilisée dans le contrôleur. La méthode "save" est utilisée pour insérer un nouvel objet "Commandes" dans la base de données. La méthode "update" est utilisée pour mettre à jour un objet "Commandes" existant dans la base de données.

La méthode "delete" est utilisée pour supprimer un objet "Commandes" de la base de données. La méthode "getOne" est utilisée pour récupérer un objet "Commandes" en fonction de son identifiant.

La méthode "getAll" est utilisée pour récupérer tous les objets "Commandes" de la base de données.

Chaque méthode utilise des requêtes SQL pour effectuer des opérations sur la base de données, telles que la sélection, l'insertion, la mise à jour et la suppression de données.

Classe CommandesContrôleur :

Contrôleur de vue pour une interface utilisateur graphique (GUI) qui permet de gérer les crédits. comporte plusieurs méthodes qui sont appelées lorsque l'utilisateur interagit avec les différents éléments de la GUI.

La méthode UpdateTable() met à jour le tableau des crédits avec les données récupérées de la base de données en utilisant la classe CommandesDAO. La méthode getDataCommandes() récupère les données de tous les crédits en utilisant la même classe CommandesDAO.

La méthode onSaveButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Sauvegarder" pour ajouter une nouvelle commande à la base de données. Elle crée un nouvel objet Commandes avec les valeurs entrées par l'utilisateur dans les champs de texte, puis l'enregistre dans la base de données en utilisant la méthode save() de la classe CommandesDAO.

La méthode onUpdateButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Mettre à jour" pour modifier une commande existante dans la base de données. Elle récupère l'objet Commande sélectionné dans le tableau, met à jour ses propriétés avec les nouvelles valeurs entrées par l'utilisateur dans les champs de texte, puis utilise la méthode update() de la classe CommandesDAO pour mettre à jour l'enregistrement correspondant dans la base de données.

La méthode onGetOneButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Obtenir". Elle récupère l'objet Commande sélectionné dans le tableau et affiche ses propriétés dans les champs de texte correspondants.

La méthode onClearInput() est appelée lorsque l'utilisateur clique sur le bouton "Effacer". Elle efface les champs de texte pour permettre à l'utilisateur d'entrer de nouvelles données.

La méthode onDeleteButtonClick() est appelée lorsque l'utilisateur clique sur le bouton "Supprimer" pour supprimer une commande existante dans la base de données. Elle récupère l'objet Commandes sélectionné dans le tableau et utilise la méthode delete() de la classe CreditDAO pour supprimer l'enregistrement correspondant de la base de données.

Enfin, les méthodes onViewProductsButtonClick(), onViewClientsButtonClick() et onViewCreditsButtonClick() sont appelées lorsque l'utilisateur clique sur les boutons "Produits", "Clients" et "Credits" respectivement pour afficher des interfaces utilisateur graphiques différentes correspondant à chaque bouton.

Ficher XML commande-view:

Le fichier commence par des instructions XML définissant la version et l'encodage du fichier. Ensuite, il y a des instructions d'importation pour les classes JavaFX nécessaires à l'interface graphique.

Le contenu principal du fichier FXML est organisé dans un conteneur VBox. Ce conteneur a une hauteur et une largeur préférées, et il a une couleur de fond et une police de caractères définies dans ses styles en ligne.

Le conteneur VBox contient deux autres conteneurs VBox et un conteneur ScrollPane. Le premier conteneur VBox contient plusieurs éléments Label et TextField pour saisir des informations de commande. Le deuxième conteneur VBox contient plusieurs boutons organisés dans une boîte HBox. Le conteneur ScrollPane contient un TableView, qui sera utilisé pour afficher une liste de commandes.

6.HelloApplication

Une classe Java qui étend la classe Application de JavaFX et permet de lancer une application JavaFX en utilisant la vue (fichier FXML) "admin-view.fxml". Le code utilise la méthode "start" de la classe Application, qui charge la vue "admin-view.fxml" en utilisant l'objet FXMLLoader, crée une scène avec cette vue, définit le titre de la fenêtre et affiche la fenêtre en utilisant la méthode "show" de la classe Stage. La méthode "main" de la classe HelloApplication lance l'application JavaFX en appelant la méthode "launch" de la classe Application.

Conclusion

En conclusion, la réalisation de l'application desktop Java Esalaf a permis de mettre en place une solution efficace de gestion des crédits, des commandes, des produits et des clients. Grâce à l'utilisation de JavaFX et JDBC, l'interface utilisateur est intuitive et facile à utiliser, tandis que la base de données MySQL offre une grande fiabilité et une sécurité accrue. Ce projet m'a permis de développer les compétences en programmation Java, en développement d'applications desktop et en gestion de bases de données, tout en offrant une solution pratique pour la gestion de ce projet.