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 5](#_Toc474850069)

[1.1 Titre 5](#_Toc474850070)

[1.2 Description 5](#_Toc474850071)

[1.3 Matériel et logiciels à disposition 5](#_Toc474850072)

[1.4 Prérequis 5](#_Toc474850073)

[1.5 Cahier des charges 5](#_Toc474850074)

[~~1.5.1~~ ~~Objectifs et portée du projet (objectifs SMART)~~ 5](#_Toc474850075)

[~~1.5.2~~ ~~Caractéristiques des utilisateurs et impacts~~ 5](#_Toc474850076)

[~~1.5.3~~ ~~Fonctionnalités requises (du point de vue de l’utilisateur)~~ 5](#_Toc474850077)

[~~1.5.4~~ ~~Contraintes~~ 5](#_Toc474850078)

[~~1.5.5~~ ~~Travail à réaliser par l'apprenti~~ 5](#_Toc474850079)

[~~1.5.6~~ ~~Si le temps le permet …~~ 5](#_Toc474850080)

[~~1.5.7~~ ~~Méthodes de validation des solutions~~ 5](#_Toc474850081)

[~~1.6~~ ~~Les points suivants seront évalués~~ 6](#_Toc474850082)

[~~1.7~~ ~~Validation et conditions de réussite~~ 6](#_Toc474850083)

[2 Planification Initiale 7](#_Toc474850084)

[3 Analyse 7](#_Toc474850085)

[3.1 Opportunités 7](#_Toc474850086)

[3.2 Document d’analyse et conception 8](#_Toc474850087)

[3.3 Conception des tests 13](#_Toc474850088)

[3.4 Planification détaillée 13](#_Toc474850089)

[4 Réalisation 13](#_Toc474850090)

[4.1 Dossier de Réalisation 13](#_Toc474850091)

[4.2 Modifications 13](#_Toc474850092)

[5 Tests 14](#_Toc474850093)

[5.1 Dossier des tests 14](#_Toc474850094)

[6 Conclusion 14](#_Toc474850095)

[6.1 Bilan des fonctionnalités demandées 14](#_Toc474850096)

[6.2 Bilan de la planification 14](#_Toc474850097)

[6.3 Bilan personnel 14](#_Toc474850098)

[7 Divers 14](#_Toc474850099)

[7.1 Journal de travail 14](#_Toc474850100)

[7.2 Bibliographie 14](#_Toc474850101)

[7.3 Webographie 14](#_Toc474850102)

[8 Annexes 15](#_Toc474850103)

# Spécifications

## Titre

A compléter par un titre cours et pertinent. Cela peut être une reprise ou compléter le titre de la première page …

## Description

A compléter par une explication du contexte, de la situation, des raisons générales de la mise en route d’un tel projet. Le lecteur doit pouvoir comprendre les motivations du lancement du projet…

## Matériel et logiciels à disposition

A compléter par ce qui est nécessaire pour le démarrage …

## Prérequis

A compléter par une description des compétences, des connaissances et de la formation minimum pour être à même de réaliser le projet …

## Cahier des charges

Le document fourni par le chef de projet fait foi. Il doit être mis en annexe 🡺 Lien sur CDC

### ~~Objectifs et portée du projet (objectifs SMART)~~

~~A compléter. Il s’agit d’ébaucher des réponses aux questions de l’acronyme CQQCOQP (Combien, Quoi, Qui, Comment, Où, Quand, Pourquoi)~~

### ~~Caractéristiques des utilisateurs et impacts~~

~~A compléter… Il s’agit d’identifier le(s) profil(s) de(s) utilisateur-trice(s) type, et les conséquences que cela va avoir sur la conception (couleurs, ergonomie, utilisation, etc.)~~

### ~~Fonctionnalités requises (du point de vue de l’utilisateur)~~

~~A compléter par une espèce de mode d’emploi du produit. S’il s’agissait d’une montre, décrire qu’à part l’heure, il y aura la possibilité d’utiliser un chronomètre, un réveil, …~~

### ~~Contraintes~~

~~Sécurité, backups, disponibilité, système utilisé, interfaces avec autres logiciels, etc.~~

### ~~Travail à réaliser par l'apprenti~~

~~Décrire à quoi doit ressembler le travail produit, ce qu’il faudra rendre …~~

### ~~Si le temps le permet …~~

~~Objectifs complémentairesau cas où le projet n’est pas assez ambitieux dans le temps imparti…~~

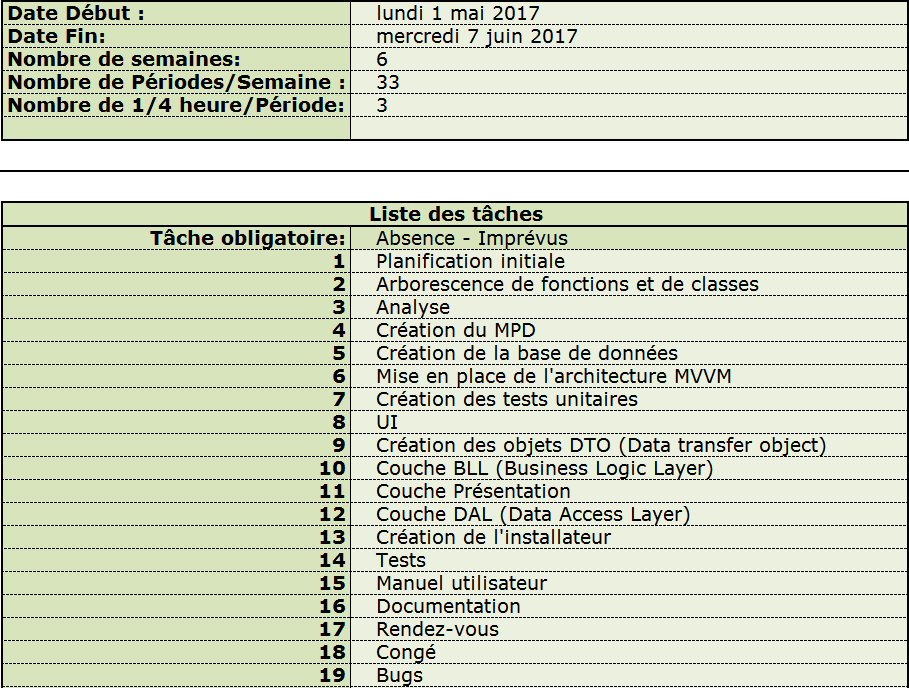
### ~~Méthodes de validation des solutions~~

~~Comment les tests vont être entrepris, quels tests doivent être entrepris, etc.…~~

## Les points suivants seront évalués

## Validation et conditions de réussite

# Planification Initiale



*Gantt disponible en annexe*

# Analyse

## Opportunités

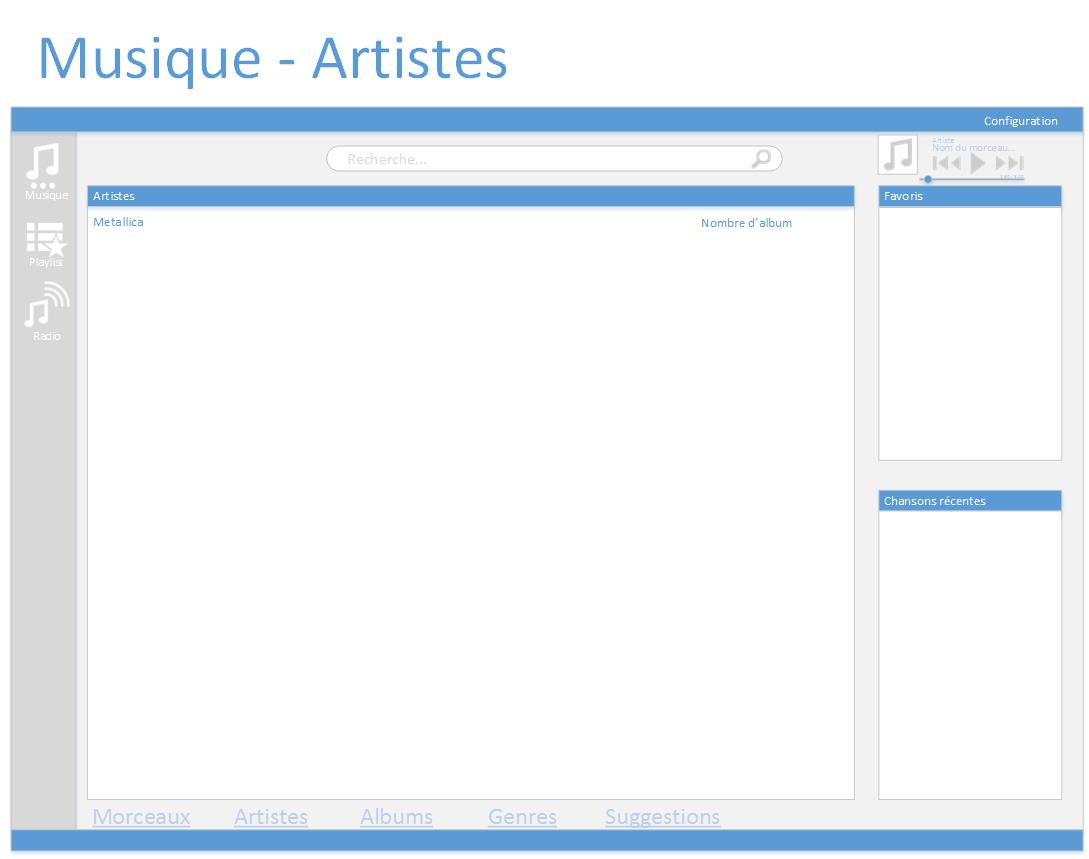
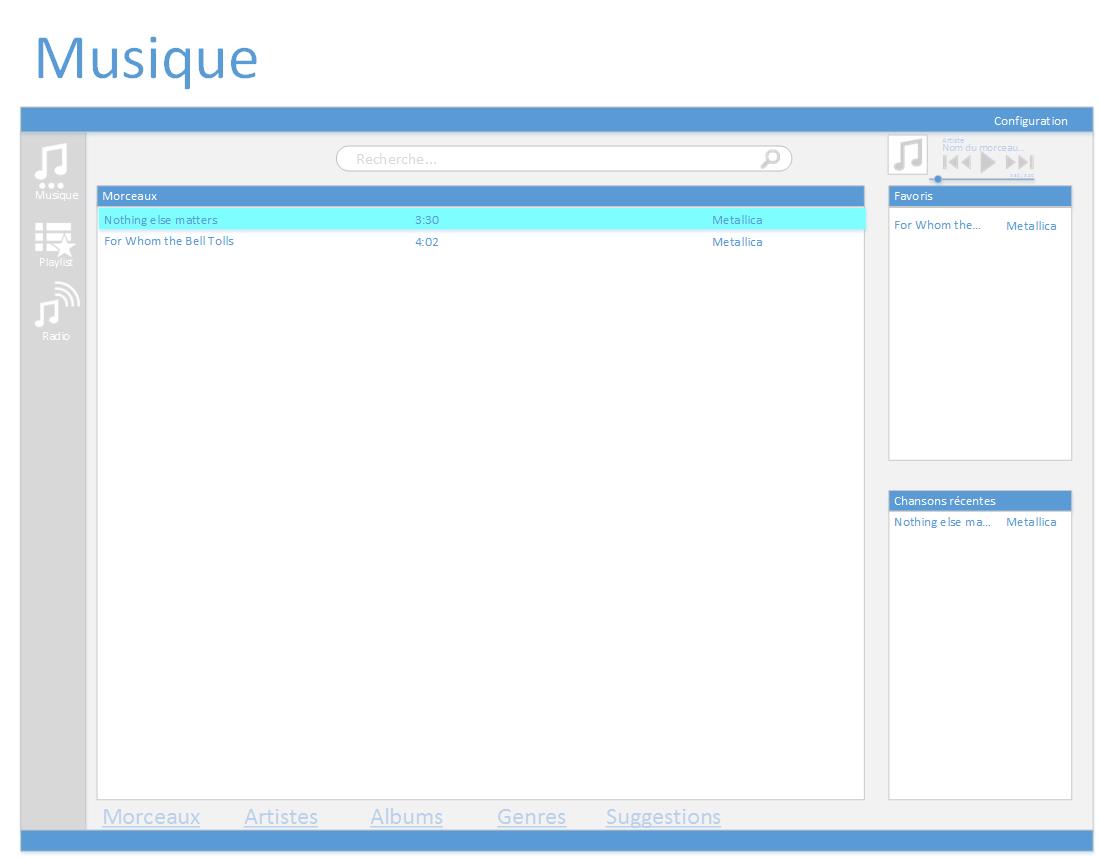
Ce projet me permettra de développer mes capacités en développement .NET. Il me donnera l’opportunité d’approfondir mes connaissances dans l’architecture MVVM.

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

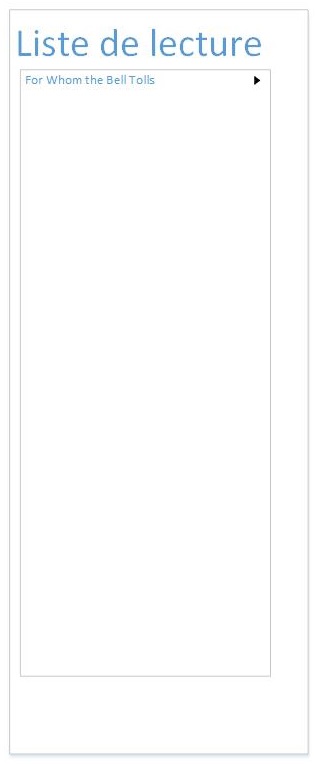
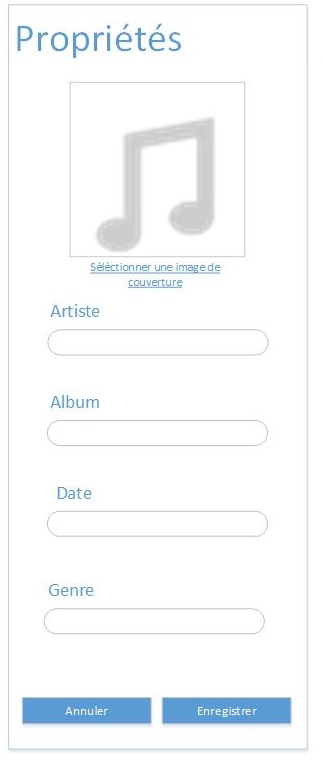
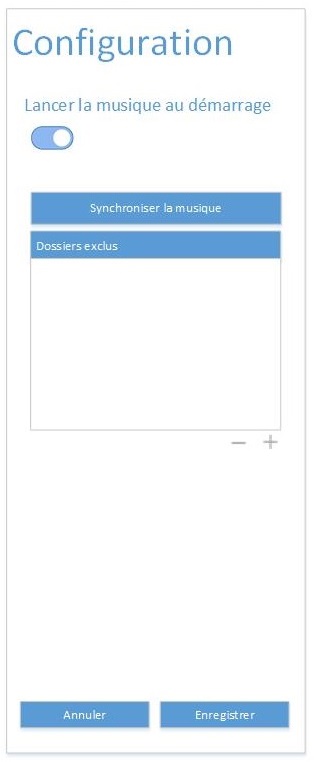
La sauvegarde du contexte de l’application sera un point difficile. Il y a plusieurs options pour pouvoir garder le contexte applicatif après la fermeture de l’application. Je pense crée une classe qui contient toute les données à sauvegarder. Celle-ci sera enregistré dans la base de données. À l’ouverture 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



### H:\TPI\Maquette\Album.jpgH:\TPI\Maquette\Genre.jpgH:\TPI\Maquette\Suggestion.jpgH:\TPI\Maquette\Playlist.jpgH:\TPI\Maquette\Radio.jpgH:\TPI\Maquette\Recherche.jpgH:\TPI\Maquette\Encours.jpg



### 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 »

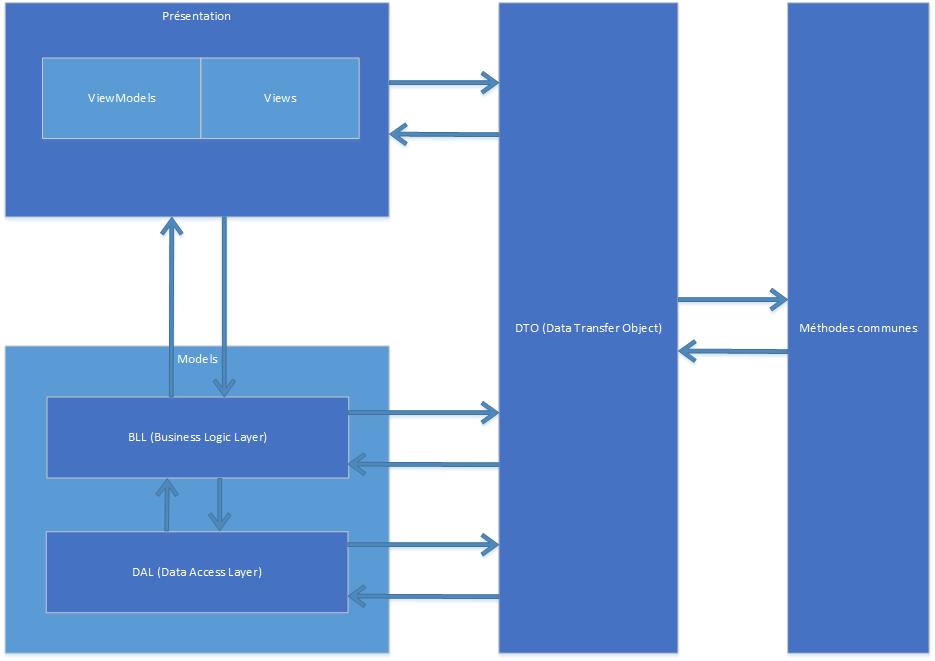


Figure 1 Illustration de l'architecture

### 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*
      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***

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***

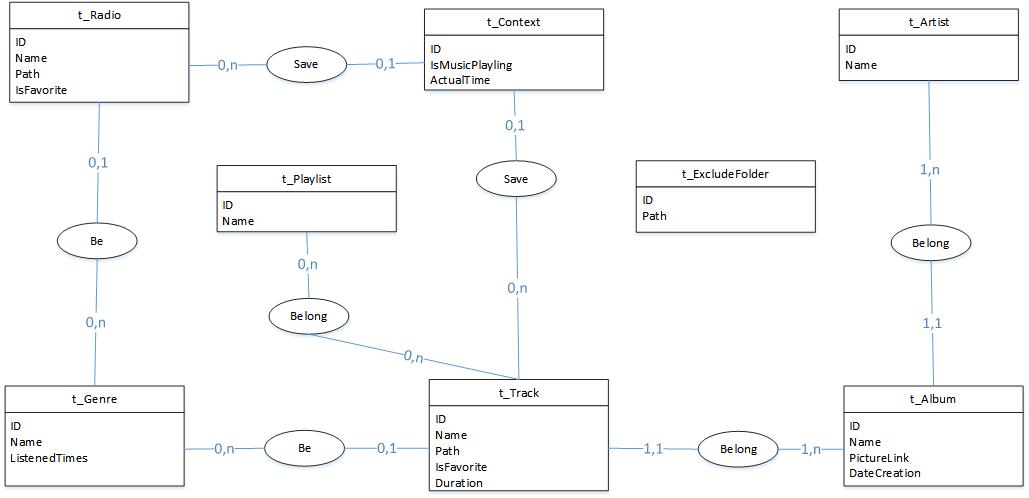
******

Figure 2 Modèle conceptuel de données

***MLD***

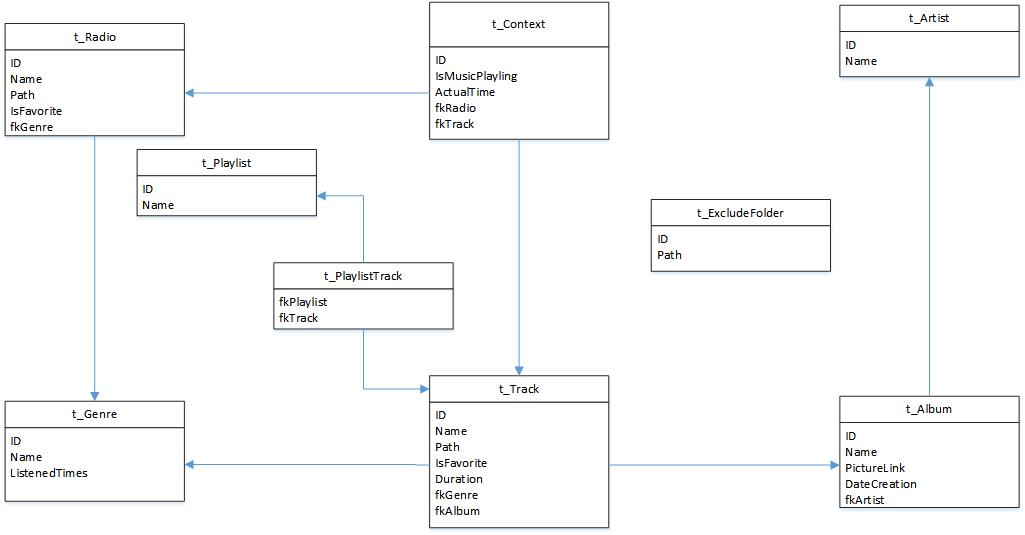
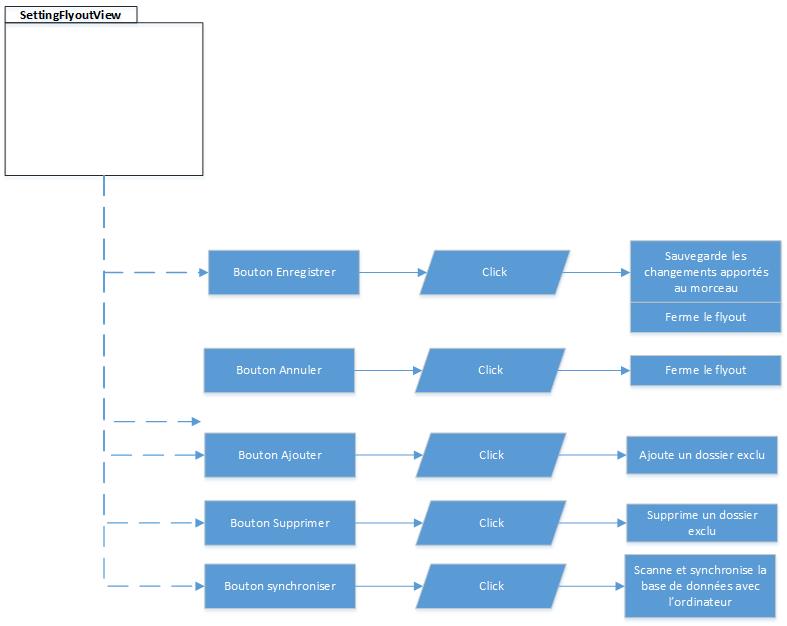
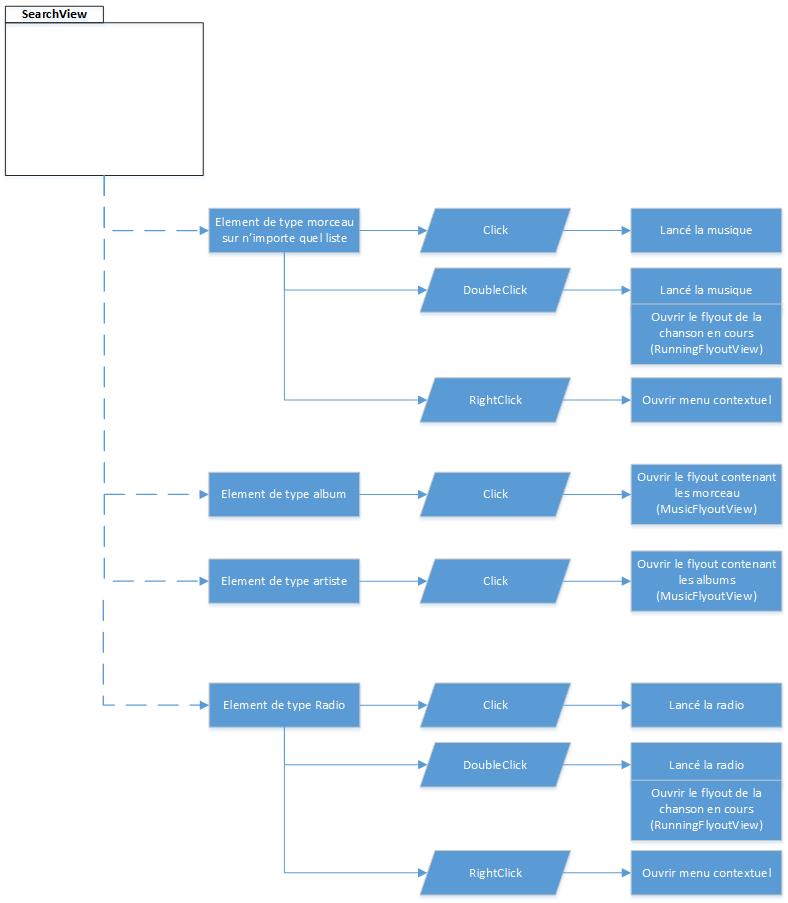
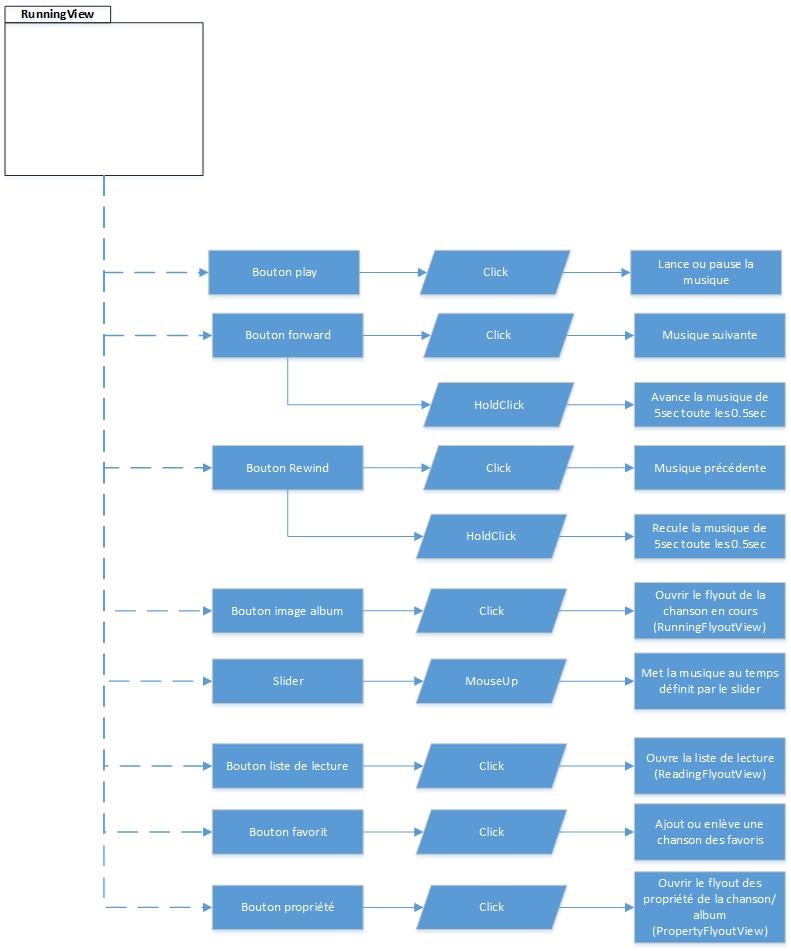
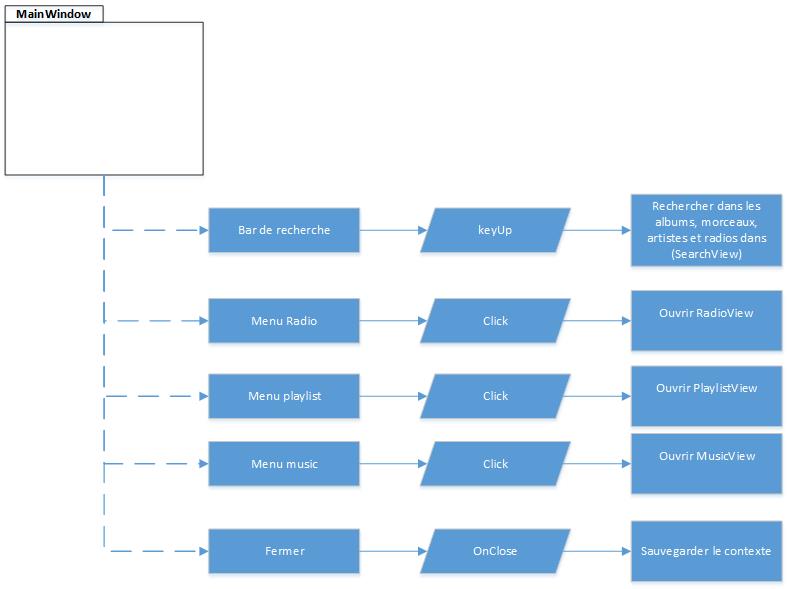
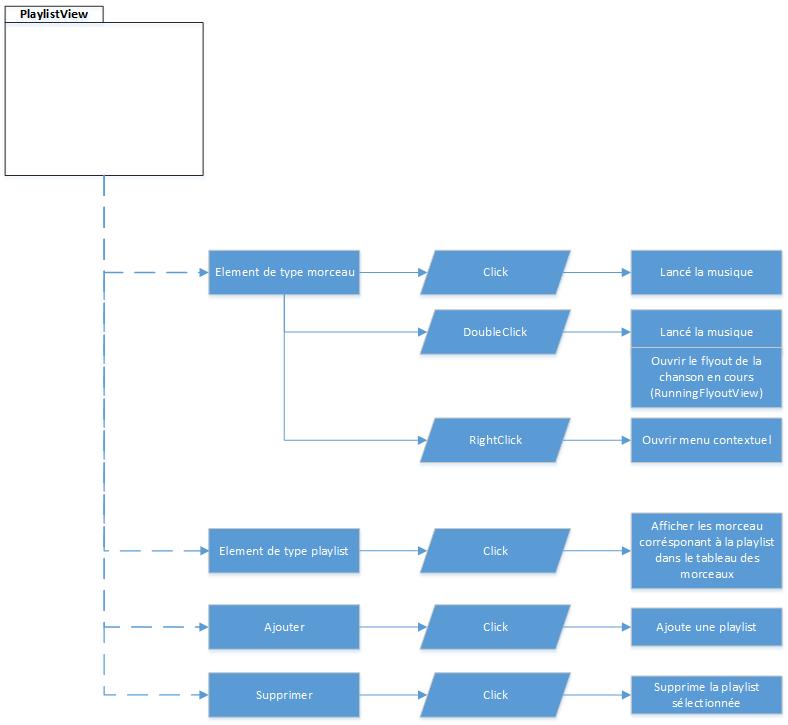
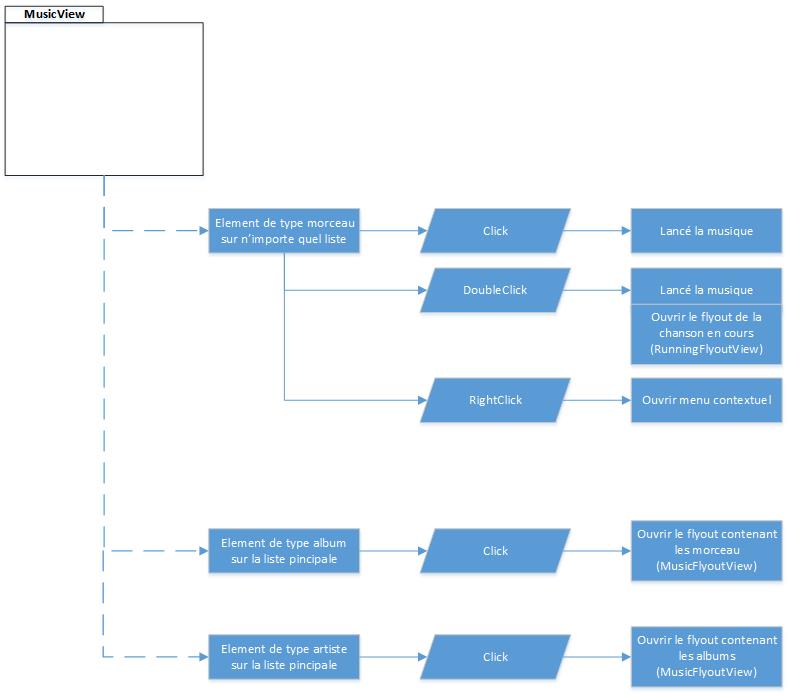
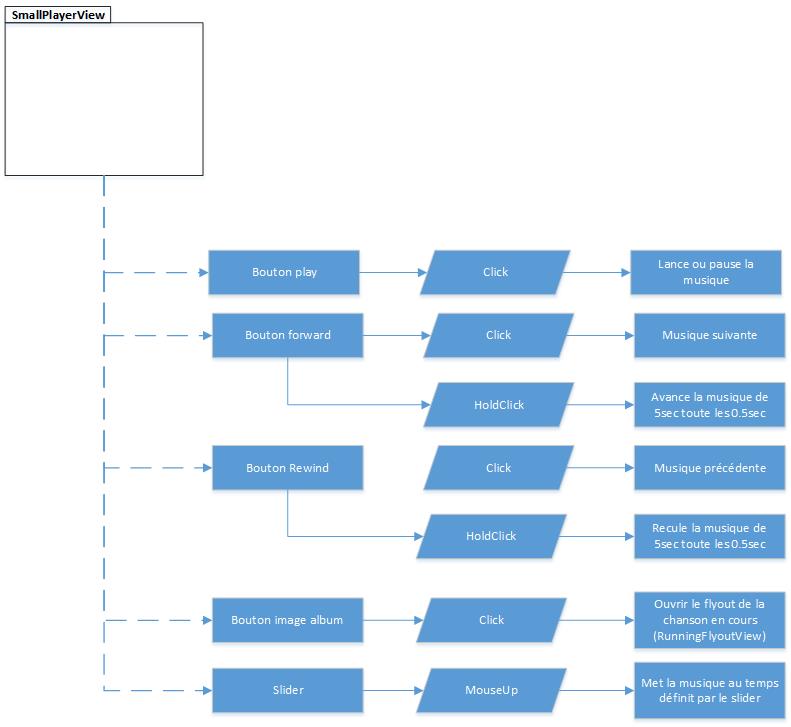


Figure 3 Modèle logique de données

### Schéma évènementiel

### H:\TPI\Schéma évènementiel\RadioView.jpg H:\TPI\Schéma évènementiel\PropertyFlyoutView.jpg

## Conception des tests

Ce paragraphe permet de spécifier la stratégie de test qui sera menée au point 5.1

Qui, quand, avec quelles données, dans quel ordre, etc.

Avec quels matériels, quels services, etc.

## Planification détaillée

A ce stade, après l’analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, …) peut être finalisé.

Le planning détaillé doit s’inscrire dans le planning initial. Il faut que l’on puisse situer cette planification détaillée par rapport à la planification initiale.

# Réalisation

## Dossier de Réalisation

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

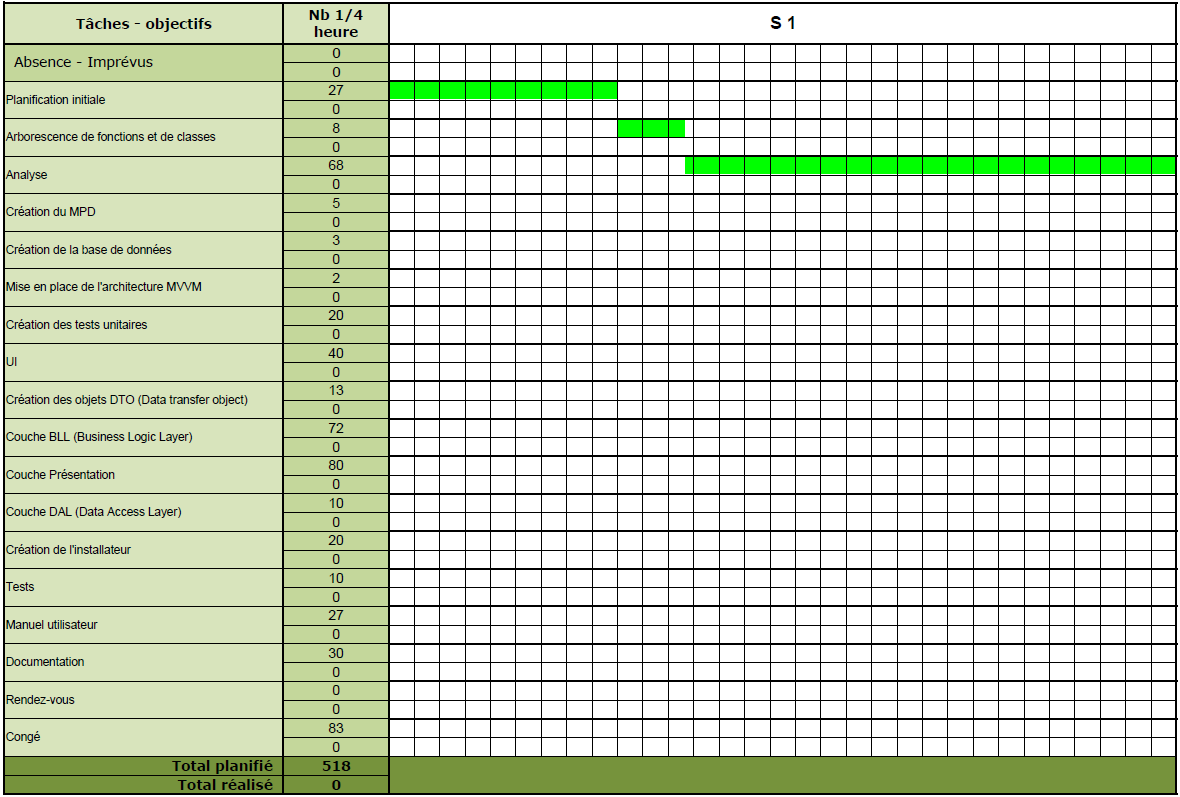
Références des sites Internet consultés durant le projet.

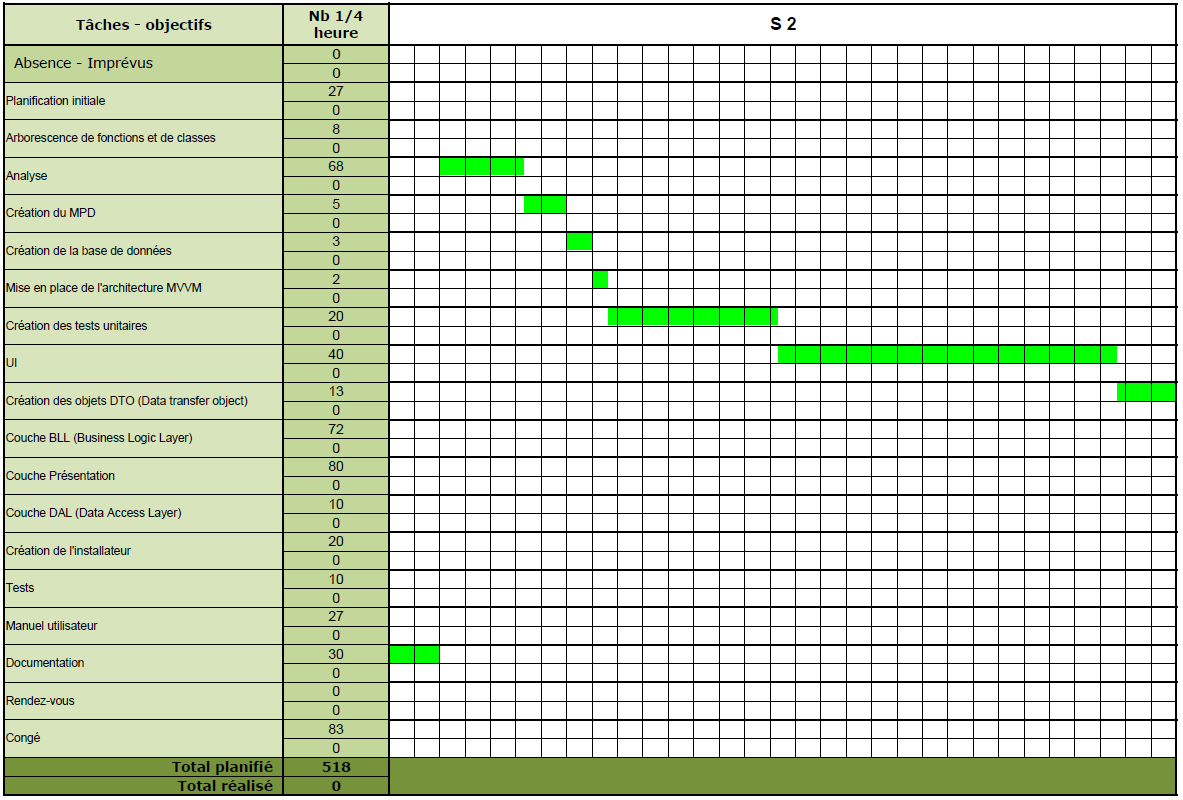
# Annexes

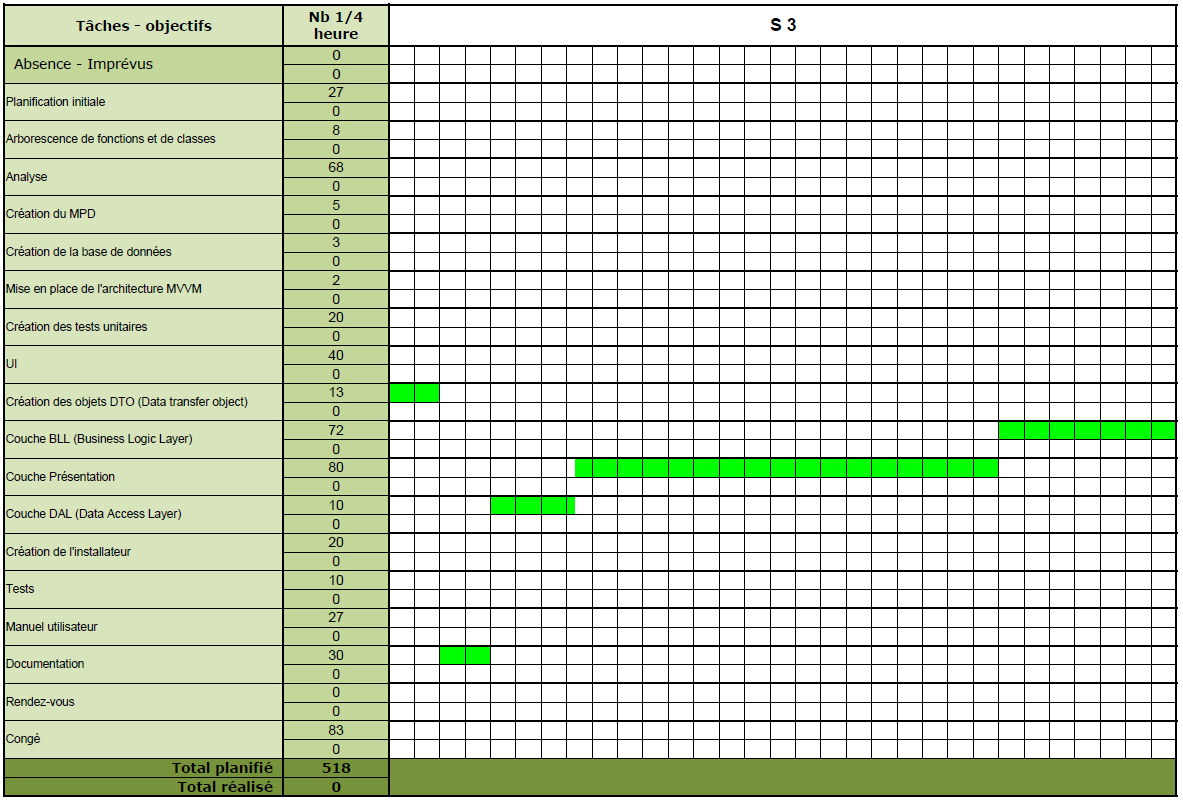
Cahier des charges

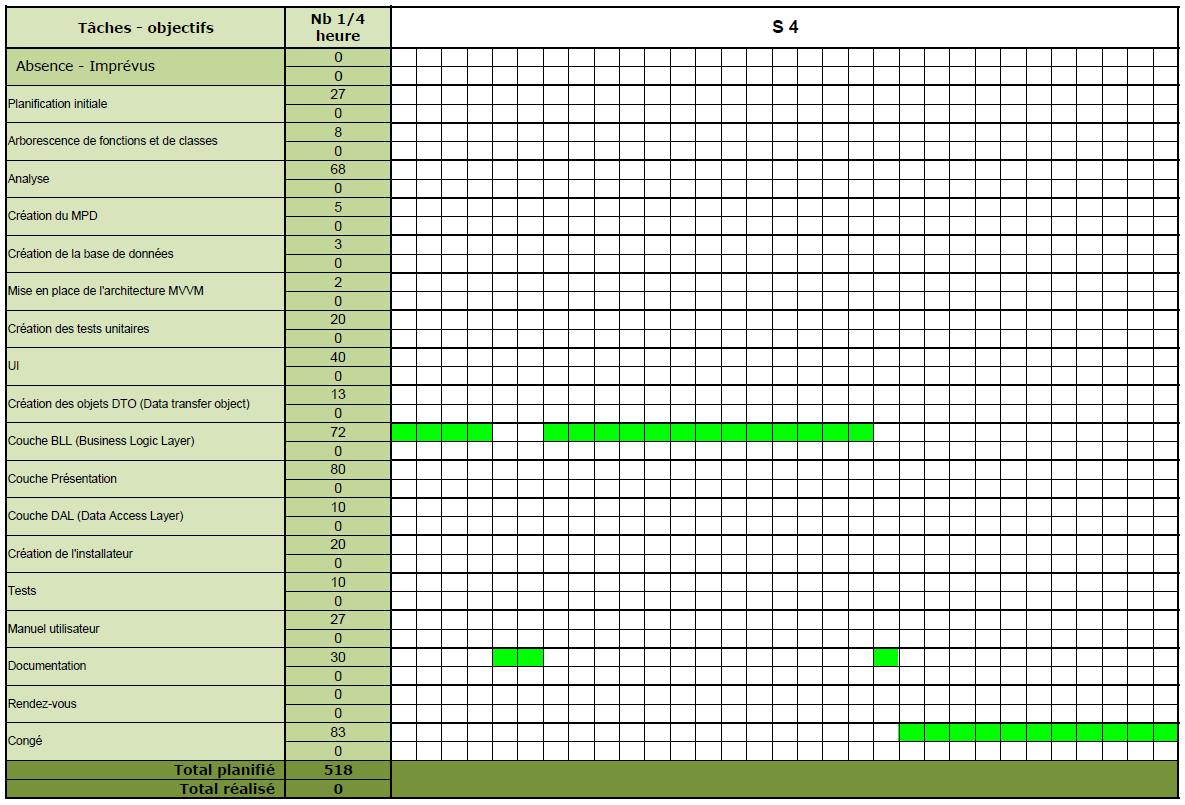
## Cahier des charges

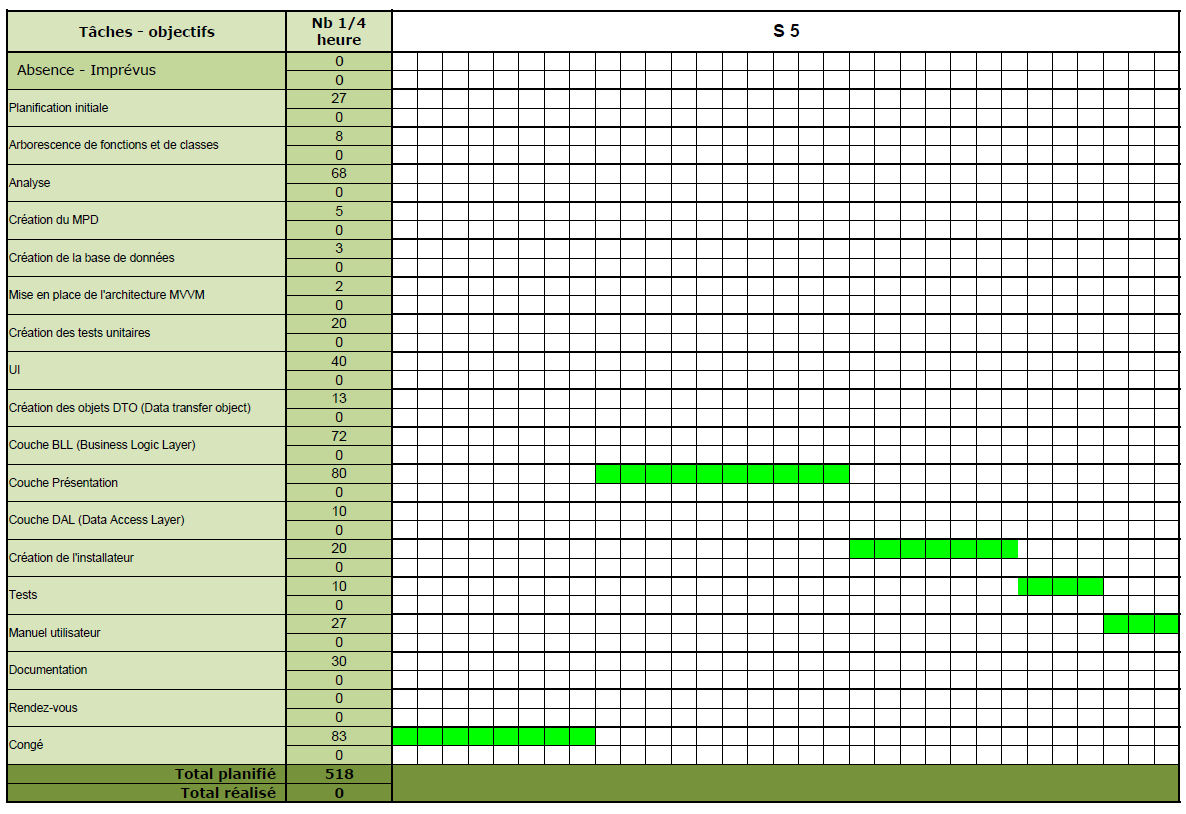
## Gantt

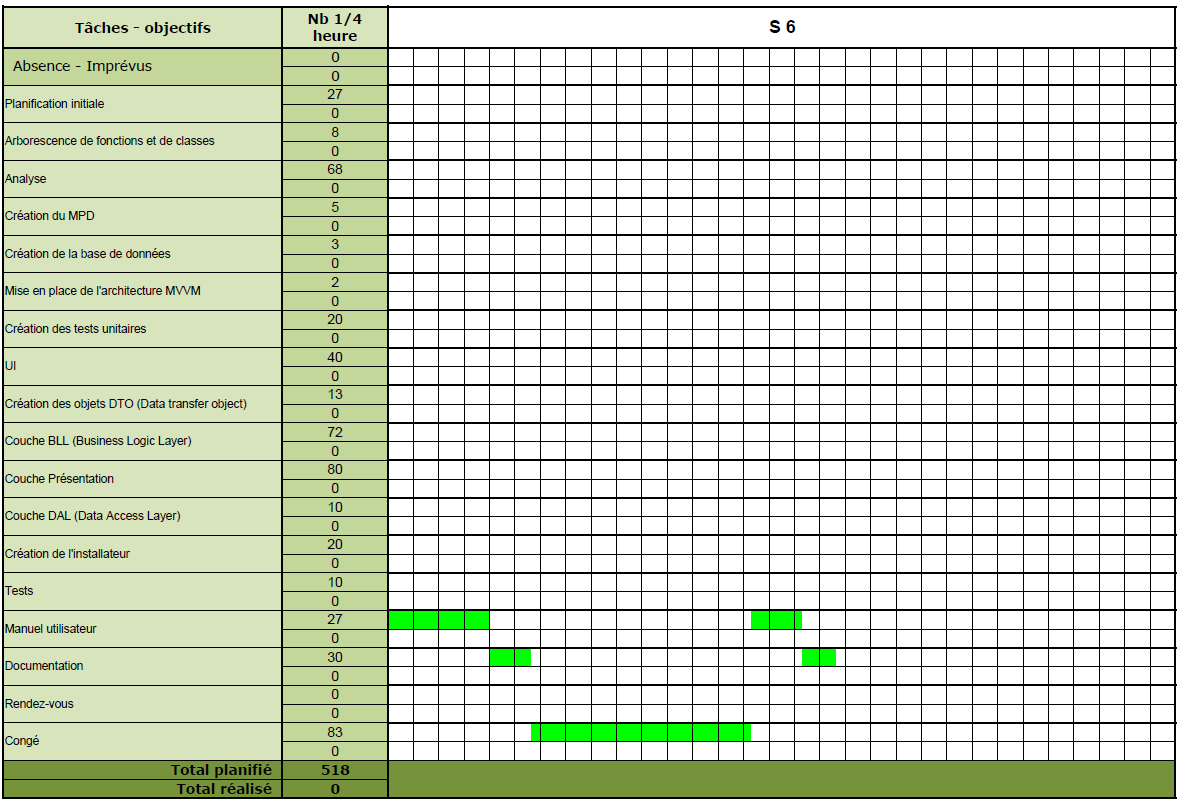












Listing du code source (partiel ou, plus rarement complet)

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.