Compte rendu conception et programmation orientée objet

Pour le29/05/2020

Antonin Devidal & Cyril Toffin

Table des matières

Diagramme de classe

Description du Modele

Musique :

Classe qui définit une Musique avec ses attributs (Nom, Image, Lien vers une vidéo YouTube, Date de création, ainsi que les types musicaux associés).

Album :

Classe qui définit un Album ainsi que ses attributs (Nom et Image de l'album mais aussi une liste de musique), elle contient une fonction qui permet d'ajouter une musique à cet album.

Playlist :

Classe qui définit ce qu'est une playlist ainsi que ses attributs (Nom et Image et une liste de musique). La fonction AjouterMusique permet d'ajouter une musique à cette playlist, à noter que si cette musique est déjà présente alors, on la retire de la playlist.

Artiste :

Classe qui définit un Artiste par ses attributs comme un Nom, une Image, une Description mais aussi une liste d'album et une liste de musique (qui sert à « stocker » les musiques qui n'appartiennent pas à un album).

Artistethèque :

Cette classe a comme attribut une liste contenant tous les artistes de l'application. Avec la méthode Rechercher, on peut comme son nom l'indique, rechercher une musique, un album ou un artiste dans l'Artistetheque. Comme pour les classe Album et Playlist on peut ajouter un artiste à la liste sachant que s'il est déjà présent, on le retire de celle-ci.

TypeMusicaux :

TypeMusicaux est une énumération de type byte permettant de définir le type musical d'une œuvre. Elle est de type byte car elle permet les combinaisons de valeur. Par exemple la valeur 4 associe la valeur Rap et 2 associe Rock, grâce à cette méthode la valeur 6 est égale au style Pop Rock.

Discothèque :

Discothèque est une classe qui va regrouper 3 dictionnaires contenant 2 listes chacun est un Artiste qui va permettre d'associer un utilisateur à un artiste. Elle contient deux méthodes utilisant la généricité. LA première (AimerObj) permet d'ajouter à une des liste un objet qu'on aime que ce soit une musique ou un artiste. Et la deuxième (AjouterObjetDernierementEcoute) va ajouter à la liste passée en paramètre une musique, un artiste ou une playlist. La généricité est très importante car elle permet d'éviter de recopier ces méthodes pour chaque objet et donc optimiser le code.

Serialisation :

Dernière classe mais pas des moindres, Serialisation qui permet de … sérialiser et de désérialiser des données en binaire. Contenant deux méthodes, DeserialisationBin retourne l'objet que l'on a choisi de désérialiser à l'aide du chemin de celui-ci. Et a contrario, SerialistationBin va sauvegarder l'objet passé en paramètre à l'aide du chemin ou l'on doit sauvegarder les données.

Description de la partie Graphique

MainWindow :

Cette fenêtre permet à l'utilisateur de se connecter à son compte. Si les identifiants sont corrects, alors cette fenêtre se ferme et FenetrePrincipale s'ouvre.

Fenetre principale :

Cette fenêtre est la colonne vertébrale de la partie graphique. En effet, elle se repose sur deux colonnes, l'une contenant un menu pour une navigation rapide et l'autre permet d'accueillir tous les user controls du projet correspondant aux diverses fonctionnalités de l'application. Tous les user controls que nous allons décrire seront affichés dans la partie droite de la fenêtre. Les boutons situés sur la gauche de la page nous rediriges respectivement vers les pages : PageAccueil , MesPlaylists, MesArtistes, MesMusiques.

--- Tous les UserControl qui vont être décrits sont liés à une Discotheque et une Artistetheque. ---

UcIcons :

Ce dossier contient différents userControls correspondant à l'affichage en miniature d'une Musique, une Playlist, un Artiste, ou un Album. Plusieurs variantes existent permettant d'afficher un de ces objets en ligne ou sous forme d'un carré.

PageAccueil :

Cette page est la première affichée quand on arrive sur la fenêtre principale. Elle affiche 3 listboxs contenant les musiques, artiste, playlists dernièrement écoutées. On peut aussi faire une recherche grâce à l'élément associé en haut de la page. On peut aussi en haut à droite de la page, accéder aux pages pour ajouter un album, une musique ou encore une playlist (voir pages AjouterMusique, AjouterAlbum, AjouterPlaylist). En cliquant sur l'une des musiques/artiste/album, on atteint sa page associée.

MesArtitstes :

Cet User Control affiche la liste des artistes que l'on a aimés. En cliquant sur un des Artistes on peut accéder à sa page (PageArtiste)

MesMusiques :

Comme pour MesArtistes, ici on affiche les musiques que l'on a aimé. En cliquant sur une des Musiques on peut accéder à sa page (UneMusique)

MesPlaylist :

Cet UserControl affiche les playlists de l'utilisateur. En cliquant sur une des Playlists on peut accéder à sa page (UnePlaylist)

PageArtiste :

PageArtiste contient un master Details car on affiche les informations relatives à un artiste et la liste de ses albums et des musique triées par date de création. A l'arrivée sur la page on appelle la méthode AjouterObjetDernierementEcoute de la Discothèque.

UneMusique :

Sur UneMusique, on peut lire les informations relatives à la musique (créateurs, styles, date de création) mais on peut aussi regarder le clip de cette musique grâce au navigateur web par défaut du Wpf. On peut aussi ajouter cette musique dans une playlist en appuyant sur le bouton associé, cela appellera la méthode AjouterMusique dans la classe Playlist. A l'arrivée sur la page on appelle la méthode AjouterObjetDernierementEcoute de la Discothèque. On peut accéder à la page de l'artiste quand on clique sur son nom.

UnAlbum :

Page qui affiche les détails de l'album en question ainsi que les musiques qu'il contient avec le nom du créateur. En cliquant sur une des Musiques on peut accéder à sa page (UneMusique)

UnePlaylist :

Cette page affiche la liste des musiques contenu dans PlaylistMusique. A l'arrivée sur la page on appelle la méthode AjouterObjetDernierementEcoute de la Discothèque. En cliquant sur une des Musiques on peut accéder à sa page (UneMusique)

PageRecherche :

Page qui permet l'affichage des musiques, albums et artistes correspondant à la recherche que l'on a effectué dans la barre de recherche situé en haut de la PageAccueil. Pour la recherche on appelle la méthode Rechercher de l'artistetheque. En cliquant sur l'une des musiques/artiste/album, on atteint sa page associée.

AjouterAlbum :

On peut ajouter un album grâce à cette page. Il faut choisir le nom de l'album et la pochette si souhaité. D'ailleurs l'image choisi sera copiée dans le fichier SpoDeezer/images/pochettes tout en associant un nom par rapport au créateur et au nom de l'album du style NomDeLArtiste_NomDeLAlbum. Quand on appui sur le bouton Confirmer, on est redirigé sur PageAccueil.

AjouterMusique :

Comme pour un album, cette page permet d'ajouter une musique en choisissant les créateurs, le style de la musique et l'album auquel elle appartient. L'image de la musique sera la pochette de l'album. Si on ne choisit pas d'album, elle sera ajoutée dans la liste MusiquesArtiste et aura alors une image par défaut. Quand on appui sur le bouton Confirmer, on est redirigé sur PageAccueil.

AjouterPlaylist :

De même, en choisissant une image (même fonctionnement que pour ajouter un album) et un nom, la playlist va être ajouté dans la liste MesPlaylists dans la discothèque. Quand on appui sur le bouton Confirmer, on est redirigé sur PageAccueil.

Description de Data

Stub :

Stub contient toutes les données en dur et est lié à la classe Serialisation du Modele pour les méthodes pour Sérialiser et désérialiser.

Description de MaConsole

Program :

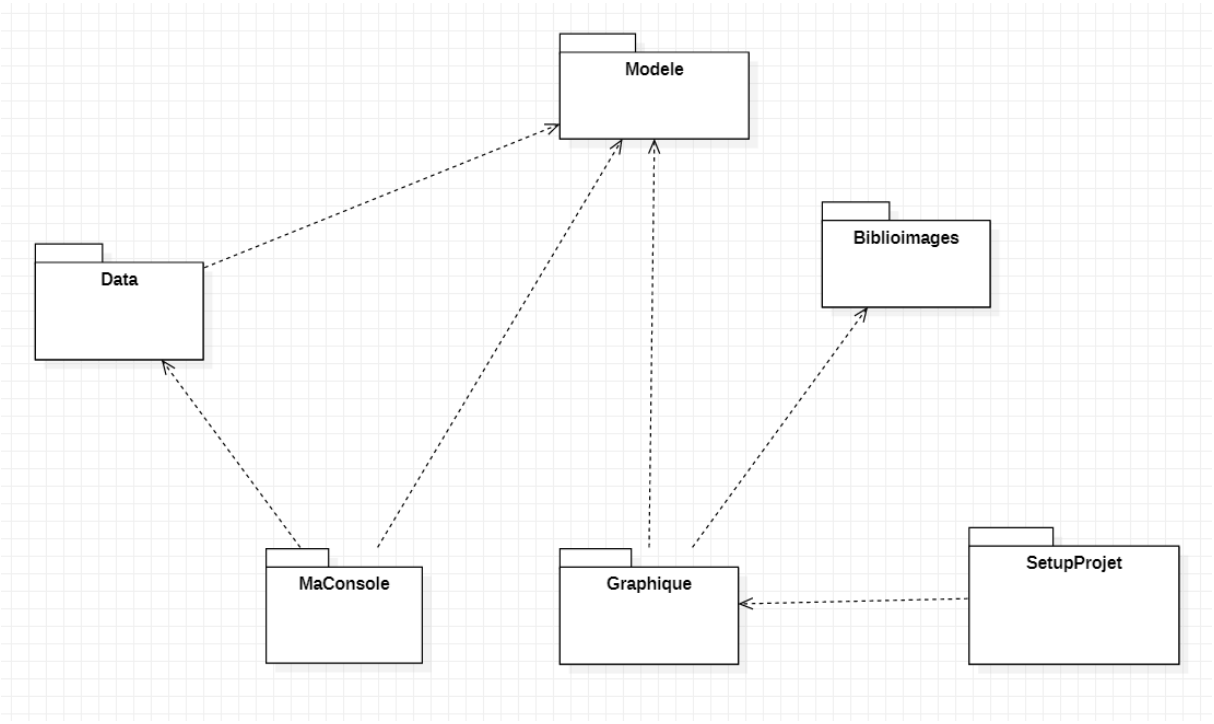
Lié au Stub, il permet de faire des tests sur les méthodes du Modele.

Description de BiblioImage

- Contient toutes les images présentes par défaut dans l'application.

Diagramme de paquetage

Diagramme



Description du diagramme de paquetage

Modele :

Il contient toutes les classes utiles aux fonctionnements de l'application, il n'a pas de liaisons.

BilioImage :

Contient en ressource toutes les images de base de l'application.

Data :

Contient toutes les données en dur pour le programme et elle est liée au Modele pour pouvoir utiliser les objets nécessaires à la création des données.

MaConsole :

Ce package permet de faire des tests pour vérifier que toutes les fonctions marchent. Il est lié à Data pour pouvoir générer des données et lié au Modele pour pouvoir tester les différentes méthodes.

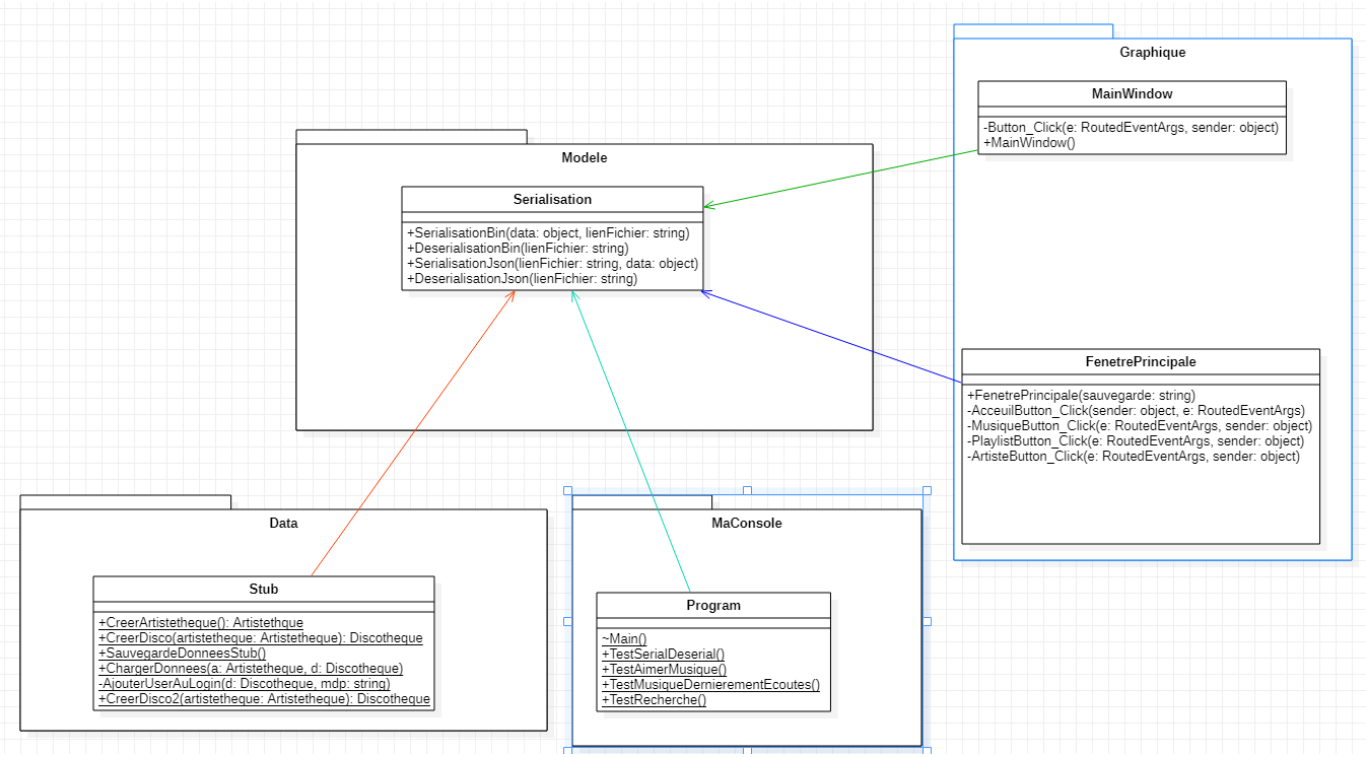
Graphique :

Il s'occupe de tout l'aspect graphique de l'application. Il est lié à BiblioImages pour récupérer les images nécessaires au fonctionnement de l'application. Il est aussi lié au Modele pour pouvoir utiliser les différents objets et méthodes associées.

SetupProjet :

SetupProjet est un package permettant de créer un installeur pour notre application. Pour cela , il dépend de Graphique et de ses dépendances.

Persistance au sein du diagramme de classe



Dans cette image, nous pouvons voir que le cœur de la sérialisation se trouve dans le Modele. En effet la classe Sérialisation contient quatre méthodes SerialisationBin, DeserialisationBin, SerialisationJson et DeserialisationJson qui correspondent à la sérialisation et à la désérialisation sous forme binaire et

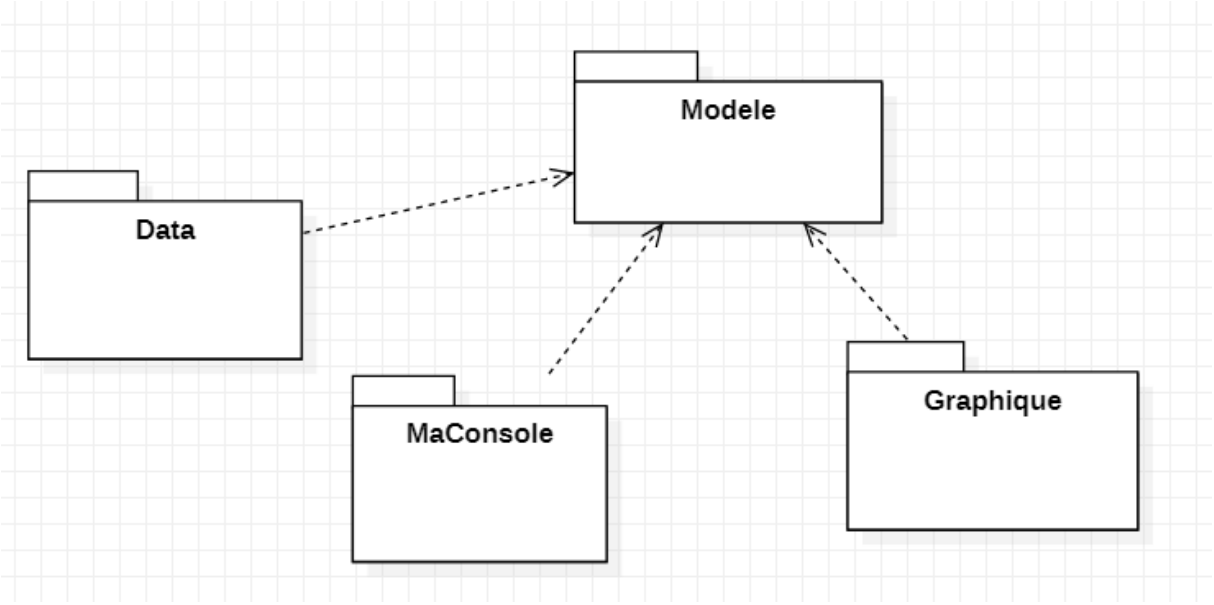
en json. La sérialisation en binaire est utilisé pour sauvegarder les données des utilisateurs et la collection des artistes de l'application, alors que celle en json est utilisée pour contenir les identifiants et mot de passes des utilisateurs lors de la connexion à l'application (voir partie personnelle Antonin Devidal).

Les méthodes de persistance en binaires et en json sont utilisées dans la classe Stub (package Data) car c'est ici que sont créées les données du programme.

Programme (package Console) utilise aussi la sérialisation pour effectuer des tests de bon fonctionnement de celles-ci.

MainWindow quant à lui utilise seulement la méthode de désérialistaion en json pour authentifier un utilisateur, alors que FenetrePrincipale utilise seulement la méthode de désérialisation binaire pour charger les données d'un utilisateur en particulier en plus de la collection d'artistes.

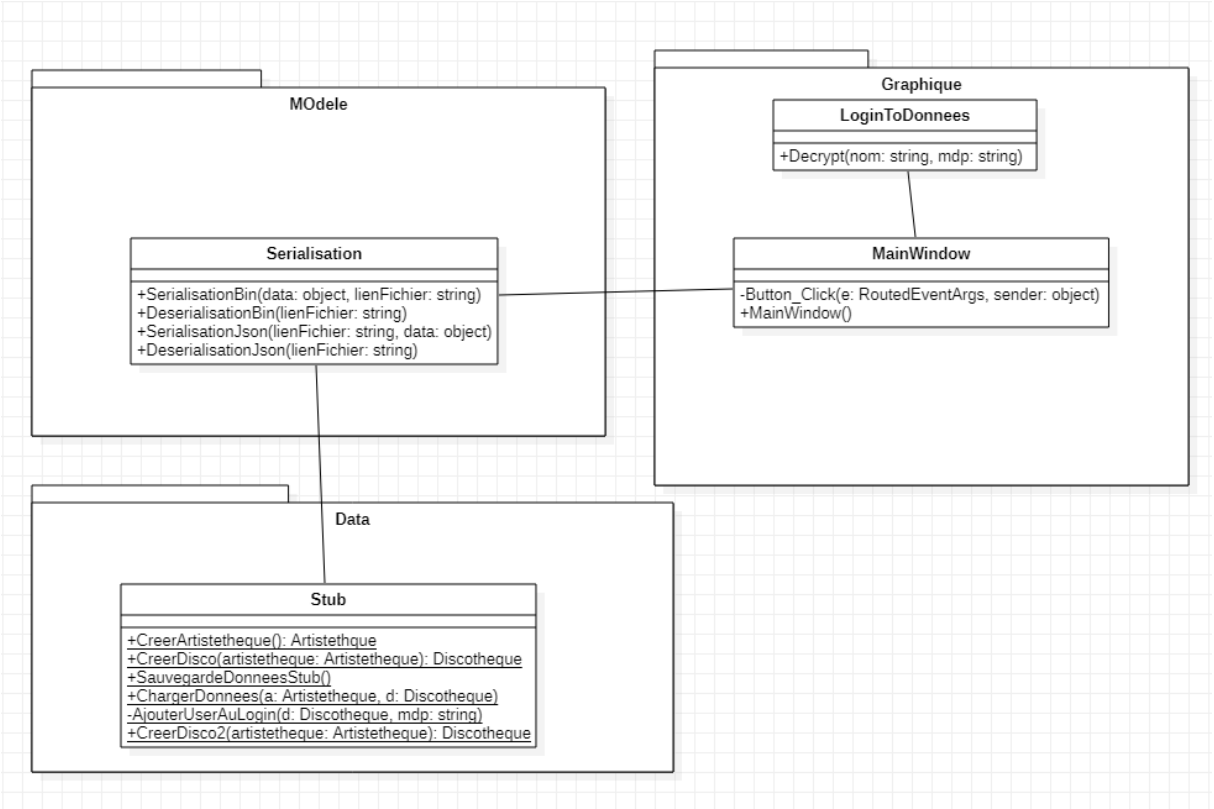
Persistance au sein du diagramme de paquetage



Dans ce diagramme de paquetage simplifié, nous avons mit en évidence les liaisons en les différents package de l'application du point de vue de la sérialisation. Nous pouvons voir que Data MaConsole et Graphique font appel aux différentes méthodes de sérialisation et désérialisation contenu dans le package Modele qui est l'élément clef de cette partie.

Parties personnelles

Partie d'Antonin



Ma partie consiste à la gestion des utilisateurs et des mots de passes. Pour la gestion des utilisateurs, j'ai décidé de créer un fichier json ou seront renseignés les informations des utilisateurs nécessaires au bon démarrage de l'application. Nous retrouvons alors dedans :

- Le nom en tant que clef
- Un dictionnaire en tant que valeur

Dans ce dictionnaire seront renseignés l'emplacement du fichier de sauvegarde de l'utilisateur et son mot de passe. Les informations des utilisateurs sont rentrées dans le fichier login.json au moment de la création de l'utilisateur (package Data : Stub :AjouterUserAuLogin).

Le fichier apparait sous la forme :

```
"Utilisateur":{
  "mdp": "monMotDePasse",
  "save": "MonEmplacementDeSauvegarde"
}
```

Lorsque l'utilisateur se connecte à l'application, il rentre alors sont identifiant et son mot de passe dans les cases associées. Alors, le programme va rechercher dans le fichier login si une de ses valeurs est égale au nom de l'utilisateur et si oui, il va regarder si le mot de passe correspond.

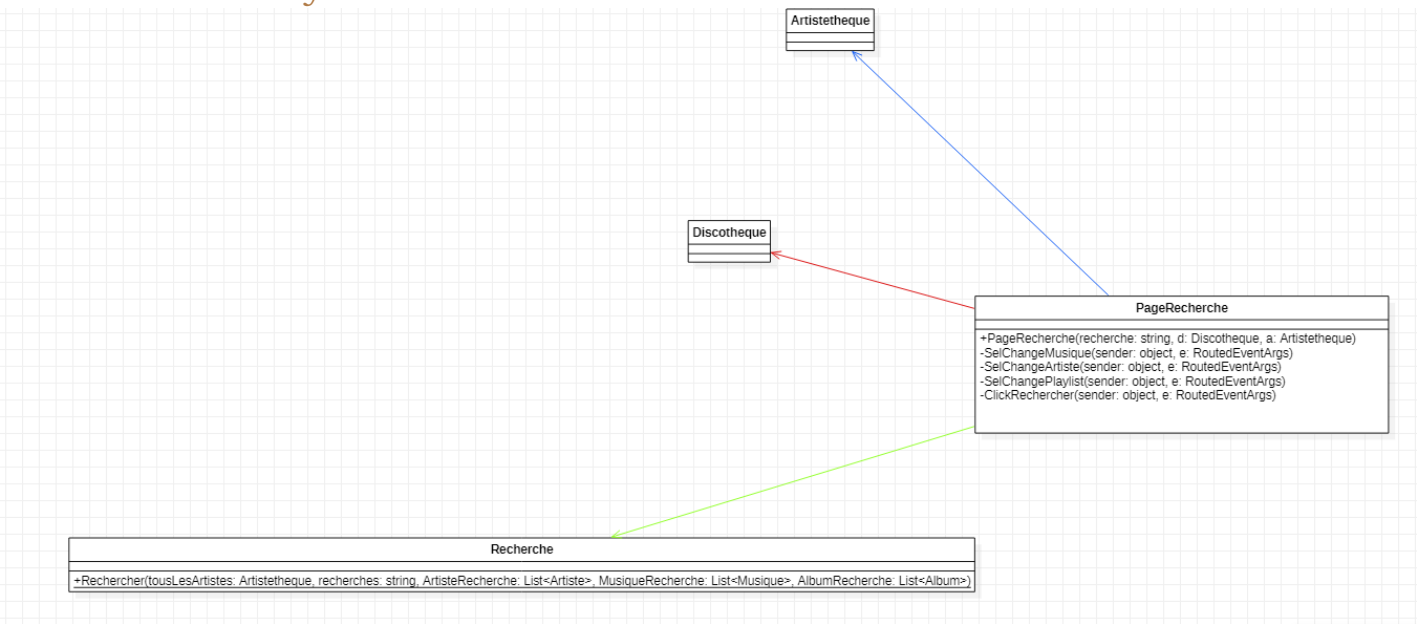
Cependant, comme montré sur l'image ci-dessus, le mot de passe des utilisateurs est affiché en clair dans le fichier et donc la sécurité de l'application est remise en question. C'est là que vient le cœur de ma partie personnelle : le chiffrement des données. Pour ce faire, j'ai utilisé le Triple Data Encryption Standard (TDES pour faire court). Après avoir importé l'API.Net Security.Cryptography, j'ai commencé par effectuer le chiffrement des mots de passes dans le fichier login. J'ai alors créé une clef (key) et un vecteur d'initialisation (iv), je transforme ensuite le mot de passe de l'utilisateur (un string) en tableau d'octet pour pouvoir le chiffrer. Je peux alors crypter mon mot de passe. Une fois cela fait, pour l'ajouter au fichier login il faut que je transforme le résultat (tableau d'octet) en chaine de caractère. Pour cela, il suffit d'appeler la méthode Convert.ToBase64string.

Ce qui nous donne le résultat :

```
"Utilisateur":{
  "mdp": "jmrOZTQXj67inpAFSr5MmQ==",
  "save": "MonEmplacementDeSauvegarde"
}
```

Le déchiffrement n'est guère plus compliqué car il suffit de répéter les étapes précédemment effectuées en sens inverse (voir package Graphique : converters\LoginToDonnees).

Partie de Cyril



Cyril a fait la fonction Recherche, cette fonction permet de rechercher des données dans l'application, grâce à cette recherche on peut trouver un artiste, une musique et un album avec la barre de recherche se situant sur la page principale. La fonction va chercher les artistes dans Artistetheque qui contient la liste de tous les créateurs de musique