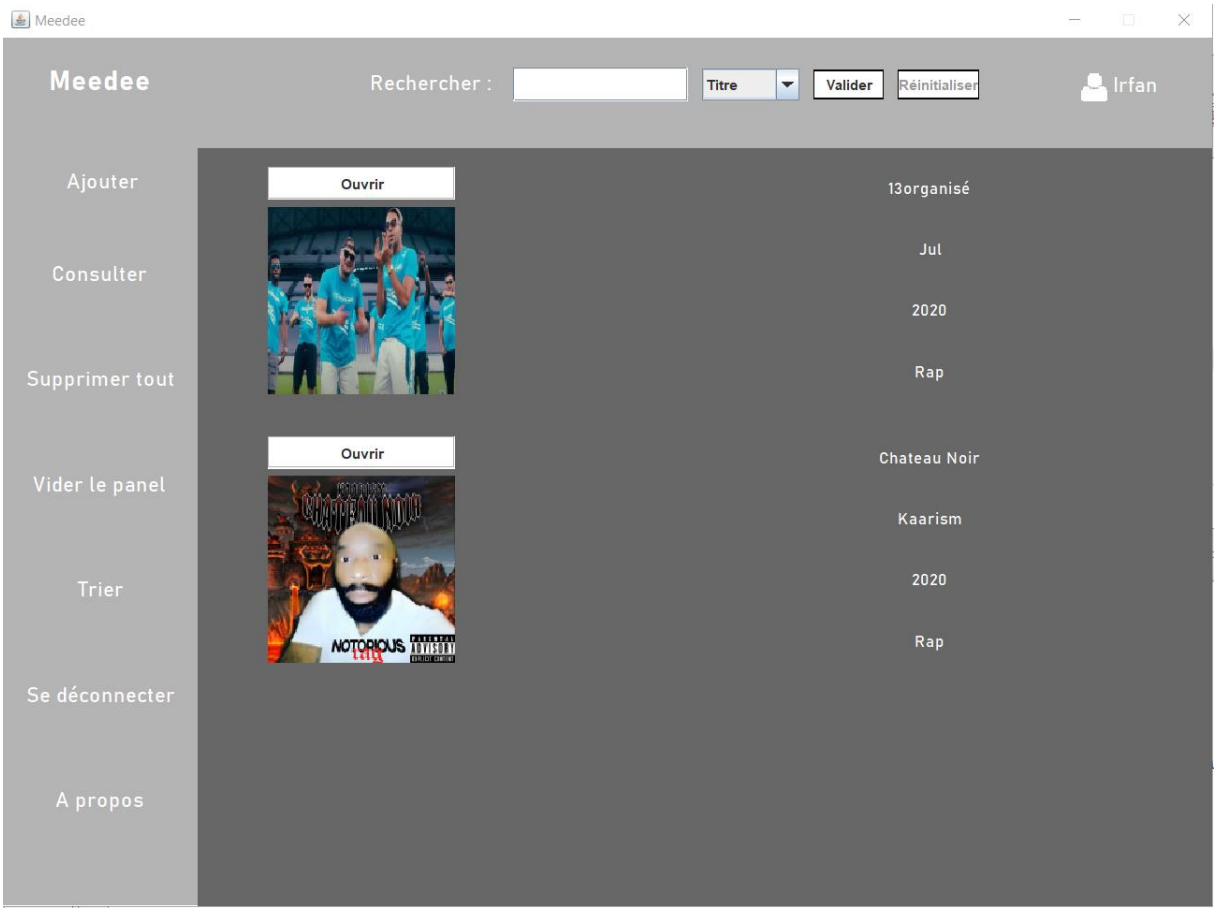
PROJET JAVA 2020-2021

Sujet 11 : Gestionnaire d'Albums de Musique

Professeur : M. Sébastien THON



BOUHENAF Irfan – MOLINA Romain

IUT Aix-Marseille site d'Arles

SOMMAIRE

INTRODUCTION

Dans le cadre de la deuxième année du DUT Informatique, nous devons réaliser une application en Java, on devait choisir un des 11 sujets disponibles.

Nous avons donc choisi le gestionnaire d'album de musiques car ce sujet nous intéressait.

Nous devons aussi utiliser une base de données afin de faire fonctionner notre application

I- Analyse

A) Classes utilisées

Pour la réalisation de ce projet nous avons utilisé plusieurs classes ayant chacune un rôle bien précis que sont :

- Connexion : Formulaire pour se connecter à l'application - hérite de JDialog

- LigneAlbum : classe représentant un album de musique - hérite de JPanel

- Fenetre : La fenêtre principale - hérite de JFrame

- Formulaire : Classe abstraite servant aux formulaires – hérite de JDialog

- FormulaireAjout : Formulaire pour ajouter un album – hérite de Formulaire

- FormulaireModifier : Formulaire pour modifier un album – hérite de Formulaire

- FenetreTri : Boîte de dialogue servant à trier les albums – hérite de JDialog

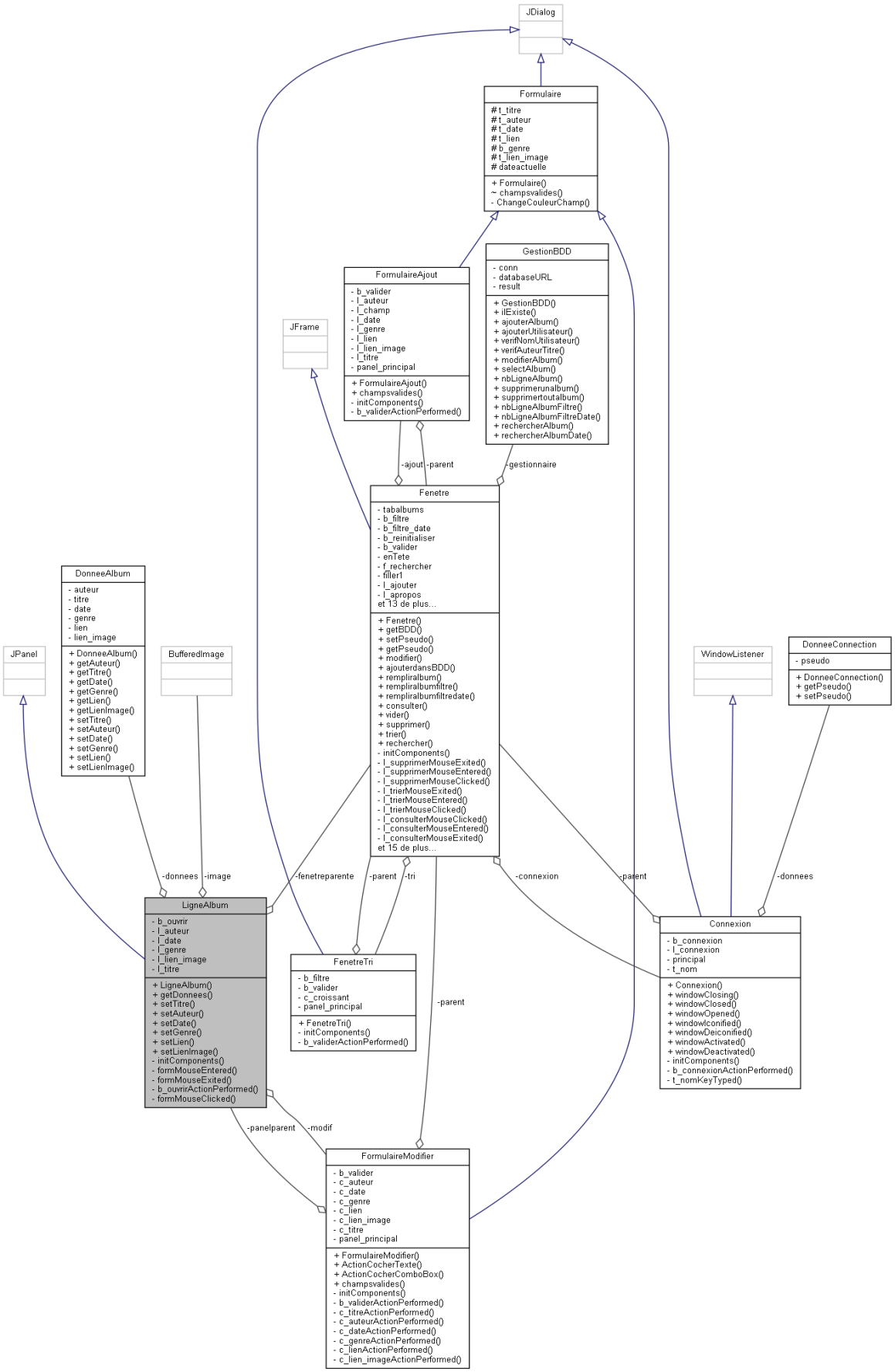
- GestionBDD : Gestionnaire de la base de données

- DonneeConnexion : Classe servant à stocker les données de connexion

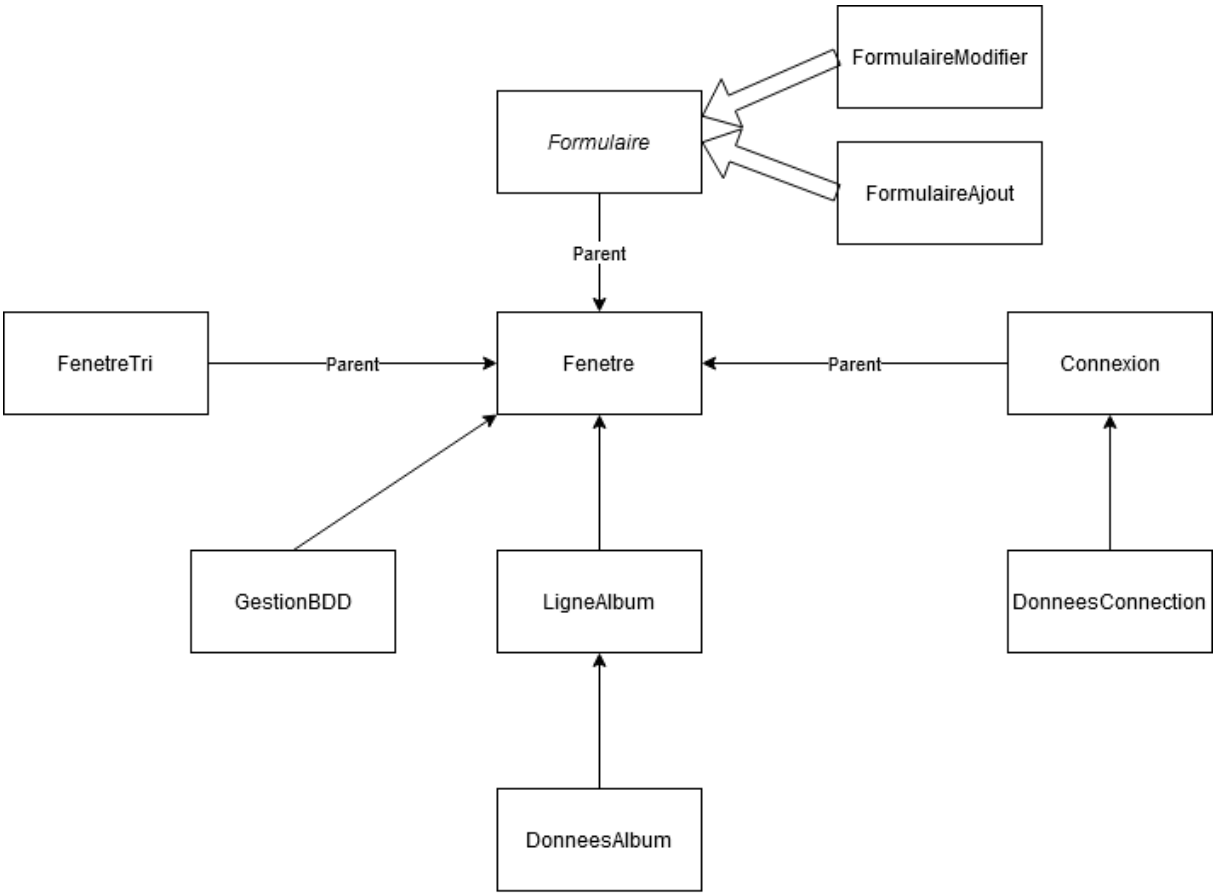
- DonneeAlbum : Classe servant à stocker les données des albums de musique

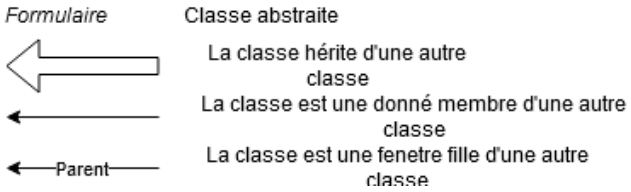
- Main : Classe où il y a la méthode main, cette dernière est obligatoire dans un programme pour qu'une application fonctionne bien

B) Diagramme des classes



C) Relation entre les classes

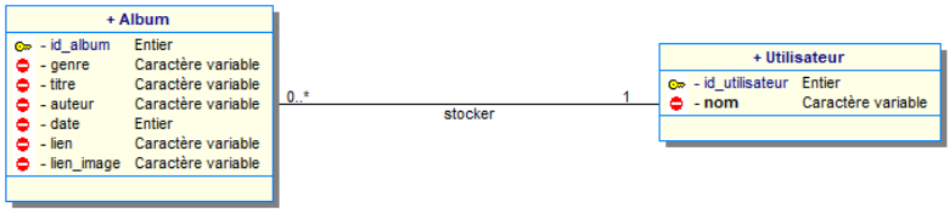




D) La base de données

Pour que notre application puisse fonctionner correctement, nous avons utilisé une base de données JDBC avec le moteur SQLite. Contrairement aux serveurs de bases de données traditionnels comme MySQL ou PostgreSQL, il n'utilise pas le schéma habituel client-serveur mais il est directement intégré aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier.

Voici le diagramme UML de la base de données :



Le schéma relationnel de la BDD est :

Album(id_album,genre,titre,auteur,date,lien,lien_image,#id_utilisateur)

Utilisateur(id_utilisateur,nom)

D'après ce diagramme, un utilisateur stocke 0 ou plusieurs albums et un album est stocké par un utilisateur.

Une base de données pour ce genre d'application est utile notamment pour sauvegarder des données notamment en cas de fermeture de l'application.

E) Fonctionnement global

La classe « Main » contient la méthode main où dedans, il y a une instance de la classe Fenetre, sans ça, l'application ne fonctionnerait pas.

La classe « Connexion » sert à faire connecter l'utilisateur à l'application, pour ce faire il vérifie si le pseudo existe dans la base de données.
S'il existe, il utilise ce pseudo pour se connecter sinon il ajoute le pseudo dans la BDD et se connecte avec.

La classe « Fenetre » est la fenêtre principale de l'application et cette classe permet d'afficher l'application et à gérer l'affichage des albums. Pour ce faire nous avons utilisé un Vector de LigneAlbum pour manipuler l'affichage des albums dans un JPanel

Pour ajouter un album dans la base de données, nous avons utilisé la classe de « FormulaireAjout » héritant de « Formulaire » qui est une classe abstraite, nous avons crée une classe abstraite afin d'éviter la redondance des données. Grâce aux différents champs remplis, un album sera ajouté dans la base de données et une boîte de dialogue de type JOptionPane apparaîtra pour notifier l'utilisateur que l'album a été ajouté avec succès si l'ajout de l'album de données a été un

succès. Dans le cas contraire, les champs incorrects seront affichés en rouge et l'utilisateur devra les corriger

Pour modifier un album, nous avons utilisé la classe « FormulaireModifier » héritant de « Formulaire », comme la classe « FormulaireAjout », Grâce aux différents champs remplis, un album sera modifié dans la base de données et dans l'application. Dans le cas contraire, les champs incorrects seront affichés en rouge et l'utilisateur devra corriger les champs incorrects.

Pour trier tous les albums, nous avons utilisé la classe « FenetreTri », grâce à une JComboBox, l'utilisateur peut choisir quel champ sera trié (titre, auteur, année, genre) et avec une JCheckBox, l'utilisateur peut trier dans l'ordre croissant ou décroissant, en appuyant sur le bouton « Valider », les albums seront triés dans la fenêtre.

La classe « LigneAlbum » est un JPanel contenant des JLabel où ces derniers seront remplis grâces aux données de la BDD (titre, auteur, image, genre, année etc..) et cette classe utilise les données de la classe DonneeAlbum pour remplir les JLabel. La classe LigneAlbum utilise la fenêtre principale pour désallouer une instance quand on veut supprimer un album dans l'application mais aussi quand il faut spécifier un parent pour une instance de la classe FormulaireModifier par exemple

La classe « DonneeConnection » transmet le pseudo de l'utilisateur à la fenêtre lorsque l'utilisateur réussit à se connecter à l'application

La classe « GestionBDD » est le gestionnaire de la base de données, c'est ici où il y a une connexion entre l'application et la base de données et inversement mais aussi, il y a toutes les requêtes SQL utiles pour l'application.

Vu que la fenêtre principale gère les albums dans l'application, elle utilise le gestionnaire de BDD pour gérer les albums grâce à des méthodes et le formulaire d'ajout utilise une de ces méthodes pour ajouter l'album dans la BDD et le formulaire de modification utilise aussi une de ces méthodes pour modifier un album dans la base de données et dans l'application.

II - Réalisation

A) Choix techniques

Tout d'abord, nous avons décidé de pouvoir gérer les albums de musique uniquement et pas leurs musiques afin que l'application soit plus simple d'utilisation. Aussi, le placement des éléments de l'application s'est fait à l'aide de l'IDE NetBeans mais aussi « à la main » c'est-à-dire que certains éléments de l'application sont placés manuellement car il y a des contraintes de placement avec NetBeans

Nous avons décidé de stocker les données de connexion et des albums dans des classes séparés pour pouvoir utiliser ces classes dans d'autres scripts par exemple

Cependant, le traitement des informations et l'affichage de ces derniers sont mélangés avec NetBeans, ce sont les fenêtres qui gèrent les évènements utilisateurs.

B) Présentations de certains algorithmes « complexes »

Nous allons présenter les différents algorithmes permettant de consulter les albums contenus dans la BDD directement dans l'application :

C'est cette méthode ci-dessous contenue dans la classe Fenetre qui permet de consulter les albums contenus dans la base de données

Méthode consulter() :

```
/**
 * Cette fonction permet de consulter les albums contenus dans la base de données
 */
public void consulter()
{
    rempliralbum();
    gestionnaire.selectAlbum(donnees.getPseudo(),tabalbums);
    panel_tableau.revalidate();
    panel_tableau.repaint();
}
```

Méthode rempliralbum() :

```
public void rempliralbum()
{
    if (gestionnaire.nbLigneAlbum(donnees.getPseudo()) == 0)
    {
        JOptionPane.showMessageDialog(this,"La recherche n'a pas aboutie");
    }
    else
    {
        tabalbums.clear();
        panel_tableau.removeAll();
        for (int i=0; i < gestionnaire.nbLigneAlbum(donnees.getPseudo()); i++)
        {
            tabalbums.addElement(new LigneAlbum(this));
            panel_tableau.add(tabalbums.lastElement());
        }
    }
}
```

Tout d'abord, en fonction du nombre de tuples (ligne dans une BDD) contenues dans la table « Album » de la BDD, un certain nombre de panel d'albums seront instanciés dans le Vector d'albums (s'il y a 45 tuples contenus dans la table « Album », alors 45 Albums seront instanciés), le tableau et le panel sont vidés avant l'allocation dynamique d'albums pour éviter que si on avait déjà consulté des albums, un même album ne sera pas affiché plusieurs fois.

La méthode nbLigneAlbum permet de compter le nombre de tuples contenu dans la table « Album » et retourner ce nombre.

Si aucun tuple est contenu dans la table « Album », alors un message d'erreur sera affiché grâce à la méthode statique de la classe showMessageDialog de la classe JOptionPane

On ajoute ensuite un album dans le panel contenant les albums à chaque itération.

Méthode selectAlbum(String nom, Vector<LigneAlbum> tab) :

Cette méthode permet de récupérer les valeurs des tuples dans base de données à l'aide d'une requête SQL et les stocker dans les instances de la classe DonneeAlbum et cette dernière permet de stocker les données d'un album. Ces données seront utilisés par les albums pour les afficher dans l'application.

A chaque modification du JPanel contenant les albums, on rafraîchit ce dernier à l'aide des méthodes revalidate() et repaint()

C) Format de fichier de données crée

Nous avons aussi créé un fichier .db pour utiliser SQLite et nous avons aussi téléchargé un driver JDBC qui permet de faire fonctionner la BDD

Des fichiers .form sont aussi crée pour permettre d'éditer l'application grâce à l'IDE NetBeans

III - Utilisation

A) Mode d'emploi

On peut ajouter un album en cliquant sur « Ajouter », on peut consulter la liste des albums d'un utilisateur qui sont stockés dans la base de données en cliquant sur « Consulter ». En cliquant sur « Supprimer tout » on supprime tous les albums d'un utilisateur dans la base de données et dans l'application, à la différence de « vider dans le panel » qui lui permet de supprimer les albums d'un utilisateur uniquement dans l'application. En cliquant sur « trier » on ouvre une fenêtre permettant de trier les albums de musique en fonction du genre, du nom de l'album, du nom de l'auteur, de la date de sortie, et du genre. On peut se déconnecter en cliquant sur le bouton « Se déconnecter », on peut avoir le nom des créateurs de l'application en cliquant sur le bouton « A propos ».

Tout en haut il y a une barre de rechercher permettant de faire une recherche, on peut choisir le type de rechercher (par titre, auteur, date, genre). En faisant clique gauche sur un album on peut modifier des informations à son sujet, en faisant clique droite dessus on le supprime.

Dans l'ordre naturel des choses pour une première utilisation, on se connecte, on ajoute un album à l'aide du bouton « Ajouter » et on clique ensuite sur consulter pour voir la liste de ses albums. A partir de là le plus dur est fait et maintenant il suffit d'ajouter tous vos albums favoris dans l'application

Pour écouter un album, il faut appuyer sur le bouton « Ouvrir » dans l'Album, ça va ouvrir un Lien YouTube qu'on a spécifié (S'il n'y a pas de lien, le bouton sera grisé)

B) Configuration Requise

Pour faire fonctionner le logiciel il est nécessaire d'avoir Java et de préférence Java 15 mais aussi une connexion internet

C) Mode d'emploi de la documentation

Nous avons prévu deux types de documentations pour ce projet java.

Il y a d'abord la documentation style javadoc que l'on peut créer en exécutant la commande dans le répertoire PROJET_JAVA_BOUHENAF_MOLINA « javadoc -encoding utf8 -docencoding utf8 -charset utf8 -d doc src/*.java » cela va générer un dossier doc

Les paramètres encoding et doencoding permettent d'afficher les accents/caractères spéciaux sur le site généré par la documentation, ce qui est quand même plus agréable

Le deuxième style de documentation est le style fait par Doxygen, il y a encore pour cette documentation deux méthodes possibles :

1- Vous possédez le logiciel DoxyWizard, auquel cas il vous suffit juste de faire File->Open et de sélectionner le fichier nommé « Doxyfile » dans le répertoire src, vous allez ensuite dans l'onglet « Run » et cliquez sur « Run doxygen », une fois le chargement finis, cliquez sur « Show HTML output »
2- Vous ne possédez pas « DoxyWizard », dans ce cas suivez le tutoriel pour installer Doxygen (https://www.doxygen.nl/manual/install.html), une fois ceci fait, allez dans le répertoire src, ouvrez un invité de commande et tapez « doxygen <config_file> » (config_file étant le nom du fichier de configuration qui est dans notre cas « Doxyfile », vous avez maintenant un dossier « html », vous pouvez ouvrir n'importe qu'elle page html de ce dossier mais il est préférable de commencer par le début en ouvrant « index.html ».

   Si vous ne possédez pas DoxyWizard et que vous ne voulez pas installer Doxygen sur votre machine, pas d'inquiétude, nous fournirons la documentation en dur séparément du projet (il y a de toute façon la javadoc que vous pourrez générer tout seul si vous possédez java).

IV - Conclusion

A) Bilan

Nous ne conseillerons pas NetBeans comme IDE Java. En effet, il y a des contraintes comme le placement des éléments, la gestion des layouts et la difficulté d'utilisation de cet IDE.

Au début du développement de ce projet, nous avions galéré à bien développer notre projet car notre base de données ne fonctionnait pas et ça a été un calvaire pour la faire fonctionner et nous avions mis une dizaine de jours pour bien utiliser NetBeans.

Cependant, le développement de ce projet a été amusant.

B) Optimisation possibles

On pourrait séparer le stockage de données, la partie visible de l'interface graphique et le traitement des actions de l'utilisateur afin d'optimiser le code.

On pourrait aussi améliorer la recherche d'albums, par exemple, on pourrait faire une recherche approximative ou bien précise

On pourrait aussi faire une classe à part qui s'occupe de la gestion d'albums dans l'application comme ça la fenêtre principale s'occupe juste d'afficher les éléments de l'application

On pourrait préremplir les champs quand on modifie l'album

On pourrait aussi stocker les musiques

C) Extensions possibles

On pourrait penser pouvoir mettre l'application sur mobile, rajouter un mot de passe pour chaque utilisateur pour mieux respecter la privacité. On peut aussi rajouter l'option de directement écoute l'album sur l'application en téléchargeant le mp3 à partir du lien donné