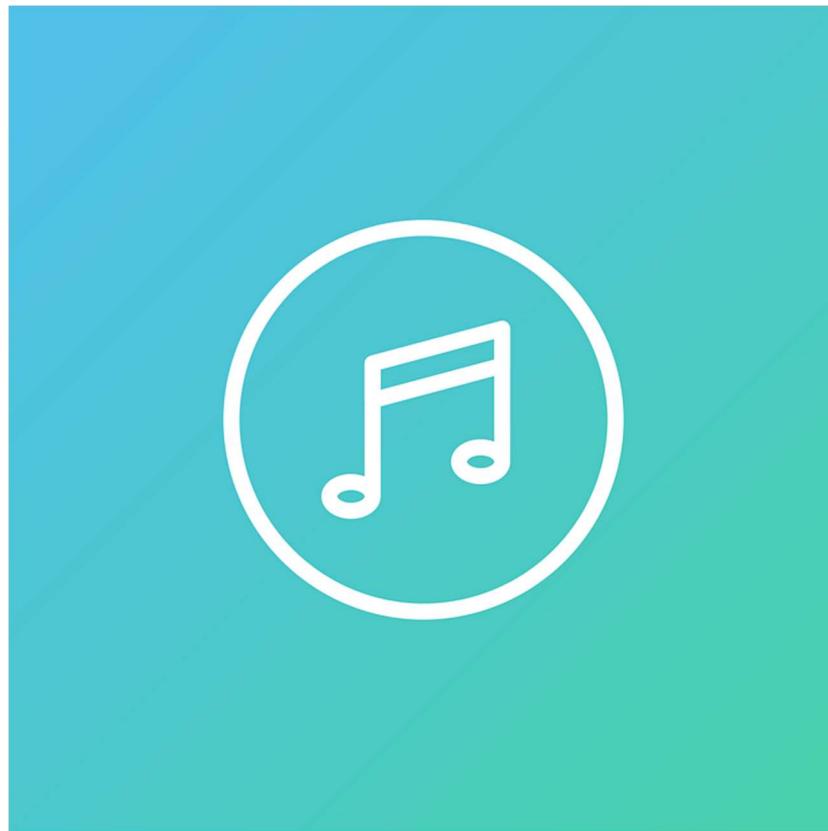


Gestion audio



Eric-Nicolas Preisig / CIN4A

ETML - Lausanne

110 heures

Chef de Projet

M. Bertrand Sahli – bertrand.sahli@vd.ch

Experts

M. Sylvain Rollinet – sylvain.rollinet@gmail.com

M. Jonathan Melly – jonathan.melly@vd.educanet2.ch

Table des matières

1 SPÉCIFICATIONS.....	4
1.1 TITRE	4
1.2 DESCRIPTION.....	4
1.3 MATÉRIEL ET LOGICIELS À DISPOSITION	4
1.4 PRÉREQUIS.....	5
1.5 CAHIER DES CHARGES	5
1.6 LES POINTS SUIVANTS SERONT ÉVALUÉS.....	5
1.7 VALIDATION ET CONDITIONS DE RÉUSSITE	5
2 PLANIFICATION INITIALE.....	6
3 ANALYSE	7
3.1 OPPORTUNITÉS.....	7
3.2 DOCUMENT D'ANALYSE ET CONCEPTION	8
3.2.1 <i>Maquette</i>	8
3.2.2 <i>Architecture</i>	13
3.2.3 <i>Librairies utilisées</i>	14
3.2.4 <i>Structure du code</i>	16
3.2.5 <i>Base de données</i>	18
3.2.6 <i>Schéma évènementiel</i>	20
3.3 CONCEPTION DES TESTS	27
4 PLANIFICATION DÉTAILLÉE.....	29
5 RÉALISATION	31
5.1 DOSSIER DE RÉALISATION.....	31
5.1.1 <i>MPD</i>	31
5.1.1 <i>Dossiers exclus</i>	31
5.1.2 <i>Lecture des radios</i>	31
5.1.3 <i>Stream radio</i>	32
5.2 MODIFICATIONS.....	32
5.3 MDP FINAL	33
5.4 MVVM	34
5.5 EXTRAIT DE CODE	35
6 TESTS	36
6.1 DOSSIER DES TESTS	36
6.2 TESTS UNITAIRES	39
7 CONCLUSION	40
7.1 BILAN DES FONCTIONNALITÉS DEMANDÉES	40

7.2	BILAN DE LA PLANIFICATION.....	41
7.3	BILAN PERSONNEL	42
7.4	REMERCIEMENT	42
8	WEBOGRAPHIE	42
9	DIVERS	43
9.1	JOURNAL DE TRAVAIL	43
10	ANNEXES.....	45
10.1	CAHIER DES CHARGES.....	45
10.2	GANTT.....	48

1 SPÉCIFICATIONS

Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le 1er jour.

Le cahier des charges est approuvé par i — CQ VD. Il est en outre présenté, commenté et discuté avec le candidat. Par sa signature, le candidat accepte le travail proposé.

Le candidat a connaissance de la feuille d'appréciation avant de débuter le travail.

Le candidat est entièrement responsable de la sécurité de ses données.

En cas de problèmes graves, le candidat avertit au plus vite les deux experts et son chef de projet.

Le candidat a la possibilité d'obtenir de l'aide, mais doit le mentionner dans son dossier de projet.

À la fin du délai imparti pour la réalisation du TPI, le candidat doit transmettre par courrier électronique le dossier de projet aux deux experts et au chef de projet. En parallèle, une copie papier du rapport doit être fournie sans délai en trois exemplaires. Cette dernière doit être en tout point identique à la version électronique.

1.1 Titre

Gestionnaire de playlists audio et de flux radios

1.2 Description

Développer une application desktop en C# selon le pattern MVVM afin de gérer des playlists audios et des flux radios publics diffusés sur Internet. Il s'agira notamment :

- d'assurer l'installation de l'application sur les OS Microsoft 7 à 10.
- d'exploiter les données audios locales (sans les dupliquer).
- de créer un lecteur audio « user-friendly ».
- de stocker les données de l'application dans une base de données SQLite.

1.3 Matériel et logiciels à disposition

PC standard de l'ETML (Windows 7 – 64 bits)

Player de machines virtuelles (vmware ou équivalent)

Suite Microsoft Office 2013

IDE C# (Visual Studio Community 2015 ou 2017, SQLite Precompiled Binaries for Windows)

1.4 Prérequis

Compétences élémentaires en base de données (SQL) et en programmation (C#) → Modules 103, 104, 105, 303.

Compétences bureautiques → Modules 301 et 302

1.5 Cahier des charges

Cahier des charges complet en annexe

1.6 Les points suivants seront évalués

- La méthodologie de travail ainsi que la conformité des tâches réalisées par rapport au planning initial.
- Le choix des « métadonnées » nécessaires à l'indexation des chansons.
- La pertinence du modèle de données réalisé.
- La mise en œuvre de tests (notamment les tests unitaires) pour vérifier les fonctionnalités implémentées.
- La journalisation des tâches ainsi que la rédaction régulière de la documentation.

1.7 Validation et conditions de réussite

La grille d'évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité...).

En plus de cela, le travail sera évalué sur les trois points spécifiques suivants :

- Le point spécifique n° 1 → L'extraction de métadonnées de divers format audio.
- Le point spécifique n° 2 → La recherche de Webradio.
- Le point spécifique n° 3 → Le (re) scan des contenus audio.

2 PLANIFICATION INITIALE

Date Début :	lundi 1 mai 2017
Date Fin:	mercredi 7 juin 2017
Nombre de semaines:	6
Nombre de Périodes/Semaine :	33
Nombre de 1/4 heure/Période:	3

Liste des tâches	
Tâche obligatoire:	
	Absence - Imprévus
1	Planification initiale
2	Arborescence de fonctions et de classes
3	Analyse
4	Création du MPD
5	Création de la base de données
6	Mise en place de l'architecture MVVM
7	Création des tests unitaires
8	UI
9	Création des objets DTO (Data transfer object)
10	Couche BLL (Business Logic Layer)
11	Couche Présentation
12	Couche DAL (Data Access Layer)
13	Création de l'installateur
14	Tests
15	Manuel utilisateur
16	Documentation
17	Rendez-vous
18	Congé
19	Bugs

Gantt disponible en annexe

3 ANALYSE

3.1 Opportunités

Ce projet me permettra de développer mes capacités en développement .NET. Il me donnera l'opportunité d'approfondir mes connaissances dans l'architecture MVVM.

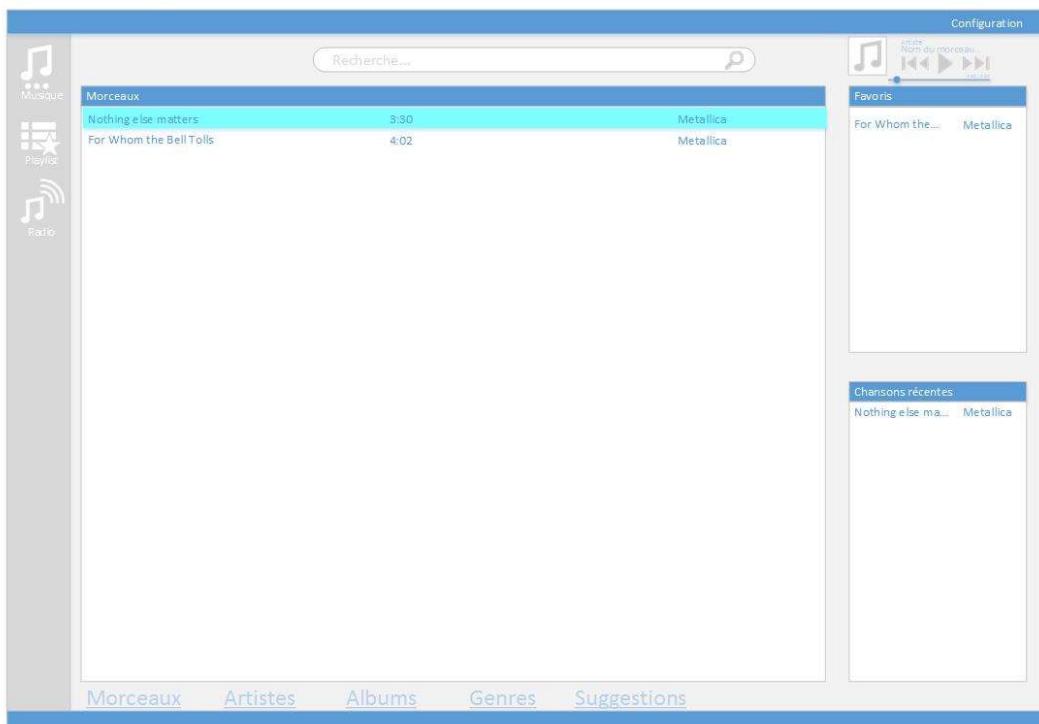
La principale difficulté sera de fournir une interface simple et ergonomique. Je pense donc partir sur un design flat et minimaliste, permettant ainsi une interface simple et intuitive. J'utiliserais pour cela la librairie Mahapps qui me permettra de fournir une interface épurée dans le style de Windows 8.

La sauvegarde du contexte de l'application sera un point difficile. Il y a plusieurs options pour pouvoir garder le contexte applicatif après la fermeture de l'application. Je pense créer une classe qui contient toutes les données à sauvegarder. Celle-ci sera enregistrée dans la base de données. À l'ouverture de l'application, je n'aurais alors plus qu'à interpréter les données de la classe pour les restaurer au bon endroit.

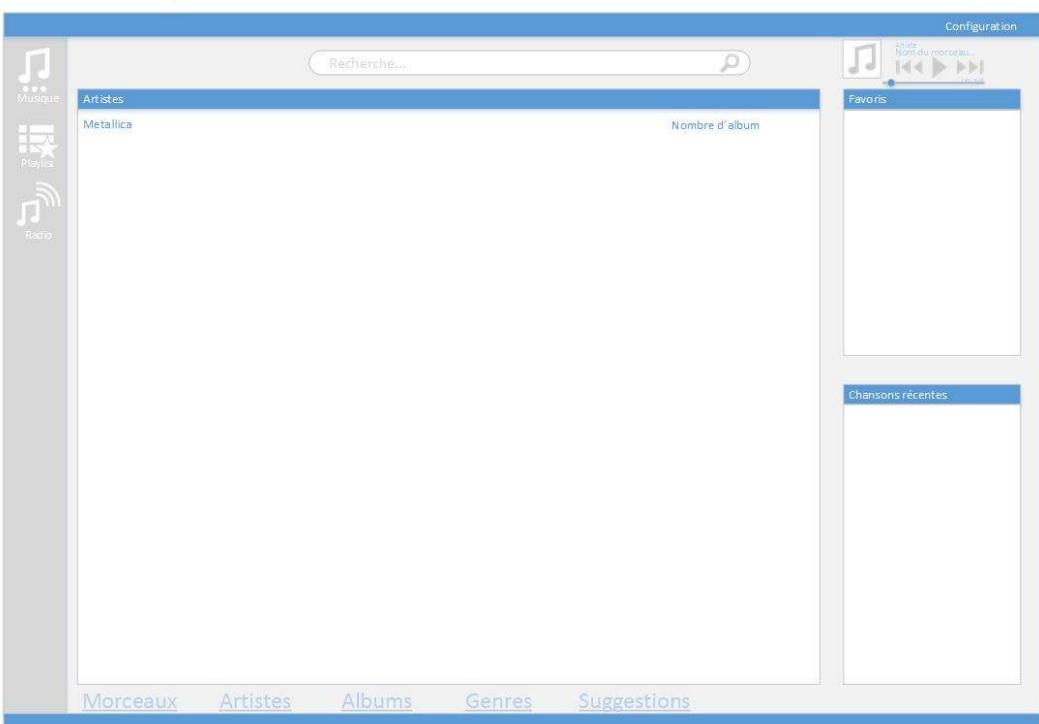
3.2 Document d'analyse et conception

3.2.1 Maquette

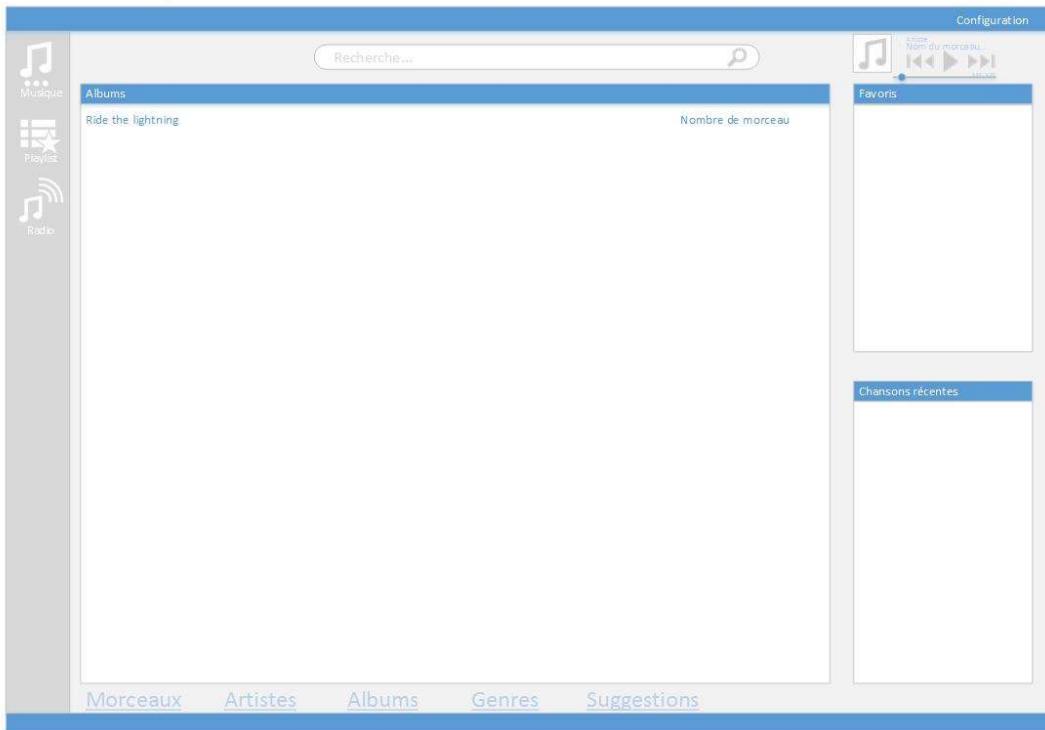
Musique



Musique - Artistes



Musique – Album



Albums

Ride the lightning

Nombre de morceau

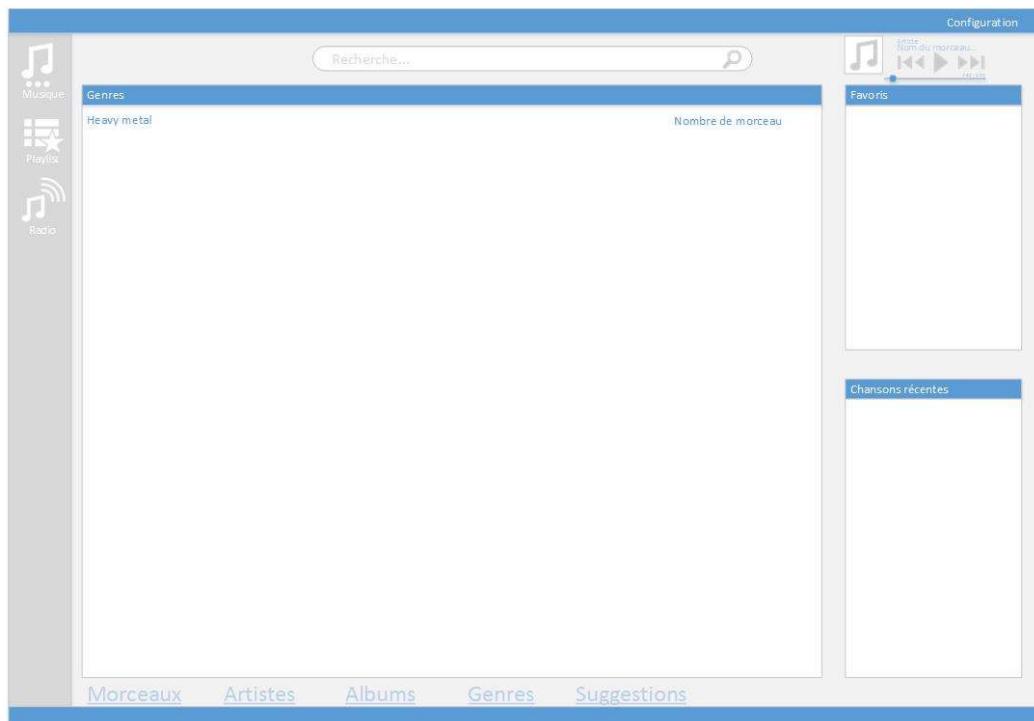
Configuration

Favoris

Chansons récentes

Morceaux Artistes Albums Genres Suggestions

Musique – Genres



Genres

Heavy metal

Nombre de morceau

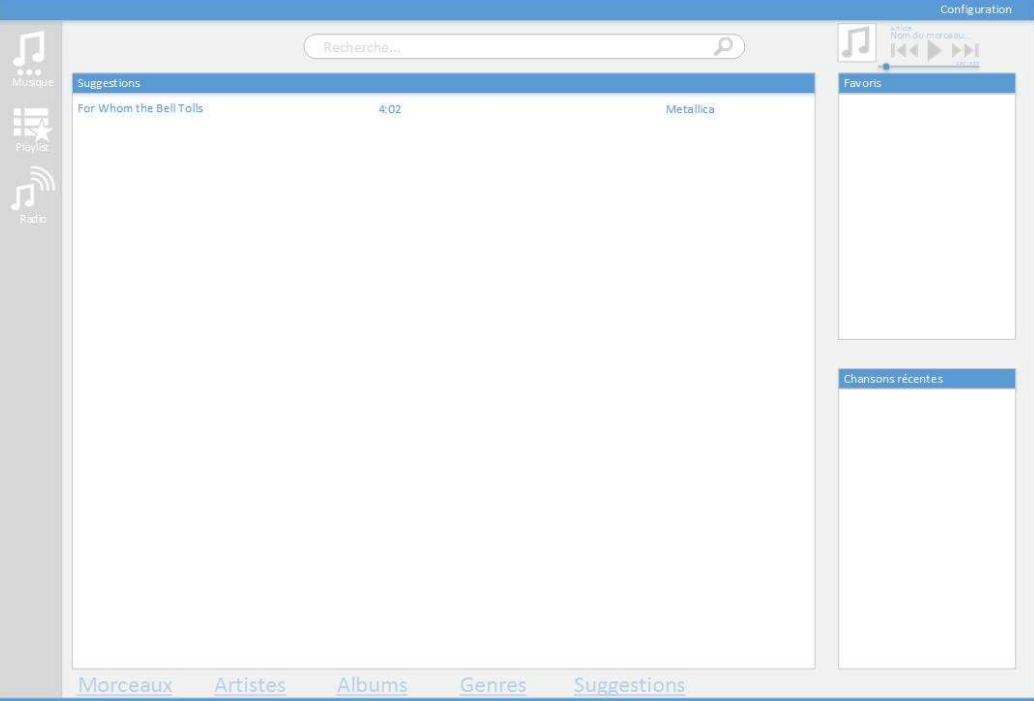
Configuration

Favoris

Chansons récentes

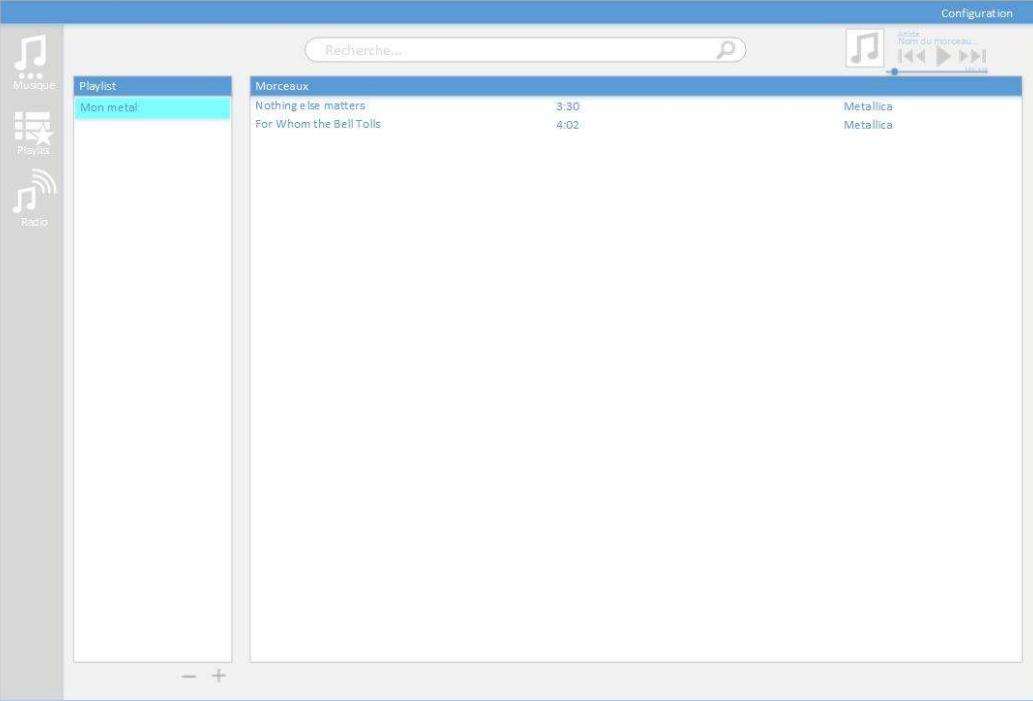
Morceaux Artistes Albums Genres Suggestions

Musique – Suggestions



The screenshot shows a music application interface. On the left, there's a sidebar with icons for Musique, Playlist, and Radio. The main area has a search bar at the top. Below it, a 'Suggestions' section displays a song: 'For Whom the Bell Tolls' by Metallica (4:02). To the right is a 'Configuration' panel with a 'Favoris' section and a 'Chansons récentes' section below it. At the bottom, there are tabs for Morceaux, Artistes, Albums, Genres, and Suggestions, with 'Morceaux' being the active tab.

Playlist



The screenshot shows a music application interface similar to the previous one, but with a 'Playlist' section visible. The sidebar on the left shows 'Musique' is selected. The main area now shows a 'Playlist' section with a highlighted entry 'Mon metal'. This section contains two songs: 'Nothing else matters' and 'For Whom the Bell Tolls' both by Metallica. The configuration panel on the right is identical to the first screenshot. At the bottom, there are minus and plus signs, likely for managing the playlist.

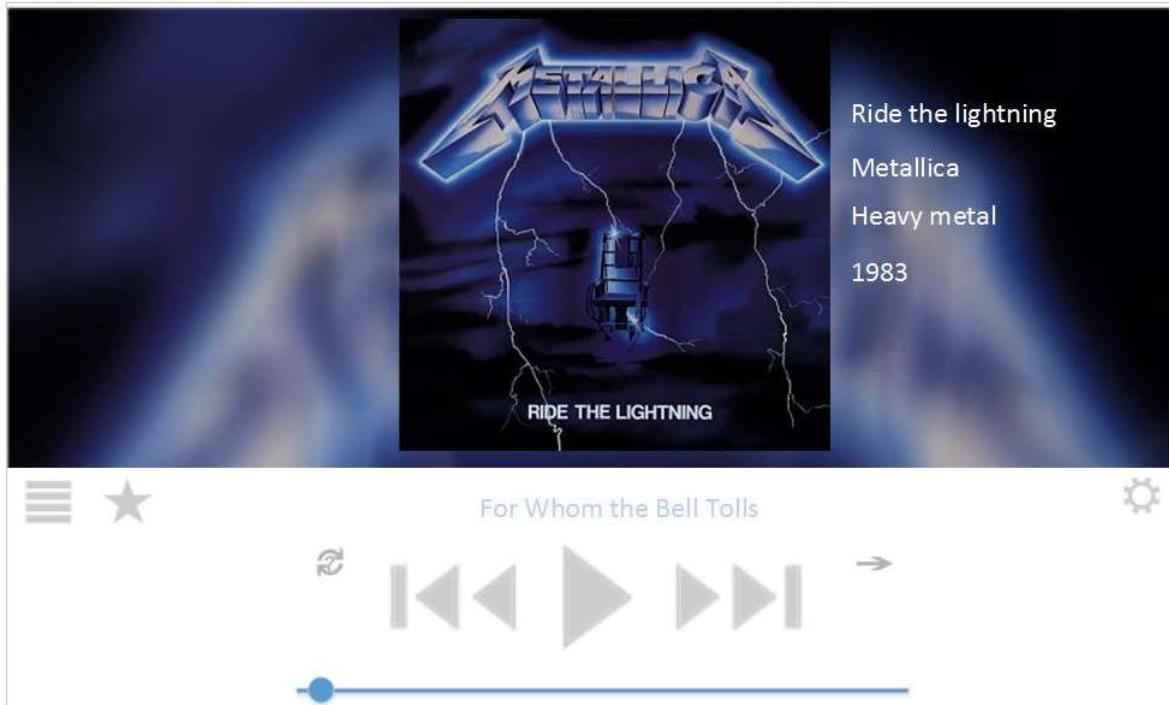
Radio

The screenshot shows a radio application interface. On the left is a vertical sidebar with icons for Musique, Playlis, and Radio. The main area has a blue header bar with a search bar labeled "Recherche...". Below the header, there are two main sections: "Radios" and "Configuration". The "Radios" section lists "AC-DC WALLYradio" (selected), "Throwback Country", "Heavy metal", and "Classic country". The "Configuration" section includes a "Favoris" list with "Throwback Country" and a "Radio récentes" list with "Throwback Country".

Recherche

The screenshot shows a search results page for the query "xxxxxxxxx". The top part displays the search bar with the result "Recherche : xxxxxxxxx". Below the search bar are two main sections: "Morceaux" and "Artistes". The "Morceaux" section shows a track: "For Whom the Bell Tolls" by Metallica (4:02) in Heavy metal. The "Artistes" section shows Metallica with the label "Nombre d'album". Below these are sections for "Albums" (showing an image of the "Ride the lightning" album cover) and "Radios" (listing "AC-DC WALLYradio" under Heavy metal).

Morceau en cours



The player interface displays the album cover of 'Ride the lightning' by Metallica, featuring a horse head with lightning bolts. To the right, song details are shown: 'Ride the lightning', 'Metallica', 'Heavy metal', and '1983'. Below the cover, the song title 'For Whom the Bell Tolls' is displayed, along with standard media control icons (refresh, back, forward, volume, and settings).

Configuration

Lancer la musique au démarrage

Synchroniser la musique

- Dossiers exclus**
- +

Annuler Enregistrer

Propriétés



Sélectionner une image de couverture

Artiste

Album

Date

Genre

Annuler Enregistrer

Liste de lecture

For Whom the Bell Tolls

Annuler Enregistrer

3.2.2 Architecture

Je vais utiliser une architecture en multicouche d'une profondeur de 3 couches. Il y aura :

La couche DAL, qui s'occupera de récupérer les données depuis la base de données ainsi que depuis l'API de Web radio.

La couche BLL, qui se chargera de formater les données brutes reçues par la couche DAL

La couche Présentation, servira d'interface humain-machine elle aura la charge d'afficher les données à l'utilisateur ainsi que de récupérer ses entrées.

Quant au design paterne, j'utiliserai MVVM. Les view et les viewModel seront inclus dans la couche de Présentation. Le modèle sera l'ensemble de la couche DAL et BLL combiné.

Une partie DTO sera commune à toutes les couches, elle aura pour rôle de fournir le modèle des objets. Si une méthode est commune à toutes les couches, elle pourra être placée dans la zone « Méthode commune ».

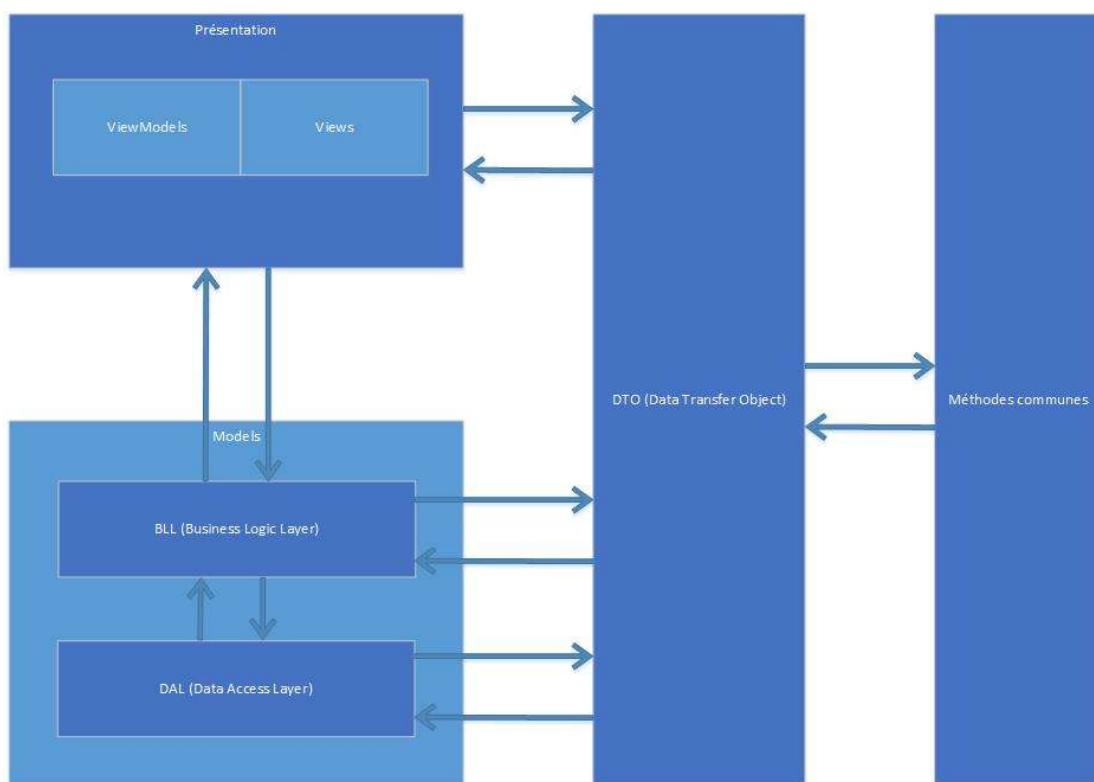


Figure 1 Illustration de l'architecture

3.2.3 Librairies utilisées

API ShoutCast

ShoutCast est une plateforme de radio amateur qui propose plus de 50 000 radios.

Elle propose aussi une API que j'utiliserais pour gérer la partie radio de l'application.

Il faut savoir que l'API Shoutcast n'est pas 100% sûre, et que parfois les données qu'elle fournit ne sont pas valides. Soit la radio a expiré, ou soit un problème passager empêche la bonne récupération du stream

MvvmLight

Permet la simplification des fonctionnalités propre à MVVM, comme le « Messenger », les « Command », et les mises à jour de la vue

EntityFramework

ORM (Object Relational Mapping). L'ORM permet de simplifier l'utilisation des entités de la base de données en les faisant passer pour des objets en définissant une correspondance entre eux.

Mahapps.Metro

Ajoute des éléments graphiques pour rendre l'interface plus moderne

NAudio

Permet la lecture des fichiers et flux audio, j'ai choisi cette librairie, car elle est reconnue comme étant une bonne librairie. Elle acceptait tous les formats de fichier dont je pensais avoir besoin.

NUnit

Ajoute des fonctionnalités aux tests unitaires

System.Data.SQLite

Permet l'utilisation d'une base de données SQLite. Cette librairie inclus les binaires précompilés.

SQLite.CodeFirst

Permet d'utiliser les actions de EntityFramework avec une base de données SQLite

3.2.4 Structure du code

Le code sera structuré de manière suivante :

1. DAL Projet Data access layer

1.1. API Dossier

1.1.1. Shoutcast Contient toutes les fonctions concernant l'API de ShoutCast

1.2. Database Dossier

1.2.1. Configuration Contient la configuration de la base de données, ainsi que le SEED

1.2.2. DbContext Crée un contexte de la base de données en utilisant une connexion string

1.2.3. Repository Class générique permettant d'effectuer les actions de base sur la base de données (CRUD)

2. BLL Projet Business logic layer

2.1. AlbumData Permet d'effectuer des actions sur les données relatives aux albums

2.2. ArtistData Permet d'effectuer des actions sur les données relatives aux artistes

2.3. RadioData Permet d'effectuer des actions sur les données relatives aux radios

2.4. TrackData Permet d'effectuer des actions sur les données relatives aux morceaux

2.5. SearchData Permet d'effectuer des actions de recherches

2.6. FavoriteData Permet d'effectuer des actions sur les données relatives aux favoris

2.7. GeneralData Permet d'effectuer des actions sur les données relatives aux fonctionnalités générales de l'application

3. Presentation Projet Presentation

3.1. Helper Dossier contenant les classes utiles à la logique de l'application

3.1.1. Context Gère le contexte applicatif du lecteur, contiendra par exemple la liste de lecture

3.1.2. MusicPlayer Fonctions relative au lecteur de musique (play, pause, etc.)

3.1.3. MusicSync Fonctions relative à la synchronisation des pistes musicales depuis l'ordinateur

3.1.4. RightClick Fonctions relative au clic droit

3.2. View Dossier contenant les vues de l'application

3.2.1. Flyout Dossier contenant les flyout de l'application

3.2.1.1. MusicFlyoutView Flyout qui contiendra une liste soit d'album, soit d'artistes, soit de morceaux

3.2.1.2. PropertyFlyoutView Flyout contenant les propriétés d'un morceau

3.2.1.3. ReadingFlyoutView Flyout contenant la liste de lecture

3.2.1.4. RunningFlyoutView Flyout contenant la vue du morceau en cours

3.2.1.5. SettingFlyoutView Flyout contenant les options de l'application

3.2.2. List Dossier contenant les listes de l'application

3.2.2.1. AlbumListView Liste d'albums

3.2.2.2. ArtistListView Liste d'artistes

3.2.2.3. FavoriteListView Liste des favoris

3.2.2.4. GenreListView Liste des genres

3.2.2.5. PlaylistListView Liste des playlists

3.2.2.6. ReadingListView Liste des morceaux en lecture

3.2.2.7. RecentListView Liste des dernières radios écoutées

3.2.2.8. SuggestionListView Liste de suggestion

3.2.2.9. TrackListView Liste de morceaux

3.2.2.10. RadioListView Liste de radios

3.2.3. MusicView Vues de la page musique

3.2.4. PlaylistView Vues de la page playlist

3.2.5. RadioView Vues de la page radio

3.2.6. SearchView Vues de la page recherche

3.2.7. SmallPlayerView Vues du petit lecteur de musique

3.3. ViewModel Dossier contenant les viewmodels

3.3.1. MainViewModel ViewModel parent de tous les autres ViewModels, il est utile en cas de partage entre ViewModels

3.3.2. MainWindowViewModel Gère l'affichage de la fenêtre principale

3.3.3. MusicViewModel Gère l'affichage de toutes les vues musiques (morceaux, artistes, albums et genre)

3.3.4. PlayerViewModel Gère l'affichage du petit player, ainsi que l'affichage du grand player dans la vue de la musique en cours

3.3.5. PlaylistViewModel Gère l'affichage de la vue playlist

3.3.6. PropertyFlyoutViewModel Gère l'affichage de la vue des propriétés

3.3.7. RadioViewModel Gère l'affichage de la vue radio

3.3.8. SettingFlyoutViewModel Gère l'affichage de la vue de configuration

3.3.9. ViewModelLocator ViewModel locator, fait des singletons des ViewModels si l'on veut l'utiliser plusieurs fois avec le même contexte

3.4. MainWindow Vue de la fenêtre layout, qui sera active sur toutes les vues

4. DTO Projet Data access object

4.1. Entity Dossier contenant toutes les entités qui seront dans la base de données

4.1.1. Album Table Album

4.1.2. Artist Table Album

4.1.3. Track Table Album

4.1.4. BaseEntity Contient les propriétés communes à toutes les tables

4.1.5. Context Table Context

4.1.6. Playlist Table Playlists

4.1.7. ExcludeFolder Table ExcludeFolder

4.1.8. Genre Table Genre

4.1.9. Radio Table Radio

4.2. Audio Contient les propriétés communes entre la table Track et la table Radio

5. Shared Projet qui contient les classes susceptibles d'être utilisées dans plusieurs layers

5.1. MusicFile Effectue des transformations sur les fichiers audios

3.2.5 Base de données

Normes ETML

Je n'ai pas respecté les normes de codage ETML au niveau de la base de données, car avec l'utilisation d'EntityFramework, la base de données devrait au final correspondre à ma structure de données dans le DTO. Certaines classes étant héritées, les noms des entités ne correspondraient plus au nom de leur table.

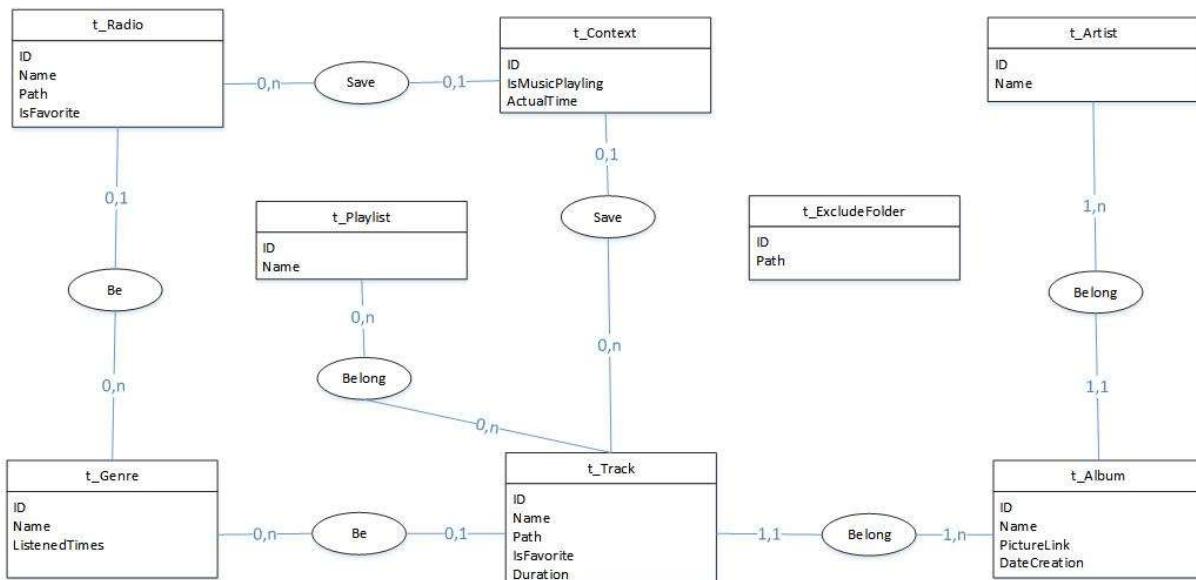
MCD

Figure 2 Modèle conceptuel de données

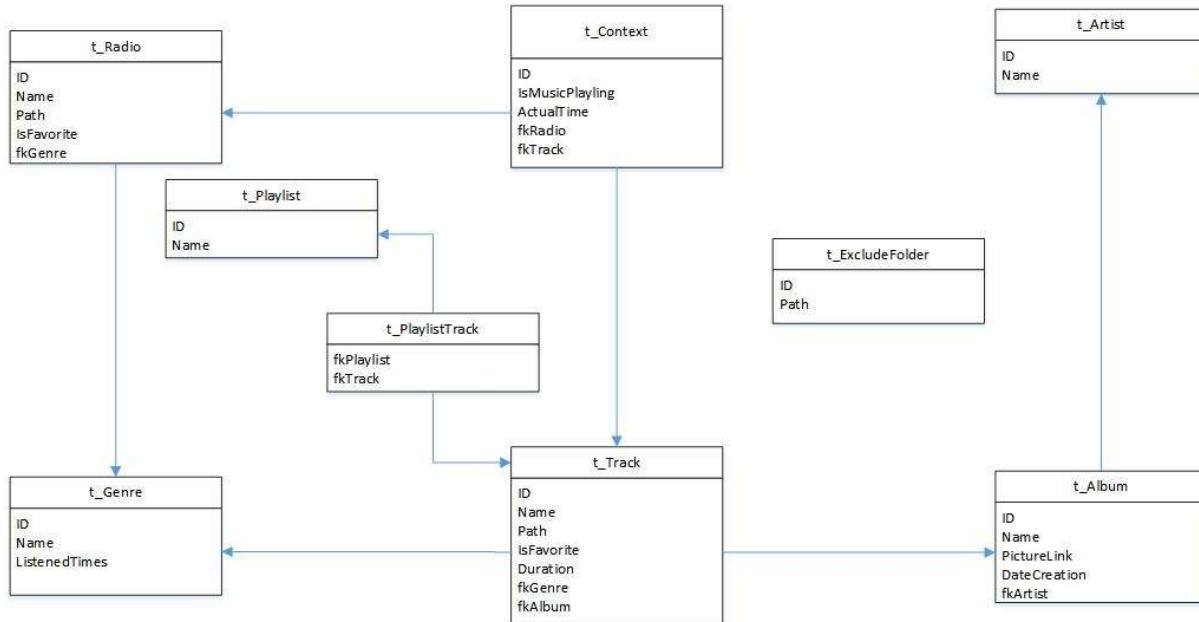
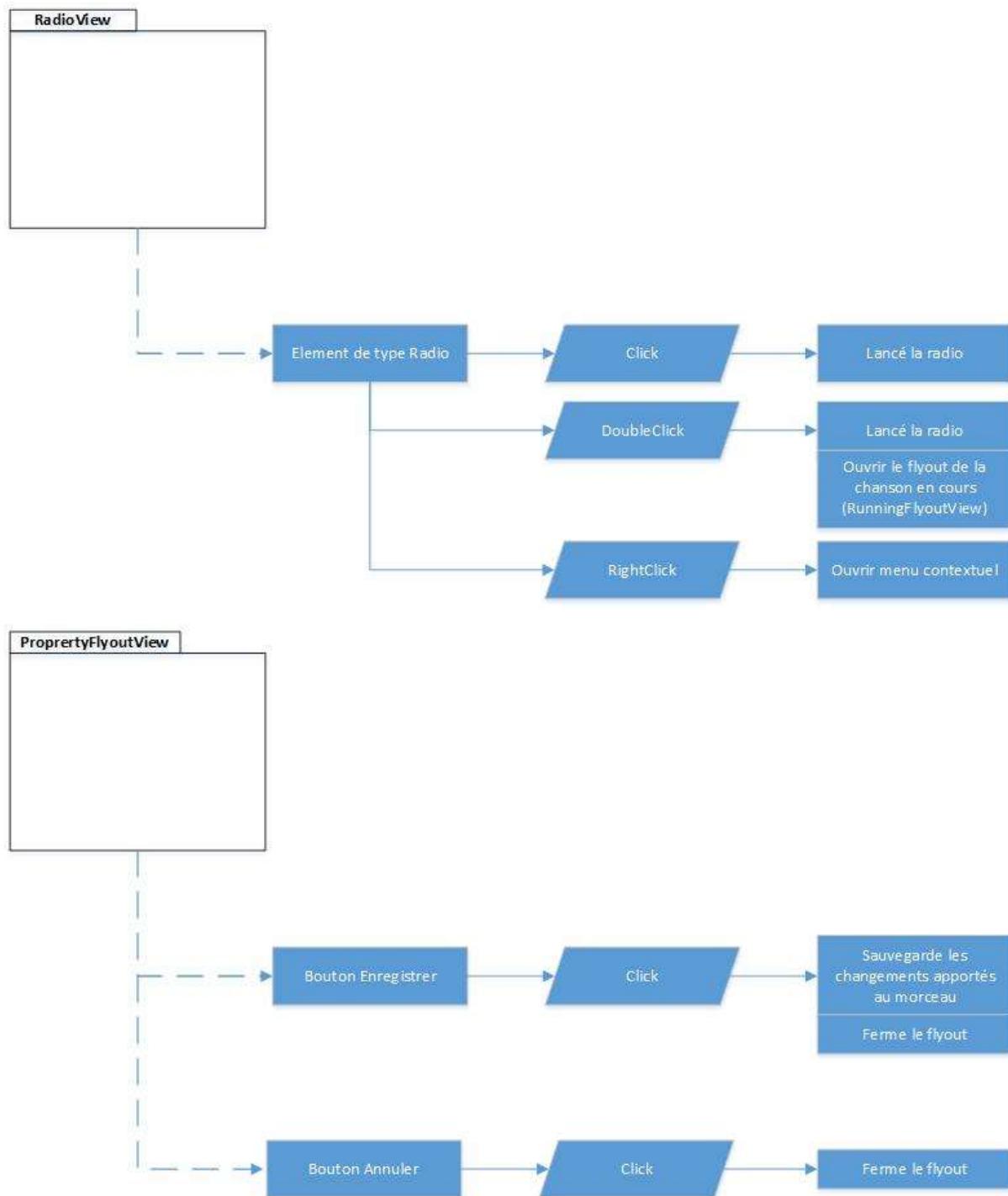
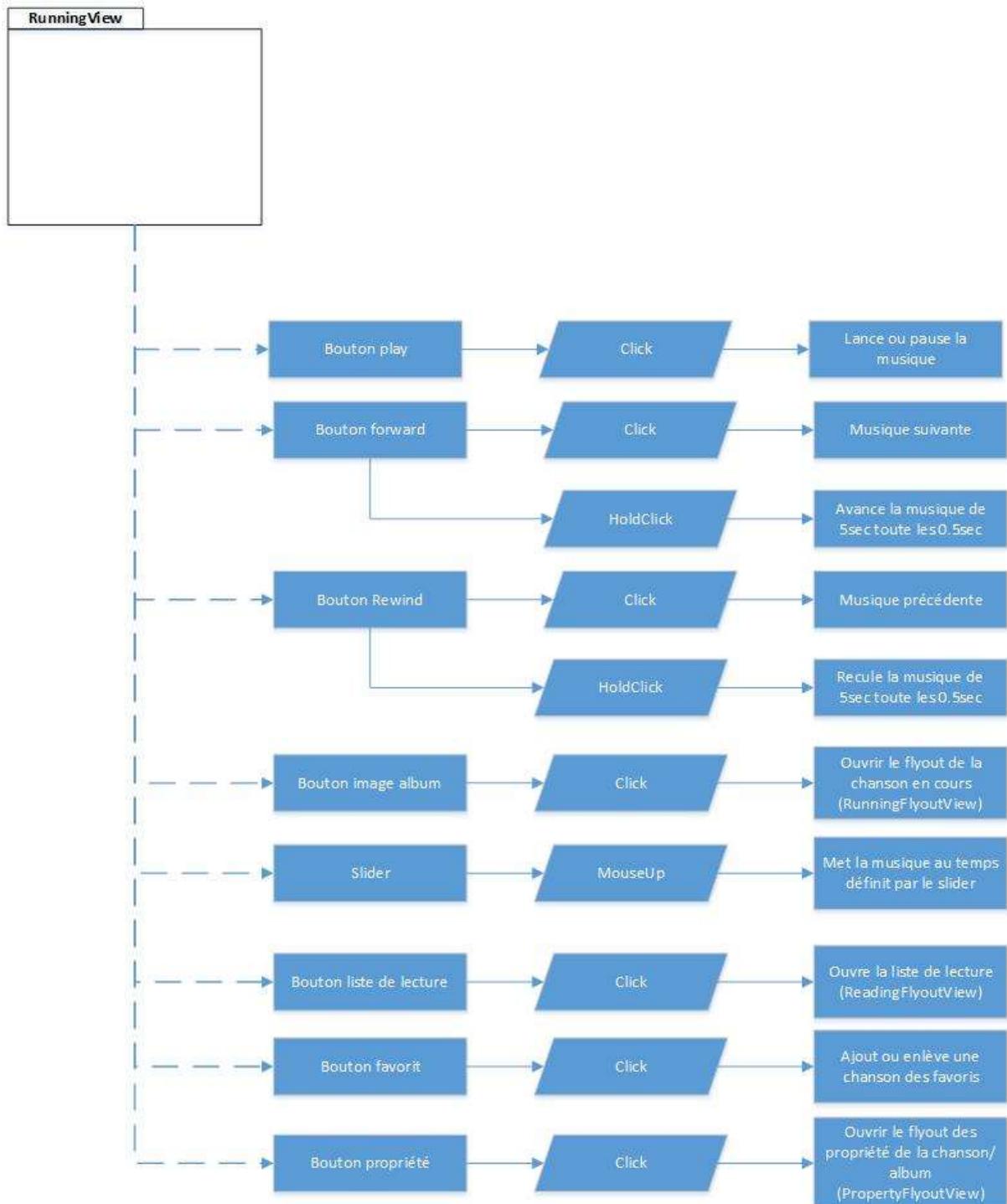
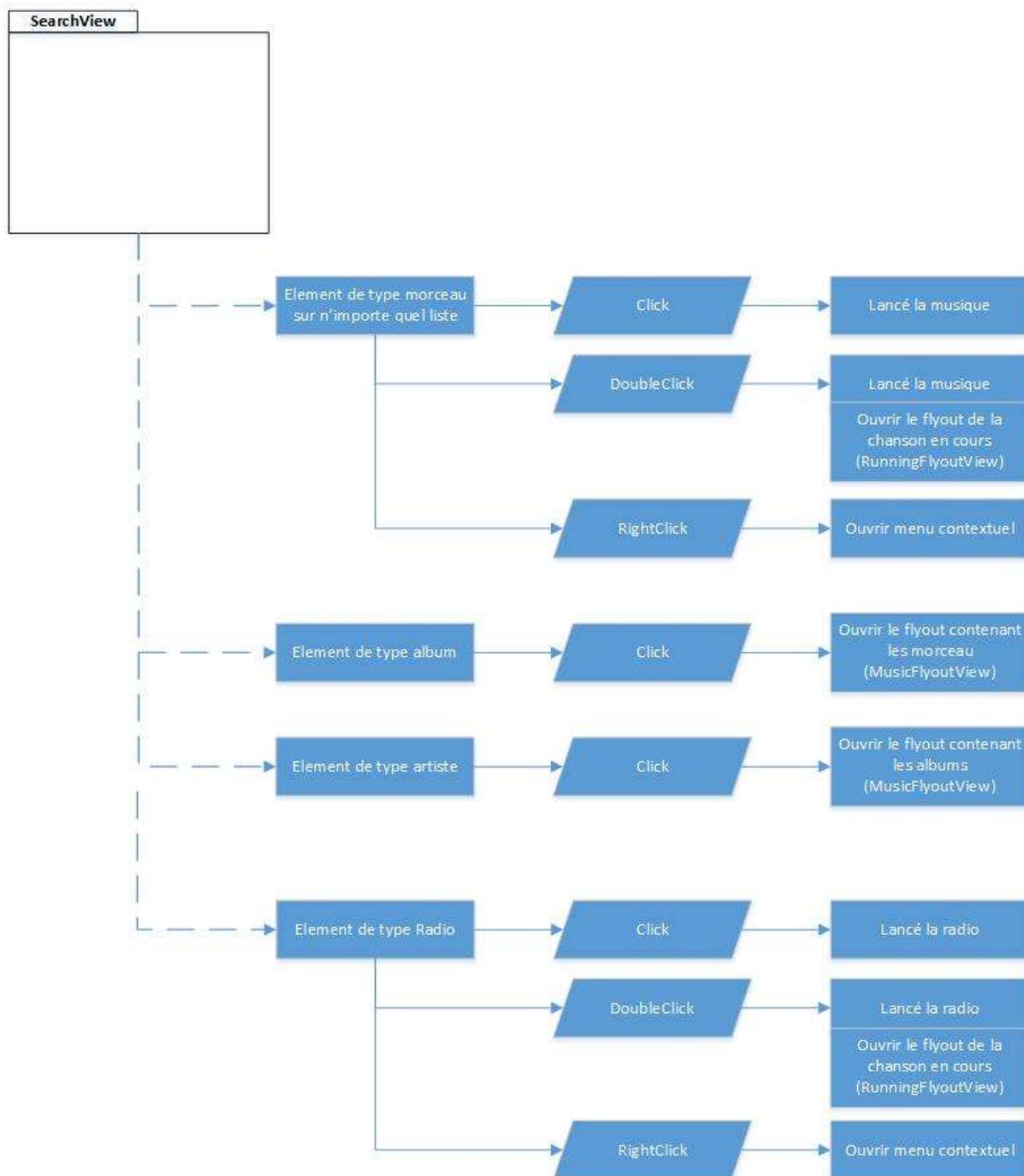
MLD

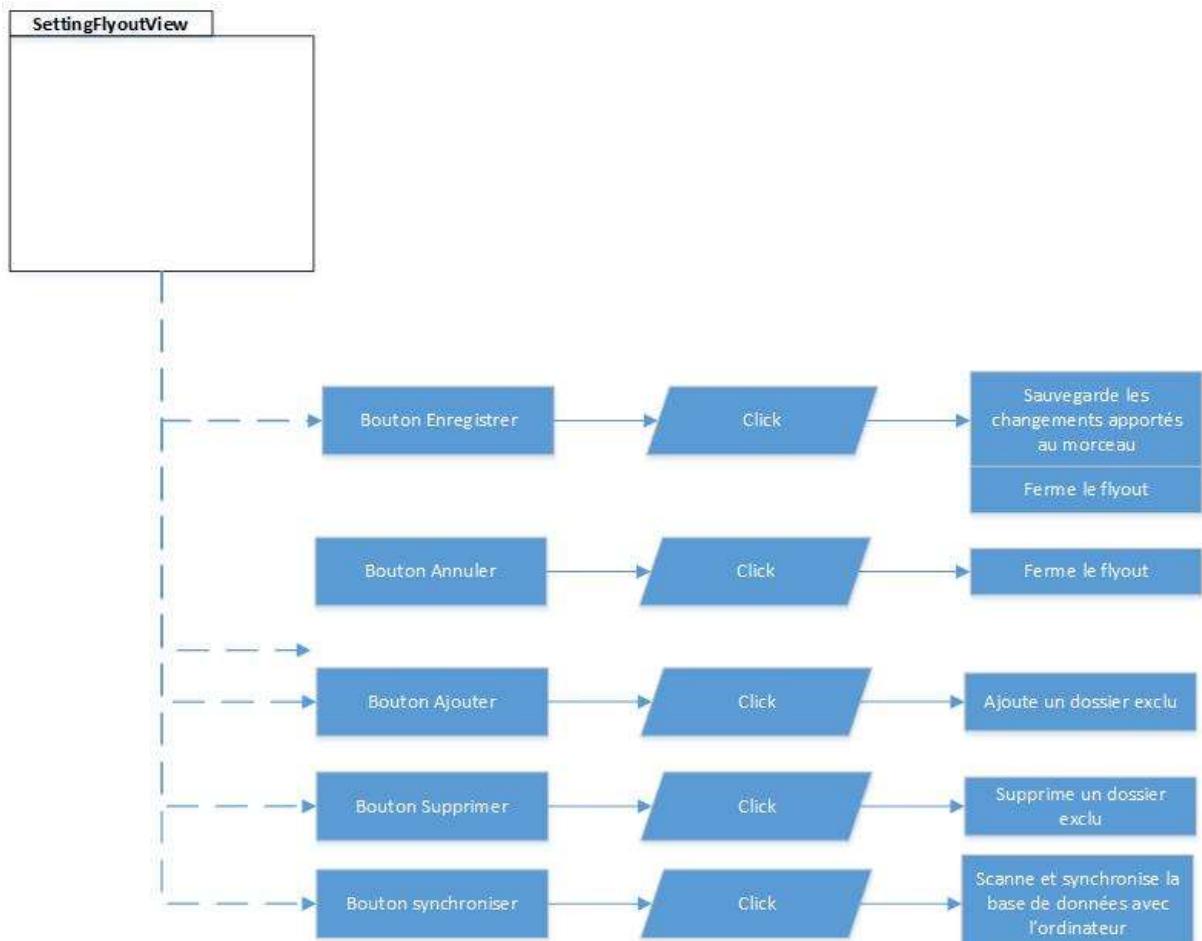
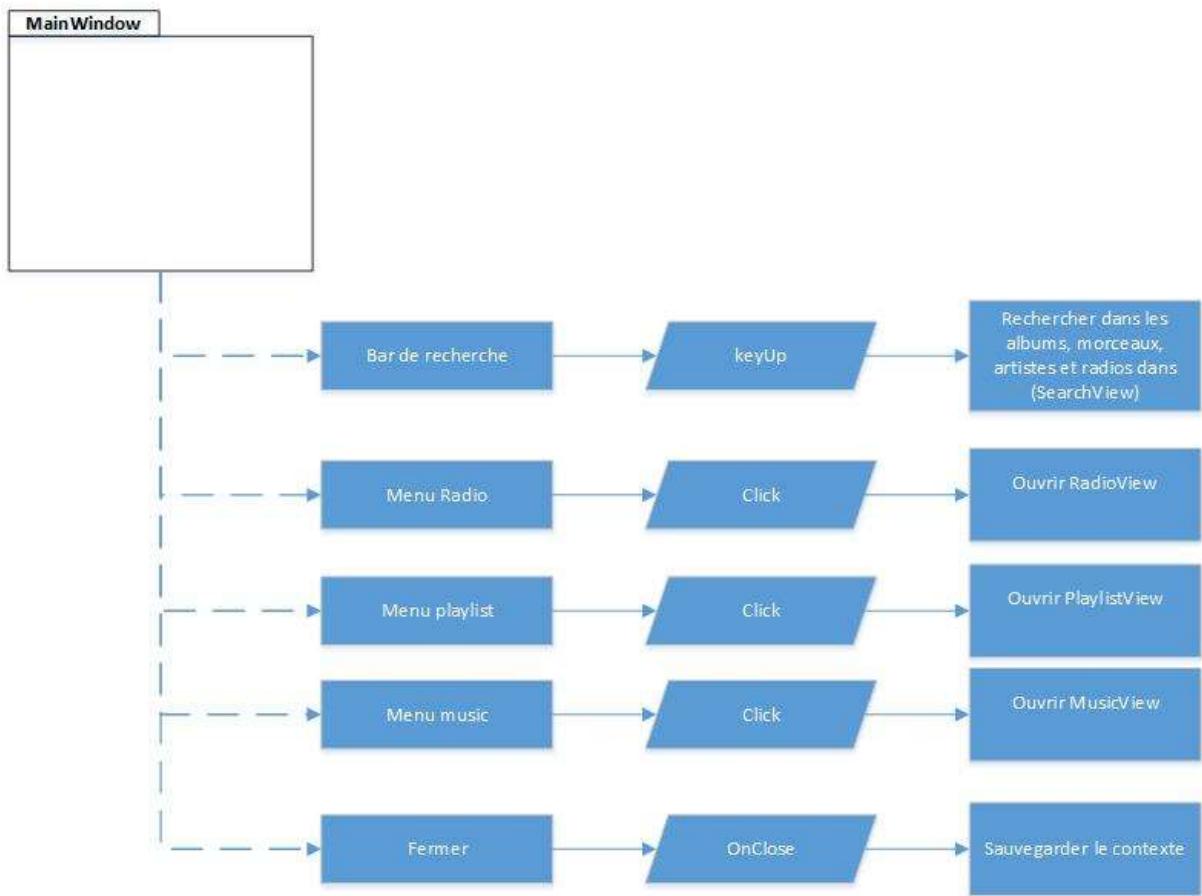
Figure 3 Modèle logique de données

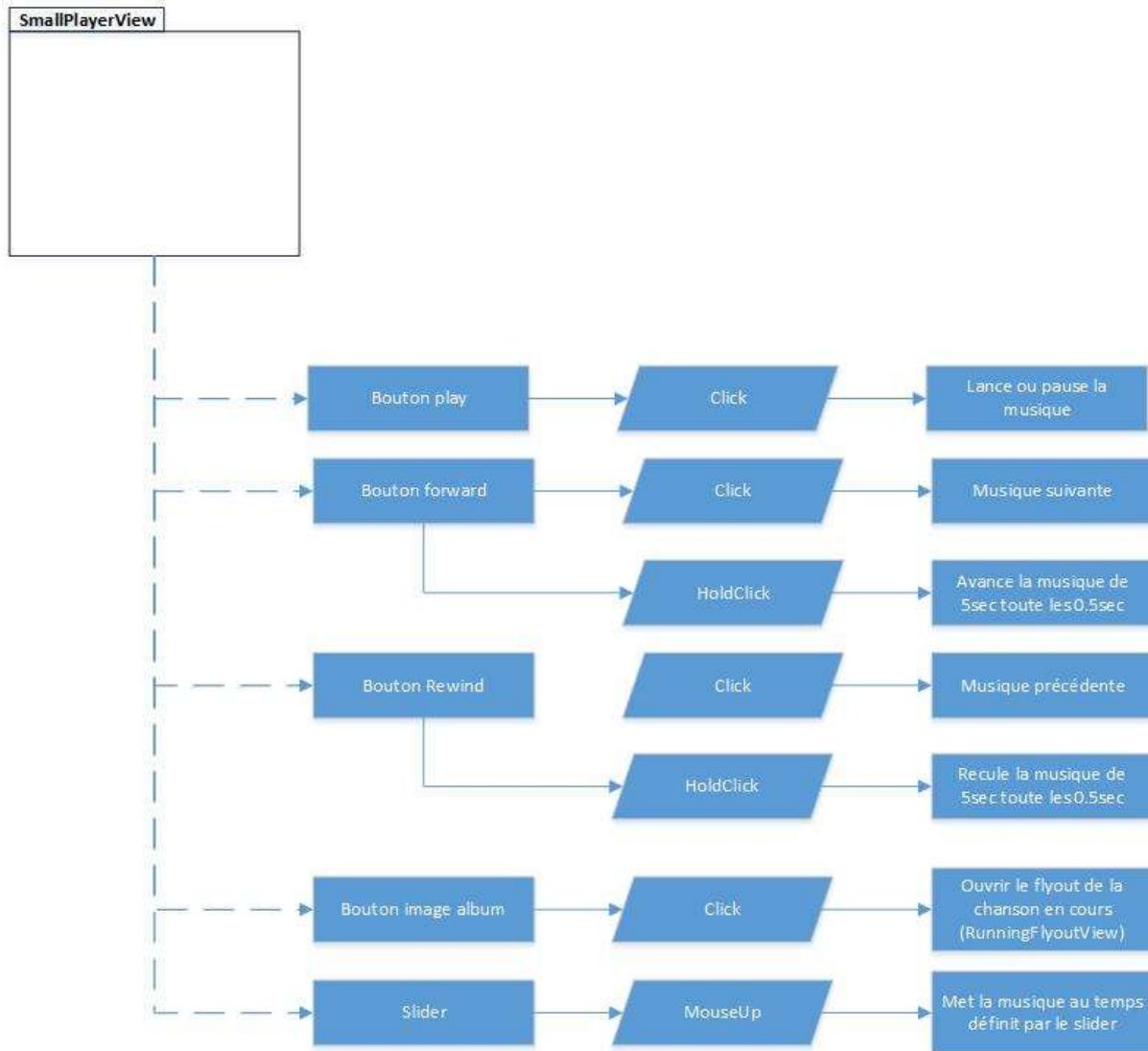
3.2.6 Schéma évènementiel

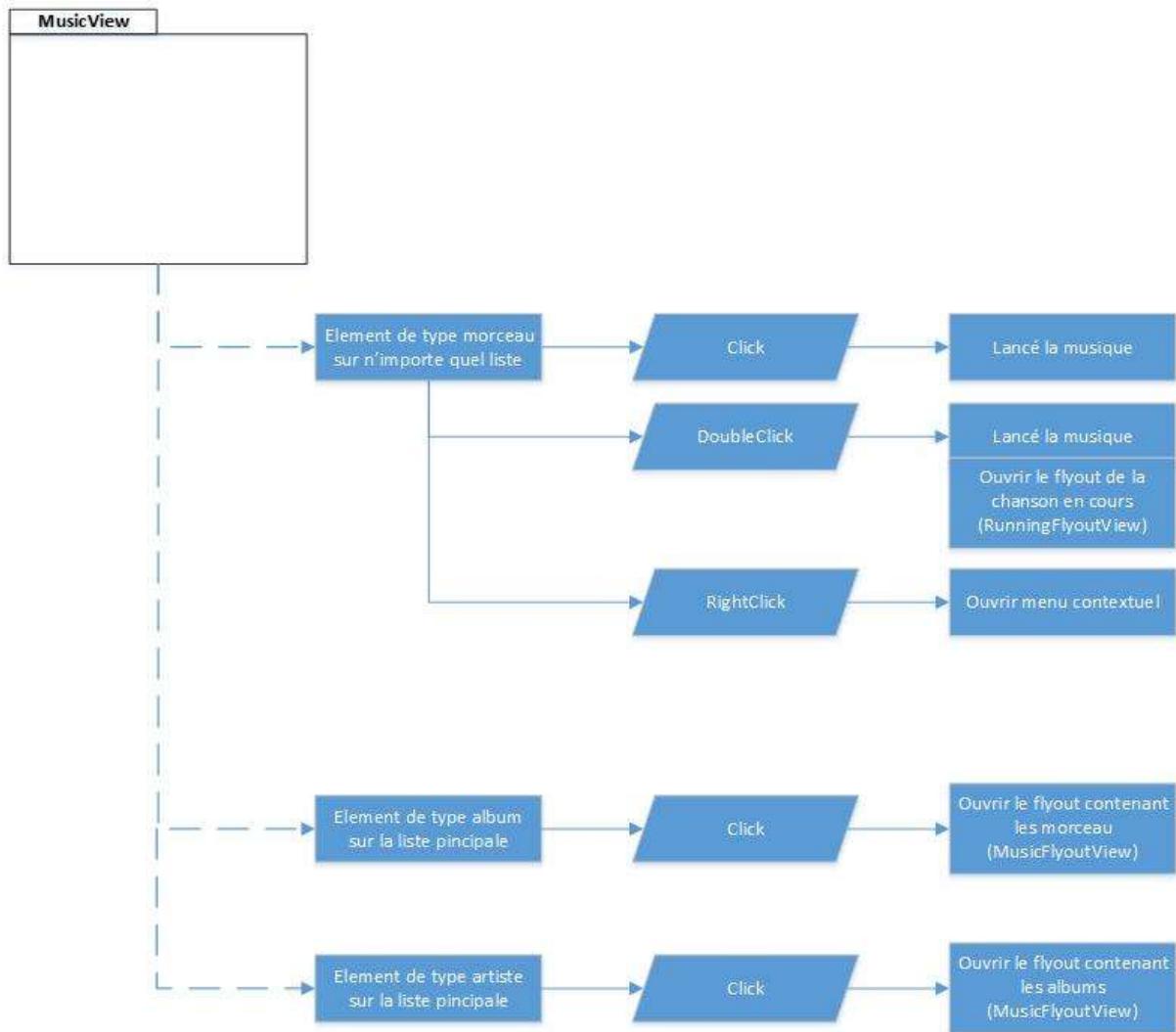


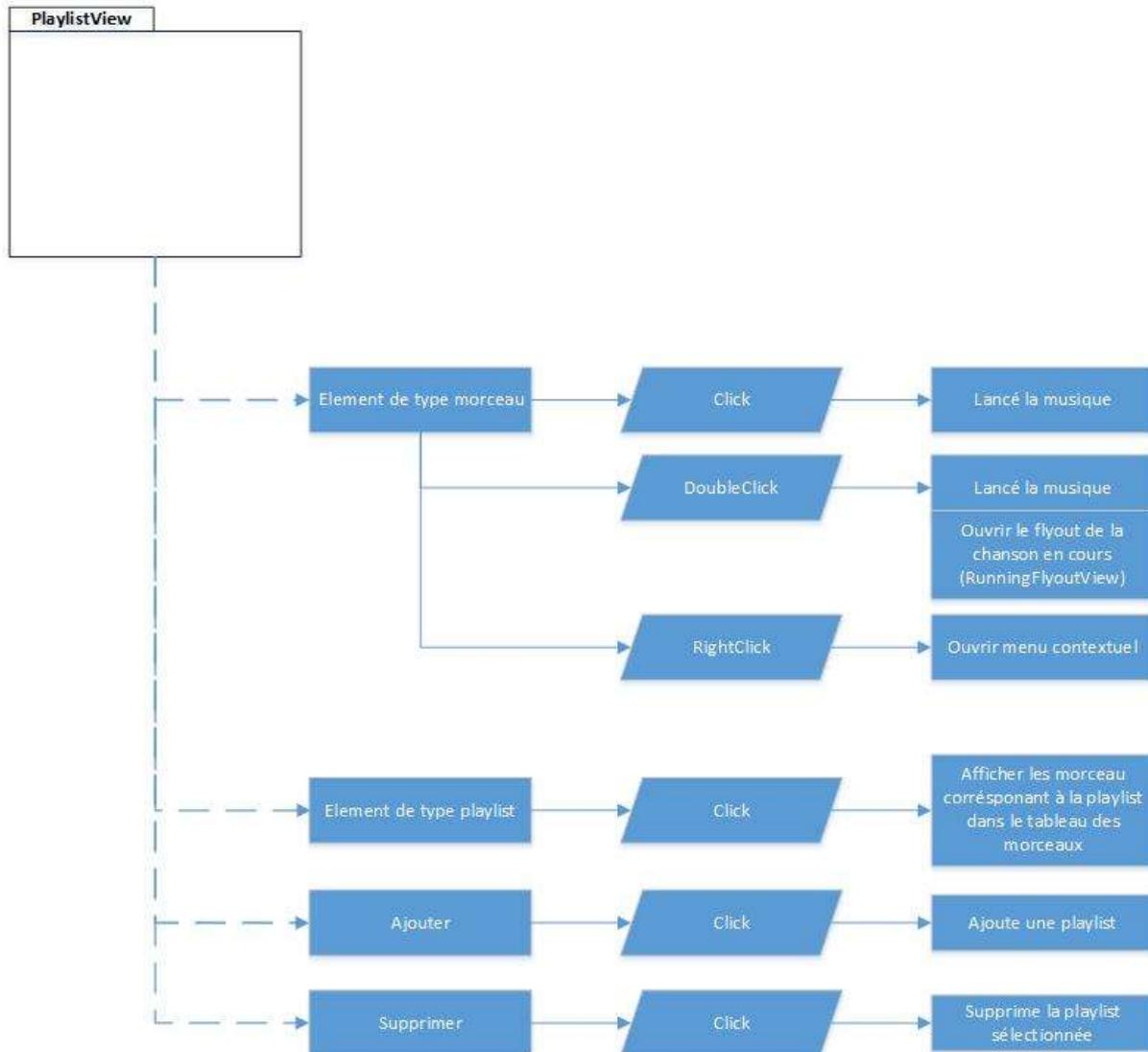












3.3 Conception des tests

Les tests seront effectués grâce à un projet de tests unitaire créé dans la solution. Je n'utilise que de la musique classique pour tester mon application (fichiers non soumis à la loi du copyright).

Suite aux conseils de monsieur Sahli, l'application devra supporter au moins le MP3, le WAV et le WMA. Ces trois formats permettront de couvrir 90 %¹ des flux renvoyés par les Web radio.

Je pratique le développement de type TDD (Test Driven Development), donc tous les tests sont écrits avant la création de l'application. Tous les tests, au départ, sont échoués. Plus le développement avance, plus les tests devront se passer correctement.

AudioTest

<i>LoadData()</i>	Crée une classe artiste, album et morceau.
<i>MusicTestFormat()</i>	Teste les 3 formats obligatoires en essayant de les jouer (WAV, MP3, WMA)
<i>MusicOperationTrack()</i>	Teste les opérations telles que l'avance rapide, la pause
<i>MusicOperationRadio()</i>	Teste les opérations telles que la pause
<i>CheckTrackInfo()</i>	Vérifie que les données reçues par la fonction sont identiques aux données du fichier de musique
<i>FavoriteAddRemovTrackAndRadio()</i>	Ajoute puis supprime une radio et un morceau de la liste des favoris
<i>CreatePlaylist()</i>	Création d'une playlist
<i>CheckPlaylist</i>	Teste les opérations de bases sur la playlist (précédent, suivant) ; ajoute puis supprime un morceau de la playlist ; supprime la playlist.

¹ Selon M.Sahli

ContextTest

<i>CreateAndSetContext()</i>	Crée un contexte puis essaie de l'attribuer à l'application.
<i>ReadingListOperation()</i>	Teste la liste de lecture avec les opérations de base telles que suivant et précédent ; teste l'ajout et la suppression d'élément dans la liste de lecture.

DatabaseTest

<i>CheckConnection()</i>	Teste la connexion à la base de données
--------------------------	---

RadioTest

<i>CheckAPIConnection()</i>	Vérifie que la connexion avec l'API s'effectue correctement
<i>CheckApiData()</i>	Teste que les données reçues par l'API sont correctes
<i>CheckRadioInfo()</i>	Vérifie que les données de l'élément radio correspondent aux données attendues

SyncTest

<i>AnalyseFolderTree()</i>	Vérifie que l'arborescence de fichier contient et analyse bien tous les dossiers
<i>SyncFolder()</i>	Tente de synchroniser un dossier test

4 PLANIFICATION DÉTAILLÉE

Semaine 4		
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?
Couche BLL (Business Logic Layer)	15	Gestion des erreurs dans la BLL, et renvoi de message à l'utilisateur de manière user friendly
Couche BLL (Business Logic Layer)	24	Réserve pour les imprévus et documentation de la couche BLL
Documentation	3	
Congé	57	Asc.
Total semaine	99	Max. 99

Semaine 5		
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?
Couche Présentation	10	Ecriture de la logique de l'application : la gestion du contexte de l'application lors de la fermeture et du démarrage
Couche Présentation	5	Gestion de la configuration de l'application
Couche Présentation	15	Réserve pour les imprévus et documentation de la couche présentation
Création de l'installateur	20	Utilisation de ClickOnce pour créer un installateur simple
Tests	10	Test de l'application sur différents windows
Manuel utilisateur	21	
Documentation	5	
Congé	13	Examen ECG
Total semaine	99	Max. 99

Semaine 6		
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?
Congé	13	Pentecôte
Manuel utilisateur	6	
Documentation	4	Finalisation de la documentation, impression etc
Total semaine	23	Max. 99

Semaine	1	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?
Planification initiale	27	
Arborescence de fonctions et de classes	8	Création de l'arborescence de fonctions et classes
Analyse	9	Explication de la structure de classes et du fonctionnement du code
Analyse	8	Création d'une maquette de l'UI
Analyse	15	Schéma événementiel
Analyse	8	Création du MCD et MLD
Analyse	8	Création de la structure des tests
Analyse	10	Explication des tests
Documentation	6	
Total semaine	99	Max. 99

Semaine	2	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?
Analyse	10	Planification détaillée
Création du MPD	5	
Création de la base de données	3	Création de la DB dans SQLite
Mise en place de	2	Installation de MVVMLight et de MahApps
Création des tests	2	Création des données de tests
Création des tests	18	Génération des tests unitaires
UI	10	Création du layout (MainWindow)
UI	3	Création des listes
UI	3	Création du petit player
UI	10	Création des flyouts
UI	14	Création des propriétés dans les viewmodels
Création des objets		Création des objets DTO
DTO (Data transfer object)	13	
Documentation	6	
Total semaine	99	Max. 99

Semaine	3	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?
Couche DAL (Data Access Layer)	3	Liens avec la base de données
Couche DAL (Data Access Layer)	3	Configurer entity framework pour lier les données de la db avec les objets DTO
Couche DAL (Data Access Layer)	4	Ajout des dépendances pour faire fonctionner SQLite
Couche Présentation	10	Création des liens entre l'interface et les viewmodels via les Commands
Couche Présentation	3	Ajout des actions à affectuer lors des événements
Couche Présentation	10	Vérification des données entrées par l'utilisateur
Couche Présentation	8	Création et gestion du clic droit
Couche Présentation	8	Écriture de la logique de l'application : de la musique
Couche Présentation	11	Écriture de la logique de l'application : la gestion de la synchronisation
Couche BLL (Business Logic Layer)	15	Réception, et mise en forme des données envoyées par la vue
Couche BLL (Business Logic Layer)	18	Vérification des données reçues par la vue
Documentation	6	
Total semaine	99	Max. 99

5 RÉALISATION

5.1 Dossier de Réalisation

5.1.1 MPD

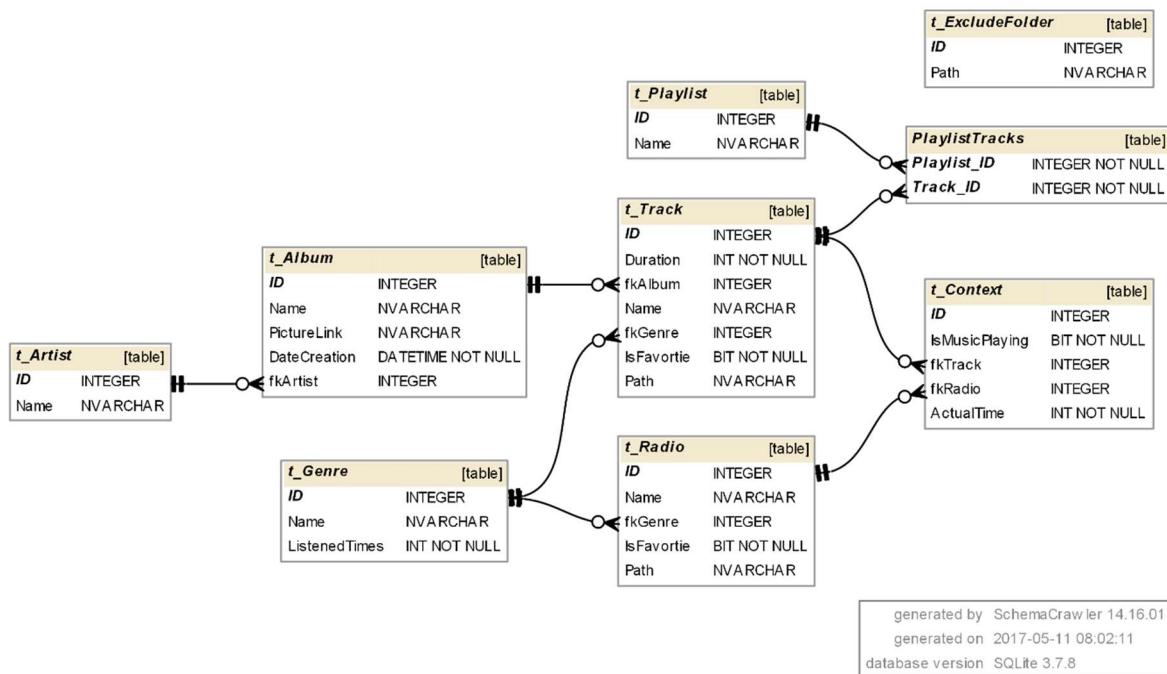


Figure 4 Model physique de données

5.1.1 Dossiers exclus

Au début de mon application, j'étais parti sur le fait de gérer la synchronisation par rapport aux dossiers exclus. Après mûre réflexion, nous avons décidé d'inclure les dossiers à analyser plutôt qu'à les exclure.

5.1.2 Lecture des radios

Pour la lecture de radio, j'ai rencontré un problème lors de la lecture des flux AAC, ces flux étant encodés en AAC (format que NAudio ne supporte pas).

Pour lire les flux AAC, j'avais l'option d'utiliser la fonction utilisant la DLL Média Fondation, celle-ci est une librairie créée par Microsoft utilisant les DLL de Windows Media Player. Cependant, bien qu'ayant une fonction permettant de lire des streams, celle-ci dispose le stream après la première itération, ce qui me laissait avec seulement quelque seconde de musique.

Pour résoudre ce problème, moi et Monsieur Melly avons choisi la solution de créer un flux alternatif encodé en MPEG (*format mp3*) à partir du flux AAC. Pour ce faire, j'ai utilisé la DLL LAME, permettant de convertir un flux AAC en MPEG.

5.1.3 Stream radio

J'ai rencontré une difficulté toute particulière lors de la lecture des streams radio. Les streams utilisant la version 2.x de Shoutcast étaient lus sans problème. Tandis que ceux utilisant la version 1 me retournaient une erreur de type : « Serveur violation ».

Ce problème était dû à un problème de header. Pour résoudre ce problème, j'ai ajouté cette ligne pour empêcher que l'erreur survienne

```
<system.net>
  <settings>
    <httpWebRequest useUnsafeHeaderParsing ="true"/>
  </settings>
</system.net>
```

Synchronisation.

La synchronisation sur un ordinateur de moyen de gamme prend environ 20 secondes pour 300 morceaux. D'une synchronisation pour simplement mettre à jour les fichiers, le temps passe à 15 secondes.

5.2 Modifications

Modification de la structure de la base de données

17.05.2017 : j'ai supprimé le dossier commun de la gestion du clic droit. J'ai trouvé plus pratique de le gérer directement dans les vues concernées.

17.05.2017 : J'ai créé un fichier PlaylistData pour gérer les playlists plus simplement. Précédemment, les playlists étaient gérées directement dans le TrackData ce qui pouvait porter à confusion.

18.05.2017 : J'ai dû modifier la structure de mon élément radio en ajoutant notamment les données venues de l'API (description, le format, l'URL du logo et l'ID Shoutcast).

19.05.2017 : Ajout de la donnée : LastListen, dans le modèle des radios pour avoir la date de dernière lecture. Celle-ci sera utile lors de la récupération des radios récente.

5.3 MDP final

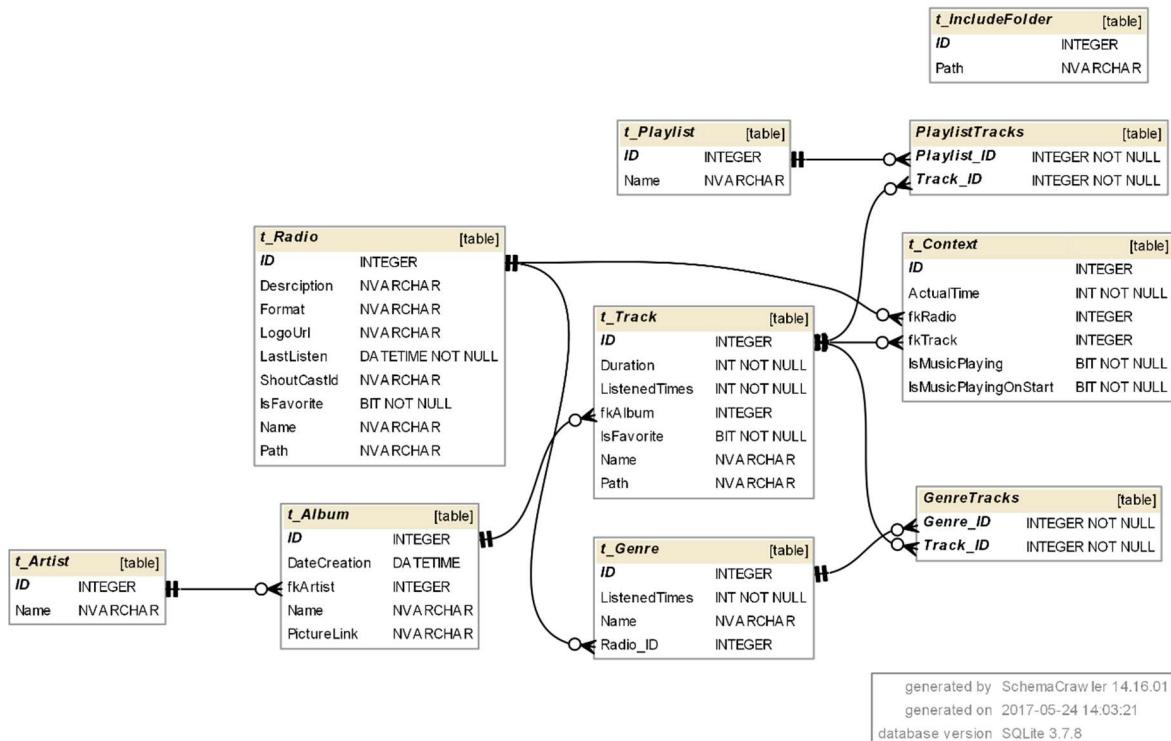


Figure 5 Model physique de données à la fin du projet

Sur le MPD final, nous pouvons voir quelques changements, notamment la table « **t_Radio** » qui se retrouve bien plus fournie.

La table **t_ExcludeFolder** qui est devenue **t_IncludeFolder**.

La table **t_Context** a également gagné l'attribut « **IsMusicPlaylingOnStrat** » qui représente le choix de si oui ou non la musique doit directement se lancer au démarrage.

Il y a également les entités **listenedTimes** dans « **t_Track** » et « **t_Genre** », qui ne sont pas utilisées, mais que j'ai mises pour respecter le cahier des charges. Dans le futur, si elles sont utilisées, elles serviront à traquer les habitudes de l'utilisateur.

5.4 MVVM

J'ai choisi le pattern MVVM, car il permet de travailler sur les vues indépendamment des ViewModel. Les vues sont alors indépendantes, donc si un autre logiciel utilise la même vue, je n'aurai qu'à faire un simple copier/coller. La logique elle aussi est bien plus réutilisable, car non polluée par les éléments de la vue.

Le paterne me permet également d'utiliser la même logique sur plusieurs vues différentes et vice-versa. Cet avantage me permet d'économiser énormément de lignes de codes.

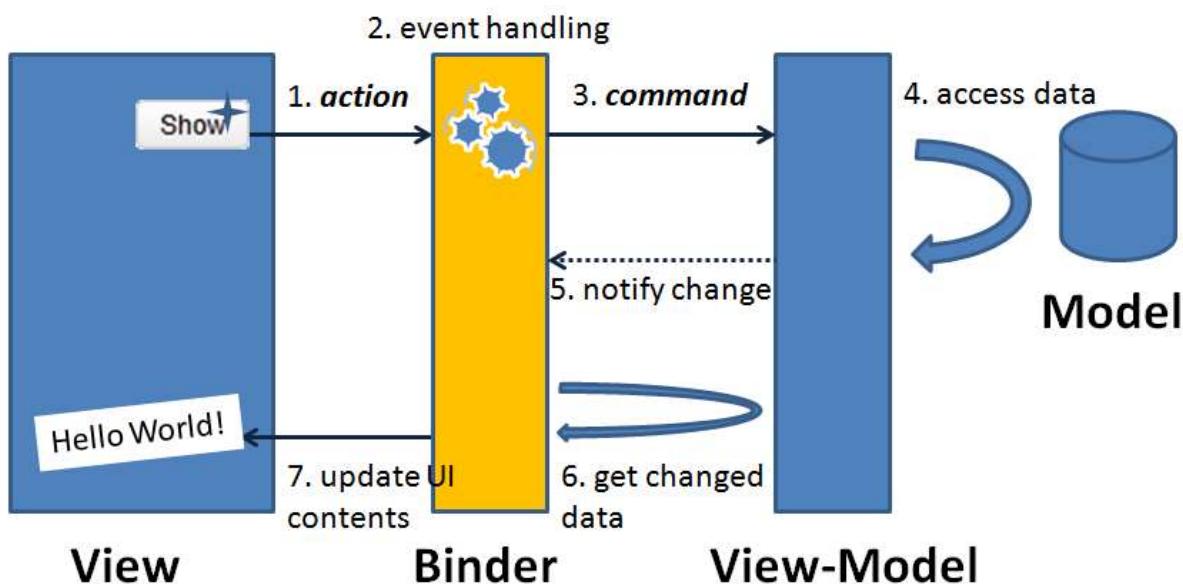


Figure 6 Schéma MVVM

https://www.zkoss.org/_w/images/f/fb/SmallTalk_MVVM_HELLO_FLOW.png

5.5 Extrait de code

```

295     /// <summary>
296     /// Move the position of the cursor on the slide bar
297     /// </summary>
298     /// <param name="changeValue"></param>
299     /// <param name="actionOnClick"></param>
300     private void MovePosition(int changeValue, Action actionOnClick)
301     {
302         _isMouseHoleded = true;
303
304         //wait that it's actually a true holding
305         Thread.Sleep(150);
306
307         //if it was a click, stop it there
308         Application.Current.Dispatcher.Invoke(() =>
309         {
310             if (Mouse.LeftButton == MouseButtonState.Released)
311                 actionOnClick();
312         });
313
314         //Check if still holding
315         while (_isMouseHoleded)
316         {
317             Thread.Sleep(100);
318
319             Application.Current.Dispatcher.Invoke(() =>
320             {
321                 Mouse.SetCursor(Cursors.Hand);
322                 if (Mouse.LeftButton != MouseButtonState.Released)
323                 {
324                     SliderValue += changeValue;
325                     return;
326                 }
327
328                 Mouse.SetCursor(Cursors.AppStarting);
329                 _isMouseHoleded = false;
330             });
331         }
332     }

```

Cette fonction bouge le curseur du slider sur la barre du temps. Elle est appelée lors que l'utilisateur appuie en continu sur avance rapide ou retour rapide.

À la ligne 305, on attend 150 ms pour faire la différence entre un clic et un appui long. Ensuite, de la ligne 308 à 312, on vérifie le statut du clic, 150 ms plus tard. Si le clic est relâché, lancer la fonction assignée au clic (musique suivante ou précédente). Sinon tant que le clic est enfoncé, chaque 100 ms, on avance ou recule la musique d'un certain temps donné.

Le Application.Current.Dispatcher sert à exécuter la commande dans le thread principal de l'application. Donc même en étant dans un thread différent, les éléments visuels sont accessibles et modifiables.

6 TESTS

6.1 Dossier des tests

Version de l'application testée	1.0.0
Date du test	29.05.2017
Nom du testeur	Eric Preisig

Scénario 1 : Synchronisation

Étape	Description	Remarque
Quoi	La musique des dossiers cibles se synchronise	Lors du premier démarrage, on demande à l'utilisateur de choisir s'il veut synchroniser son répertoire de musique
Quand	Lors du premier démarrage ou lors du clic sur le bouton synchroniser depuis la configuration	
Résultat	Les musiques qui ne sont plus dans les dossiers cibles se suppriment et les dossiers cibles sont analysés récursivement. Vérifie aussi que les fichiers déjà présents dans la base de données existent toujours.	

Scénario 2 : Lecture de musique

Étape	Description	Remarque
Quoi	Lit une musique	
Quand	Lorsque l'utilisateur clique sur un morceau depuis la page musique	
Résultat	La musique se lance, le volet de la musique s'ouvre et la liste de lecture est mise à jour.	

Remarque :

Si la musique a été déplacée, le logiciel avertit l'utilisateur, puis supprime la musique de la base de données ainsi que de la liste de lecture

Scénario 3 : Lecture de radio

Étape	Description	Remarque
Quoi	Lit une radio	
Quand	Lorsque l'utilisateur clique sur une radio depuis la page radio	
Résultat	La radio se lance	

Remarque :

Si la musique a été déplacée, le logiciel avertit l'utilisateur, puis supprime la musique de la base de données ainsi que de la liste de lecture

Scénario 4 : Recherche

Étape	Description	Remarque
Quoi	La recherche de morceaux, artistes, albums et radios	
Quand	L'utilisateur appuie sur la touche « enter » après avoir écrit un mot dans la barre de recherche	
Résultat	La recherche s'effectue, elle est rapide. La recherche des radios ne bloque pas l'application et apparaît peu après le reste.	

Remarque :

La recherche est rapide, mais elle peut devenir plus lente selon le nombre de fichiers. J'ai testé la recherche avec environ 4000 fichiers, avec un résultat prenant moins de 2 secondes.

Scénario 5 : L'application démarre

Étape	Description	Remarque
Quoi	Le contexte se récupère correctement	
Quand	Quand l'application s'ouvre	
Résultat	Si l'utilisateur écoutait un morceau, celui-ci se lance correctement et reviens au moment précédent l'extinction du logiciel. Si l'utilisateur écoutait une radio, celle-ci se lance correctement	

Remarque :

Si une erreur intervient pendant la récupération (la radio ou le morceau n'existe plus, la radio est indisponible...), le contexte est alors supprimé

Scénario 6 : Suppression

Étape	Description	Remarque
Quoi	Suppression d'un morceau	
Quand	Quand l'utilisateur effectue un clic droit sur un morceau et le supprime	
Résultat	S'il le supprime de disque dur, le fichier est alors supprimé du disque. La liste des morceaux se met à jour, ainsi que la liste de lecture. En cas de suppression de la bibliothèque, le fichier est simplement supprimé de la bibliothèque de l'application (ainsi que des listes qui utilisaient le morceau)	

Remarque :

Si le fichier n'est pas supprimable, une erreur se produit

Scénario 6 : Mise en favoris

Étape	Description	Remarque
Quoi	Met un morceau ou une radio en favoris	
Quand	Lorsque l'utilisateur clique sur l'étoile	
Résultat	Met le morceau ou radio en favoris, et l'affiche dans la liste des favoris de sa vue.	

6.2 Tests unitaires

J'ai dû supprimer plusieurs tests, car la manière dont ils fonctionnaient n'était pas compatible avec la manière dont le logiciel fonctionne. Je pensais pouvoir créer des ViewModel dans les tests unitaires, malheureusement, je ne peux pas. Aussi toutes les interactions avec la musique sont bloquées (play/pause, etc.).

J'ai aussi supprimé toutes les interactions avec la base de données dans les tests, et essayé de les adapter grâce à du code en dur.

7 CONCLUSION

7.1 Bilan des fonctionnalités demandées

Fonctionnalité	Résultat
<i>Synchronisation au premier lancement</i>	OK
<i>Habitudes d'écoutes de l'utilisateur</i>	OK
<i>Synchroniser à la demande</i>	OK
<i>10 dernière radio écoutée</i>	OK
<i>Rechercher des chansons</i>	OK
<i>Concevoir des playlists</i>	OK
<i>Jouer des chansons (play, pause, slider, suivant, précédent)</i>	OK
<i>Lecture continue, aléatoire, répétée</i>	OK
<i>Recherche des radios sur Internet</i>	OK
<i>Information sur la chanson en cours</i>	OK
<i>Lire un flux de radio</i>	OK
<i>Reprendre le dernier contexte applicatif</i>	OK

Suite aux tests de M.Sahli, une liste de fonctionnalité et de changement pratique a été faite.
Si je devais continuer le logiciel, j'améliorerais ces points précis.

Remarques et amélioration

- Ne pas bloquer l'application lors de la synchronisation et permettre à l'utilisateur de voir les morceaux synchronisés
- Se souvenir des tris effectués sur les vues lors du changement de vue
- Prendre en compte les tris pendant la création de la liste de lecture
- Afficher le morceau en cours en surbrillance et déplacer le scroll à celui-ci

7.2 Bilan de la planification

Voici la liste de tâches qui m'ont pris significativement plus ou significativement moins de temps que planifier. (Le temps est en quart d'heure).

Tâche	Prévus	Effectuer	Raison
Planification initiale	27	14	J'ai fini plus tôt mon analyse initiale. Ce qui m'a permis de commencer mon analyse plus tôt
Création de la base de données	3	0	J'ai configuré EntityFramework pour la créer automatiquement
Mise en place de l'architecture MVVM	2	0	Je l'ai fait durant la création de l'arborescence de classe
Création de tests unitaire	20	5	Je n'avais pas fait beaucoup de tests unitaires auparavant, donc j'ai surestimé le temps
UI	40	23	J'ai utilisé plusieurs fois les mêmes vues, j'ai donc gagné beaucoup de temps
BLL Présentation	72 80	44 104	J'ai placé plusieurs fonctions du BLL à la couche présentation
Installateur	20	5	L'utilisation de ClickOnce (installateur intégré à Visual Studio) m'a permis de créer l'installateur rapidement. Cependant, j'ai eu beaucoup de bugs à résoudre par rapport à celui-ci
Manuel utilisateur	27	9	J'ai surestimé le temps de travail que me prendrait la documentation utilisateur

7.3 Bilan personnel

Si le projet était à refaire, je ne changerais que l'installateur : ClickOnce, car bien que très pratique au niveau du déploiement (il se dépolit très en un clic, et n'a pas besoin des droits administrateurs pour s'installer). Il introduit énormément de bugs, notamment au niveau des dépendances dont le programme a besoin.

Ce projet m'a permis d'apprendre à gérer des flux audios en continu. Il m'a aussi appris à utiliser la librairie NAudio, celle-ci pouvant être très utile dans d'autre projet contenant de la musique.

Si ce projet était à continuer, j'ajouterais en priorité une fonction permettant de remplir automatiquement les métadonnées des morceaux. Permettant ainsi à l'utilisateur d'éviter la tâche fastidieuse de les remplir à la main.

Ensuite, je ferai une option permettant le partage de musique entre utilisateurs, car cette option me semble inédite dans le milieu des logiciels audio.

7.4 Remerciement

Je remercie **M. Sahli**, mon maître de projet, qui en plus de m'assister tout au long du TPI, a aussi testé mon application.

Je remercie **M. Melly**, qui m'a aidé à résoudre un problème qui empêchait la lecture de certains flux radio.

Je remercie **Eric Taylor** qui m'a fourni un ensemble de musique libre de droits. Ce qui m'a permis de tester mon application.

8 WEBOGRAPHIE

Mes sources principales lors de ce TPI étaient :

Stack Overflow: <https://stackoverflow.com>

Microsoft MSDN: <https://msdn.microsoft.com>

Microsoft MSDN forum : <https://social.msdn.microsoft.com>

9 DIVERS

9.1 Journal de travail

Semaine	1	Date: lundi, 1 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Planification initiale	14	Discussion avec le maître de projet et avec M.Rollinet; Rédaction de la planification initiale	
Documentation Analyse	2	Réflexion et mise en page de la future documentation	Rapport CH_3.2.1 : Maquette
Arborescence de fonctions et de classes	15	Création de la maquette sur viso (toutes les pages de l'application ainsi que les flyout)	Voir le GIT : https://github.com/ericpreisig/Preisig-TPI/tree/Dev/GestionAudio
Analyse	5	Création de l'arborescence de fonctions et classes	
Analyse	8	Explication de l'architecture et création du DTO	Rapport CH_3.2.2 : Architecture
Analyse	7	Explication de la structure des classes, description du fonctionnement de toutes les classes	Rapport CH_3.2.3 : Librairie utilisée & Rapport CH_3.2.4 Structure du code
Analyse	15	Création du MCD et du MLD	CH_3.2.5 Base de données
Documentation	3	Schéma événementiel de toutes les vues	CH_3.2.6 Schéma événementiel
Documentation	3	Mise en page des schémas dans la documentation	
Documentation	3	Résolution de problème lié à Word	
Analyse	6	Correction de problèmes sur le MLD et le MCD (il manquait la table playlist), j'ai aussi du refaire les schémas sur viso car l'application que j'utilisais (Jmerise) créait une table intermédiaire erronée.	
Analyse	6	Création de la structure des tests	
Total semaine	99	Max. 99	
Semaine	2	Date: lundi, 8 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Documentation	2	Ajustement du journal de travail en ajoutant des références	
Analyse	2	Création de la structure des tests	CH_3.3 Conception des tests
Analyse	6	Explication des tests	
Analyse	5	Rédaction de la planification détaillée	
Création de la base de données	0	Je laisserais finalement Entity Framework faire la base de données pour moi	
Création du MPD	0	Attente de la base de données pour créer un MPD automatiquement	
Création du MPD	2	Recherche de logiciel pour créer automatiquement le MPD (à tester: schemaspy, schemaCrawler)	
Création des tests unitaires	5	Création des tests unitaires	Voir GIT : projet UnitTest
UI	5	Création du layout (menu/bar de recherche)	Voir GIT : projet Presentation/MainWindow
UI	3	Création des listes avec des datagrids	Voir GIT : projet Présentation/View/List
UI	6	Création des vues flyout	Voir GIT : projet Présentation/View/Flyout
UI	7	Création du reste des vues	Voir GIT : projet Présentation/View
Couche DAL (Data Access Layer)	4	Liens avec la base de données et avec les DTO	
Couche DAL (Data Access Layer)	6	Création des fonctions de récupération de données	Voir GIT : projet DAL
Couche Présentation	5	Affichage des données des morceaux et navigation sur la page Musique	Voir GIT : projet Présentation/ViewModel/MusicViewModel
Rendez-vous	2	Rendez-vous avec M.Sahlí, détection d'un bug lors du changement des métadonnées avec Windows	
Couche DAL (Data Access Layer)	5	Gestion des images d'albums	
Couche Présentation	6	Création des fonctions pour lire la musique	Voir GIT : projet Présentation/View/SmallPlayerView
Couche Présentation	10	Création du petit lecteur de musique avec les fonctionnalités de base (bar du temps/avance rapide/retour rapide/précédent/suivant, etc.)	Voir GIT : projet Présentation/Helper/MusicPlayer
Couche Présentation	8	Création des fonctions du lecteur de musique principales (musique continue, aléatoire) et interférage	
Couche BLL (Business Logic Layer)	4	Ajout de la gestion des favoris	
Couche Présentation	2	Ajout du flyout de la liste de lecture	
Documentation	4		
Total semaine	99	Max. 99	

Semaine		3	Date: lundi, 15 mai 2017
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment?	Liens, références, ...
Couche Présentation	4	Création du viewmodel playlist, avec la gestion des playlists	Voir GIT : projet Présentation/ViewModel/PlaylistViewModel
Couche Présentation	6	Gestion du click droit (ajouter/supprimer de la playlist)	
Bugs	3	Résolution d'un problème qui faisait que la musique s'ouvrait lors du clic droit.	
Couche BLL (Business Logic Layer)	10	Création de la liaison avec l'api shoutcast, avec récupération des top radios, ainsi que la récupération du fichier radio	Voir GIT : projet Présentation/ViewModel/RadioViewModel
Couche BLL (Business Logic Layer)	7	Lire les stream radio de type mpeg	Voir GIT : projet Shared/MusicFile
Couche BLL (Business Logic Layer)	7	Lire les stream radio de type aacp en transformant le stream aacp en stream MPEG	Voir GIT : projet Shared/MusicFile
Couche Présentation	12	Mise en relation des radios et des morceaux, ils utilisent maintenant la même classe parente	
Couche BLL (Business Logic Layer)	2	Récupération des images de radios	
Couche Présentation	3	Ajout des radios en favoris	
Bugs	8	Résolution d'un bug que les radios ne se lançaient pas parfois	Voir GIT : projet Shared/MusicFile
UI	2	Adaptation des players pour pouvoir inclure les données des radios	Voir GIT : projet Présentation/View/Flyout/RunningFlyoutView
Couche BLL (Business Logic Layer)	5	Recherche de radio avec l'API de shoutcast	Voir GIT : projet DAL/API/Shoutcast
Couche Présentation	3	Création de la liaison de la vue recherche avec le viewmodel	
Couche Présentation	8	Effectuer la recherche parmi tous les éléments de la base de données ainsi que shoutcast	Voir GIT : projet Présentation/ViewModel/SearchViewModel
Couche Présentation	5	Création et gestion de la vue pour la configuration de l'application	Voir GIT : projet Présentation/ViewModel/SettingFlyoutViewModel
Couche BLL (Business Logic Layer)	5	Faire que la synchronisation se déroule correctement et ne fais pas de doublon, empêche maintenant l'utilisateur d'accéder aux autres pages pendant la synchronisation	
Couche Présentation	6	Sauvegarde du contexte ainsi que récupération de celui-ci	Voir GIT : projet Présentation/ViewModel/MainWindowViewModel
Documentation	3		
Total semaine	99	Max. 99	
Semaine		4	Date: lundi, 22 mai 2017
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment?	Liens, références, ...
Couche Présentation	2	Récupération du contexte s'il est en mode radio	Voir GIT : projet Présentation/ViewModel/MainWindowViewModel
Couche BLL (Business Logic Layer)	2	Il est maintenant possible d'avoir plusieurs genres par music	
Couche Présentation	2	Ajout de la suppression de music [de la bibliothèque, ou du disque]	
Couche Présentation	2	Ajout de la gestion du volume de la musique	
Couche Présentation	10	Meilleure gestion du clic droit, le menu contextuel ne s'affiche plus lors du clic droit sur le header des datagrids	
Bugs	4	un bug dupliquait les radios dans les dernières radios écoutees	
Bugs	10	Résolution du bug des flux radio. En plus, au cas où le flux n'as pas été bien transmis, il y a 3 essais avant d'annoncer une fail	
Couche Présentation	2	Ajout d'un icône	
Documentation	4		
Bugs	4	Amélioration de la recherche lors d'un grand nombre de données	
Congé	57	Asc.	
Total semaine	99	Max. 99	
Semaine		5	Date: lundi, 29 mai 2017
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment?	Liens, références, ...
Couche BLL (Business Logic Layer)	2	Lire les fichiers radio en format xspf au lieu des m3u, car je les trouve plus stables	
Bugs	3	après la lecture de beaucoup de radios, quand l'utilisateur revenait dans la vue musique, l'application plantait	
Couche Présentation	2	Ajout de la possibilité de changer de musique depuis la liste de lecture	
Couche Présentation	3	Si la radio à un problème, jouer l'ancienne radio à la place et ne pas afficher le flyout	
Bugs	2	Si la musique a été déplacée, elle disparaît maintenant de la liste de lecture	
Documentation	2	Création des scénarios de tests	
Création de l'installateur	6	Création d'un installateur avec clickonce	
Tests	10	Résolution des bugs d'installation sur les machines Windows 7. Notamment un bug de DLL manquante qui arrivait sur le Windows 7 sans la librairie «Redistributable Visual C++ 2015»	
Manuel utilisateur	6	Rédaction du manuel utilisateur	
Rendez-vous	3	Rendez-vous avec M.Sahlí	
Couche Présentation	1	Comme demandé par M.Sahlí, le logiciel recherche les morceaux selon l'artiste et l'album en plus du nom et du genre	
Couche Présentation	3	Pour éviter des lenteurs dans la recherche, il faut maintenant appuyer sur enter ou sur la loupe	
Tests	2	Exécution des scénarios de tests	
Manuel utilisateur	3	Correction orthographique	
Documentation	7	Ajout de commentaires de code pour le rendre correct selon format EML	
Bugs	9	Optimisation du code, suppression des références non utilisée, suppression des fonctions non utilisée et formatage du code	
Documentation	3	Écriture de commentaires de fichier	
Bugs	5	Grande optimisation de la recherche	
Documentation	4	Rédaction des Scénarios de tests	
Tests	5	Effectuer les Scénarios	
Documentation	5	Rédaction de la conclusion	
Congé	13	Examen ECG	
Total semaine	99	Max. 99	
Semaine		6	Date: lundi, 5 juin 2017
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Documentation	10		
Congé	13	Pentecôte	
Total semaine	23	Max. 99	

10 ANNEXES

10.1 Cahier des charges

1 INFORMATIONS GENERALES

Candidat :	Nom : PREISIG mailto:mailto:preisiger@etml.educanet2.ch	Prénom : ERIC-NICOLAS Téléphone : -	
Lieu de travail :	ETML, Sébeillon 12 1004 Lausanne		
Chef de projet :	Nom : SAHLI mailto:mailto:bertrand.sahl@vd.ch	Prénom : BERTRAND Téléphone : 021 316 02 62	
Expert 1 :	Nom : ROLLINET mailto:mailto:sylvain.rollinet@gmail.com	Prénom : SYLVAIN Téléphone : -	
Expert 2 :	Nom : MELLY mailto:mailto:jonathan.melly@vd.educanet2.ch	Prénom : JONATHAN Téléphone : -	
Dates de réalisation :	Du lundi 1 mai à 8h au mercredi 7 juin à 11h25		
Horaire de travail : (Basé sur l'horaire officiel)	Lundi Mardi Mercredi Jeudi Vendredi	08h00-11h25 - 08h00-12h15 13h10-16h35 08h00-11h25 12h20-16h35 08h00-12h15 13h10-16h35	- - Exo CG 31 mai 2017 après-midi Pont de l'Aso, 25 mai 2017 Pont de l'Aso, 26 mai 2017
Présentation :	Entre le mercredi 14 et jeudi 15 juin 2017		
Nombre d'heures :	Environ 110 heures		
Planning (en H ou %)	Analyse: 15%, Implémentation: 40%, Tests: 20%, Documentation: 25%		

2 PROCÉDURE

Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le 1er jour.

Le cahier des charges est approuvé par i-CQ VD. Il est en outre présenté, commenté et discuté avec le candidat. Par sa signature, le candidat accepte le travail proposé.

Le candidat a connaissance de la feuille d'appréciation avant de débuter le travail.

Le candidat est entièrement responsable de la sécurité de ses données.

En cas de problèmes graves, le candidat avertit au plus vite les deux experts et son chef de projet.

Le candidat a la possibilité d'obtenir de l'aide, mais doit le mentionner dans son dossier de projet.

A la fin du délai imparti pour la réalisation du TPI, le candidat doit transmettre par courrier électronique le dossier de projet aux deux experts et au chef de projet. En parallèle, une copie papier du rapport doit être fournie sans délai en trois exemplaires. Cette dernière doit être en tout point identique à la version électronique.

3 TITRE

Gestionnaire de playlists audio et de flux radios

4 SUJET

Développer une application desktop en C# selon le pattern MVVM afin de gérer des playlists audio et des flux radios publics diffusés sur Internet. Il s'agira notamment :

- d'assurer l'installation de l'application sur les OS Microsoft 7 à 10.
- d'exploiter les données audios locales (sans les dupliquer).
- de créer un lecteur audio « user friendly ».
- de stocker les données de l'application dans une base de données SQLite.

5 MATERIEL ET LOGICIEL À DISPOSITION

PC standard de l'ETML (Windows 7 – 64bit)

Player de machines virtuelles (vmware ou équivalent)

Suite Microsoft Office 2013

IDE C# (Visual Studio Community 2015 ou 2017, SQLite Precompiled Binaries for Windows)

6 PRÉREQUIS

Compétences élémentaires en base de données (SQL) et en programmation (C#) → Modules 103, 104, 105, 303.

Compétences bureautiques → Modules 301 et 302

7 DESCRIPTIF DU PROJET

Concevoir une interface graphique « user friendly » inspirée de Spotify.

Modéliser et implémenter une base de données permettant de soutenir les fonctionnalités de l'application. Soit au minimum:

- o la bibliothèque des contenus audios du PC (vide au 1^{er} lancement).
- o les habitudes d'écoute de l'utilisateur (ses chansons favorites, ses styles musicaux préférés, ses listes de lecture, les 10 dernières radios écoutées).

Développer les fonctionnalités suivantes :

- o au 1^{er} lancement, scanner les contenus audios du PC et indexer tous les contenus audios découverts dans la bibliothèque « locale ».
- o à la demande, (re)scanner les contenus audios du PC et indexer tous les nouveaux contenus audios découverts dans la bibliothèque « locale ».
- o rechercher, dans « locale », des chansons (au minimum par titre ou groupe).
- o concevoir des playlists (au minimum les identifier et les jouer) et y ajouter la/les chansons sélectionnées (les moyens de sélection seront ergonomiques et efficaces).
- o jouer des chansons (prévoir les actions habituelles : pause, play, slider, stop, ...)
- o jouer des listes de lecture (en plus des actions habituelles, prévoir les actions : précédent, suivant, lecture continue, lecture aléatoire, lecture répétée, ...)
- o rechercher des radios diffusées sur Internet (au minimum par nom ou style musical).
- o se souvenir des 10 dernières radios sélectionnées.
- o écouter le flux de la WebRadio sélectionnées.
- o fournir les informations de la chanson en cours (au minimum le titre, le groupe, l'album, le style musical, une image significative et le passage du temps).
- o reprendre le dernier contexte applicatif à l'ouverture de l'application.

8 POINTS ÉVALUÉS DURANT LE PROJET

- La méthodologie de travail ainsi que la conformité des tâches réalisées par rapport au planning initial.
- Le choix des « métadonnées » nécessaires à l'indexation des chansons.
- La pertinence du modèle de données réalisé.
- La mise en œuvre de tests (notamment les tests unitaires) pour vérifier les fonctionnalités implémentées.
- La journalisation des tâches ainsi que la rédaction régulière de la documentation.

9 LIVRABLES

Le candidat est responsable de livrer à son chef de projet et aux deux experts :

- Une planification initiale (à envoyer par mail le 3 mai à 12h15 au plus tard)
- Un rapport de projet (à envoyer par mail, en l'état, toutes les fins de semaine)
- Un journal de travail (à envoyer par mail, en l'état, toutes les fins de semaine)
- Le code source de l'application
- Une archive autoinstallable de l'application compatible MS Windows 7 à MS Windows 10.
- Un guide d'utilisation

10 POINTS TECHNIQUES ÉVALUÉS SPÉCIFIQUES AU PROJET

La grille d'évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, ...).

En plus de cela, le travail sera évalué sur les trois points spécifiques suivants :

- Le point spécifique n° 1 → L'extraction de métadonnées de divers format audio.
- Le point spécifique n° 2 → La recherche de WebRadio.
- Le point spécifique n° 3 → Le (re)scan des contenus audios.

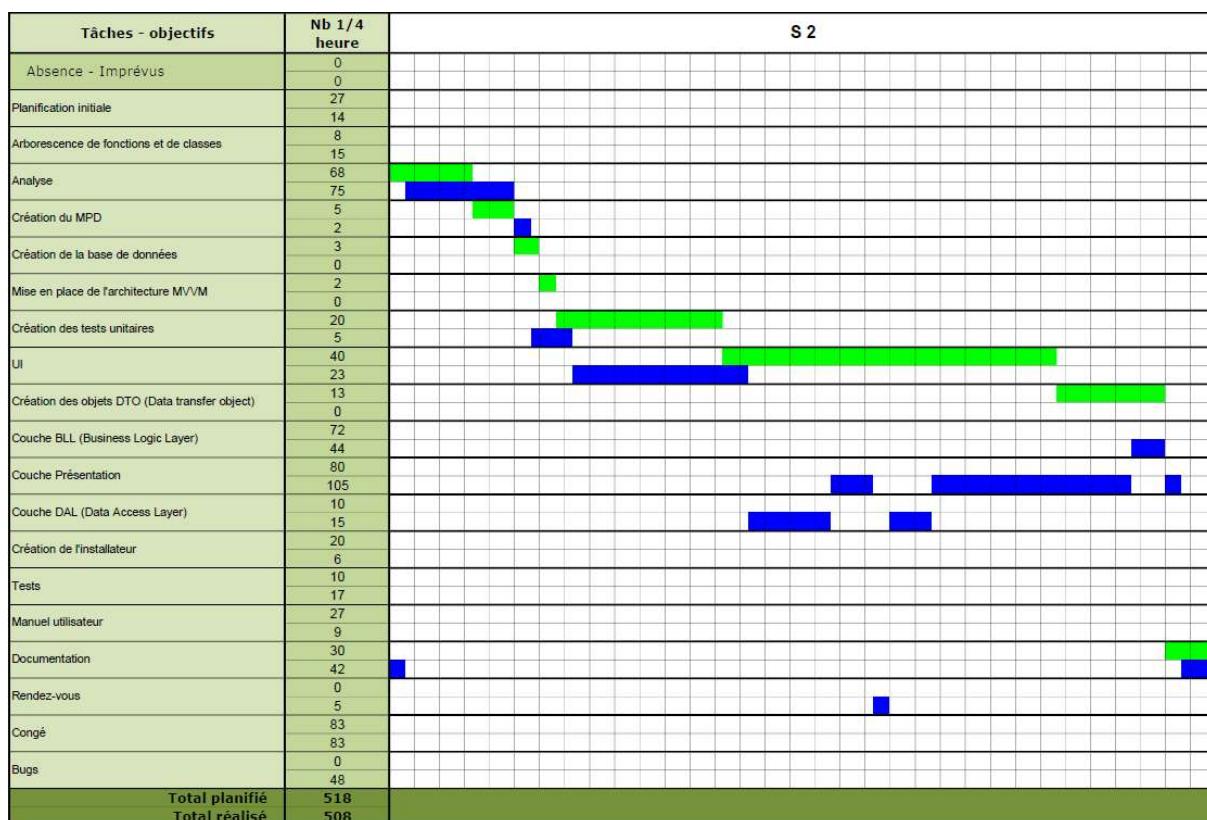
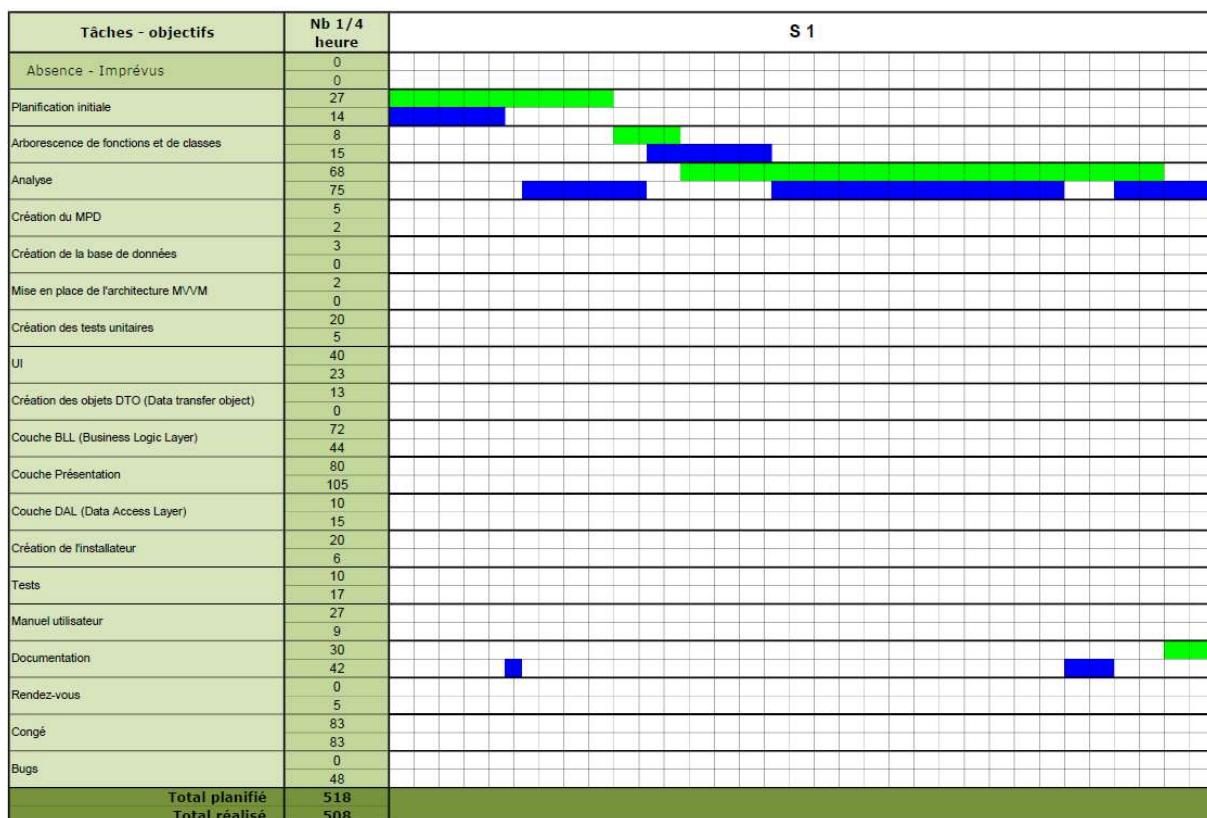
11 VALIDATION

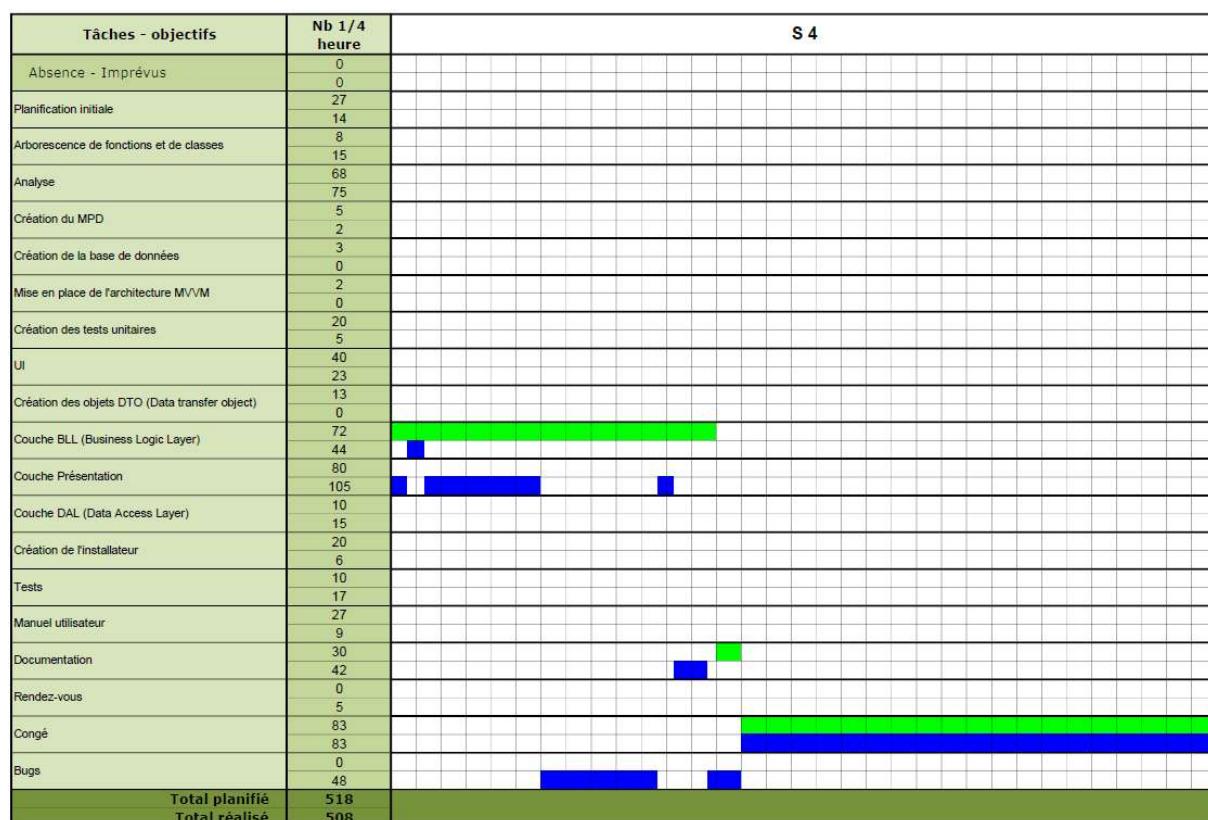
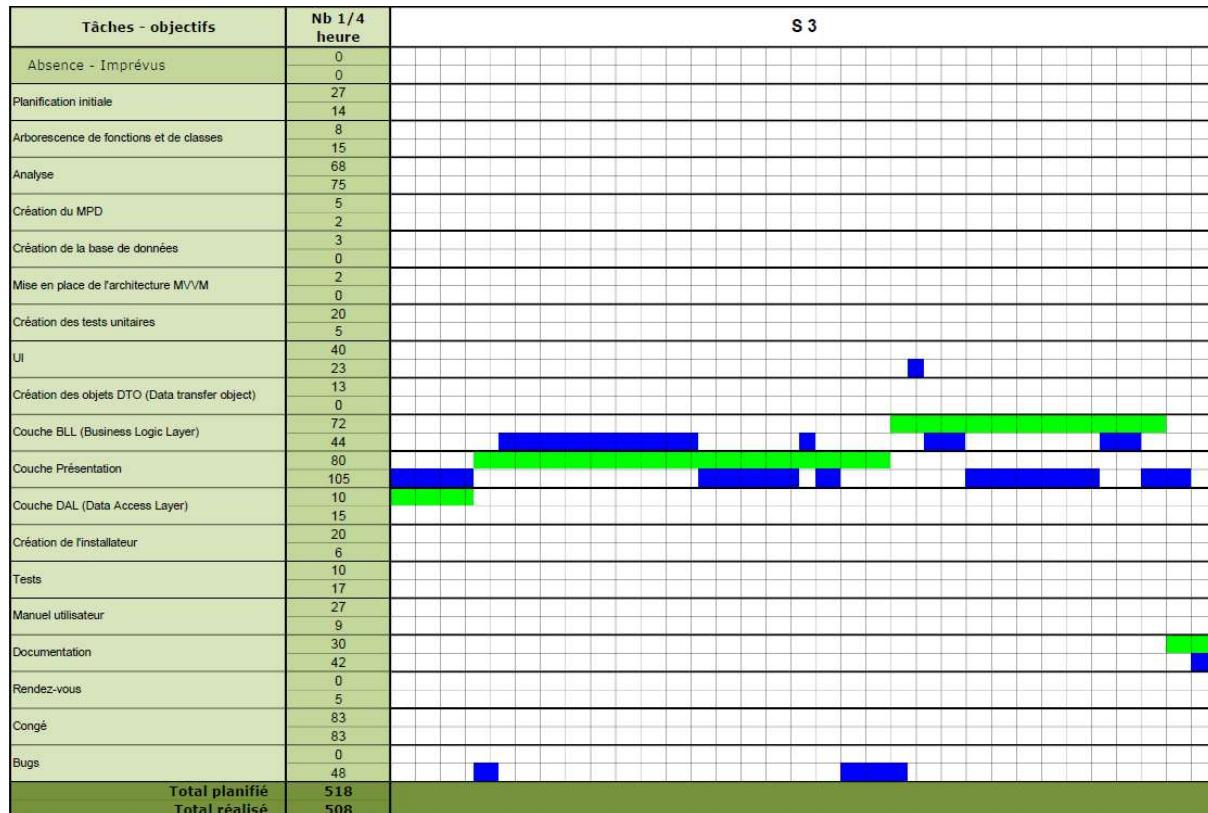
	Lu et approuvé le :	Signature :
Candidat :		
Expert n°1 :		
Expert n° 2 :		
Chef de projet :		

10.2 Gantt

En vert : les périodes planifiées

En bleu : les périodes effectuées





Tâches - objectifs	Nb 1/4 heure	S 5
Absence - Imprévu	0	
Planification initiale	27	
Arborescence de fonctions et de classes	14	
Analyse	8	
Création du MPD	15	
Création de la base de données	68	
Mise en place de l'architecture MVVM	2	
Création des tests unitaires	0	
UI	20	
Création des objets DTO (Data transfer object)	5	
Couche BLL (Business Logic Layer)	40	
Couche Présentation	13	
Couche DAL (Data Access Layer)	0	
Création de l'installateur	23	
Tests	72	
Manuel utilisateur	6	
Documentation	10	
Rendez-vous	17	
Congé	15	
Bugs	20	
Total planifié	518	
Total réalisé	508	

Tâches - objectifs	Nb 1/4 heure	S 6
Absence - Imprévu	0	
Planification initiale	27	
Arborescence de fonctions et de classes	14	
Analyse	8	
Création du MPD	15	
Création de la base de données	68	
Mise en place de l'architecture MVVM	2	
Création des tests unitaires	0	
UI	20	
Création des objets DTO (Data transfer object)	5	
Couche BLL (Business Logic Layer)	40	
Couche Présentation	13	
Couche DAL (Data Access Layer)	0	
Création de l'installateur	23	
Tests	72	
Manuel utilisateur	6	
Documentation	10	
Rendez-vous	17	
Congé	20	
Bugs	5	
Total planifié	518	
Total réalisé	518	