

Gestion d'albums de musique

Conception et programmation objet avancées



Groupe :

TRONC Clément

DI PIAZZA Hugo

Table des matières

Introduction.....	3
Analyse	3
Classes utilisées	3
Relations entre les classes.....	3
Fonctionnement global	5
Réalisé	5
Choix techniques	5
Présentation de certains algorithmes "complexes" utilisés	6
La BDD en SQLite.....	6
Utilisation	6
Mode d'emploi	6
Configurations requises	8
Conclusion	8
Bilan	8
Optimisations possibles	9
Extensions possibles	9

Introduction

Sujet : Gestion d'albums de musique

Ce logiciel permettra de gérer une collection d'albums de musique. Il devra être possible d'ajouter un album, d'en supprimer, de modifier, de consulter, et de sauvegarder. Il devra être possible d'effectuer une recherche par titre, par artiste, par date, par genre. Lors de la consultation, on pourra trier les résultats par titre d'albums, par artiste, par date, par genre. On utilisera une base de données locale avec JDBC.

On devra donc créer une bibliothèque d'album de musique visuel qui permet de stocker ses albums préférés. Le tout lié à une base de données couplé au Java.

Analyse

Classes utilisées

Nos Classes

Principal : La classe main qui permet de lancer le programme. Il lance une instance de FenetreGraphique qui permet de lancer le programme graphiquement.

FenetreGraphique : La fenêtre principal du programme. Qui rassemble l'ensemble des éléments et gère les éléments tel que la bar de recherche ou la barre à gauche. Elle instancie aussi Montrage est l'élément central du programme.

Ajouter : Formulaire permettant l'adhésion de nouveau album dans le programme et la base de données.

Montrage : Classe centré autour du CardLayout pour afficher un ensemble d'album et voir aussi la présentation individuelle d'un album.

Modifier : Formulaire permettant la modification d'un album déjà existant.

Bibliothèque : La classe qui rassemble les fonctions de requêtes SQL ainsi que la fonction de création des Albums de musique.

BDDConnection : La classe qui établit la connexion entre l'application et la base de données JDBC.

Librairie

JDBC : rend possible l'utilisation de base de données en SQLite

Relations entre les classes

Schéma simplifié de la relation des classes entres elles

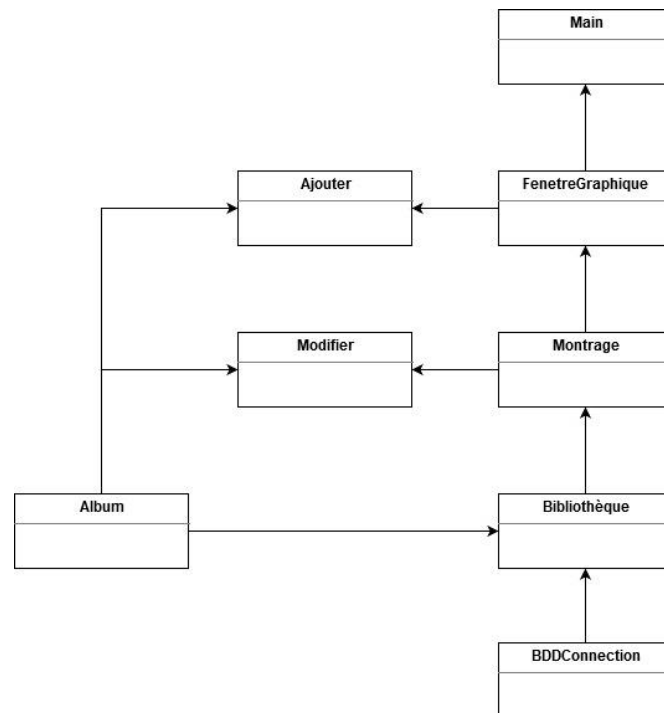


Diagramme UML

```

classDiagram
    class Principal {
        +main()
    }
    class ActionListener {
        +ActionListener()
    }
    class JFrame {
        +JFrame()
    }
    class JDialog {
        +JDialog()
    }
    class JPanel {
        +JPanel()
    }
    class FenetreGraphique {
        +fenetre : JFrame
        +Ajouter : JButton
        +BarRecherche : JTextField
        +Control : JPanel
        +PanelMenu : JPanel
        +PanelRecherche : JPanel
        +Regroup : JPanel
        +Retour : JButton
        +Valider : JButton
        +AjouterUnAlbum()
        +actionPerformed()
        +FenetreGraphique()
        +TransformButton()
    }
    class Modifier {
        +Valider : JButton
        +fenetre_ajouter : JFrame
        +tdAnnée : JTextField
        +tdArtiste : JTextField
        +tdDuree : JTextField
        +tdGenre : JComboBox
        +tdImage : JTextField
        +tdRecherche : JTextField
        +tdTitre : JTextField
        +Modifier()
        +actionPerformed()
        +testImage()
    }
    class Ajouter {
        +Valider : JButton
        +fenetre_ajouter : JFrame
        +tdAnnée : JTextField
        +tdArtiste : JTextField
        +tdDuree : JTextField
        +tdGenre : JComboBox
        +tdImage : JTextField
        +tdRecherche : JTextField
        +tdTitre : JTextField
        +Ajouter()
        +actionPerformed()
        +testImage()
    }
    class Montrage {
        +Assemble : JPanel
        +Header : JPanel
        +Modifier : JButton
        +Status : JLabel
        +Suppression : JButton
        +Trie : JComboBox
        +principal : JPanel
        +recherche_active : boolean
        +recherche_vide : boolean
        +voulu : Vector<Album>
        +Accueil()
        +Ajout()
        +AlbumPresentation()
        +ChangementPanel()
        +Filtre()
        +ImageURL()
        +Modifier()
        +Montrage()
        +Recherchet()
        +Refresh()
        +RetourArtiste()
        +SetLigne()
        +TrieArt()
        +TuaAlbum()
        +actionPerformed()
        +getPrincipal()
        +newLigne()
        +newLigne()
        +newLigne()
        +newLigne()
        +newCard()
    }
    class Bibliotheque {
        +tabalbum : Vector<Album>
        +InitBDD()
        +ajouter Album()
        +getTabalbum()
        +modifier Album()
        +supprimeAlbum()
        +Bibliotheque()
    }
    class Album {
        +artiste : String
        +anne : String
        +duree : String
        +genre : String
        +image : String
        +nb_pistes : String
        +nom : String
        +Album()
        +getArtiste()
        +getDuree()
        +getGenre()
        +getImage()
        +getNb_pistes()
        +getNom()
    }
    class BDDConnection {
        +connection : Connection
        +statement : Statement
        +BDDConnection()
        +getStatement()
    }

    Principal --> ActionListener
    ActionListener --> FenetreGraphique
    FenetreGraphique --> JFrame
    FenetreGraphique --> JDialog
    FenetreGraphique --> JPanel
    FenetreGraphique --> Modifier
    FenetreGraphique --> Ajouter
    FenetreGraphique --> Montrage
    FenetreGraphique --> Bibliotheque
    FenetreGraphique --> Album
    FenetreGraphique --> BDDConnection
    Modifier --> FenetreGraphique
    Ajouter --> FenetreGraphique
    Montrage --> FenetreGraphique
    Bibliotheque --> FenetreGraphique
    Album --> FenetreGraphique
    BDDConnection --> FenetreGraphique
  
```

Notre application de gestion d'albums de musique comporte plusieurs fonctionnalités, dans un premier temps l'utilisateur peut retrouver toute la liste des albums contenu dans sa bibliothèque où il pourra les consulter mais aussi en ajouter d'autres en cliquant sur le bouton "Ajouter" qui fera apparaitre une nouvelle fenêtre dans laquelle l'utilisateur devra fournir certaines informations. Les albums sont affichés sous formes de cartes d'albums sur lesquelles l'utilisateur peut cliquer pour pouvoir voir des informations supplémentaires, il pourra également modifier ou supprimer l'album sélectionné. Pour les autres fonctionnalités l'utilisateur peut trier les albums par titre, par artiste, par date, par genre ou rechercher par les mêmes critères.

Choix techniques

Une des difficultés du projet à était le panneau principal, celui qui affiche tous les éléments. En effet il était compliqué de concevoir un Panel qui s'efface et réapparaissait constamment. Puis par hasard au fil des recherches on a trouvé le CardLayout pour un JPanel. Il pallia nos problèmes car il propose un système de couches tel des onglets que l'on peut ajouter ou retirer à notre guise. Suite

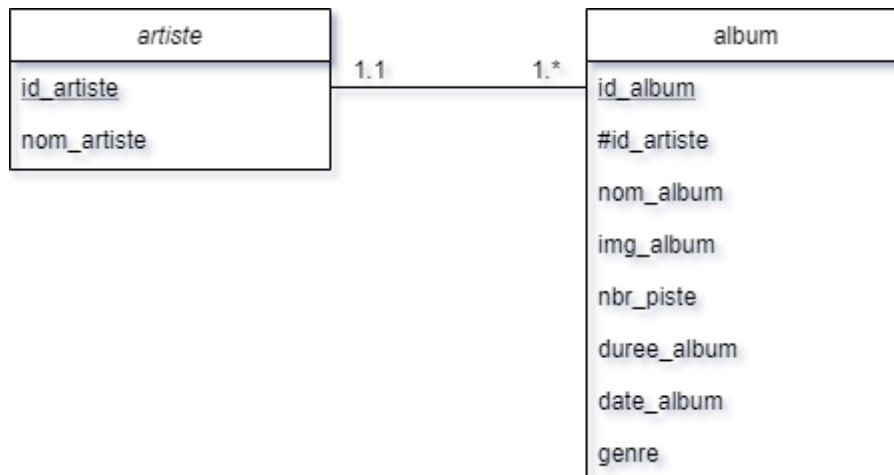
à cela nous créâmes la classe Montrage qui sera centralisé autour de ce CardLayout et des diverses fonctions qu'il devait présenter

Présentation de certains algorithmes "complexes" utilisés

La BDD en SQLite

Il fallait pour le sujet utilisé une base de données. Pour ce faire on a utilisé le logiciel SQLite Studio qui permet de créer des bases de données assez simplement au format db.

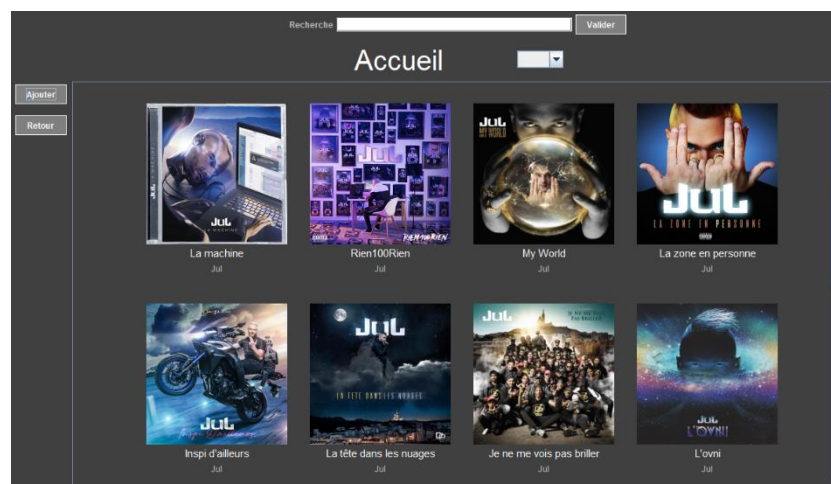
Diagramme de la BDD :



Utilisation

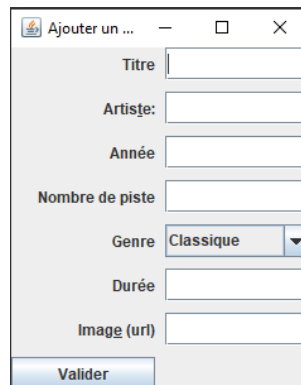
Mode d'emploi

Présentation globale au lancement de l'application.



L'interface global, outre la barre de recherche, possède les options Ajouter et Retour.

Ajouter permet par le biais d'un formulaire l'ajout d'un nouvel album dans la base de données.



Ajouter un ...

Titre

Artiste:

Année

Nombre de piste

Genre Classique

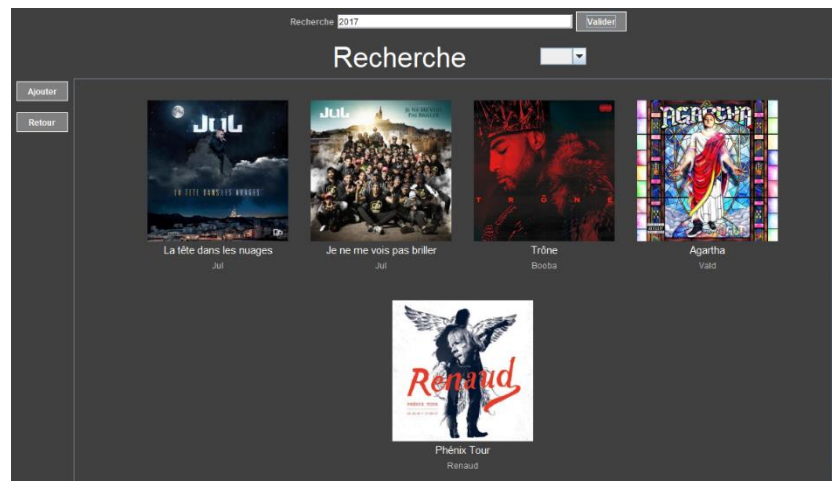
Durée

Image (url)

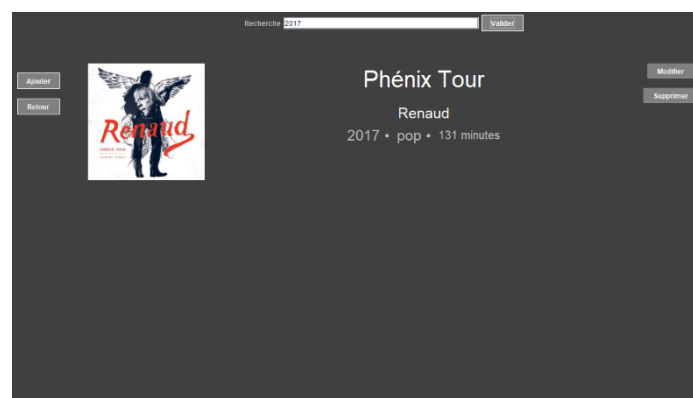
Valider

Le bouton Retour permet lors de la navigation dans l'application, de revenir en arrière dans le programme. Il permet notamment d'effacer les onglets et de revenir jusqu'à la page d'accueil.

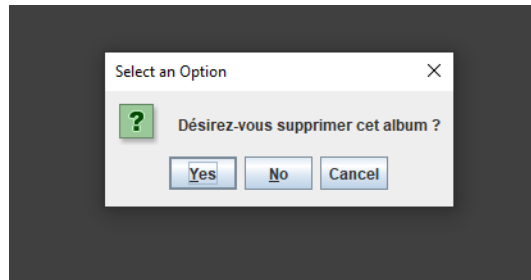
La recherche effectuée par la barre de recherche et le bouton Valider. La recherche marche avec les noms d'albums, les noms artistes, le genre ou les dates.



Cliquer sur un album permet l'affichage d'informations supplémentaires, l'option Modifier et l'option Supprimer.



Le bouton supprimer affichera une validation pour savoir si l'utilisateur veut réellement supprimer l'album. À la suite de cela l'album s'efface de la base de données et s'enlève de l'affichage.



Le bouton modifier redonne le formulaire d'Ajouter avec les champs préremplis par l'album à modifier. Une fois les champs modifiés, la pression du bouton Valider changera dans la base de données l'album voulu.

La liste déroulante permet la trie par catégorie listé. Marche dans l'accueil et dans la recherche.



Configurations requises

- Un ordinateur
- Une connexion internet stable (pour les images)

Conclusion

Bilan

L'application marche bien dans l'ensemble. La liaison avec la base de données est bonne et l'application permet la bonne visualisation des albums dans son ensemble.

Optimisations possibles

L'optimisation la plus idéale aurait pu être niveau image. En effet à chaque rafraichissement des albums toutes les images se retélécharge ce qui pose un problème pour les faibles connections. L'idéal aurait été de sauvegarder toutes images télécharger pour les redistribuer par la suite.

Extensions possibles

Une extension possible serait la compatibilité avec Spotify pour trouver les albums et les lire.

Une autre voie importante aurait été l'implantation de la gestion des morceaux d'un album. C'est à dire la tracklist entière morceau par morceau d'un album. Avec durée, artiste, titre etc... sur tous les morceaux d'un album