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 3](#_Toc481763288)

[1.1 Titre 3](#_Toc481763289)

[1.2 Description 3](#_Toc481763290)

[1.3 Matériel et logiciels à disposition 3](#_Toc481763291)

[1.4 Prérequis 4](#_Toc481763292)

[1.5 Cahier des charges 4](#_Toc481763293)

[1.6 Les points suivants seront évalués 4](#_Toc481763294)

[1.7 Validation et conditions de réussite 4](#_Toc481763295)

[2 Planification Initiale 5](#_Toc481763296)

[3 Analyse 5](#_Toc481763297)

[3.1 Opportunités 5](#_Toc481763298)

[3.2 Document d’analyse et conception 7](#_Toc481763299)

[3.2.1 Maquette 7](#_Toc481763300)

[3.2.2 Architecture 12](#_Toc481763301)

[3.2.1 Librairie utilisées 13](#_Toc481763302)

[3.2.2 Structure du code 13](#_Toc481763303)

[3.2.3 Base de données 16](#_Toc481763304)

[3.2.1 Schéma évènementiel 17](#_Toc481763305)

[3.3 Conception des tests 24](#_Toc481763306)

[3.4 Planification détaillée 24](#_Toc481763307)

[4 Réalisation 24](#_Toc481763308)

[4.1 Dossier de Réalisation 24](#_Toc481763309)

[4.2 Modifications 24](#_Toc481763310)

[5 Tests 24](#_Toc481763311)

[5.1 Dossier des tests 24](#_Toc481763312)

[6 Conclusion 24](#_Toc481763313)

[6.1 Bilan des fonctionnalités demandées 24](#_Toc481763314)

[6.2 Bilan de la planification 25](#_Toc481763315)

[6.3 Bilan personnel 25](#_Toc481763316)

[7 Divers 25](#_Toc481763317)

[7.1 Journal de travail 25](#_Toc481763318)

[7.2 Bibliographie 25](#_Toc481763319)

[7.3 Webographie 25](#_Toc481763320)

[8 Annexes 26](#_Toc481763321)

[8.1 Cahier des charges 26](#_Toc481763322)

[8.2 Gantt 29](#_Toc481763323)

# 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.

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.

## Titre

Gestionnaire de playlists audio et de flux radios

## Description

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.

## Matériel et logiciels 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)

## Prérequis

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

Compétences bureautiques 🡺 Modules 301 et 302

## Cahier des charges

Cahier des charges complet en annexe

## 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.

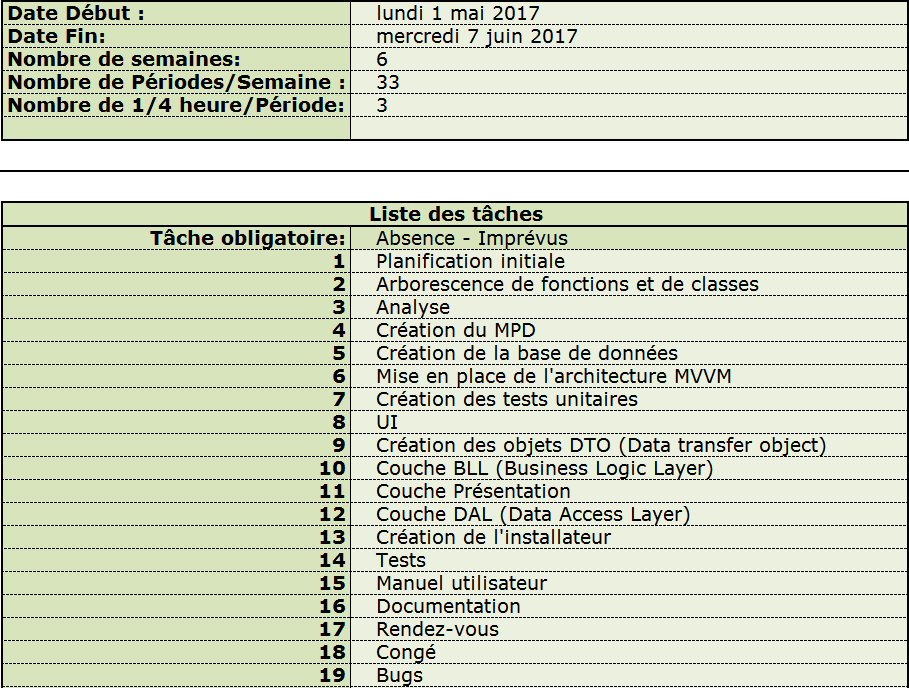
## Validation et conditions de réussite

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 audios.

# Planification Initiale



*Gantt disponible en annexe*

# Analyse

## Opportunités

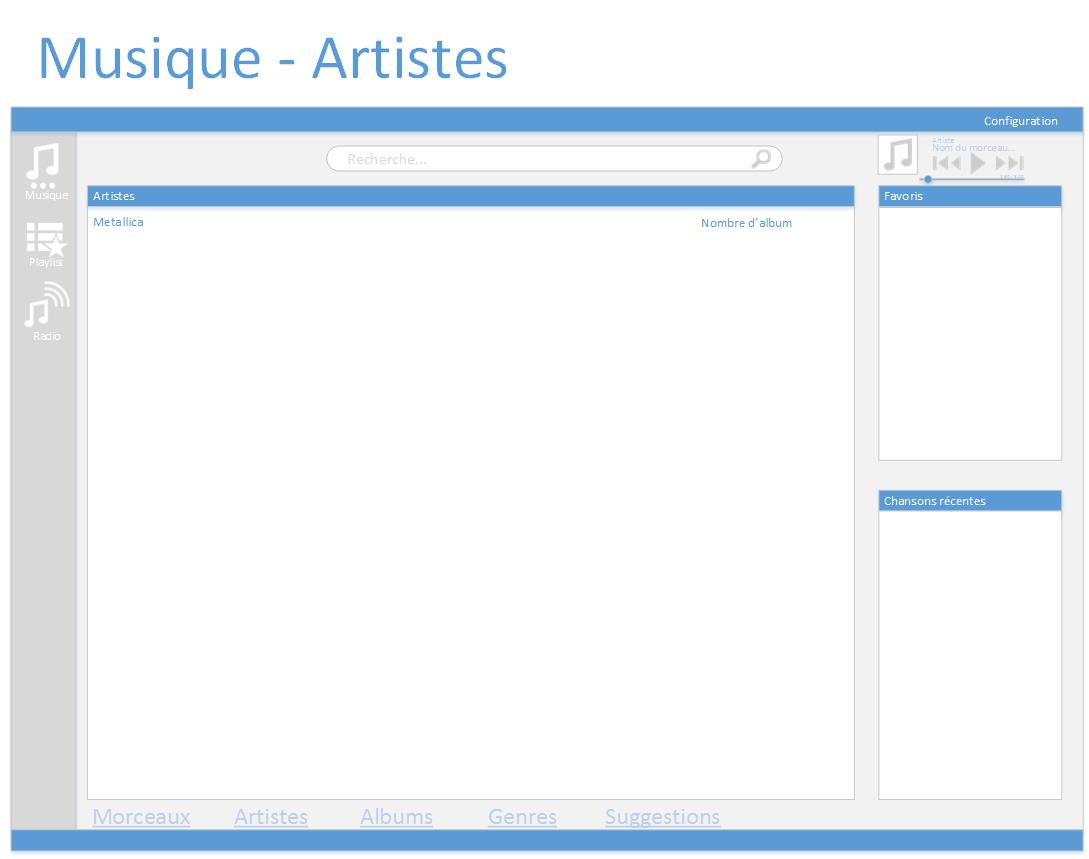
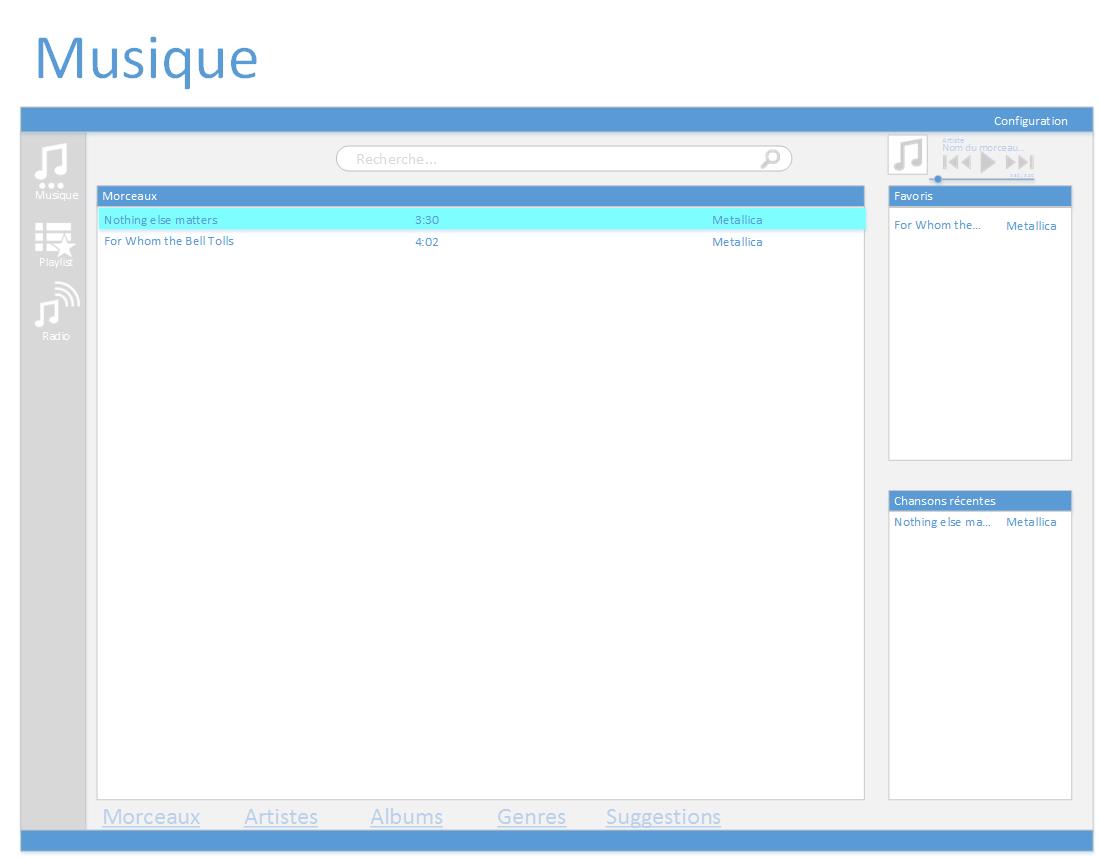
Ce projet me permettra de développer mes capacités en développement .NET. Il me donnera l’opportunité d’approfondir mes connaissances dans l’architecture MVVM.

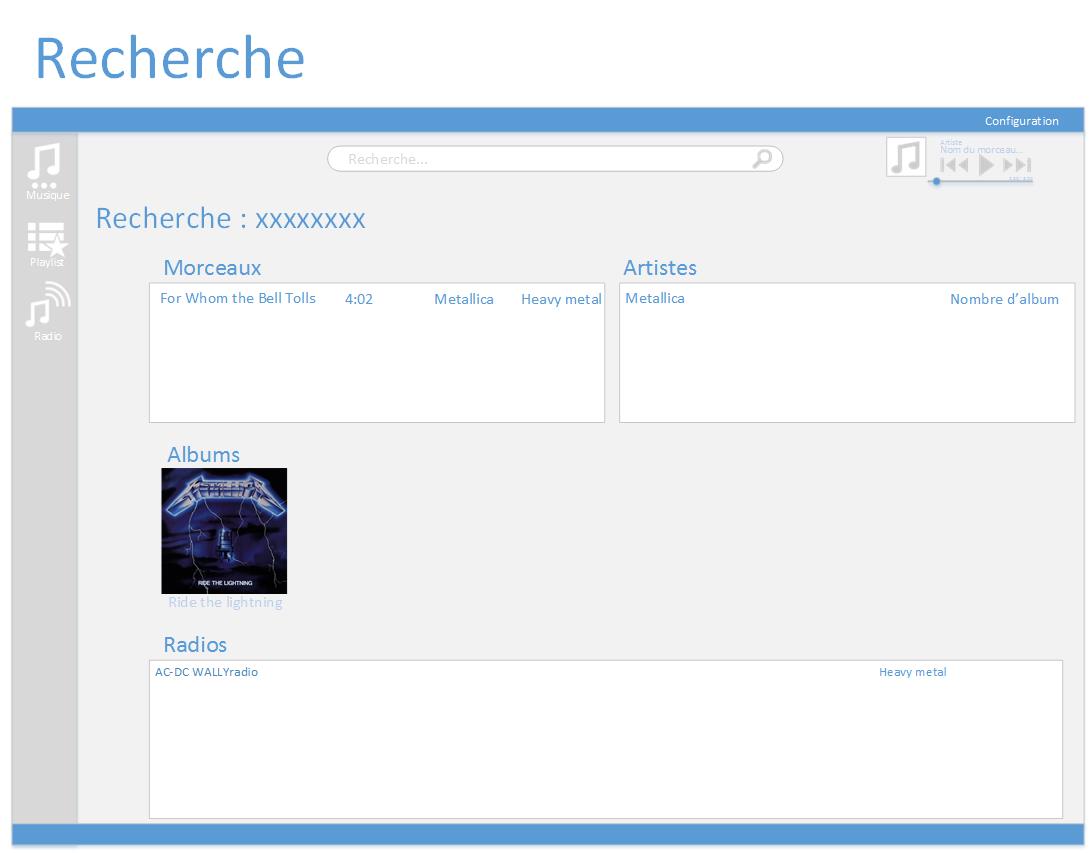
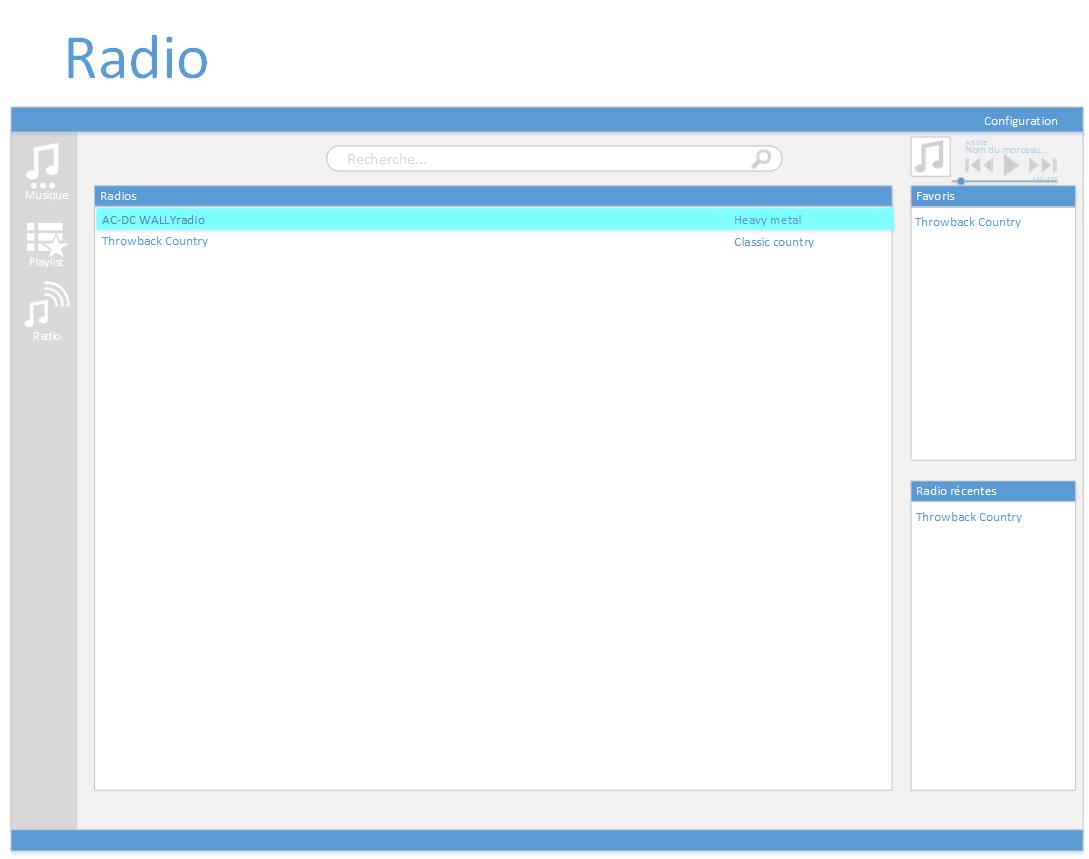
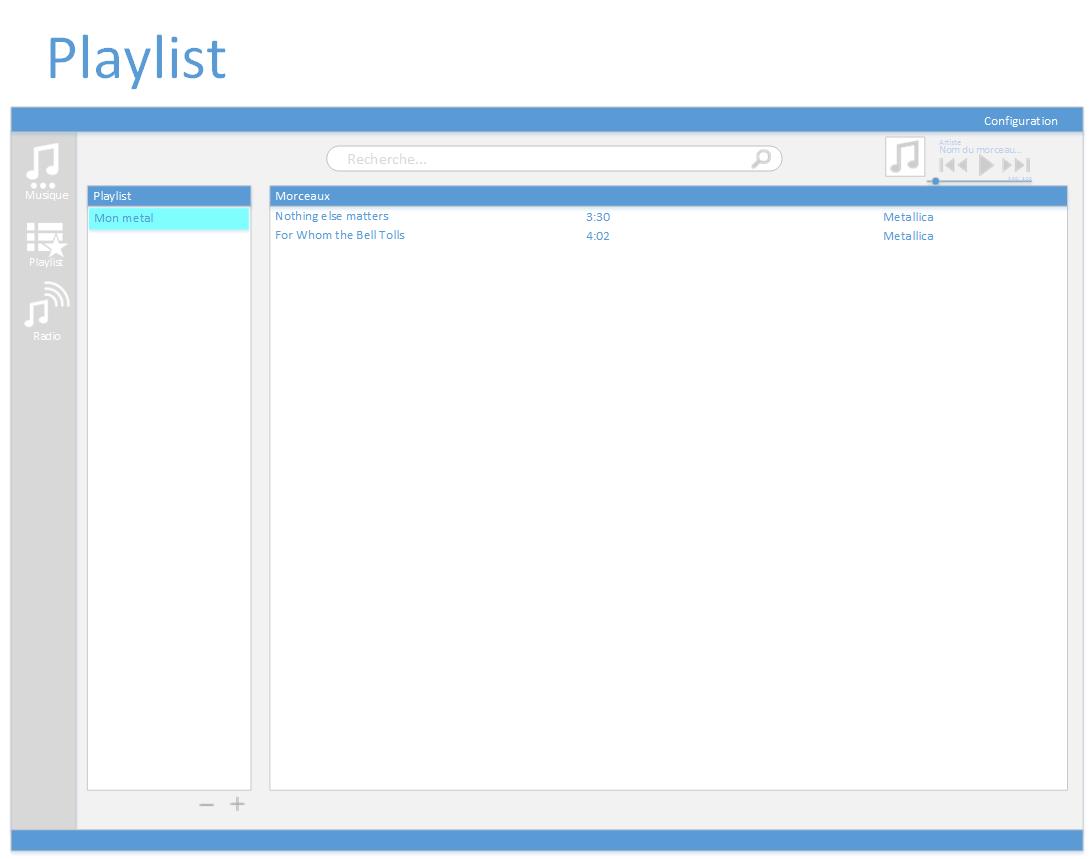
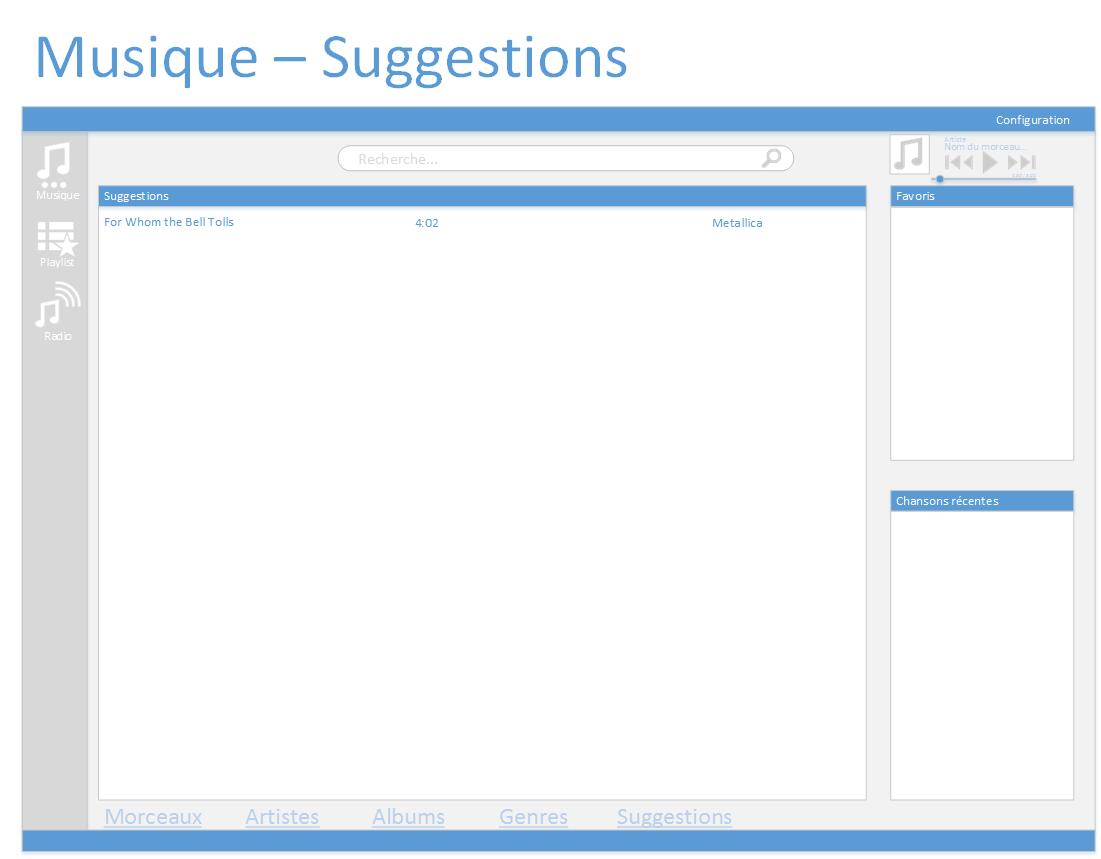
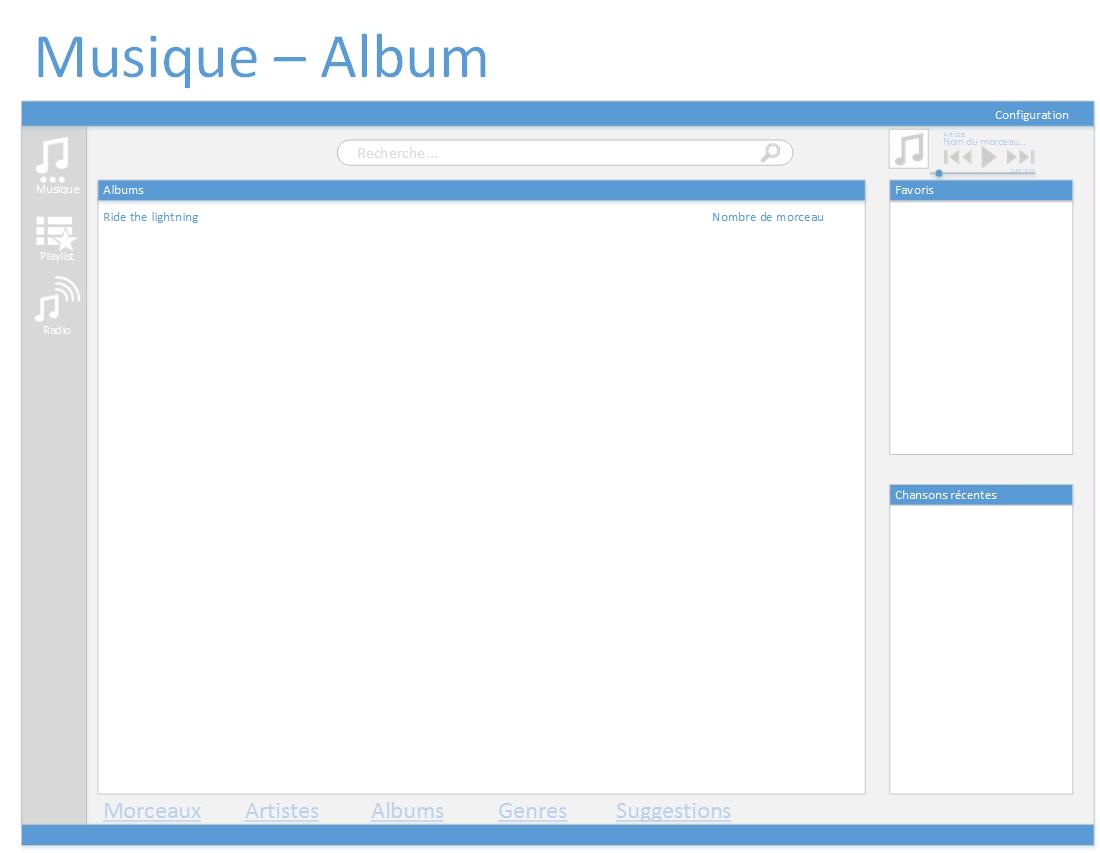
La principale difficulté sera de fournir une interface simple et ergonomique. Je pense donc partir sur un design flat et minimaliste, permettant ainsi une interface simple et intuitive. J’utiliserais pour cela la librairie Mahapps qui me permettra de fournir une interface épurée dans le style de Windows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

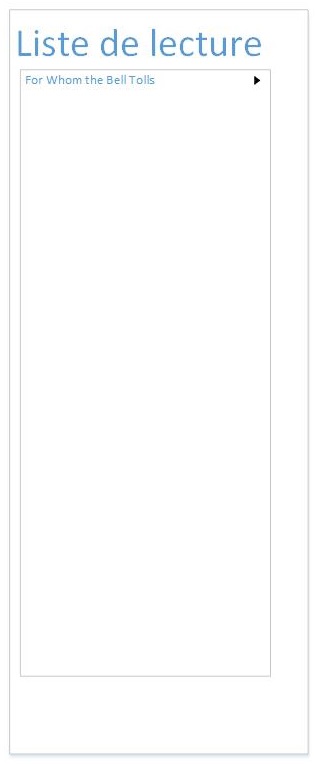
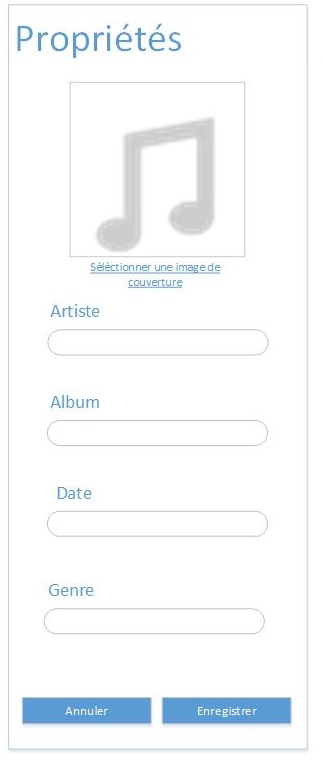
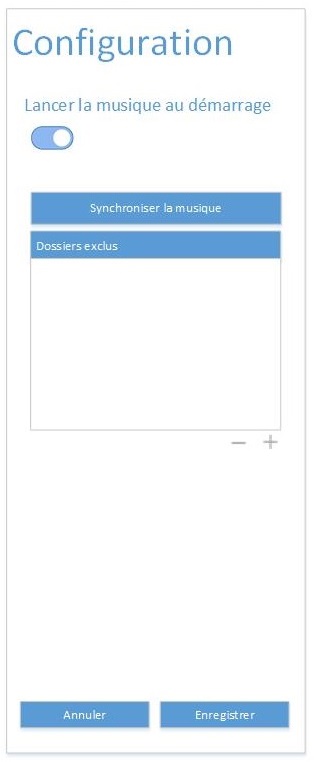
La sauvegarde du contexte de l’application sera un point difficile. Il y a plusieurs options pour pouvoir garder le contexte applicatif après la fermeture de l’application. Je pense crée une classe qui contient toute les données à sauvegarder. Celle-ci sera enregistré dans la base de données. À l’ouverture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

## Document d’analyse et conception

### Maquette







### 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 radios.

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’utilisateurs ainsi que de récupérer ses entrées.

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

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

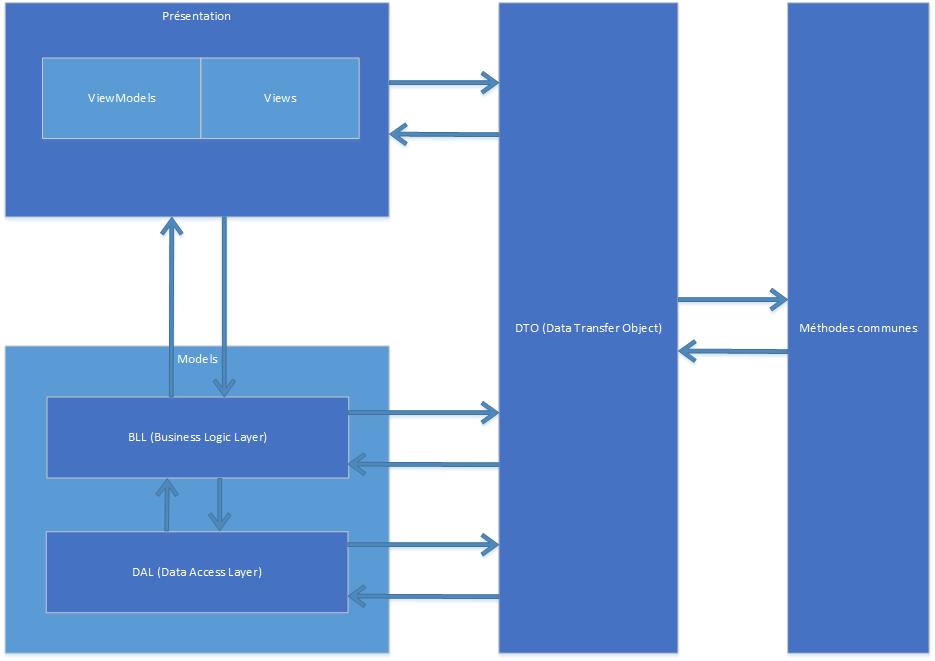


Figure Illustration de l'architecture

### Librairie 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

***MvvmLight***

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

***EntityFramwork***

ORM

***Mahapps.Metro***

Ajoute des éléments graphiques pour rendre l’interface plus moderne

***NAudio***

Permet la lecture des fichiers et flux audio

***NUnit***

Ajoute des fonctionnalités aux tests unitaire

***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 EntityFramwork avec une base de données SQLite

### Structure du code

Le code sera structuré de manière suivante :

1. **DAL** *Projet Data access layer* 
   1. **API** *Dossier*
      1. **Shoutcast** *Contient toute les fonctions concernant l’API de ShoutCast*
   2. **Database** *Dossier*
      1. **Configuration** *Contient la configuration de la base de données, ainsi que le SEED*
      2. **DbContext** *Crée un contexte de la base de données en utilisant une connexion string*
      3. **Repository** *Class générique permettant d’effectuer les actions de base sur la base de données (CRUD)*
2. **BLL** *Projet Business logic layer* 
   1. **AlbumData** *Permet d’effectuer des actions sur les données relatives aux albums*
   2. **ArtistData** *Permet d’effectuer des actions sur les données relatives aux artistes*
   3. **RadioData** *Permet d’effectuer des actions sur les données relatives aux radios*
   4. **TrackData** *Permet d’effectuer des actions sur les données relatives aux morceaux*
   5. **SearchData** *Permet d’effectuer des actions de recherches*
   6. **FavoriteData** *Permet d’effectuer des actions sur les données relatives aux favoris*
   7. **GeneralData** *Permet d’effectuer des actions sur les données relatives aux fonctionnalités générales de l’application*
3. **Presentation** *Projet Presentation* 
   1. **Helper** *Dossier contenant les classes utiles à la logique de l’application*
      1. **Context** *Gère le contexte applicatif du lecteur, contiendra par exemple la liste de lecture*
      2. **MusicPlayer** *Fonctions relative au lecteur de musique (play, pause, etc)*
      3. **MusicSync** *Fonctions relative à la synchronisation des pistes musicales depuis l’ordinateur*
      4. **RightClick** *Fonctions relative au clic droit*
   2. **View** *Dossier contenant les vues de l’application*
      1. **Flyout** *Dossier contant les flyout de l’application*
         1. **MusicFlyoutView** *Flyout qui contiendra une liste soit d’album, soit d’artistes, soit de morceaux*
         2. **PropertyFlyoutView** *Flyout contenant les propriétés d’un morceau*
         3. **ReadingFlyoutView** *Flyout contenant la liste de lecture*
         4. **RunningFlyoutView** *Flyout contenant la vue du morceau en cours*
         5. **SettingFlyoutView** *Flyout contenant les options de l’applications*
      2. **List** *Dossier contenant les listes de l’application*
         1. **AlbumListView** *Liste d’albums*
         2. **ArtistListView** *Liste d’artistes*
         3. **FavoriteListView** *Liste des favoris*
         4. **GenreListView** *Liste des genres*
         5. **PlaylistListView** *Liste des playlists*
         6. **ReadingListView** *Liste des morceaux en lecture*
         7. **RecentListView** *Liste des dernières radios écoutées*
         8. **SuggestionListView** *Liste de suggestion*
         9. **TrackListView** *Liste de morceaux*
         10. **RadioListView** *Liste de radios*
      3. **MusicView** Vues de la page musique
      4. **PlaylistView** Vues de la page playlist
      5. **RadioView** Vues de la page radio
      6. **SearchView** Vues de la page recherche
      7. **SmallPlayerView** Vues du petit lecteur de musique
   3. **ViewModel** *Dossier contenant les viewmodels*
      1. **MainViewModel** *ViewModel parent de tous les autre ViewModels, il est utile en cas de partage entre ViewModels*
      2. **MainWindowViewModel** *Gère l’affichage de la fenêtre principale*
      3. **MusicViewModel** *Gère l’affichage de toutes les vues musique (morceaux, artistes, albums et genre)*
      4. **PlayerViewModel** *Gère l’affichage du petit player, ainsi que l’affichage du grand player dans la vue de la musique en cours*
      5. **PlaylistViewModel** *Gère l’affichage de la vue playlist*
      6. **PropertyFlyoutViewModel** *Gère l’affichage de la vue des propriétés*
      7. **RadioViewModel** *Gère l’affichage de la vue radio*
      8. **SettingFlyoutViewModel** *Gère l’affichage de la vue de configuration*
      9. **ViewModelLocator** *ViewModel locator, fait des singletons des ViewModels si l’on veut l’utiliser plusieurs fois avec le même contexte*
   4. **MainWindow** *Vue de la fenêtre layout, qui sera active sur toutes les vues*
4. **DTO** *Projet Data access object*
   1. **Entity** *Dossier contenant toutes les entités qui seront dans la base de données*
      1. **Album** *Table Album*
      2. **Artist** *Table Album*
      3. **Track** *Table Album*
      4. **BaseEntity** *Contient les propriétés communes à toute les tables*
      5. **Context** *Table Context*
      6. **Playlist** *Table Paylist*
      7. **ExcludeFolder** *Table ExcludeFolder*
      8. **Genre** *Table Genre*
      9. **Radio** *Table Radio*
   2. **Audio** *Contient les propriétés communes entre la table Track et la table Radio*
5. **Shared** *Projet qui contient les classes susceptibles d’être utilisée dans plusieurs layers*
   1. **MusicFile** *Effectue des transformations sur les fichiers audios*

### 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***

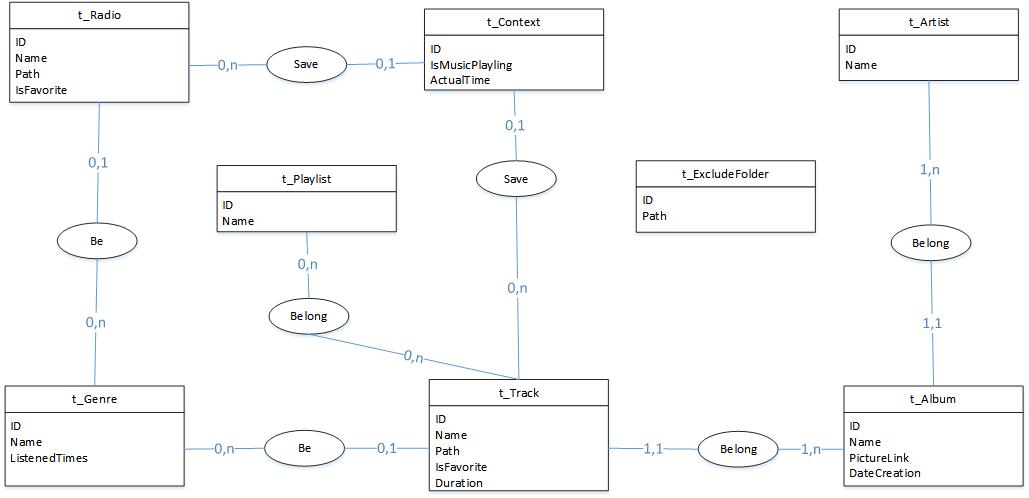
******

Figure Modèle conceptuel de données

***MLD***

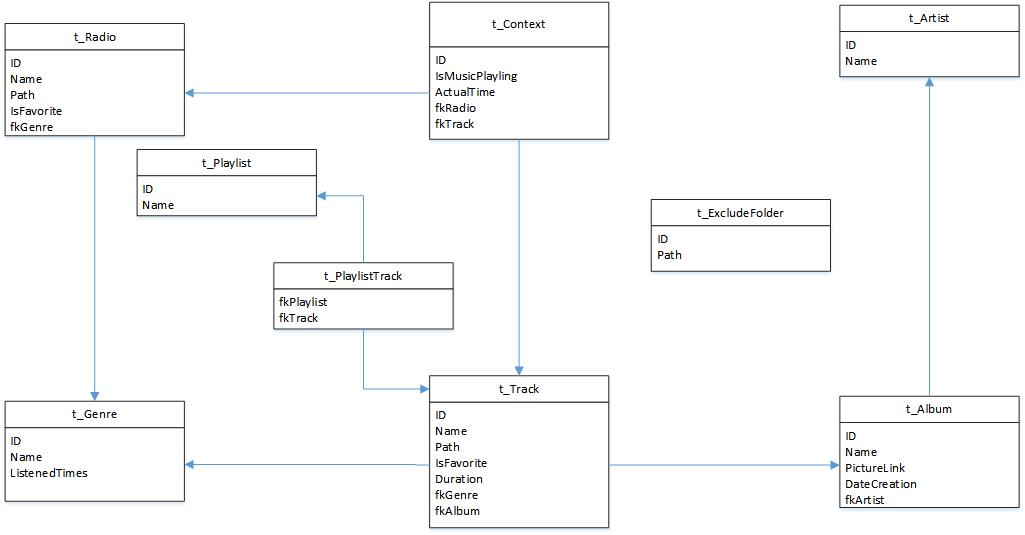
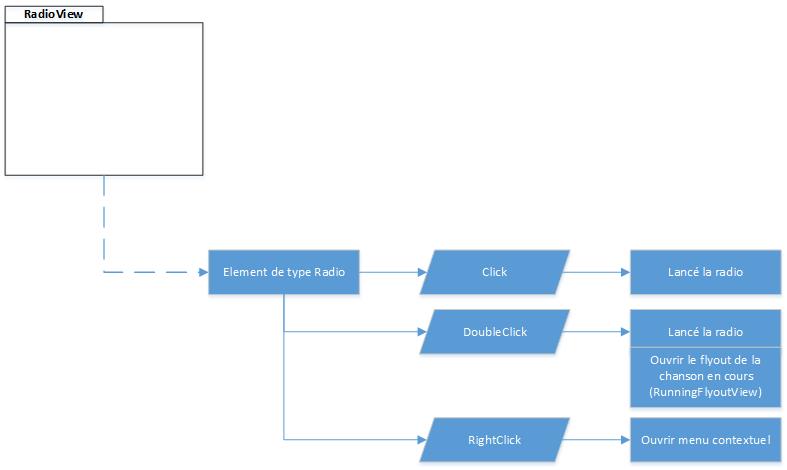
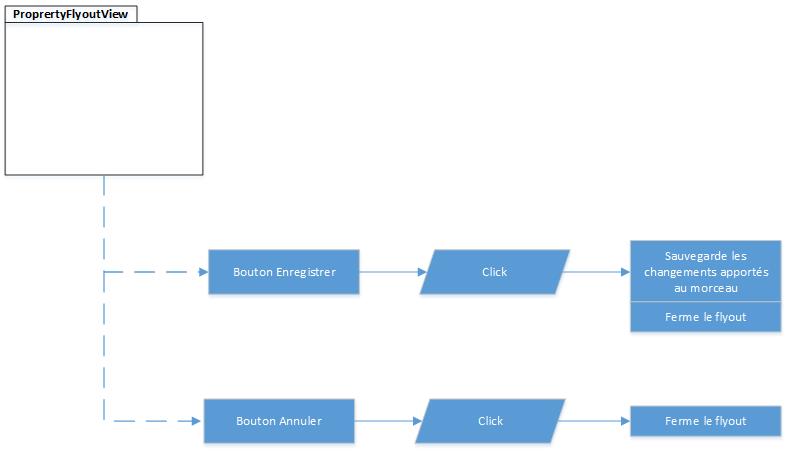
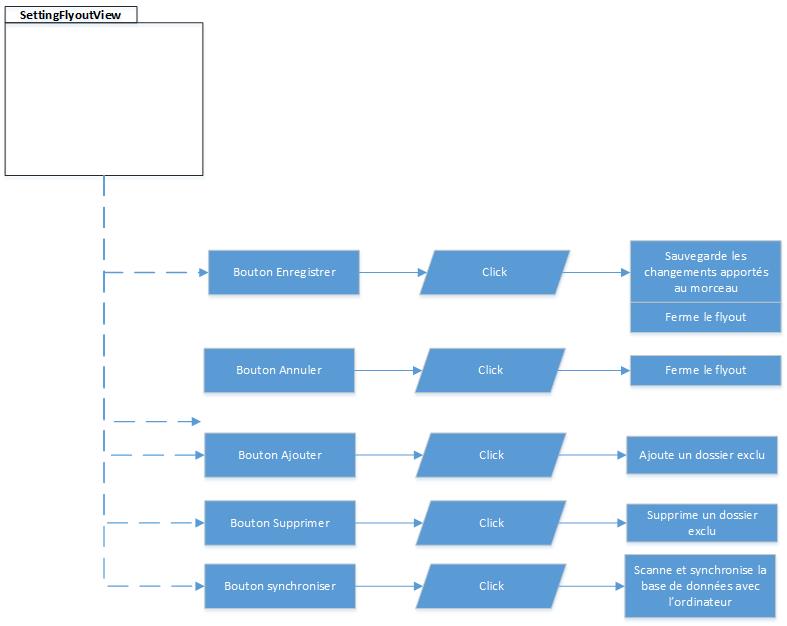
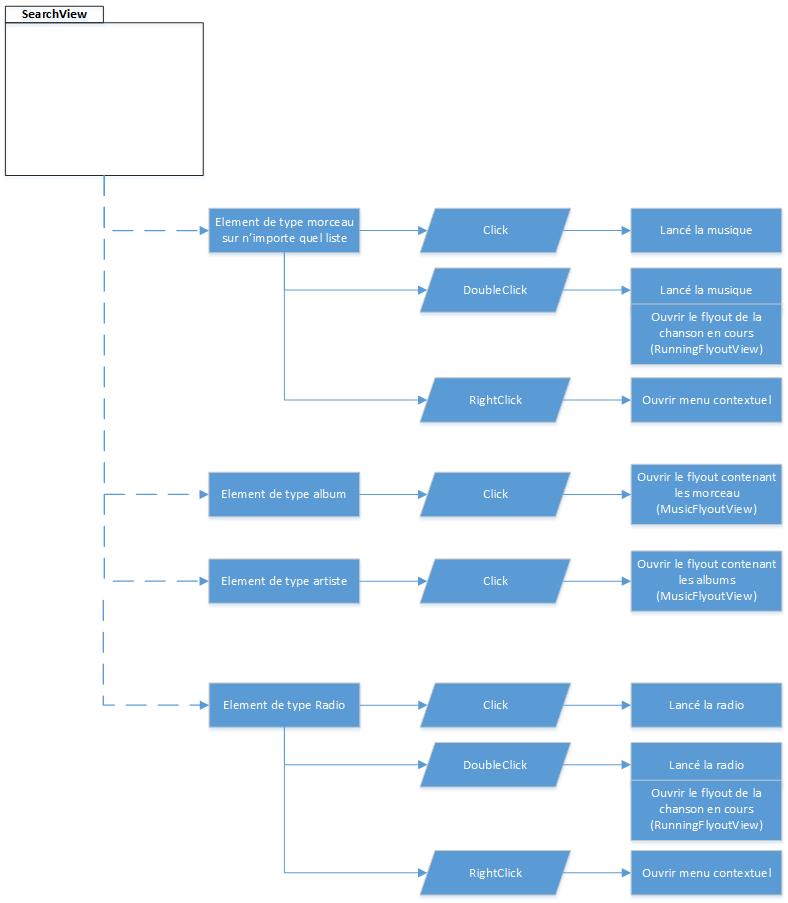
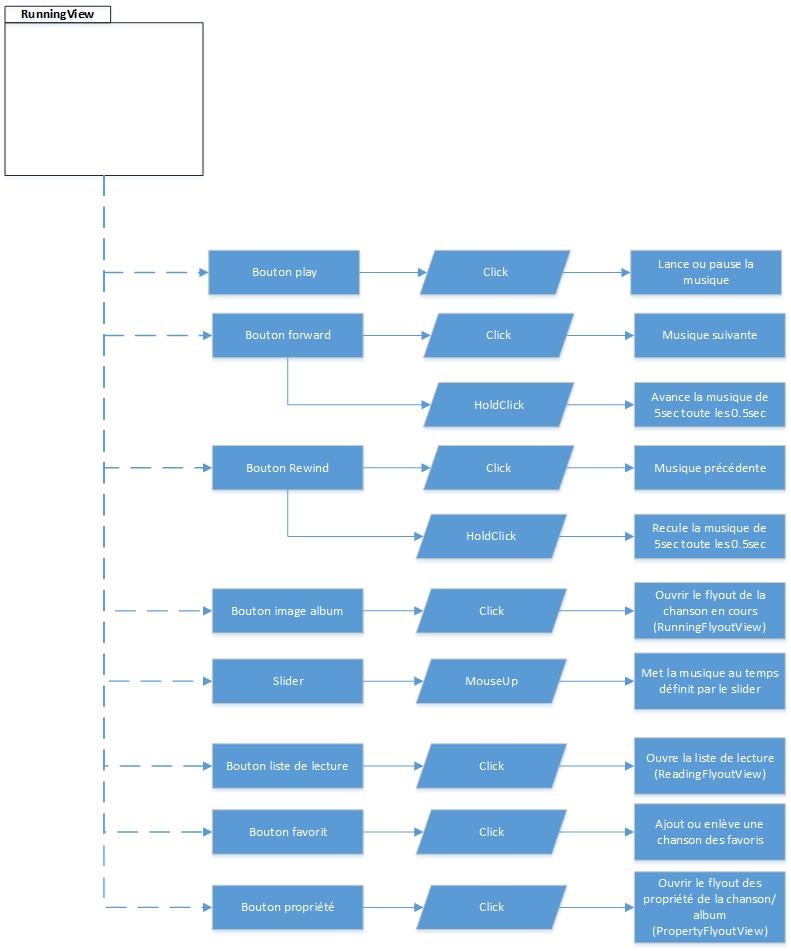
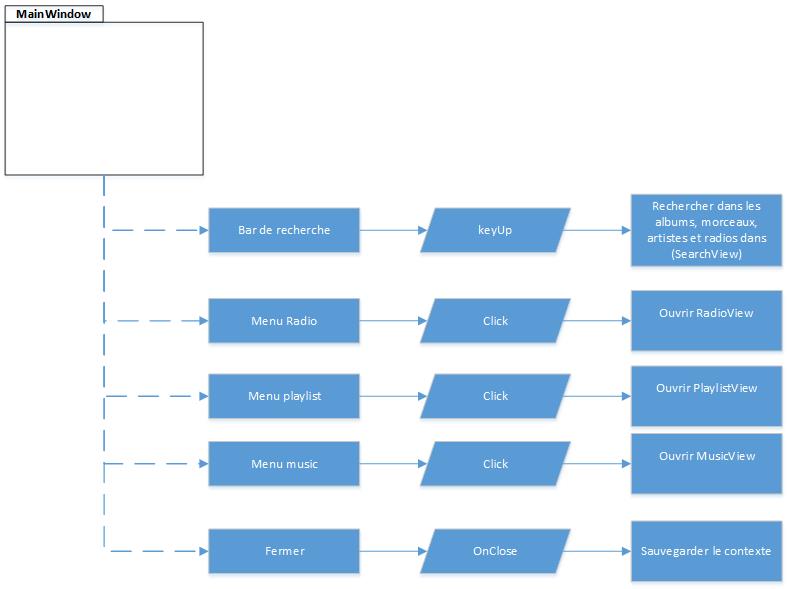
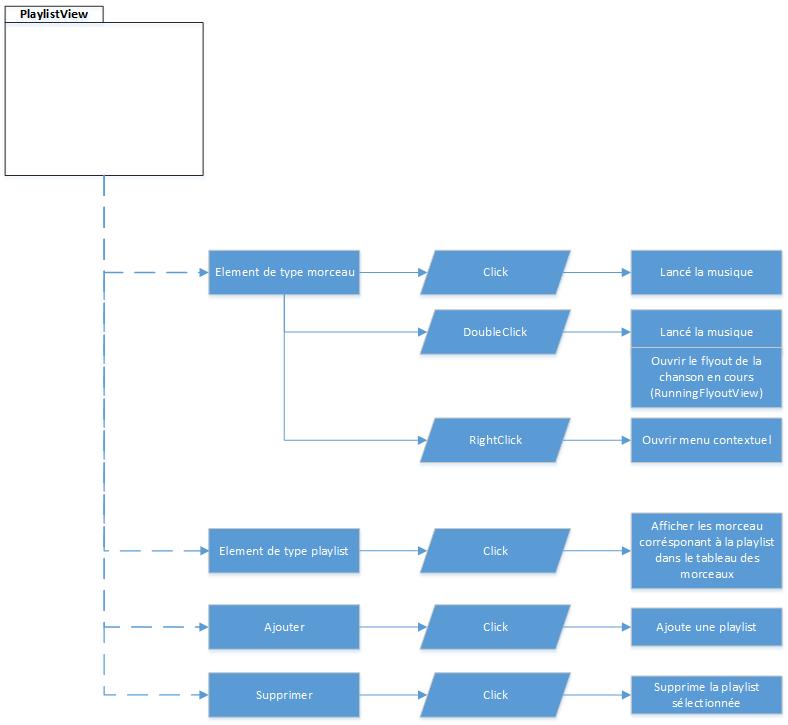
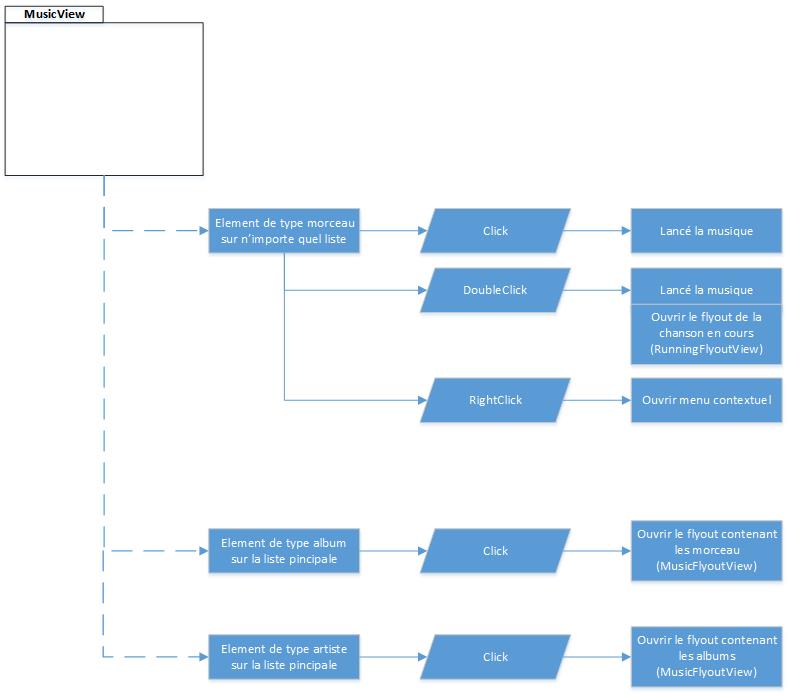
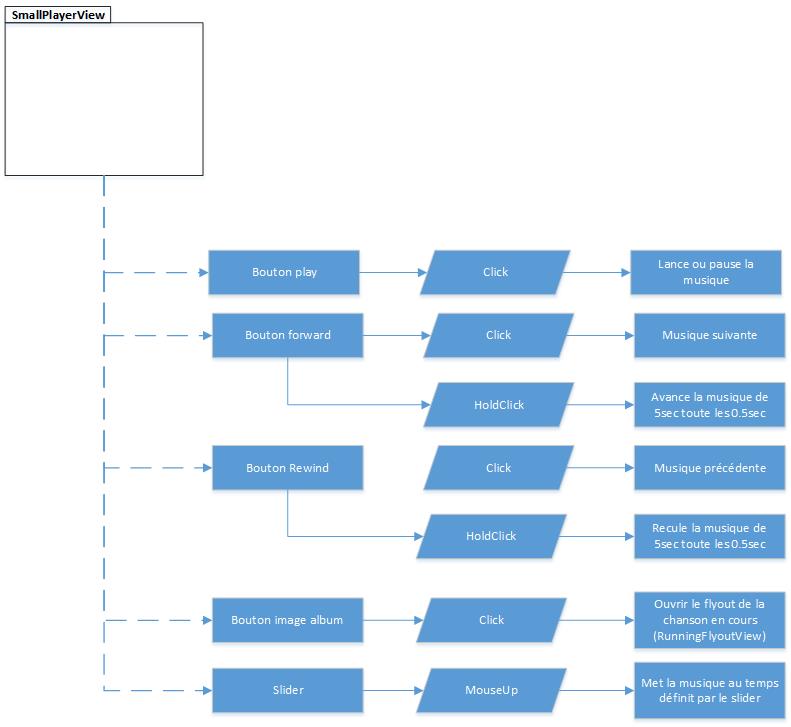


Figure Modèle logique de données

### Schéma évènementiel

## Conception des tests

Les tests seront effectué 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 au conseils de monsieur Sahli, l’application devra supporter au moins le MP3, le WAV et le WMA. Ces trois format permettront de couvrir 90%[[1]](#footnote-1) des flux renvoyer par les web radio.

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

|  |  |
| --- | --- |
| **AudioTest** |  |
| LoadData() | Crée une classe artiste, album et morceau. |
| MusicTestFormat() | Teste les 3 formats obligatoire en essayant de les jouer (WAV, MP3, WMA) |
| MusicOperationTrack() | Teste les opération telles que l’avance rapide, la pause |
| MusicOperationRadio() | Teste les opération telles que la pause |
| CheckTrackInfo() | Vérifie que les données reçus par la fonction sont identique au données du fichier de musique |
| FavoriteAddRemovTrackAndRadio() | Ajoute puis supprimer une radio et un morceau de la liste des favoris |
| CreatePlaylist() | Création d’un playlist |
| CheckPlaylist | Teste les opérations de bases sur la playlist (précédent, suivant) ; Ajout puis supprime un morceau de la playlist ; Supprime la playlist |

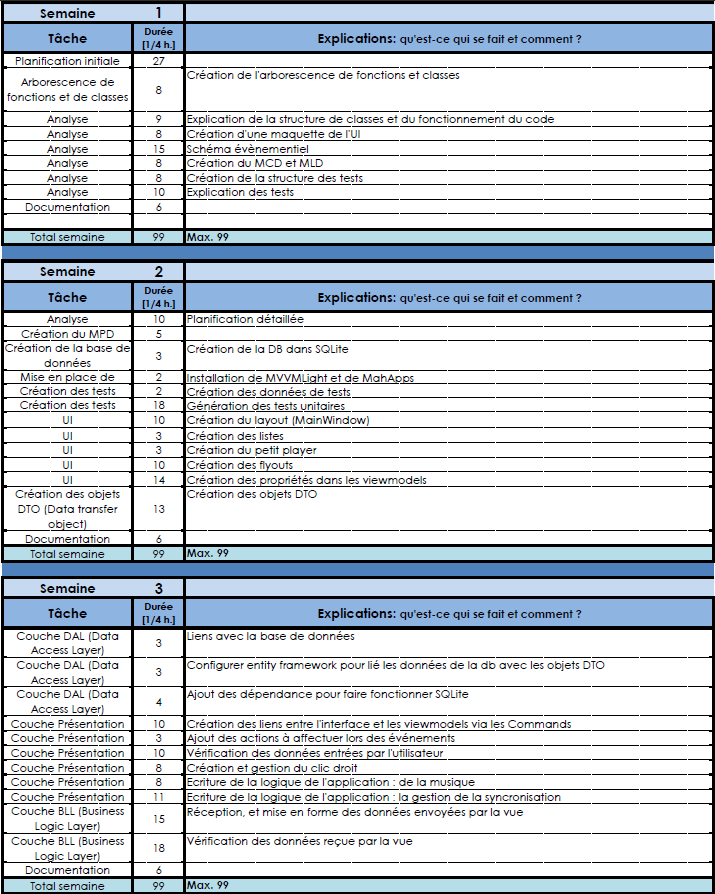
|  |  |
| --- | --- |
| **ContextTest** |  |
| CreateAndSetContext() | Crée un contexte puis essaie de l’attribuer à l’application. |
| ReadingListOperation() | Teste la liste de lecture avec les opérations de base tel que suivant et précédant; Test l’ajout et la suppression d’élément dans la liste de lecture |

|  |  |
| --- | --- |
| **DatabaseTest** |  |
| CheckConnection() | Teste la connexion à la base de données |

|  |  |
| --- | --- |
| **RadioTest** |  |
| CheckAPIConnection() | Vérifie que la connexion avec l’API s’effectue correctement |
| ChackApiData() | Test que les données reçus par l’API sont correctes |
| CheckRadioInfo() | Vérifie que les données de l’élément radio correspond au données attendues |

|  |  |
| --- | --- |
| **SyncTest** |  |
| AnalyseFolderTree() | Vérifie que l’arborescence de fichier et détecte et analyse bien tous les dossiers |
| SyncFolder() | Tante de synchroniser un dossier test |

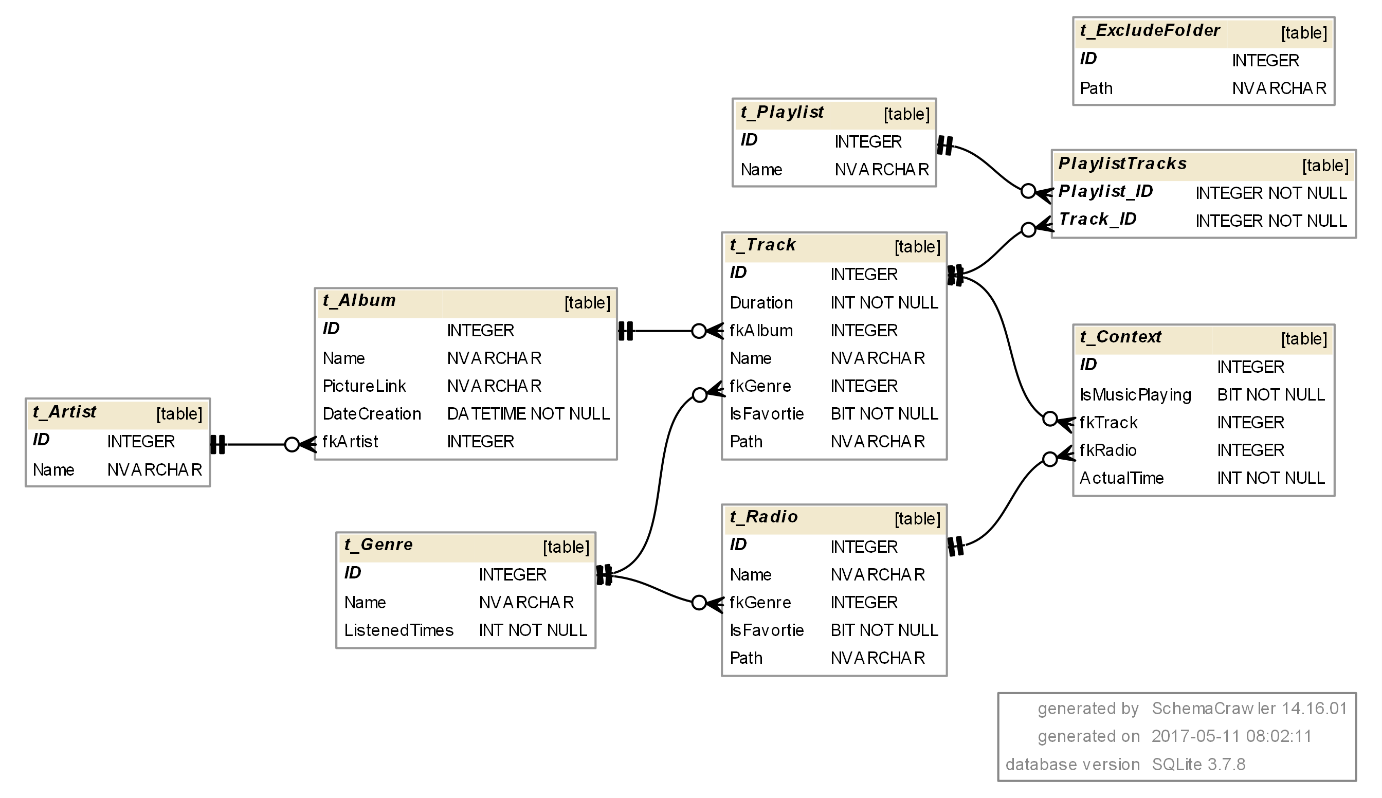
# H:\TPI\PlanificationDétailée\planif2.PNGPlanification détaillée



# Réalisation

## Dossier de Réalisation

### MPD



Cette partie permet de reproduire ou reprendre le projet par un tiers.

Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :

Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)

Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)

Code source commenté des éléments logiciels développés.

Modèle physique d’une base de données.

Arborescences des documents produits.

Schémas, plans d’adressages, plan de nommage, etc.

Il faut décrire le parcours de réalisation et justifier les choix.

## Modifications

Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.

Date, raison, description, etc.

# Tests

## Dossier des tests

On dresse le bilan des tests effectués (qui, quand, avec quelles données…) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats attendus et obtenus, ainsi que les actions à entreprendre en conséquence (et une estimation de leur durée).

Si des tests prévus dans la stratégie n'ont pas pu être effectués :

raison, décisions, etc.

Liste des bugs répertoriés avec la date de découverte et leur état:

Corrigé, date de correction, corrigé par, etc.

# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différences entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Bibliographie

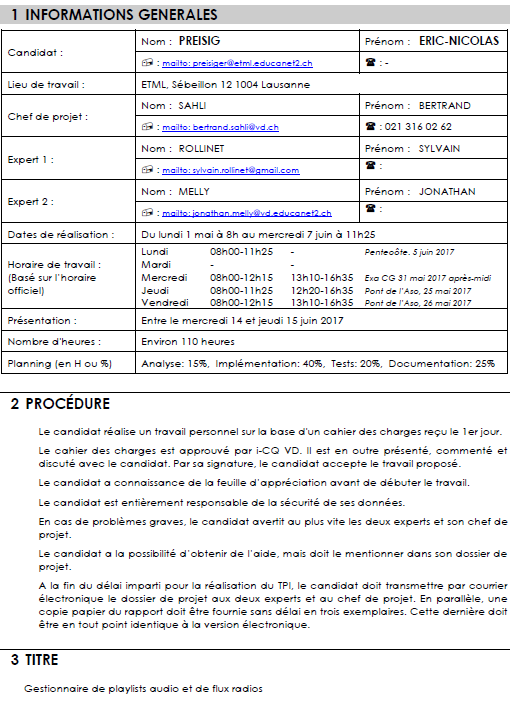
Références des livres, revues et publications utilisés durant le projet.

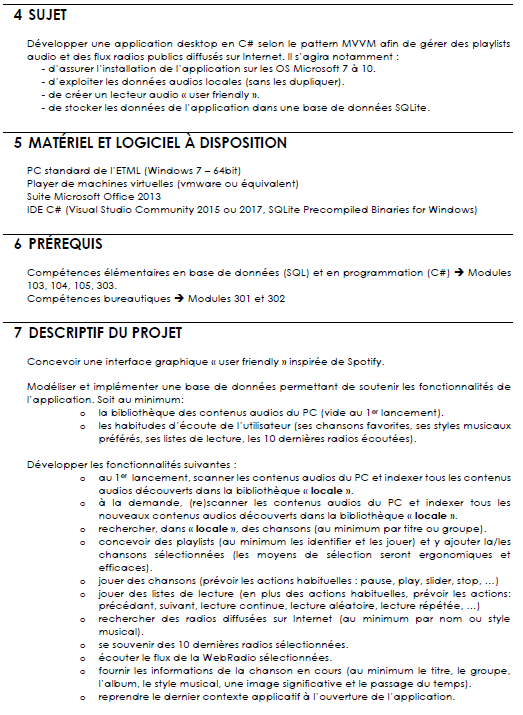
## Webographie

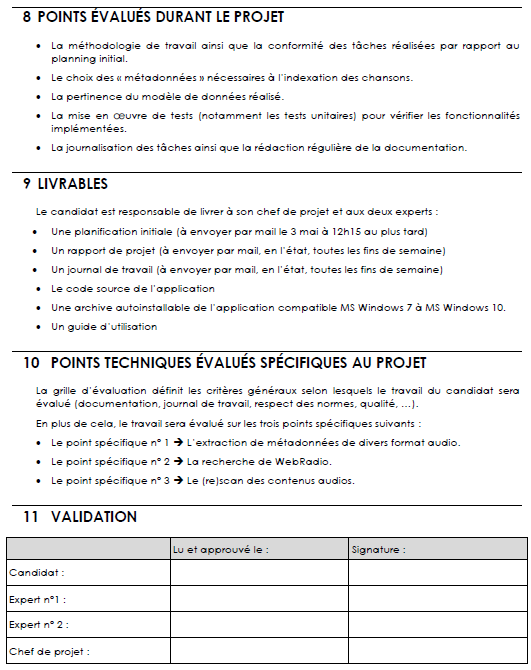
Références des sites Internet consultés durant le projet.

# Annexes

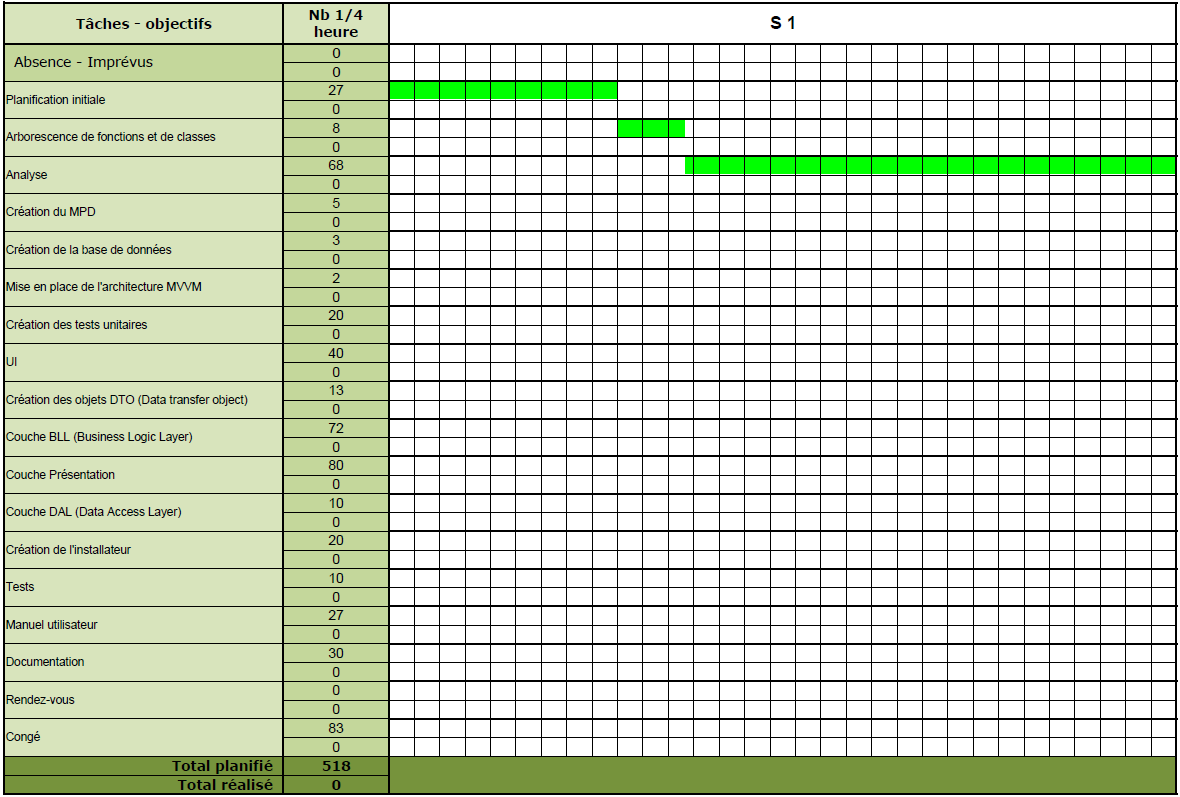
## Cahier des charges

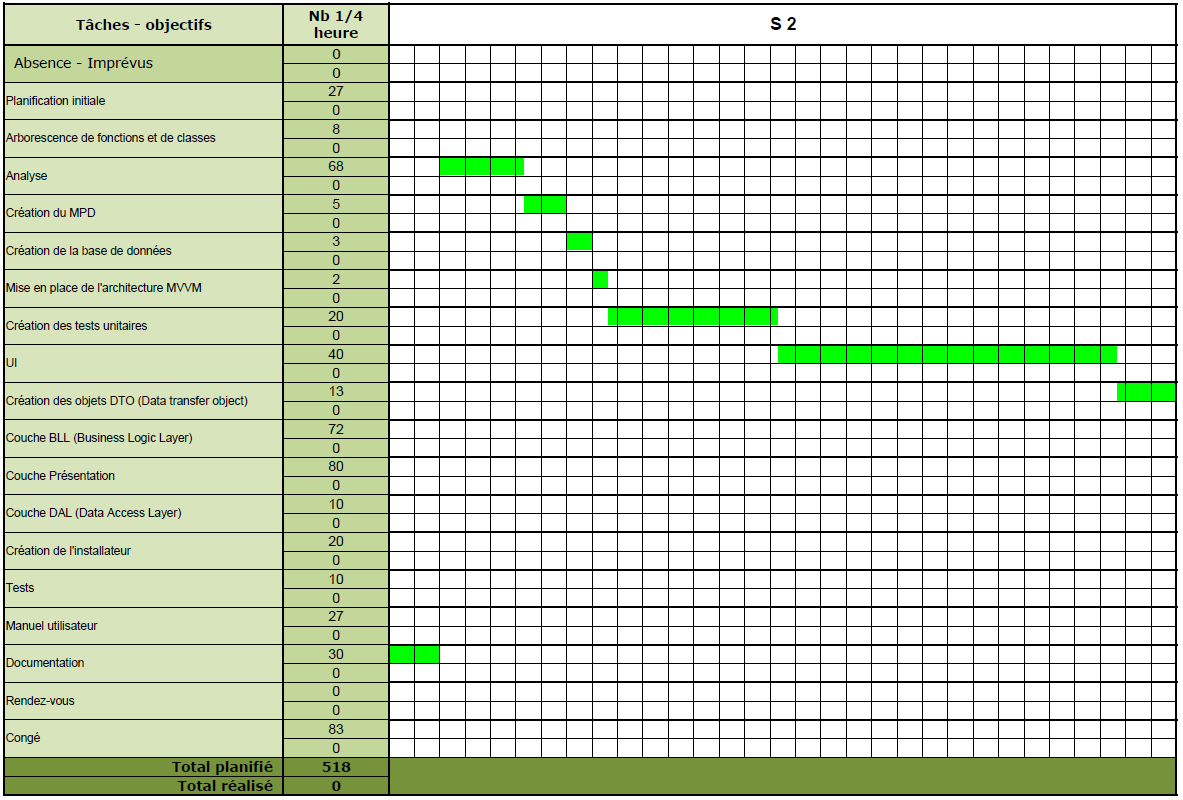


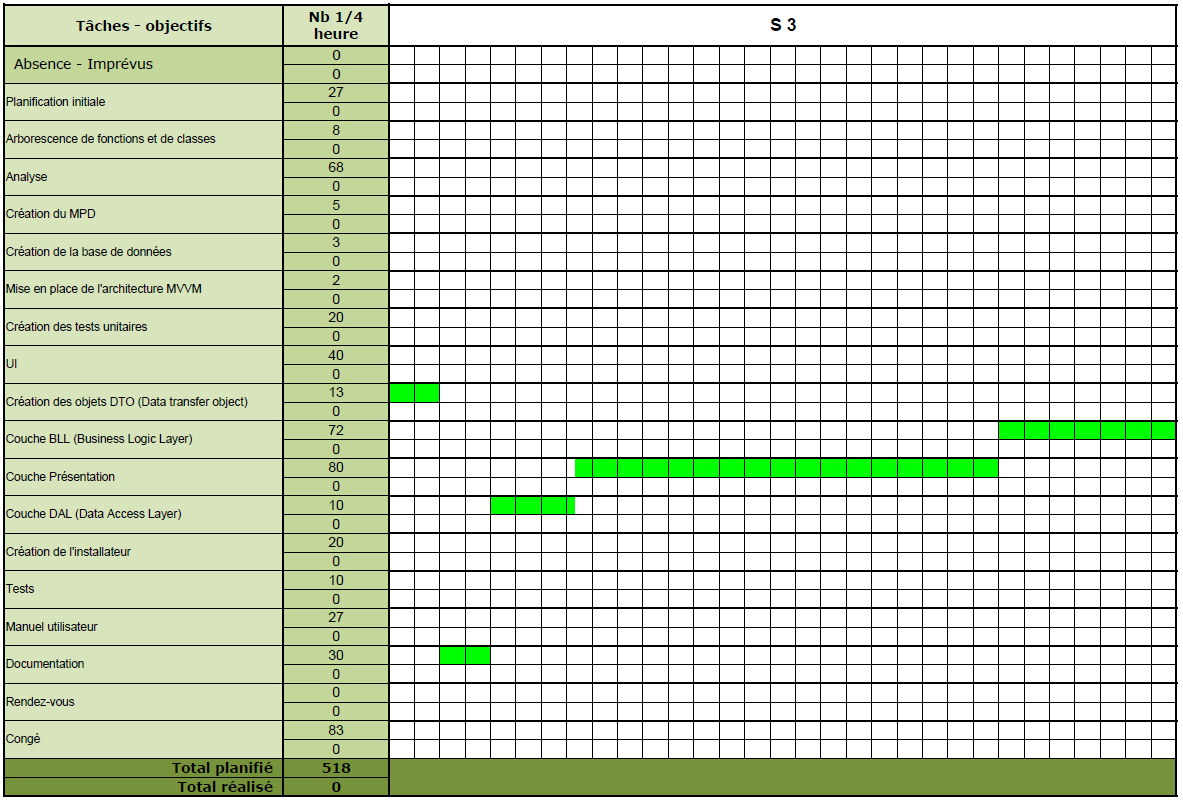


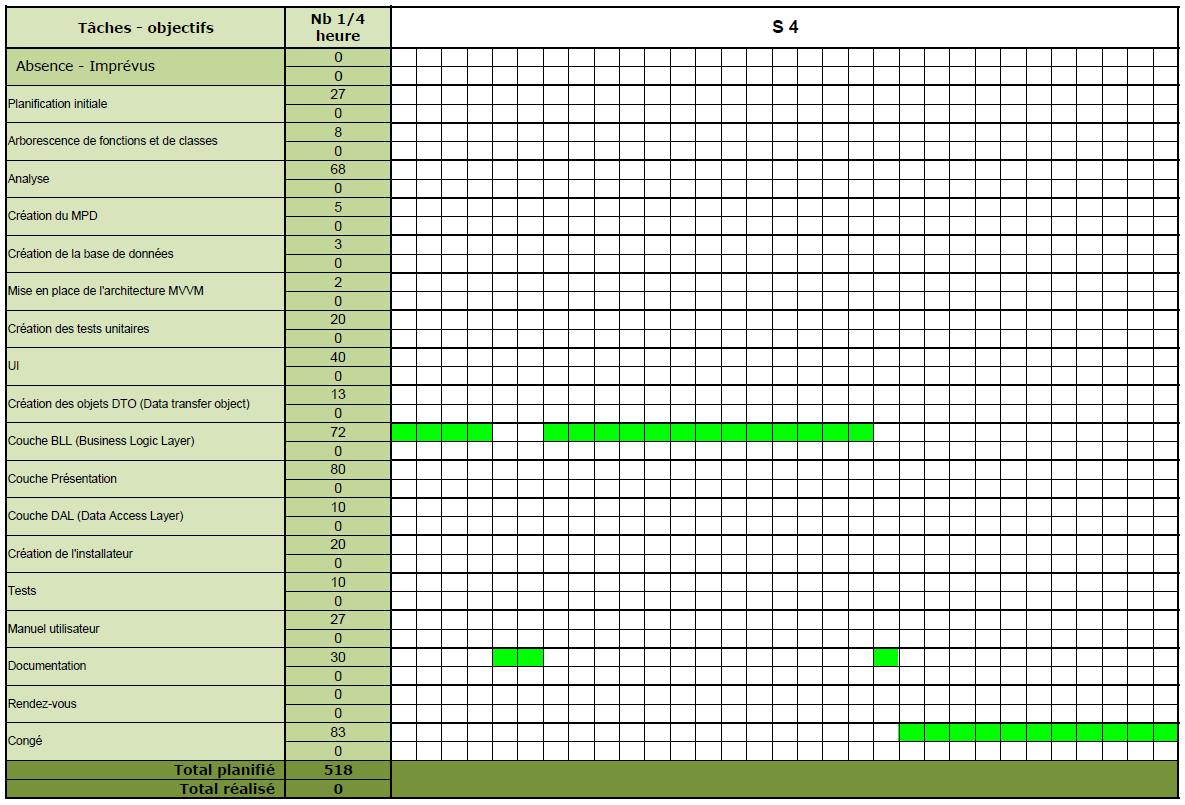


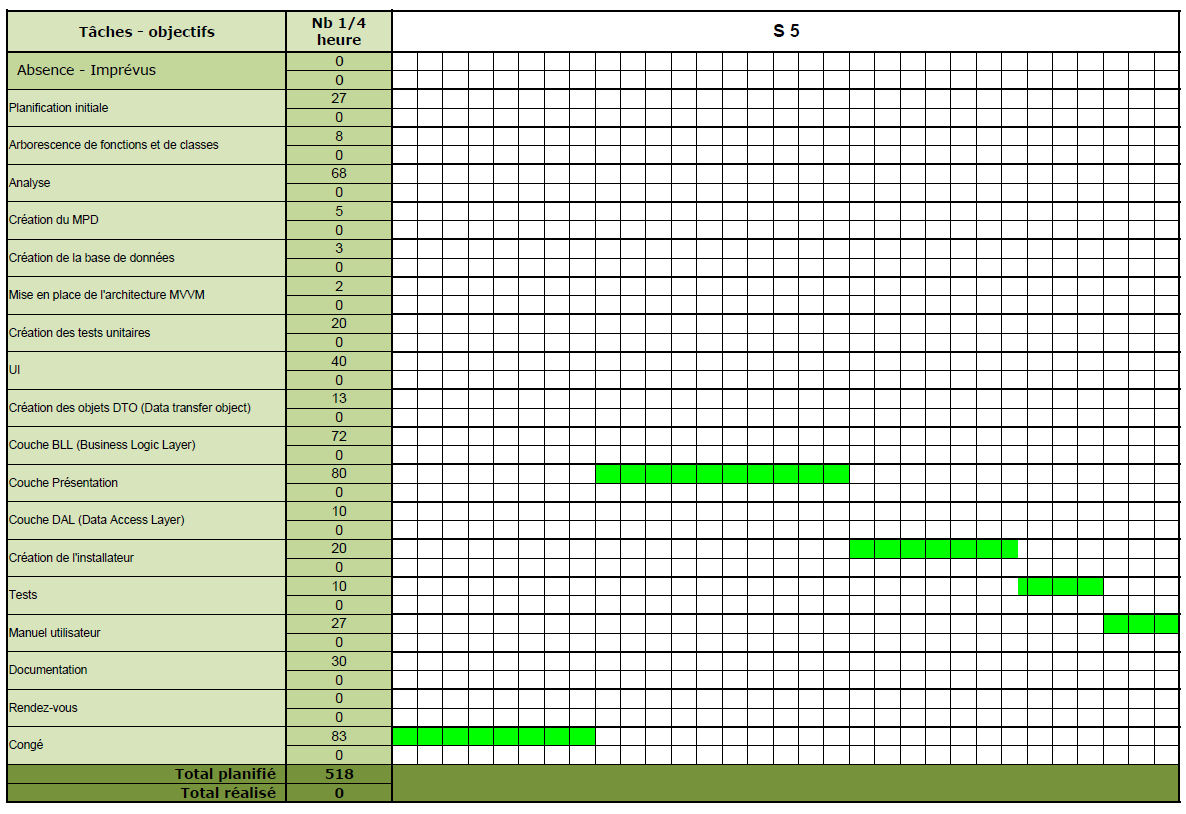
## Gantt

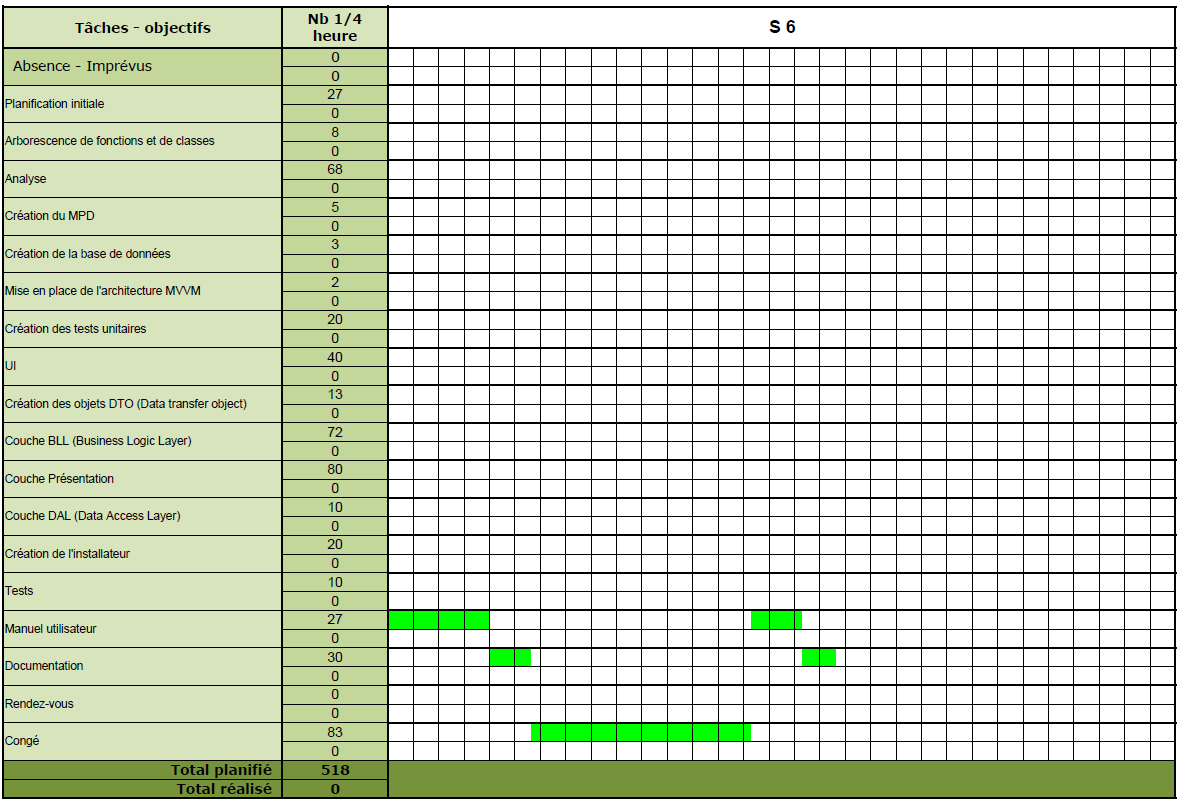












## Remerciement

Musiques : Eric Taylor

Listing du code source (partiel ou, plus rarement complet)

Guide(s) d’utilisation et/ou guide de l’administrateur

Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).

Extraits de catalogue, documentation de fabricant, etc.

1. Selon M.Sahli [↑](#footnote-ref-1)