

WebradioManager

Travail de diplôme 2014

Simon Menetrey – T.IN E2A

16/04/2014

1 Résumé

1.1 Français

1.2 English

Contenu

1	Résumé	1
1.1	Français.....	1
1.2	English.....	1
2	Introduction.....	6
2.1	Mise en situation	6
2.2	Sujet.....	6
2.3	Qu'est-ce qu'une webradio ?	6
2.4	Pourquoi ce sujet ?	6
2.5	Termes.....	7
3	Cahier des charges.....	8
4	Analyse fonctionnelle	9
4.1	Schéma de l'application	9
4.2	Interface principale	10
4.2.1	Fenêtre d'administration.....	10
4.2.2	Menu principale	10
4.3	Gestion de plusieurs webradio.....	11
4.3.1	Création d'une webradio.....	12
4.3.2	Sélection d'une webradio.....	12
4.4	Serveur de diffusion interne et externe	12
4.5	Onglet « Status »	13
4.6	Gestion des musiques/pubs	14
4.6.1	Affichage.....	14
4.6.2	Importer et indexer	14
4.6.3	Ajout à une playlist.....	15
4.7	Gestion des listes de lecture	16
4.7.1	Listes de lecture musicales	16
4.7.2	Listes de lecture publicitaires	16
4.7.3	Création	16
4.7.4	Génération automatique.....	16
4.7.5	Affichage du contenu d'une playlist.....	16
4.8	Gestion des horaires.....	17
4.8.1	Événement.....	18
4.8.2	Événement périodique	18

4.8.3	Remplissage manuel.....	18
4.8.4	Remplissage automatique	19
4.8.5	Suppression d'un événement.....	19
4.9	Gestion des transcoders.....	20
4.9.1	Création	20
4.9.2	Affichage.....	21
4.9.3	Modification	21
4.9.4	Historique de diffusion	21
4.10	Gestion du serveur	21
4.10.1	Contrôles	21
4.10.2	Configuration.....	22
4.10.3	Log	22
4.10.4	Interface web.....	22
4.10.5	Administration web	22
5	Analyse organique	24
5.1	Environnement.....	24
5.2	Diagramme de classes	24
5.2.1	Diagramme	24
5.2.2	Observateurs/Sujet	24
5.2.3	AudioType et StreamType	25
5.2.4	Classes abstraites	25
5.2.5	Stockage des webradios	25
5.3	Base de données.....	25
5.3.1	SQLite.....	25
5.3.2	Utilisation	25
5.3.3	Suppression en cascade.....	26
5.3.4	Schéma	26
5.3.5	Twebradio.....	27
5.3.6	Tsever	27
5.3.7	Tcalendar	27
5.3.8	Tcalendarevent.....	27
5.3.9	Tplaylist.....	28
5.3.10	Taudiotype.....	28
5.3.11	Tmusic.....	28

5.3.12	Tgender.....	28
5.3.13	Tplaylist_has_music.....	28
5.3.14	Thistory.....	29
5.3.15	Ttranscoder	29
5.4	Schéma de diffusion	30
5.4.1	Principe de base	30
5.4.2	Infomaniak.....	30
5.5	ShoutCast.....	31
5.5.1	Présentation	31
5.5.2	Pourquoi cet outil ?	31
5.5.3	Serveur	32
5.5.4	Transcoder.....	32
5.5.5	Schéma de fonctionnement résumé	32
5.6	Structures des dossiers/fichiers	33
5.6.1	Schéma	33
5.6.2	Exécutables Shoutcast.....	34
5.7	Initialisation de l'application	35
5.8	Gestion des processus	36
5.9	Webradio.....	36
5.9.1	Classes associée.....	36
5.9.2	Affichage des webradios disponibles	37
5.9.3	Création	37
5.9.4	Chargement	38
5.9.5	Duplication	39
5.9.6	Suppression	39
5.9.7	Génération des configurations.....	39
5.10	Transcoder.....	40
5.10.1	Définition des bitrates, taux d'échantillonnage et type d'encoder	53
5.10.2	Fichier de configuration.....	53
5.10.3	Licence MP3.....	53
5.10.4	Exécution	54
5.10.5	Statut et logs.....	54
5.10.6	Historique	54
5.10.7	Gestion des processus	54

5.11	Listes de lecture.....	44
5.11.1	Musicale	Erreur ! Signet non défini.
5.11.2	Publicitaires	Erreur ! Signet non défini.
5.11.3	Ajout à une playlist.....	44
5.11.4	Génération automatique.....	45
5.11.5	Suppression	47
5.12	Bibliothèque	40
5.12.1	Listes de lecture associées.....	Erreur ! Signet non défini.
5.12.2	Importation	40
5.12.3	Tags ID3	41
5.12.4	Analyse des tags	42
5.12.5	Indexation.....	42
5.12.6	Suppression	42
5.13	Grille horaire.....	49
5.13.1	Affichage du calendrier	51
5.13.2	Création d'événement.....	52
5.13.3	Génération automatique.....	53
5.14	Serveur de diffusion interne.....	54
5.14.1	Configuration.....	54
5.14.2	Exécution	54
5.14.3	Log	54
6	Tests.....	55
7	Plannings	56
7.1	Prévu.....	56
7.2	Final	56
8	Apports personnels	57
9	Conclusion	58
10	Améliorations possibles.....	59
11	Références.....	59
12	Annexes	59

2 Introduction

2.1 Mise en situation

Je m'appelle Simon Menetrey, j'ai 20 ans et je suis actuellement en dernière année de formation technicien ES en informatique. J'ai effectué un CFC d'informaticien en 4 ans avant de commencer ma formation actuelle.

Cette documentation concerne mon travail de diplôme réalisé pour ma dernière année en tant que technicien ES en informatique. Ce travail a pour sujet la création d'un gestionnaire de webradio.

Mon professeur de diplôme, Monsieur Garcia, m'a mis en contact avec une entreprise (KTFM) qui recherche un programme de gestion pour leur webradio. En effet, actuellement, c'est un tiers qui s'occupe de la diffusion de leur webradio via des émissions préalablement enregistrées dans leurs studios. Dans l'état actuel, l'entreprise n'a pas un contrôle direct sur la diffusion de son contenu et elle désirerait pouvoir gérer elle-même l'intégralité de leur webradio. Ainsi, elle supprimera un intermédiaire et aura pleinement contrôle de la diffusion.

En plus de cela, KTFM a aussi besoin de pouvoir avoir une traçabilité des morceaux qu'elle diffuse avec des informations précises afin de pouvoir faciliter le paiement des droits d'auteur à la Suisa¹. J'ai donc réalisé le cahier des charges avec KTFM afin de répondre au mieux à leurs besoins.



2.2 Sujet

Gestionnaire de webradio :

- Permettre de diffuser directement depuis le logiciel
- Gérer les morceaux à diffuser/listes de lecture
- Gérer les plages horaires
- Historique de diffusion

2.3 Qu'est-ce qu'une webradio ?

Une webradio est une radio diffusée sur internet via la technologie de lecture en continu. Cette technologie fournit ce que l'on appelle un « flux » que les auditeurs écoutent via leur lecteur multimédia préféré ou via un site web.

2.4 Pourquoi ce sujet ?

Je suis passionné de musique. J'ai aussi toujours recherché un outil simple et gratuit pour gérer une webradio et sa diffusion. J'ai donc imaginé une application, tout-en-un, remplissant ce besoin. J'ai donc l'espoir que mon projet me sera autant utile à moi qu'à l'entreprise KTFM ainsi que de potentielles futures entreprises intéressées.

¹ SUISA est la coopérative des auteurs et éditeurs de musique <http://www.suisa.ch/fr/>

2.5 Termes

Voici une liste de termes qui seront utilisés dans cette documentation :

- Playlist : Liste de lecture
- Stream : Flux
- Log : Journal des événements

3 Cahier des charges

Ce projet a pour but la création d'un gestionnaire de webradio (de type shoutCAST) complet. Les principales fonctionnalités sont les suivantes :

- Possibilité de gérer plusieurs webradio indépendamment
- Gestion des playlist, horaires et pubs
 - Génération automatique de playlist
 - Génération au format xml
- Gestion des serveurs de diffusions distants et transcoder interne.
 - Diffusion de la webradio
- Indexation des fichiers musicaux (tags, chemin sur le disque dur)
- Historique des morceaux joués
 - Génération d'un compte-rendu afin de faciliter la gestion des droits d'auteurs
- Serveurs de diffusion interne (local)

Options :

- Si serveur de diffusion interne activé : Serveur WEB interne contenant un mini-site
- Modifier les informations des musiques indexées.

Plus de détails dans le résumé et le *mindmap*.

Analyses

4 Analyse fonctionnelle

4.1 Schéma de l'application

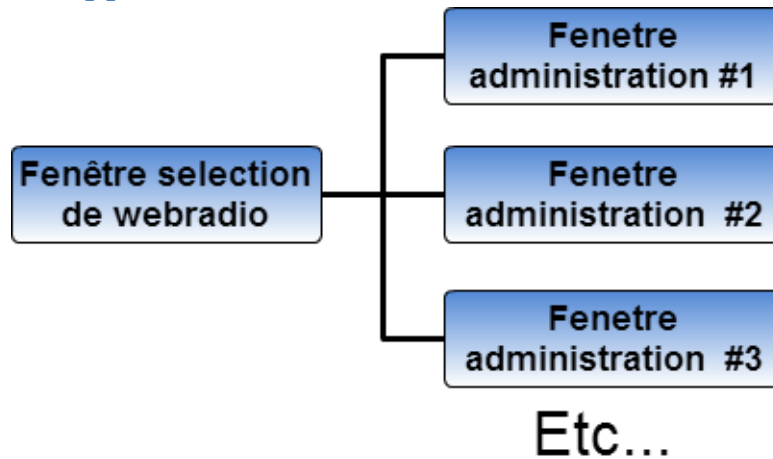


Figure 1 - Schéma application

Lors du lancement de l'application, la fenêtre de sélection de webradio se lance. L'utilisateur choisit la webradio qu'il veut gérer, puis une fenêtre d'administration s'ouvre avec les informations de la webradio sélectionnée. Il peut à tout moment réafficher la fenêtre de sélection et ouvrir une nouvelle fenêtre d'administration avec une autre webradio à gérer.

La diffusion de webradio est possible via des serveurs distants ou alors via le serveur interne (local) à l'application. Il y en a 1 par webradio.

4.2 Interface principale

4.2.1 Fenêtre d'administration



Figure 2 - Interface principale AdminView

Cette fenêtre est l'interface principale de l'application. Elle est associée à une webradio. Comme montré dans le schéma de l'application, il est possible d'avoir plusieurs fenêtres d'administration ouvertes simultanément, une pour chaque webradio lancée. Les différents onglets sont décrits dans les chapitre suivants.

4.2.2 Menu principale

TODO : complètement en fonction du menu choisi

4.3 Gestion de plusieurs webradio

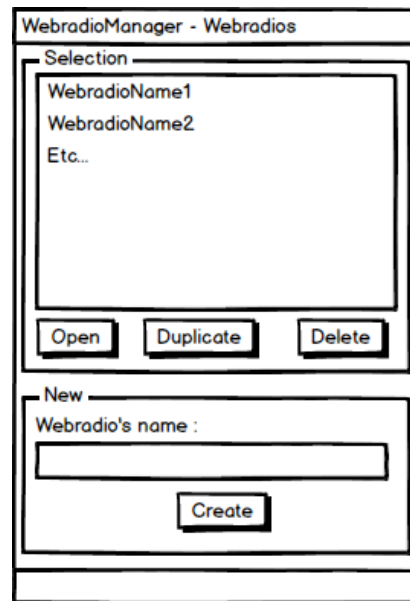


Figure 3 - Interface gestion des webradios SelectionView

La fenêtre de gestion des webradios s'affiche au démarrage de l'application pour sélectionner ou créer une webradio. Lorsque l'utilisateur clique sur « Open », la fenêtre principale s'ouvre avec les données de la webradio sélectionnée. Il peut aussi utiliser un double clic pour ouvrir une webradio.

Le bouton « Duplicate » crée une copie exacte de la webradio sélectionnée avec un suffixe « copy » à son nom. Enfin, le bouton « Delete » supprime la webradio sélectionnée.

Le fenêtre n'est pas redimensionnable et ne peut être maximisée.

Concernant la création, la partie inférieure de la fenêtre propose la création d'une nouvelle webradio. L'utilisateur lui donne un nom puis clique sur le bouton « Create ». Attention : Les webradios doivent avoir des noms différents. Un message d'erreur notifie l'utilisateur si une webradio a déjà le même nom.

L'utilisateur peut gérer autant de webradio en même tant qu'il le souhaite, en effet, à tous moments, il peut ouvrir la fenêtre de gestion des webradios et en sélectionner une autre. Cela ouvrira une nouvelle fenêtre principale, sans pour autant fermer les autres déjà ouvertes, avec les informations de la webradio fraîchement sélectionnée.

Chaque webradio possède ses propres listes de lecture, sa propre grille horaire ainsi que ses propre transcoders. Un transcoder est un outils qui permet d'envoyer un flux musical à un serveur de diffusion. Pour plus d'informations, rendez-vous au chapitre concernant la [diffusion](#).

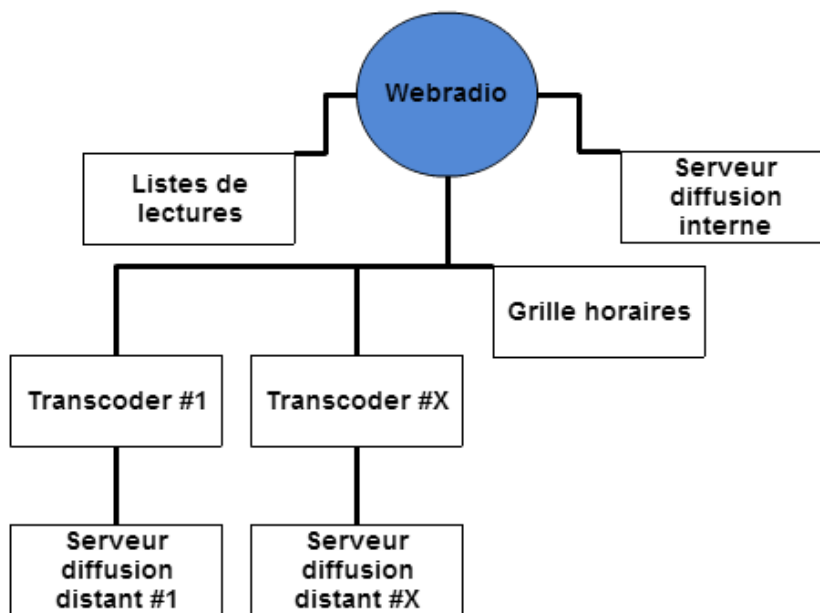
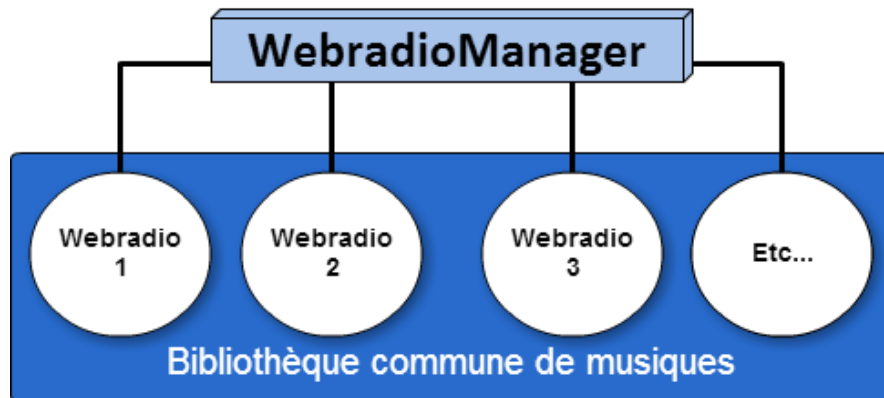


Figure 4 - Schéma webradios

4.3.1 Création d'une webradio

L'utilisateur entre le nom de la future webradio dans le champ correspondant, puis clique sur « create » pour créer la webradio et ouvrir la fenêtre d'administration liée à la nouvelle webradio fraîchement créé. Le nom d'une webradio ne peut pas dépasser 255 caractères.

4.3.2 Sélection d'une webradio

L'utilisateur sélectionne une webradio parmi la liste proposée, puis clique sur « open » pour ouvrir une fenêtre d'administration liée la webradio sélectionnée.

4.4 Serveur de diffusion interne et externe

Il faut bien différencier les serveurs de diffusion qui sont externes et ceux internes. Les externes sont des serveurs hébergés par un provider² (exemple : infomaniak³ pour KTFM). L'application va donc se servir de ces serveurs pour diffuser la webradio.

Les serveurs internes sont des serveurs de diffusion hébergés dans l'application elle-même.

² Fournisseur de services internet

³ <http://www.infomaniak.com/>

4.5 Onglet « Status »

Transcoder	Status
Transcoder1	Off
Transcoder2	On

Figure 5 - Onglet "status"

L'onglet de statut est la page d'accueil de la webradio. L'utilisateur peut y modifier le nom de la webradio via le champ en dessous du nom.

Un tableau affiche l'état des différents transcoders de la webradio actuelle.

4.6 Gestion des musiques/pubs

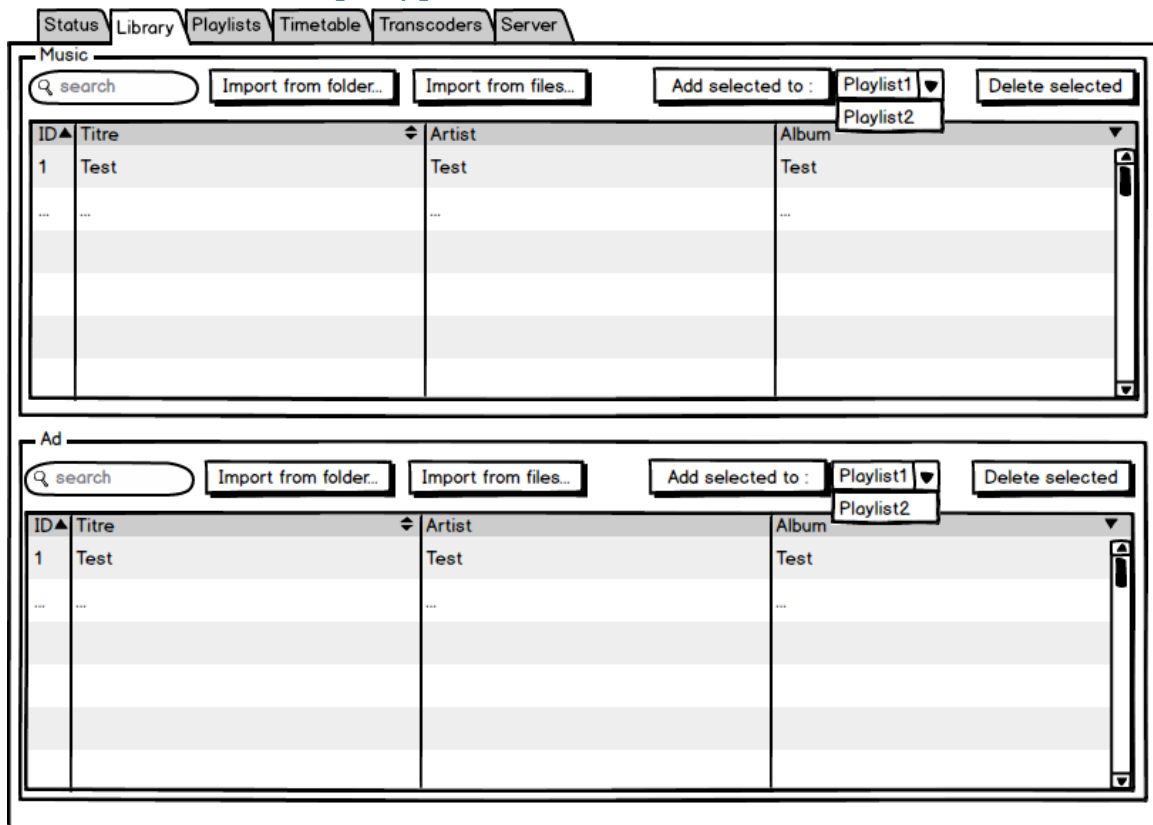


Figure 6 - Bibliothèque de musiques

Les musiques sont indexées pour former une bibliothèque commune dans l'application. Elle est commune car elle est accessible depuis n'importe quelle webradio créé. L'interface est doublée : une partie pour les musiques et une autre pour les pubs. Elles sont identiques.

4.6.1 Affichage

La partie principale de cette interface est l'affichage du contenu de la bibliothèque dans la partie inférieure. Il est affiché sous la forme d'une liste à entrées. Il est possible de sélectionner un ou plusieurs éléments.

4.6.2 Importer et indexer

L'utilisateur peut importer des fichiers musicaux dans sa bibliothèque. Le bouton « Import from folder... » ouvre une boîte de dialogue où l'utilisateur peut sélectionner le dossier à analyser afin d'importer les fichiers musicaux qui y sont présents. C'est la partie dite « d'indexation ». Cette indexation peut être récursive, c'est-à-dire que les sous-dossiers du dossier sélectionné vont être analysés aussi. Une boîte de dialogue demande donc à l'utilisateur s'il veut effectuer une analyse récursive.

Le bouton « Import from files... » permet d'importer un ou plusieurs fichiers sélectionnés manuellement. La sélection s'effectue via une boîte de dialogue Windows standard.

A gauche de ces 2 boutons, une barre de recherche permet d'effectuer une recherche dans la bibliothèque de l'application.

Le bouton de droite « Delete selected » va supprimer les éléments sélectionnés dans la liste de morceaux affichés en dessous.

4.6.3 Ajout à une playlist

Pour ajouter des morceaux à une playlist, l'utilisateur peut en sélectionner autant qu'il le souhaite dans la liste d'affichage puis sélectionner une playlist via le menu déroulant situé à droite du bouton « Add selected to ». Ce dernier permet donc de confirmer l'ajout à une playlist.

4.6.4 Supprimer

L'utilisateur a la possibilité de supprimer une ou plusieurs musique/pub d'un seul coup grâce à la sélection multiple.

4.7 Gestion des listes de lecture

La gestion des différentes listes de lecture se fait dans l'onglet « playlists ».

Figure 7 - Onglet "playlists"

4.7.1 Listes de lecture musicales

Une liste de lecture musicale contient exclusivement des musiques. Son type est « music ».

4.7.2 Listes de lecture publicitaires

Une liste de lecture publicitaire contient des pubs, des annonces ou des jingles. Son type est « ad ».

4.7.3 Création

La création d'une playlist se fait via le cadre situé en haut à gauche. Il lui faut un nom et un type. La playlist fraîchement créé est ensuite ajouté à l'une des 2 listes situées à gauche en fonction de son type.

4.7.4 Génération automatique

L'utilisateur a la possibilité de générer automatiquement une playlist en définissant une durée (minimum 1 minute et maximum 500 minutes), un nom, un genre musical et un type. Si le type « Ad » est choisi, le menu déroulant « Gender » est désactivé (une pub ne peut pas avoir de genre musical).

La liste de genre affiche les genres disponibles dans la bibliothèque de l'application. Le logiciel va prendre différentes musiques ou publicité afin de remplir le temps voulu. L'algorithme fait en sorte de s'approcher le plus possible de la durée demandée mais plus celle-ci est longue, plus cela sera possible d'être précis.

4.7.5 Affichage du contenu d'une playlist

Quand une playlist est sélectionnée (musique ou pub), son contenu est affiché dans la partie centrale de la vue. Les informations des morceaux sont affichées de même manière que pour l'onglet

« [Library](#) ». Au-dessus de cette partie, des informations concernant la playlist sont affichées telle que le temps total de lecture, le nombre de morceaux présents etc.

4.7.6 Retirer d'une playlist

L'utilisateur peut sélectionner un ou plusieurs morceaux dans la partie affichant le contenu de la playlist afin de les retirer de cette dernière. Une fois la sélection faite, le bouton « Remove selected » supprime le/les morceau(x) et le nouveau contenu de la playlist est affiché.

4.7.7 Recherche

L'utilisateur a la possibilité de recherche parmi les morceaux d'une playlist sélectionnée via le champ de recherche au-dessus de l'affichage du contenu de la playlist.

4.8 Gestion des horaires

The screenshot shows a web application interface for managing schedules. The top navigation bar includes tabs for 'Status', 'Library', 'Playlists', 'Timetable' (which is the active tab), 'Transcoders', and 'Server'. Below the navigation bar, there is a 'Create event' section. This section contains several input fields and checkboxes: 'Name' (text input), 'Playlist' (dropdown menu showing 'Playlist1'), 'Start time' (text input), 'Duration' (text input), and checkboxes for days of the week (MO, TU, WE, TH, FR, SA, SU). There are also checkboxes for 'Shuffle' and 'Priority' (set to 3). A 'Create' button is located to the right of these fields. Below the 'Create event' section is a 'Timetable' section. It features a 'Delete selection' button. The main content area of the timetable is currently empty, displaying a large yellow box with the text 'Composant calendrier semaine' (Calendar component week).

Figure 8 - Onglet "timetable"

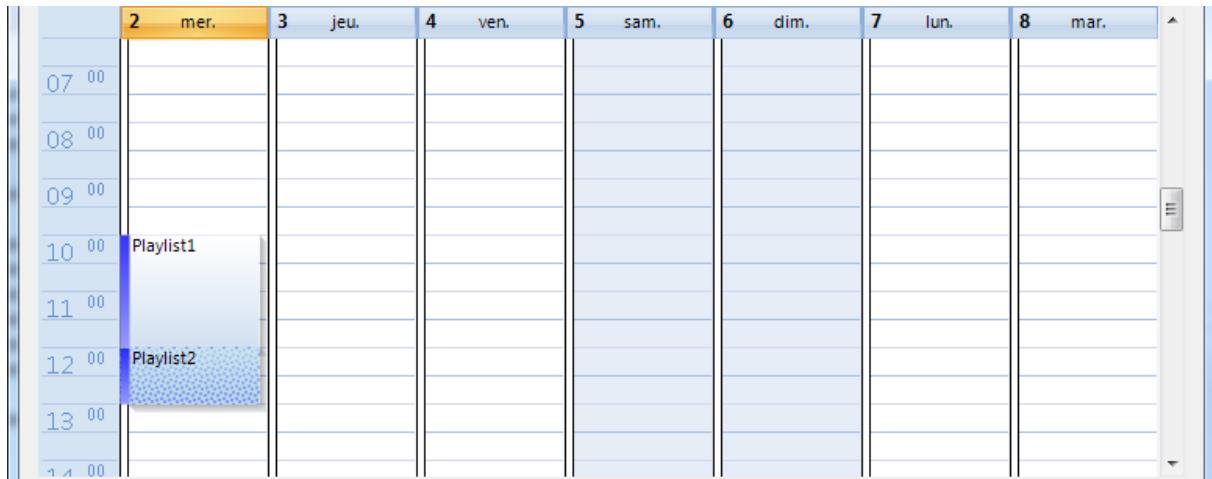


Figure 9 - Calendrier

4.8.1 Événement

Un événement est un élément du calendrier. Il est composé des informations suivantes :

- Un nom
- Une playlist (nom de la playlist qui sera jouée à l'événement)
- Un ou plusieurs jours où la playlist sera jouée
- Une heure de début (identique pour chaque jour sélectionné)
- Une durée de lecture (la playlist est jouée en boucle pendant le temps défini)
- Une option pour jouer la playlist en mode aléatoire ou non
- Une priorité (dans le cas où plusieurs événements se superposent, celui avec la plus grande priorité sera joué). De 0 à 100.

4.8.2 Événement périodique

Un événement périodique se produit à intervalles réguliers.

TODO

4.8.3 Remplissage manuel

L'utilisateur peut utiliser le formulaire de création d'événement en le remplissant à la main ou alors il a la possibilité de sélectionner sur le calendrier, la plage horaire où il désire créer un événement.

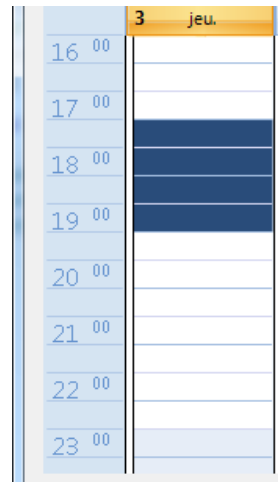


Figure 10 - Sélection multiple calendrier

La partie bleu foncée est la sélection faite par l'utilisateur. Dans ce cas, l'heure de début sera 17h30 et sa durée sera 2h car la sélection finie à 19h30. Ces informations sont directement retranscrites dans le formulaire. Evidemment, le reste des informations demandées sera à remplir à la main par l'utilisateur.

4.8.4 Remplissage automatique

TODO

4.8.5 Suppression d'un événement

Pour supprimer un événement, il faut le sélectionner dans le calendrier et cliquer sur « Delete sélection ». Il est possible de faire une sélection multiple afin de supprimer plusieurs événements en même temps ([sélection multiple calendrier](#)).

4.9 Gestion des transcoders

Figure 11 - Onglet "Transcoders"

Cet onglet permet de gérer les différents transcoders d'une webradio. Ces derniers servent à diffuser le flux audio vers un serveur de diffusion. Ils utilisent les playlists et le calendrier configuré par l'utilisateur.

4.9.1 Création

La partie de gauche offre la possibilité de créer un nouveau transcoder avec ses différents réglages. Pour des mesures de simplicités, le minimum requis est demandé à l'utilisateur. Les informations suivantes sont demandées :

- Un nom (différencier les différents transcoders par la suite)
- Le type de flux (mp3 ou ACC+)
- Le bitrate⁴ du flux
- Le sample rate⁵
- Le nom du flux
- L'URL du flux (par exemple : site web de la webradio)
- Adresse IP du serveur
- Port du serveur
- Mot de passe du serveur

⁴ Débit binaire : une mesure de la quantité de données numériques transmises par unité de temps.

⁵ Taux d'échantillonnage

4.9.2 Affichage

Dans la partie droite, la liste des transcoders de la webradio est affichée. L'utilisateur peut cliquer et sélectionner un des transcoder afin de la supprimer à l'aide du bouton « Delete ». Les informations du transcoder sélectionné sont affichées à droite de la liste, dans les champs prévus à cet effet. Le statut (on ou off) est affiché en dessous de ces dernières. Tout en bas, le log du transcoder est affiché et peut être effacé à l'aide du bouton « Clear ».

4.9.3 Modification

Les informations affichées dans les champs à droite peuvent être modifiées puis enregistrées (bouton « Update ») afin de mettre à jour les paramètres du transcoder sélectionné.

4.9.4 Historique de diffusion

L'historique de diffusion est enregistré (quelle musique est passé à quelle heure) et l'utilisateur peut l'afficher avec le bouton « show » sous la forme d'un fichier texte qui s'ouvrira dans le programme par défaut de l'ordinateur. Il est aussi possible de vider l'historique avec le bouton « clear ».

4.10 Gestion du serveur

The screenshot shows the 'Server' tab of a web application. At the top, there are tabs for 'Status', 'Library', 'Playlists', 'Timetable', 'Transcoders', and 'Server'. The 'Server' tab is active. Below the tabs, there are three main sections: 'Controls', 'Configuration', and 'Log'. The 'Controls' section contains a 'Status' label showing 'Offline' in red, and four buttons: 'Start', 'Stop', 'Show web interface', and 'Show web administration'. The 'Configuration' section contains four input fields: 'Port', 'Password', 'Admin password', and 'Max listener' (with a spinner set to 32), and a 'Save' button. The 'Log' section contains a 'Clear' button and a large text area labeled 'Logs...' with a vertical scrollbar on the right.

Figure 12 - Onglet "server"

L'onglet serveur propose un serveur de diffusion interne (local) pour la webradio. Bien entendu, l'utilisateur doit créer un transcoder qui s'y connectera afin de diffusion le flux audio.

4.10.1 Contrôles

La partie « controls » permet de démarrer ou arrêter le serveur de la webradio via les boutons « start » et « stop ». A leur droite, un bouton permet de lancer l'interface web et un autre l'administration web via le navigateur par défaut de l'utilisateur.

4.10.2 Configuration

Différents éléments sont configurables pour le serveur :

- Port : le port de connexion au serveur pour le transcoder (Défaut : 8000)
- Password : Le mot de passe de connexion au serveur pour le transcoder
- Admin password : Le mot de passe pour l'accès à l'administration web du serveur
- Max listener : Le nombre maximum de connexion au serveur (connexion client/auditeur). Par défaut : 1. Maximum : 2000.

4.10.3 Log

Le log du serveur est affiché dans la partie inférieure. Il affiche le journal d'événement du serveur.

4.10.4 Interface web

Une interface web est fournie avec le serveur. Elle permet de voir différentes informations sur le serveur et le stream. Il est aussi possible de télécharger le fichier permettant d'écouter le flux directement sur un lecteur multimédia.

The screenshot shows the SHOUTcast Stream Status web interface. At the top, the title 'SHOUTcast Stream Status' is displayed in large orange letters. To the right, the version 'SHOUTcast Server v2.2.1.109/win64' is shown. Below the title is a navigation bar with links: 'Status', 'Song History', 'Listen' (with a speaker icon), 'Stream URL', 'Admin Login', and 'Server Login'. The main content area is titled 'Current Stream Information' and displays the following details:

Server Status:	Server is currently up and private
Stream Status:	Stream is up at 56 kbps with 0 of 32 listeners
Stream Name:	My Test Server
Content Type:	audio/aacp
Stream Genre:	Misc
Stream URL:	http://www.shoutcast.com

Un historique des morceaux joué par le serveur est aussi disponible.

4.10.5 Administration web


Via l'interface web décrite précédemment, il est possible d'accéder à une section d'administration. Pour se faire, il faut cliquer sur le lien « Admin login » puis entrer le nom d'utilisateur « admin » et le mot de passe configuré pour le serveur.

SHOUTcast Listeners and Status

Uptime: 14 minutes 29 seconds SHOUTcast Server v2.2.1.109/win64

[Listeners](#) | [Log \(Tailing | Save\)](#) | [Song History](#) | [Ban List](#) | [Reserved List](#) | [Admin Logout](#) | [Server Login](#)

Current Stream Information

Stream Details	Server Status:	Server is currently up and private 
Log file: sc_serv.log	Stream Status:	Stream is up at 56 kbps with 0 of 32 listeners
Configuration file: examples/sc_serv_basic.conf	Stream Name:	My Test Server
Intro file is empty	Content Type:	audio/aacp
Backup file is empty	Stream Genre:	Misc
Idle timeouts are 30s	Stream URL:	http://www.shoutcast.com
Source connection type: v2	Stream Source:	127.0.0.1 [kick]
	Stream Uptime:	22 seconds
Stream artwork not available Playing artwork not available		

Pour l'administrateur, il est possible de :

- Voir la liste des « listeners » (clients qui écoutent la webradio depuis ce serveur)
- Voir le log du serveur
- Gérer une liste d'adresses IP bannies
- Gérer une liste d'adresses IP qui disposent d'un accès réservé (si une adresse réservée désire écouter le webradio mais que le serveur est plein, un client sera éjecté du serveur pour laisser une place au client disposant d'une adresse réservée)

Tous ces services sont fournis par l'interface d'administration web.

5 Analyse organique

5.1 Environnement

- Langage : C#/.NET
- IDE : Visual Studio 2013
- OS : Windows 7 64 bits

5.2 Diagramme de classes

5.2.1 Diagramme

TODO : PAS FINI

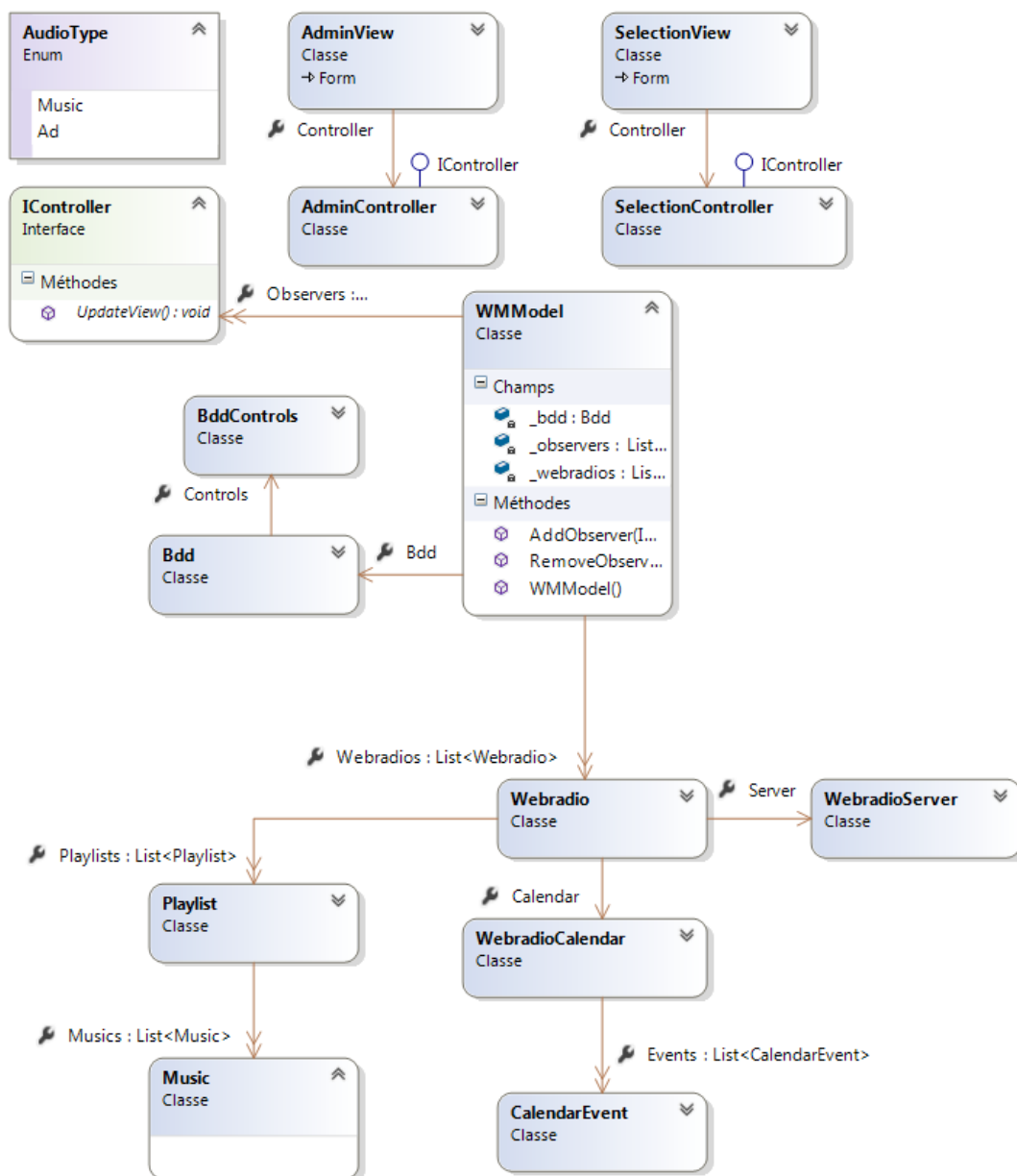


Figure 13 - Diagramme de classes

5.2.2 Observateurs/Sujet

TODO : explication du principe

5.2.3 AudioType et StreamType

TODO : expliquer que ces enum ont des valeur prise des id de la base de données

5.2.4 Classes abstraites

TODO : explication du principe

TODO : explication de l'instanciation

5.2.5 Stockage des webradios

TODO : explication des webradio dans le model sous form de dictionnary avec id

5.3 Base de données

La base de données est présente pour sauvegarder les diverses informations de l'application. La plupart des paramètres sont stockées dans des fichiers texte sous forme de configuration utilisable par les différents transcoders et servers.

5.3.1 SQLite



Figure 14 - Logo SQLite

SQLite est une bibliothèque écrite en C qui propose un moteur de base de données relationnelle accessible par le langage SQL. SQLite implémente en grande partie le standard SQL-92 et des propriétés ACID.

Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

Source : Wikipédia (<http://fr.wikipedia.org/wiki/SQLite>)

Pour mon projet, j'utilise le logiciel SQLite Studio pour éditer ma base de données et y entrer des données de test : <http://sqlitestudio.pl/>

5.3.2 Utilisation

Une bibliothèque sous forme d'une DLL⁶ est disponible pour .NET (C#) ici : <https://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki>

Une classe fourni par le site web suivant : <http://www.dreamincode.net/forums/topic/157830-using-sqlite-with-c%23/#/> permet l'interaction avec un fichier SQLite. Cette classe prendra la place de BddControls dans le diagramme de classe de l'application.

⁶ Dynamic Link Library

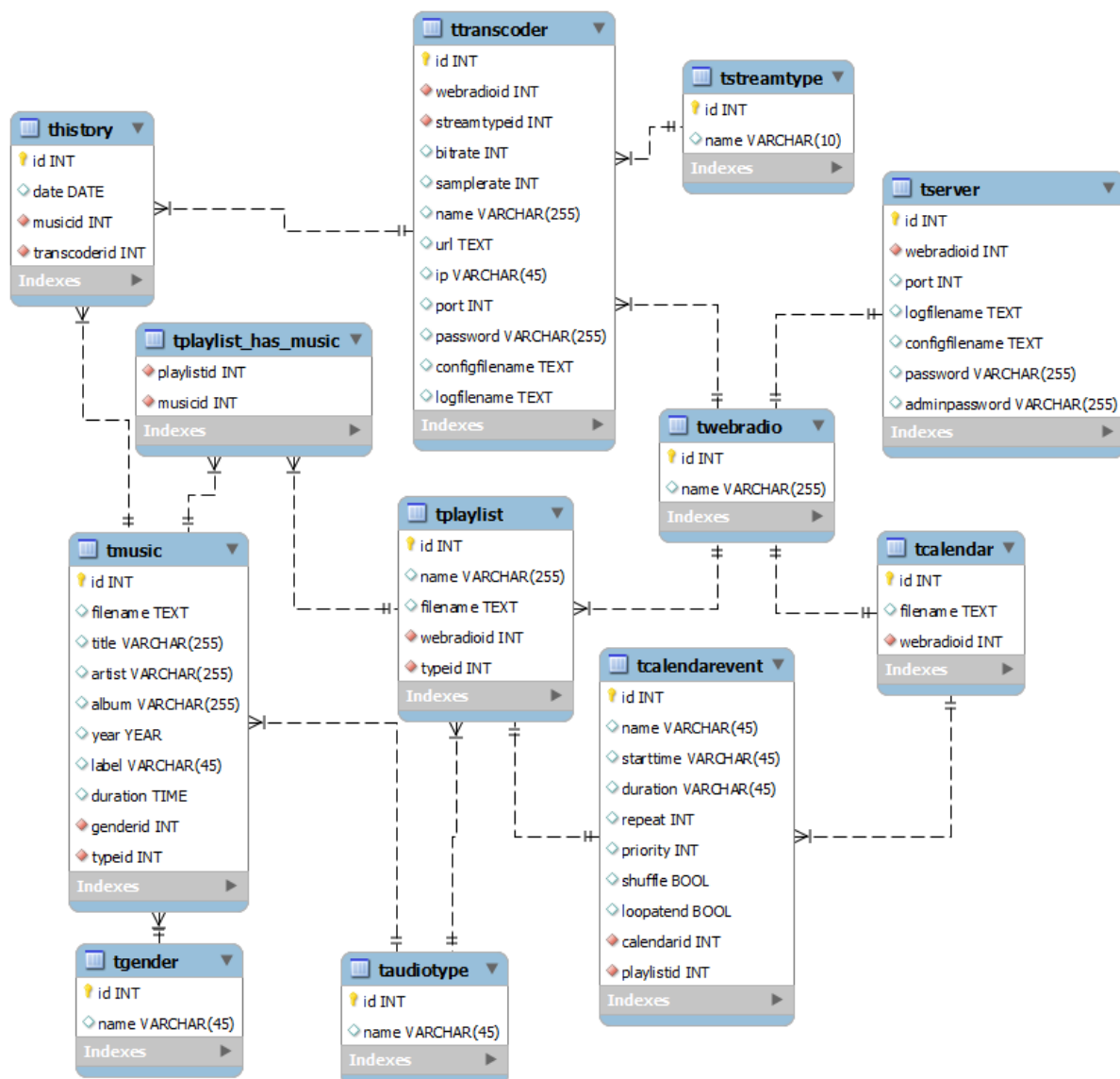
La classe Bdd utilise BddControls afin d'effectuer des requêtes sur la base de données. Bdd propose des méthodes simples comme par exemple « AddWebradio ». Ainsi la partie traitement des données se fait dans Bdd et la partie exécution dans BddControls.

La base de données sert principalement à la sauvegarde des données qui sont récupérées dans le modèle à chaque démarrage de l'application.

5.3.3 Suppression en cascade

SQLite permet la suppression en cascade. C'est-à-dire, lorsqu'un enregistrement est supprimé, toutes les références à ce dernier via des clés étrangères sont supprimées automatiquement.

5.3.4 Schéma



5.3.5 Twebradio

Nom	Type	Description
Id	Int	Identifiant unique d'une webradio
Name	Varchar(255)	Nom de la webradio

5.3.6 Tsever

Nom	Type	Description
Id	Int	Identifiant unique d'un serveur
Webradioid	Int	Clé étrangère (radio possédant ce serveur)
Port	Int	Numéro du port d'écoute du serveur0
Logfilename	Text	Chemin vers le fichier de log du serveur
Configfilename	Text	Chemin vers le fichier de configuration du serveur
Password	Varchar(255)	Mot de passe de connexion au serveur (pour une source)
Adminpassword	Varchar(255)	Mot de passe de l'administration web du serveur
Maxlistener	Int	Le nombre maximal d'auditeur sur le serveur

5.3.7 Tcalendar

Nom	Type	Description
Id	Int	Identifiant unique d'un calendrier
Filename	Text	Chemin vers le fichier XML du calendrier
Webradioid	Int	Clé étrangère (radio possédant ce calendrier)

5.3.8 Tcalendarevent

Nom	Type	Description
Id	Int	Identifiant unique d'un événement
Name	Varchar(45)	Nom de l'événement
Starttime	Varchar(45)	Heure du commencement de l'événement
Duration	Varchar(45)	Durée de l'événement
Repeat	Int	Valeur de répétition de l'événement (voir chapitre Grille horaire)
Priority	Int	Priorité de l'événement
Shuffle	Bool	Défini si l'événement lit la playlist de façon aléatoire
Loopatend	Bool	Défini si la playlist recommence une fois que tous les morceaux

		sont écoutés
Calendarid	int	Clé étrangère (Calendrier possédant cet événement)
Playlistid	Int	Clé étrangère (Playlist jouée par cet événement)

5.3.9 Tplaylist

Nom	Type	Description
Id	Int	Identifiant unique d'une playlist
Name	Varchar(255)	Nom de la playlist
Filename	Text	Chemin vers le fichier de la playlist
Webradioid	Int	Clé étrangère (webradio possédant cette playlist)
Typeid	Int	Clé étrangère (le type de la playlist)

5.3.10 Taudiotype

Nom	Type	Description
Id	Int	Identifiant unique d'un type audio
Name	Varchar(45)	Nom du type audio

Cette table est lié à l'enum « AudioType ».

5.3.11 Tmusic

Nom	Type	Description
Id	Int	Identifiant unique d'une musique
Filename	Text	Chemin vers le fichier musical
Title	Varchar(255)	Titre du morceau
Artist	Varchar(255)	Artiste du morceau
Album	Varchar(255)	Album du morceau
Year	Year	Année du morceau
Label	Varchar(45)	Label du morceau
Duration	Time	Durée du morceau
Genderid	Int	Clé étrangère (genre du morceau)
Typeid	Int	Clé étrangère (type du morceau)

5.3.12 Tgender

Nom	Type	Description
Id	Int	Identifiant unique d'un genre musical
Name	Varchar(45)	Nom du genre

5.3.13 Tplaylist_has_music

Nom	Type	Description
-----	------	-------------

Playlistid	Int	Clé étrangère (Playlist concernée)
Musid	Int	Clé étrangère (Musique concernée, qui fait partie de la playlist ayant l'identifiant Playlistid)

5.3.14 Thistory

Nom	Type	Description
Id	Int	Identifiant unique d'un élément d'historique
Date	Date	Date de l'événement dans l'historique
Musid	Int	Clé étrangère (Musique jouée)
Transcoderid	Int	Clé étrangère (Transcoder ayant joué cette musique)

5.3.15 Ttranscoder

Nom	Type	Description
Id	Int	Identifiant unique d'un transcoder
Webradioid	Int	Clé étrangère (Webradio possédant ce transcoder)
Streamtypeid	Int	Clé étrangère (Type du transcoder MP3 ou autre)
Bitrate	Int	Débit binaire du flux (en bits/s)
Samplerate	Int	Taux d'échantillonnage du flux
Name	Varchar(255)	Nom du flux
Url	Text	URL concernant le flux (site web par exemple)
Ip	Varchar(45)	Adresse IP du serveur de diffusion
Port	Int	Port du serveur de diffusion
Password	Varchar(255)	Mot de passe du serveur de diffusion
Configfilename	Text	Chemin vers le fichier de configuration du transcoder
Logfilename	Text	Chemin vers le fichier de log du transcoder

5.4 Schéma de diffusion

5.4.1 Principe de base

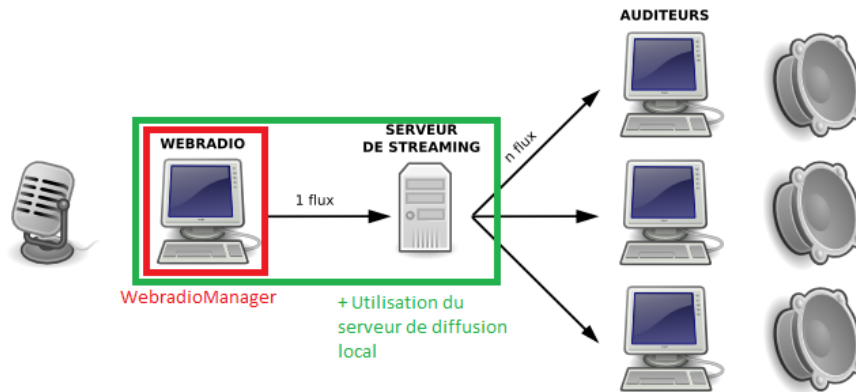


Figure 15 - Principe de base diffusion

Ce type de diffusion est appelé « client-serveur ». C'est le principe de base de streaming audio/vidéo. Dans cette application, la diffusion est possible sur des serveurs distants ou sur un serveur interne (local).

5.4.2 Infomaniak

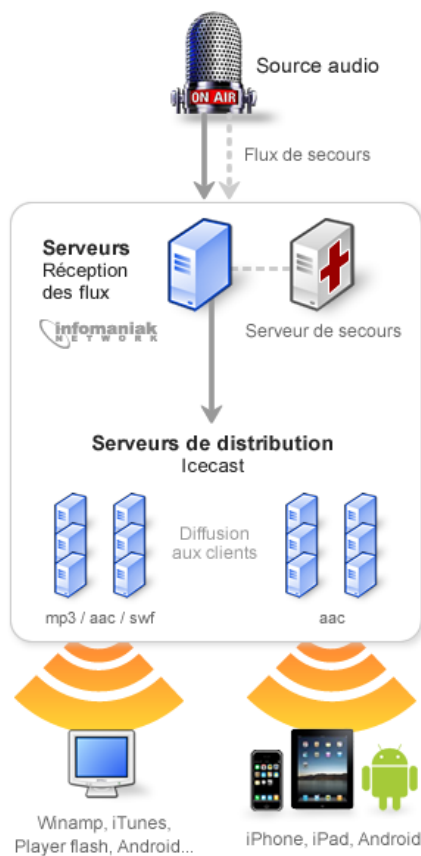


Schéma de fonctionnement
du streaming radio live Infomaniak Network SA

Figure 16 - Schéma de diffusion

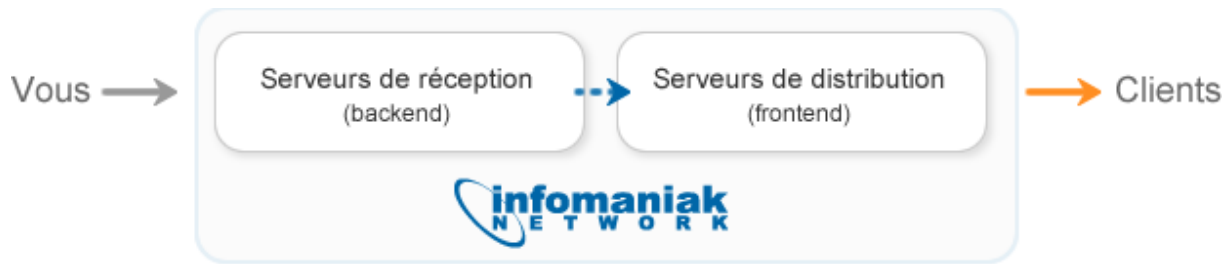


Figure 17 - Schéma de diffusion 2

Ces 2 schémas montrent le fonctionnement de la diffusion des webradio via infomaniak. Le logiciel avec les transcoders sont les « sources audio ». Infomaniak demande seulement un flux audio, peut importer le logiciel utilisé pour le diffuser. C'est ensuite leurs serveurs qui s'occuperont de diffuser le flux aux différents auditeurs.

5.5 ShoutCast



Figure 18 - Logo Shoutcast

5.5.1 Présentation

SHOUTcast est le nom d'un protocole et d'un serveur de diffusion pour webradio ou pour webtv. Il a été créé par la société Nullsoft en même temps que le logiciel client Winamp pour l'écoute. Le protocole s'appuie sur deux protocoles, HTTP et ICY pour supporter les ID tag (« title streaming »).

Source : Wikipédia (<http://fr.wikipedia.org/wiki/SHOUTcast>)

SHOUTcast a été racheté récemment par la société Radionomy (<http://www.pcworld.fr/business/actualites,societe-belge-radionomy-confirme-rachat-winamp-shoutcast,545553,1.htm>). Cela a pour répercussion que les fichiers ne sont plus disponibles sur le site web de Shoutcast. Mais malgré cela, les fichiers de la version 2 sont toujours disponibles sur le forum de Winamp⁷. Ce sont donc ces derniers qui seront utilisés pour le projet.

5.5.2 Pourquoi cet outil ?

J'ai choisi Shoutcast car il propose des outils en ligne de commande ainsi qu'une gestion facilitée via des fichiers XML⁸ ou texte basiques. Cela permet de pouvoir interagir plus facilement avec des applications externes telle que la mienne.

⁷ Lecteur multimédia développé par Nullsoft

⁸ eXtensible Markup Language : Langage de balisage extensible

5.5.3 Serveur

Shoutcast, parmi ses outils, propose un serveur en ligne de commande qui sera utilisé dans ce projet. Il permet de diffuser un flux qu'il reçoit et qui est envoyé par, par exemple, un [transcoder](#). Ce flux est distribué aux clients (auditeurs dans ce cas) qui désire écouter la webradio.

Ce serveur propose aussi une interface web pour visualiser ton statut (données sur le flux actuel, musique en cours etc.) et d'administrer (il faut être authentifié en tant qu'administrateur avec le mot de passe défini dans la configuration du serveur) le serveur. Ces détails sont expliqués dans [l'analyse fonctionnelle](#).

Plus de détails dans la partie consacrée au [serveur interne de diffusion](#).

5.5.4 Transcoder

Tout comme le serveur, le transcoder est un outil fourni par Shoutcast en ligne de commande. Il permet de diffuser un flux sur un serveur de diffusion. Ce flux peut être en MP3 ou AAC+. Il donne aussi la possibilité de créer des playlist et de les agencées dans un calendrier XML. Les détails concernant ce système sont expliqués dans la suite de cette analyse organique.

Il gère de façon indépendante les horaires, les priorités entre les playlists et la lecture des fichiers musicaux. Le programme de ce projet va s'occuper de générer les fichiers nécessaires au transcoder en fonction des paramètres définis par l'utilisateur ainsi que d'afficher ces informations de façon visuelle (exemple : calendrier) afin de faciliter la manipulation et la configuration.

5.5.5 Schéma de fonctionnement résumé

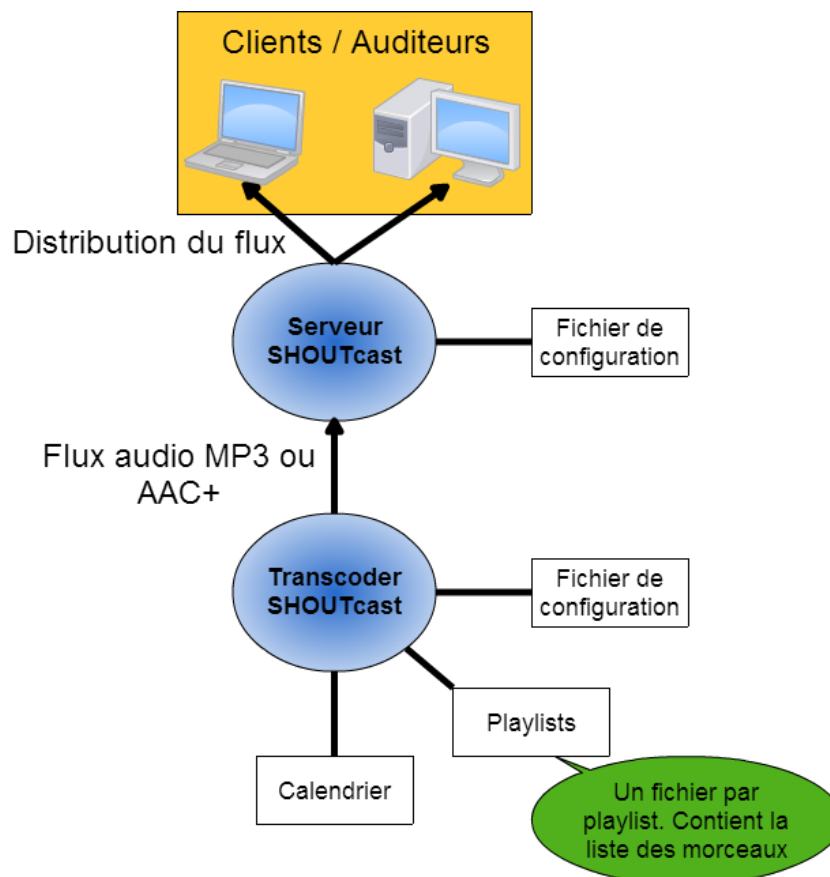


Figure 19 - Schéma shoutcast

5.6 Structures des dossiers/fichiers

5.6.1 Schéma

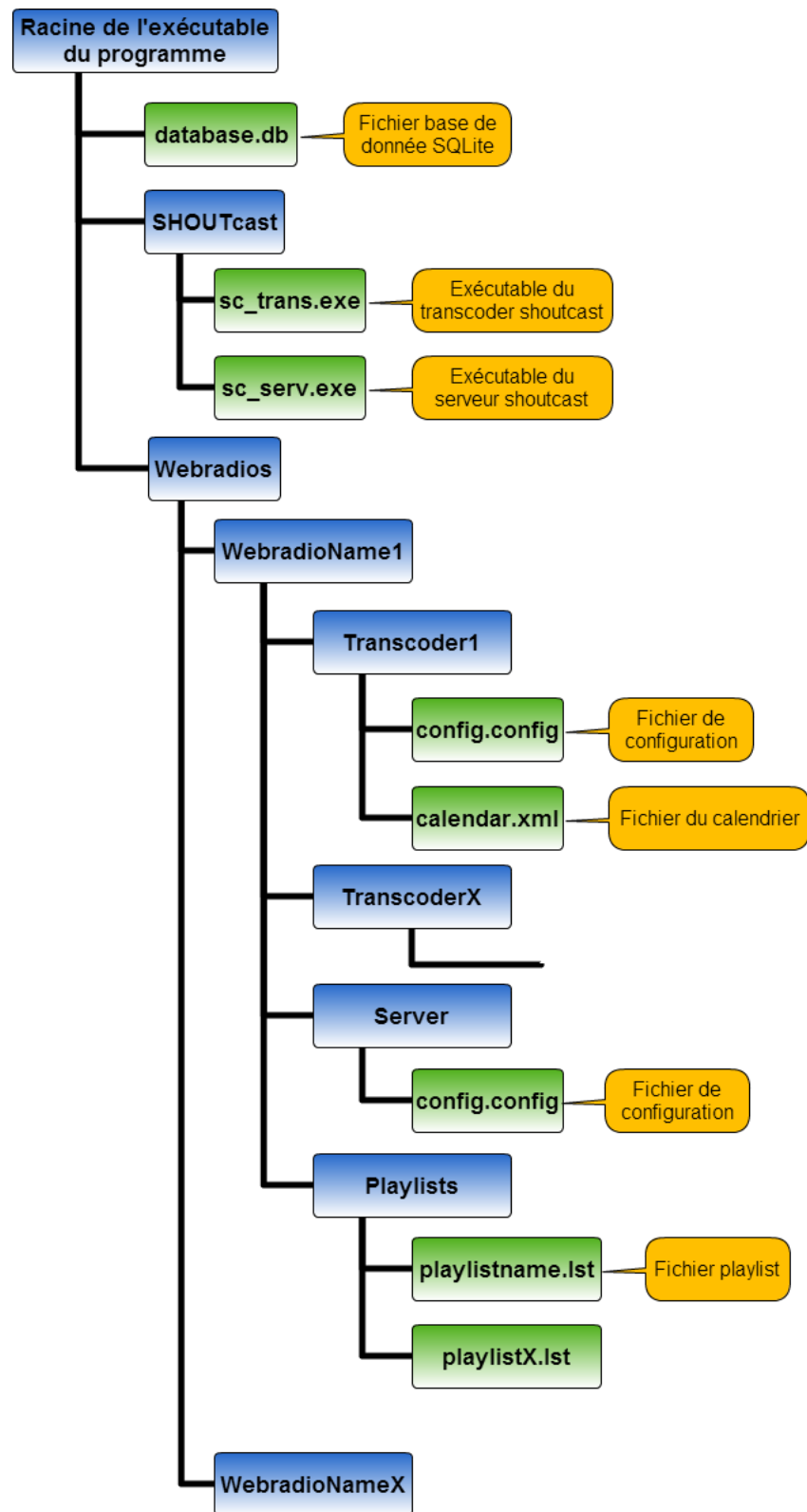


Figure 20 - Schéma structure des fichiers/dossiers

Ce schéma décrit l'organisation des fichiers dans le dossier de l'application. La base de données contient les informations pour les transcodeurs et les serveurs ainsi que le chemin vers les différents fichiers de ces dernières (configuration, calendrier etc.). Cela permet à l'application de savoir où trouver les fichiers lors des traitements.

5.6.2 Exécutables Shoutcast

Attention : Il n'y a pas un exécutable de transcodeur par transcodeur de webradio ni un exécutable de serveur par webradio. Il existe seulement un seul et unique exécutable transcodeur et serveur dans le dossier « shoutcast ». Ces exécutables peuvent être lancés avec le chemin vers un fichier de configuration en paramètre. Par exemple : A chaque fois qu'un transcodeur devra se lancer, une nouvelle instance de l'exécutable `sc_trans.exe` présent dans le dossier « shoutcast » sera lancée dans un nouveau processus et utilisera le fichier de configuration du transcodeur en question. Cela a été décidé car si une mise à jour des exécutables doit être faite, seuls les 2 exécutables du dossier « shoutcast » seront mise à jour.

Voici ce principe illustré :

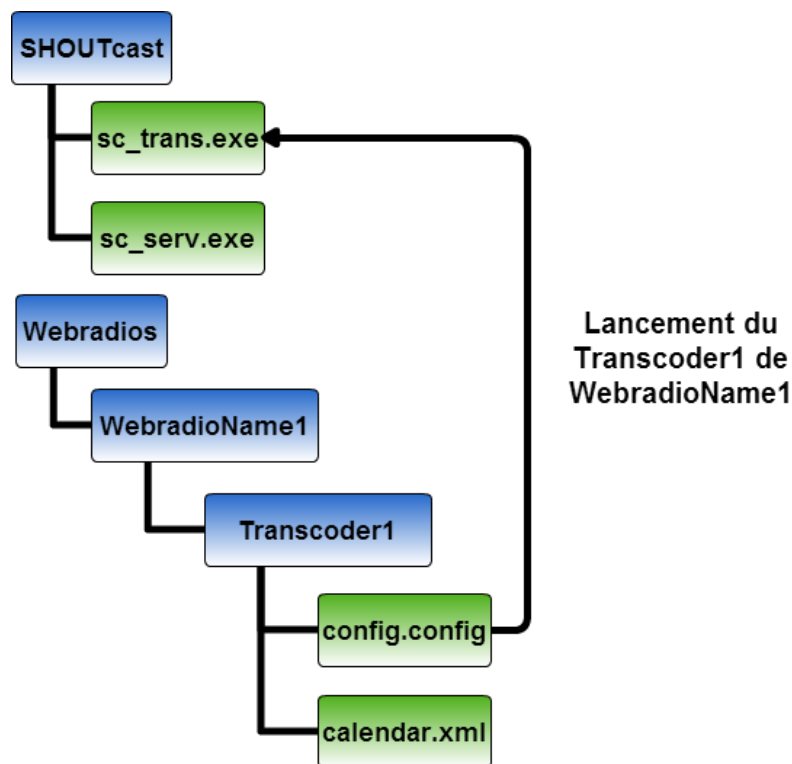


Figure 21 - Exemple lancement transcoder

5.7 Initialisation de l'application

La première fenêtre à se lancer est la SelectionView. C'est elle qui va appeler les différentes méthodes du modèle (via son contrôleur) servant à l'initialisation.

C'est le classe Bdd qui s'occupe du traitement des données pour les passer ensuite au modèle pour que ce dernier remplisse ses champs « webradios » et « library ». Le diagramme de séquence suivant explique le déroulement de façon simplifiée.

IM

Figure 22 - Diagramme de séquence initialisation application

A la fin de se diagramme, SelectionView est affichée à l'utilisateur avec les informations recueillies avec UpdateView(). Cette méthode met à jour la vue avec les informations disponibles dans le modèle la concernant.

Le modèle est donc rempli au démarrage de l'application. Toutes les informations contenues dans la base de données sont récupérées, traitées et ajoutées au modèle. Cela permet d'éviter un nombre de requêtes inutiles vers la base de données et de travailler avec les informations stockées en mémoire.

Les 2 méthodes « LoadWebradios() » et « LoadLibrary() » de la classe Bdd s'occupe du traitement des informations. La première charge toutes les informations de façon hiérarchique (une webradio a un calendrier, qui lui dispose d'événement etc...) pour chaque webradio de la base de données. La 2^{ème} charge les musiques présentes dans la bibliothèque ainsi que les playlist qui leur sont associées.

5.8 Gestion des processus

TODO : explication classe Process C# et la gestion (un processus par classes en aillant besoin, exemple : server)

TODO : tout s'eteins a la fermture

TODO : POUR CHAQUE CHAPITRE, EXPLIQUER LA CLASSE ASSOCIEE

5.9 Webradio

5.9.1 Classes associée



Figure 23 - Classe "Webradio"

TODO : mettre ajour l'image en fin de rappoort + expliquer le ToString overrid

5.9.2 Affichage des webradios disponibles

Les webradios disponibles sont affichés dans la ListBox centrale de la fenêtre SelectionView. Pour la remplir, cela est effectué dans la méthode « UpdateView() » qui va récupérer les webradios du model et remplir la liste « d'items » avec les objets de type Webradio obtenus. C'est la méthode « ToString » de la classe Webradio qui va être appelée pour afficher le nom de la webradio dans la liste.

5.9.3 Création

Une webradio ne peut pas avoir un nom qui dépasse 255 caractères. Pour se faire, la limitation est directement configurée dans la propriété « MaxLength » du TextBox permettant à l'utilisateur de nommer sa webradio.

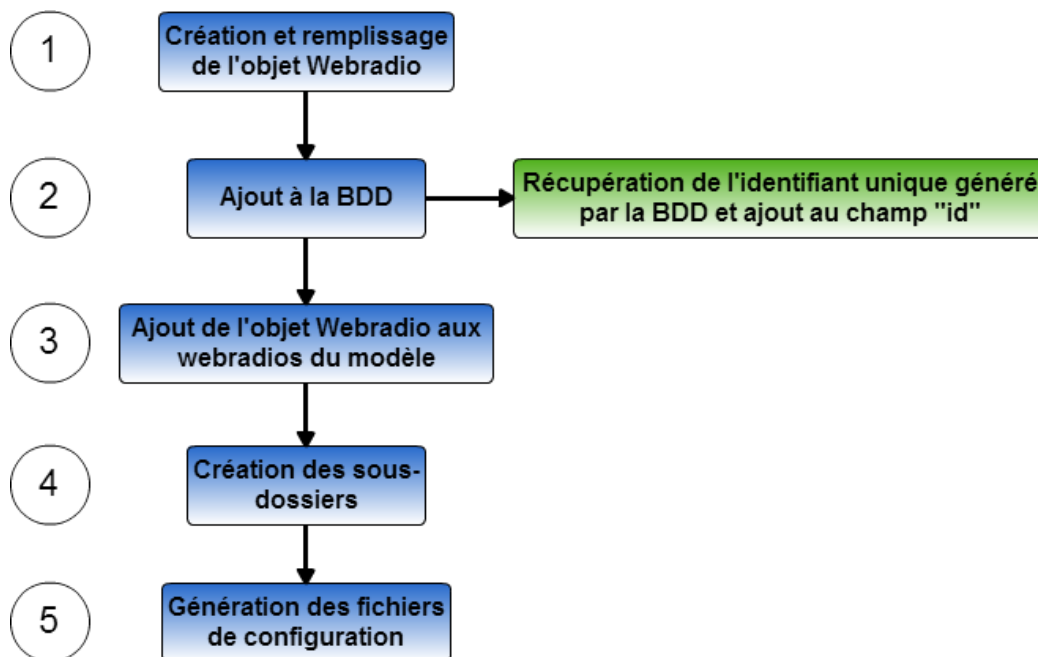


Figure 24 - Schéma création webradio

Comme présenté dans le schéma ci-dessus, la création se divise en 5 étapes, ces dernières s'effectuent dans le modèle et la méthode « CreateWebradio() » :

- **1** : Instanciation d'un nouveau objet de type Webradio avec le nom donné en paramètre à la méthode « CreateWebradio() ».
Instanciation des objets nécessaires à la webradio (WebradioServer, WebradioCalendar et les différentes listes d'objets telle que la propriété Playlists). Seul la propriété « id » de l'objet webradio n'est pas rempli car il s'agit de son identifiant dans la base de données, il sera donc rempli par la suite.
- **2** : L'objet créé est passé à la classe Bdd qui s'occupe de l'ajout de toutes ces informations dans la base de données via sa méthode « AddWebradio() » qui retourne l'identifiant qui a été attribué à cette nouvelle webradio par la base de données. Cet id est récupéré dans la méthode « CreateWebradio() » précédente et ajouté à l'objet webradio.
- **3** : L'objet webradio final est ajouté à la liste d'objets de type « Webradio » du modèle.
- **4** : Création des différents sous-dossiers pour stocker les fichiers de la nouvelle webradio.

- 5 : La méthode « GenerateConfigFiles() » est appelée afin de créer les fichiers de configuration nécessaires à la webradio. Plus d'information sur cette méthode dans [ce chapitre](#).

5.9.4 Chargement

Une webradio est chargée lorsqu'elle est sélectionnée via SelectionView. Une fenêtre de type AdminView est ensuite ouverte avec les informations de la webradio sélectionnée. Ces informations viennent du modèle. Voici le diagramme de séquence pour la création d'une nouvelle instance d'une AdminView :

25

Figure 25 - Diagramme de séquence instantiation AdminView

Après cette initialisation, le nouvel AdminController est ajouté à la liste d'observateurs du modèle.

Comme montré dans la diagramme, la méthode « UpdateView() » de AdminView est appelé afin de charger les informations dans ses différents composants. La vue contient l'id de la webradio qui lui est attribuée. C'est avec cet id qu'elle va aller récupérer la webradio en question dans le modèle via son contrôleur.

En ce qui concerne le remplissage des différentes ComboBox et ListBox, leurs Items ne seront pas de simples chaînes de caractères mais des objets complets. Par exemple, les ComboBox affichant les playlists disponibles comme dans l'onglet « Library », sont remplis avec des objets de type Playlist. Il est important que les classes ajoutées à des composants de ces types implémentent une méthode « ToString() » car c'est celle qui est appelée par le composant lorsqu'il affiche, sous forme de chaîne

de caractère, les éléments de sa liste. Cette méthode « override » celle héritée par la classe parente « Object ». De cette façon, il est facile de personnaliser les informations retournées par la classe quand un composant utilise sa méthode ToString.

Le fait d'utiliser directement des objets dans les ListBox ou ComboBox permet de garder un pointeur sur les objets présents dans le modèle. Cela permet de manipuler un objet et ses modifications seront répercutées partout où il est utilisé.

Le calendrier est un composant spécial et il est aussi mis à jour lors de l'UpdateView. Pour plus d'information sur son fonctionnement, rendez-vous [au chapitre le concernant](#).

5.9.5 Duplication

TODO : depuis model -> dis à la bdd de dupliquer et duplique dans son model

Todo : Cloneable . Un clone() par sous objet de webradio ? Modifier la méthode « AddWebradio » pour parcourir tous les éléments de la webradio donnée ? et pas que nle calendrier et le server

5.9.6 Suppression

La suppression d'une webradio s'effectue via son identifiant. La méthode « DeleteWebradio() » du modèle va en premier temps supprimer le webradio de la base de données, puis de sa liste (dictionnaire) de webradios. La suppression dans la liste se fait via la méthode « Remove » proposée par les listes de type Dictionnaire qui prend l'identifiant de la webradio à supprimer.

5.9.7 Génération des configurations

La classe Webradio dispose d'une méthode « GenerateConfigFiles() » qui appelle la méthode « GenerateConfigFile() » de chacun de ses membres (Server, Playlists etc.) ayant besoin d'un fichier de configuration.

Ces méthodes suppriment le fichier de configuration existant (si il y en a un) et en crée un nouveau avec les informations contenues dans les champs de la classe en question. En fonction de la classe, le type de fichier de configuration sera différent (simple fichier texte ou fichier XML).

La génération de configuration est appelée lors de la sauvegarde ou le changement d'informations depuis l'AdminView mais encore lors de la création d'une nouvelle webradio. Plus d'informations pour la génération de configuration de chaque composant du programme dans la suite de la documentation.

5.10 Bibliothèque

5.10.1 Classes utilisées

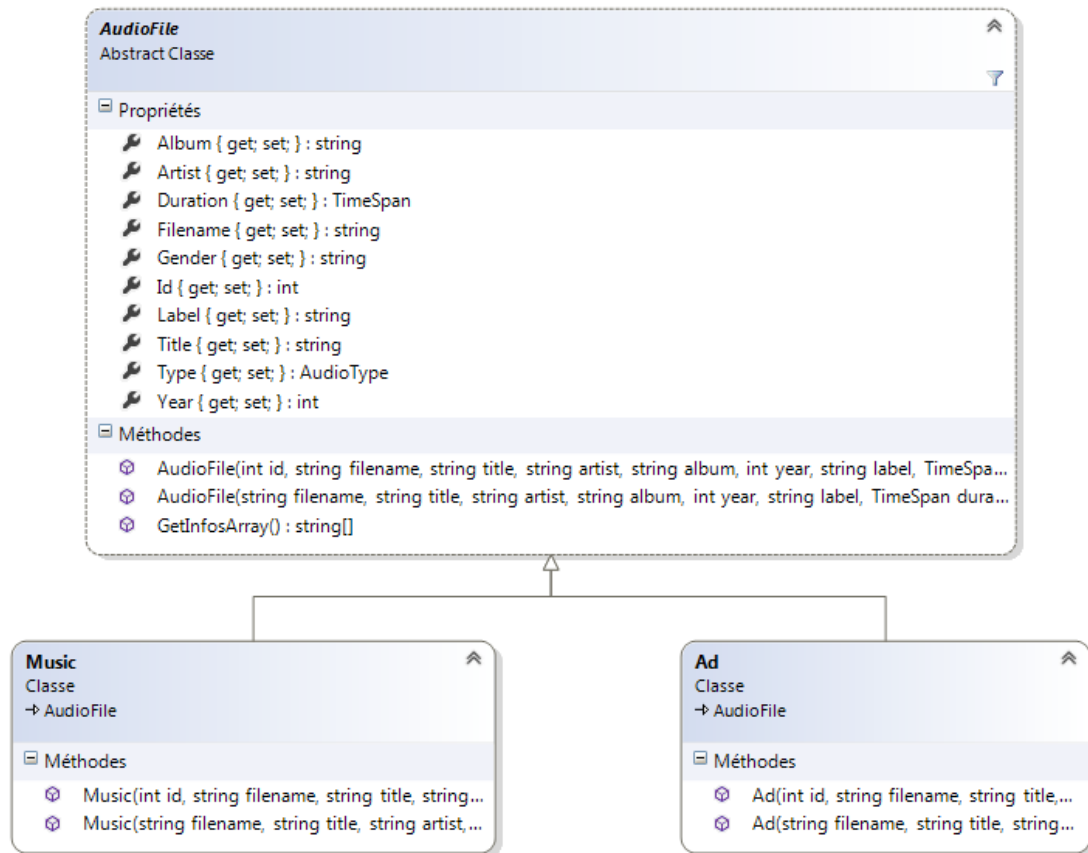


Figure 26 - Classes bibliothèque

5.10.2 MP3

A l'heure actuelle, seuls les fichiers MP3 peuvent être importés. Le raison principale est que le flux de sortie de la webradio sera de toute façon compressé donc il n'y a pas d'intérêt à importer des fichiers de meilleure qualité.

TODO : explication algo de compression

5.10.3 Importation

Depuis l'onglet « library », l'utilisateur choisi si il veut importer un dossier ou des fichiers à la section « music » ou « ad ». Chaque bouton contient une valeur dans sa propriété « tag » qui est soit « Music » ou soit « Ad » car les événements « OnClick » des boutons (regroupé par type d'importation, c'est-à-dire, par dossier ou fichiers) pointe sur une même méthode et cette propriété « tag » permet de différencier si le bouton cliqué est dans la section « music » ou « ad ».

Dans le cas d'un dossier, le traitement va rechercher tous les fichiers MP3 contenus dans le dossier sélectionné. Si l'utilisateur a désiré d'importer de façon récursive, les sous-dossiers du dossier sélectionné seront aussi analysés.

La méthode statique « GetFiles » de la classe Directory permet de récupérer un tableau de string contenant les « filename » (chemin absolu vers les fichiers) des fichiers correspondant au pattern donné en paramètre dans le dossier spécifié. Ce pattern se présente sous la forme : « *.mp3 » pour

recupérer seulement les fichiers dont l'extension est « mp3 ». Une option permet de faire cette recherche de façon récursive. Voici un exemple de code pour des fichiers mp3 et récursivement :

```
Directory.GetFiles(FBD.SelectedPath, "*.mp3", SearchOption.AllDirectories);
```

Pour une recherche non-récursive, l'option « SearchOption » doit être changée en : `SearchOption.TopDirectoryOnly`.

La vue va ensuite passer le tableau de string au modèle via son contrôleur. Le modèle va effectuer le traitement (analyse de tags ID3 et ajout à la base de donnée/modèle) avec sa méthode « ImportFilesToLibrary ». Cette méthode est générique, il suffit de lui envoyer un tableau de filenames pour fonctionner. Cela permet qu'une importation par fichiers ou dossier puisse utiliser la même méthode. Voici le schéma récapitulatif :

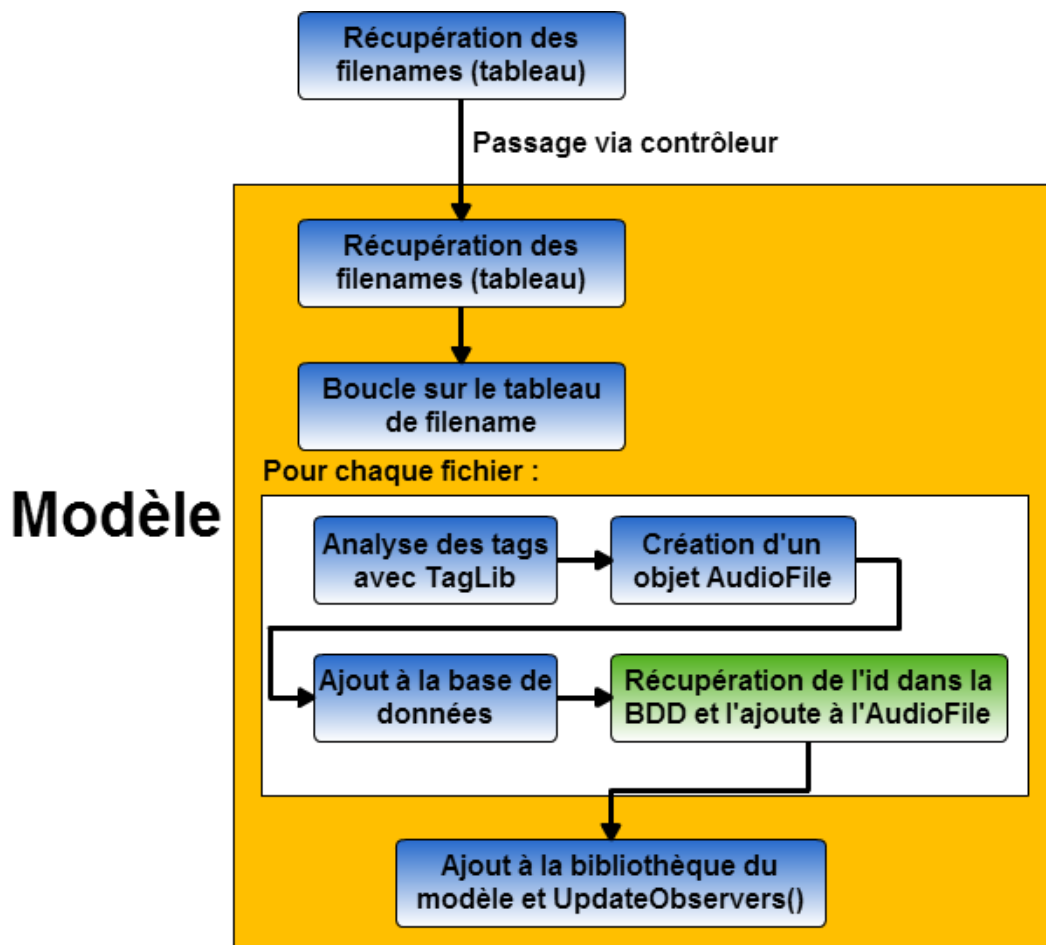


Figure 27 - Schéma importation fichier

La partie « analyse de tag » est expliquée plus précisément [ici](#). Concernant l'ajout à la base de donnée, [ce chapitre](#) décrit le procédé.

5.10.4 Tags ID3

ID3 est le nom des métadonnées pouvant être insérées dans un fichier audio comme MP3. Ces métadonnées permettent d'avoir des informations sur le contenu du fichier comme le titre, le nom de l'interprète, les commentaires, ou encore la date de sortie.

Source : Wikipédia

5.10.5 Analyse des tags

La récupération des tags ID3 est effectuée à l'aide de la bibliothèque « TagLib-Sharp » :

<https://github.com/mono/taglib-sharp>

Le fonctionnement de cette bibliothèque est simple. Pour analyser et récupérer les tags ID3 d'un fichier, elle fournit une classe File (à ne pas confondre avec la classe File fourni par .NET) qui propose une méthode statique « Create » qui prend un filename en paramètre. Cette dernière retourne donc un objet instancié de type « TagLib.File » qui contient toutes les informations sur le fichier donné.

2 types d'informations importantes se distinguent et qui seront utilisées :

- La propriété « Tag » qui contient des sous-propriétés comme « Title », « Year » etc. Elles sont les tags à proprement parlé.
- La propriété « Properties » qui contient des sous-propriétés comme « Duration », « BitRate » etc. Ce sont les informations concernant le fichier en lui-même.

TODO : la librairie lit les tag MusicBrainz

5.10.6 Indexation

Indexation est la partie qui consiste à enregistrer les informations des musiques de la bibliothèque dans la base de données. Pour se faire, quelques règles sont établies :

- Il ne peut y avoir qu'une occurrence par fichier. C'est-à-dire, pas de doublon. Plusieurs musiques peuvent avoir les mêmes informations dans les tags mais c'est le nom de fichier qui fait foi.
- Chaque genre musical à son enregistrement dans la table « tgender ». Si lors de l'ajout d'une musique/pub à la base de données, son genre n'est pas encore dans cette table, il est ajouté à cette table et son identifiant est récupéré. Dans l'autre cas, si le genre est déjà présent dans la table, son identifiant est juste trouvé dans la table.

Voici ces 2 règles représentées sous forme de code :

```
if(this.AudioFileExist(file.Filename))
    return ERROR;

int genderId = this.GetGenderId(file.Gender);
//If return error, gender doesn't exist in DB, so add it
if (genderId == ERROR)
    //Get the new id
    genderId = AddGender(file.Gender);
```

Ensuite, l'ajout s'effectue avec un nouvel enregistrement dans la table « tmusic » puis l'identifiant de ce nouvel enregistrement est retourné à la fin de la méthode.

5.10.7 Suppression

Pour la suppression, le même principe que l'importation par rapport à la différenciation entre le bouton « Delete selected » de la section « Music » et « Ad » avec la propriété « Tag ».

L'utilisateur peut supprimer plusieurs occurrences d'un seul coup, pour cela il faut parcourir la propriété « SelectedRows » du composant « DataGridView » en question. Voici la boucle :

```
foreach(DataGridViewRow row in ((type ==
AudioType.Music)?dgvMusics.SelectedRows:dgvAds.SelectedRows))
{
    if (!this.Controller.DeleteAudioFile(int.Parse(row.Cells[0].Value.ToString()),
row.Cells[row.Cells.Count - 1].Value.ToString()))
        state = false;
}
```

La présence d'un test ternaire dans le partie « in » du foreach permet de sélectionner le bon composant DataGridView en fonction de la section (Music ou Ad) dans laquelle le bouton de suppression a été pressé.

Pour chaque occurrence, la musique/pub va être supprimée de la base de données ainsi que du modèle et des playlist la concernant. Pour se faire, toutes les playlist de toutes les webradios du programme vont être bouclées et dans la liste de filename de chacune, les occurrences du filename de la musique/pub supprimée seront enlevées.

Ensuite, la suppression s'effectue dans la bibliothèque du modèle puis dans la base de données.

5.10.8 Recherche

La recherche consiste à afficher seulement les lignes aillant au moins une correspondance (n'importe quel champ de la ligne) avec la chaîne de recherche entrée par l'utilisateur. Cette chaîne est en premier temps mise en minuscule afin de ne pas prendre la casse en compte.

Une boucle parcourt les lignes du DataGridView et pour chacune, une autre boucle parcourt les cellules. Ensuite, pour chaque cellule, la valeur de cette dernière est prise, un ToString suivi d'un ToLower y est appliqué (pas de casse) et enfin la méthode « Contains » va retourner un booléen si elle trouve une occurrence dans la valeur de la cellule. Si c'est le cas, cette ligne est valide et pourra être affichée. La propriété « Visible » de la ligne est donc modifiée à true mais dans le cas contraire elle sera false. Et ainsi de suite pour chaque ligne.

```
searchString = (sender as TextBox).Text.ToLower();
foreach(DataGridViewRow row in ((type == AudioType.Music)?dgvMusics.Rows:dgvAds.Rows))
{
    foreach(DataGridViewCell cell in row.Cells)
    {
        if (cell.Value.ToString().ToLower().Contains(searchString))
        {
            valid = true;
            break;
        }
    }
    row.Visible = (valid) ? true : false;
    valid = false;
}
```

5.11 Listes de lecture

Les listes de lecture sont utilisées par le calendrier, qui lui, est utilisé par un transcoder.

5.11.1 Classes utilisées

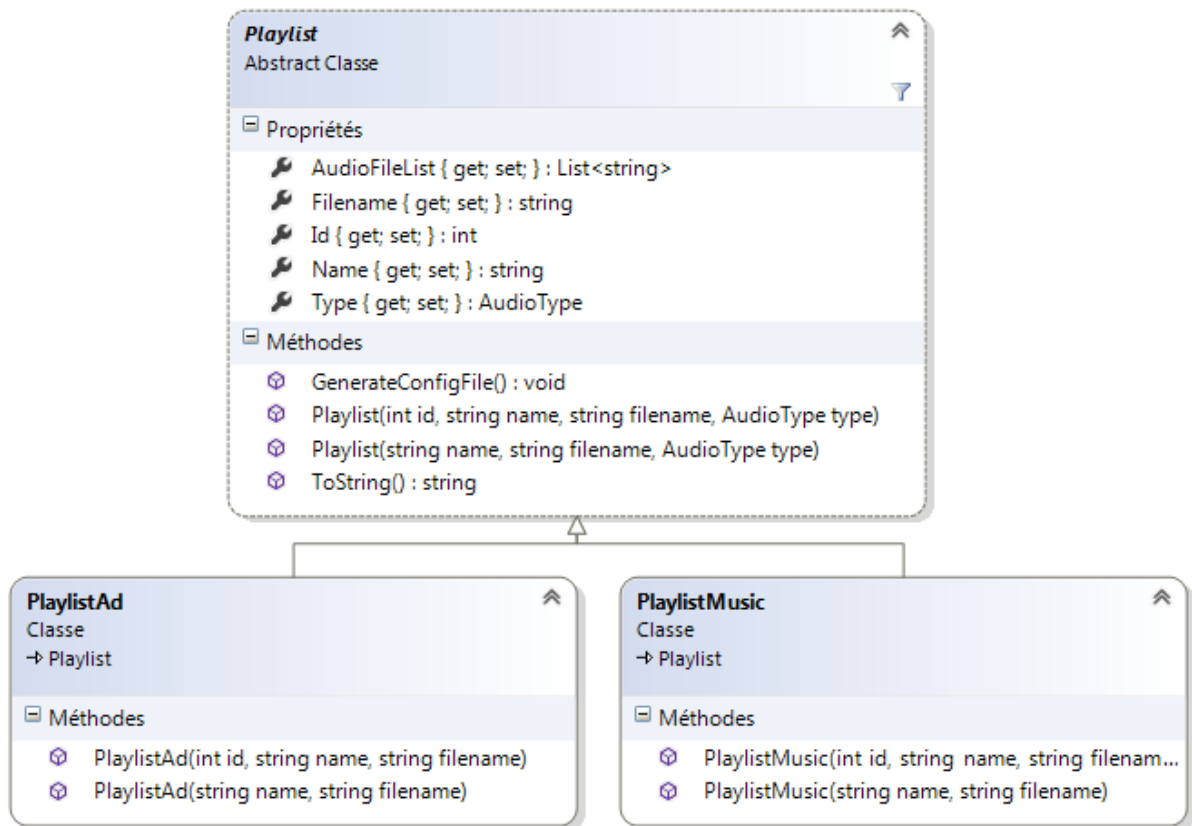


Figure 28 - Classes Playlist

5.11.2 Génération de configuration

Une liste de lecture comprend un fichier de configuration dont la forme est une simple liste des fichiers à utiliser. Elle est stockée dans un fichier texte avec l'extension « .lst ». Voici un exemple de fichier :

```

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Those_Were_The_Days_Original_Instrumental_Mix-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Everlasting_Outro-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz -
Wistfulness\Wasted_Penguinz_and_Toneshifterz-Together_Extended_Version-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Falling_Extended_Version-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Endless_Extended_Version-ToffMusic.mp3
  
```

Chaque ligne du fichier correspond au « filename » (chemin de fichier) vers la musique/pub de la playlist. C'est grâce à cela que le transcoder pourra aller chercher les musiques/pubs pour les jouer et les diffusées.

Concernant la génération, c'est le champ « AudioFileList » de la classe « Playlist » qui contient les filenames des musiques/pubs de la playlist, qui est parcouru afin de générer le fichier.

5.11.3 Création d'une playlist

Pour la création d'une nouvelle playlist, une règle importante a été établie :

- Le nom d'une playlist doit être unique (au sein de la même webradio et avec le même type (Music ou Ad)).
- Par exemple : Il peut y avoir 2 playlist qui se nomment « Test » au sein de la même webradio si chacune d'entre elles a un type différent.

Concernant la création à proprement parlé, elle se décompose en plusieurs étapes qui se déroulent dans la méthode « CreatePlaylist » du modèle :

1. Création du futur filename du fichier de configuration de la playlist :
`DEFAULT_WEBRADIOS_FOLDER + webradioName + "/" + DEFAULT_PLAYLISTS_FOLDER + name + ".lst";`
2. Instanciation d'une classes PlaylistMusic ou PlaylistAd en fonction du type de playlist créée.
3. Insertion dans la base de données. La méthode d'ajout à la BDD retourne l'identifiant dans la BDD de la nouvelle playlist. Si cet id correspond « ERROR » (constante définie), une erreur est survenue lors de l'ajout. Dans ce cas, la méthode de création retourne directement « false ». Si un identifiant valide a été retourné, il est configuré à l'objet de type playlist instancié précédemment.
4. La méthode « [GenerateConfigFile](#) » est appelé sur l'objet Playlist afin de créer son fichier de configuration (bien que vide pour le moment).
5. L'objet est ajouté au modèle (dans la propriété « Playlists » de la webradio à laquelle la playlist est ajoutée).
6. « UpdateObservers » est appelé afin de mettre à jour toutes les fenêtres concernées.

5.11.4 Génération automatique

Le principe de la génération automatique est de créer une playlist d'une durée donnée en la remplissant (aléatoirement) de musiques/pubs du genre donné. Dans le cas de pubs, le genre n'est pas pris en compte.

Il faut savoir que plus la durée demandée grande, plus il sera possible de s'en rapprocher le plus possible en utilisant les morceaux disponibles dans la bibliothèque.

Concernant l'algorithme, il consiste à parcourir la bibliothèque du modèle de façon aléatoire (avec l'objet de type Random) dans une boucle de type while. La condition de cette dernière est que la durée de la playlist doit rester plus petite que la durée demandée. Le but étant de se rapprocher le plus possible de la durée demandé sans la dépasser.

Il est déterminé que l'algorithme essaye un nombre de fois (consécutives), défini par la constante « MAX_TRY_GENERATE », de remplir la liste de lecture. Si ce nombre de fois est dépassé, il est considéré qu'il n'est plus possible de remplir sans dépasser la limite de durée et la boucle est donc arrêtée.

A chaque tour de boucle, une musique/pub est « piochée » et il est testé si le temps actuel de la playlist + le temps de la musique sélectionnée dépasse la durée demandée. Si c'est le cas, le nombre d'essais s'incrémente de 1 et la boucle refait un tour avec l'instruction « continue ». Si ce n'est pas le

cas, le compteur d'essais est remis à zéro et la boucle continue son traitement sans interruption. Ensuite, un test vérifie si le type est Ad OU si le type est Music ET du genre demandé, c'est en passant ce test que la musique/pub est ajouté à la playlist ainsi que sa durée qui est additionnée à la durée actuelle de la playlist. Une liste d'entier est aussi utilisée pour stocker les identifiants des morceaux ajoutés à la playlist afin de pouvoir les utiliser lors de l'ajout à la base de données.

Voici un schéma récapitulatif :

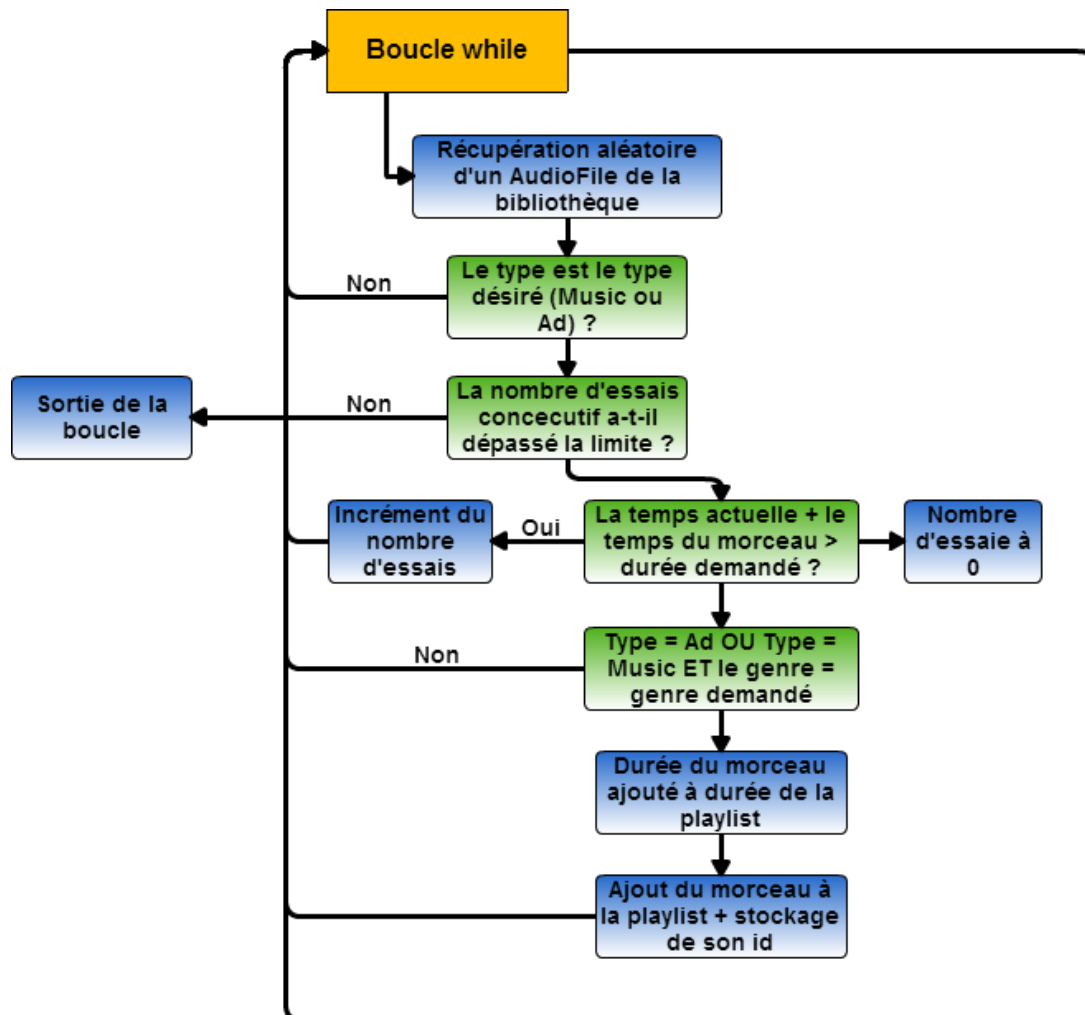


Figure 29 - Algorithme génération playlist

Au final, après la boucle, un test vérifie qu'il y a bien des musiques/pubs présentent dans la playlist. Si n'est pas le cas (l'algorithme n'a pas réussi à combler la durée malgré les essais), la méthode retourne false. Dans le cas contraire, la playlist est ajoutée à la base de données et au modèle. Pour finir, le fichier de configuration de la playlist est généré et la méthode « UpdateObservers » est appelée.

5.11.5 Ajout à une playlist

L'ajout de musique/pub s'effectue depuis l'onglet « Library ».

Les 2 boutons d'ajout (celui dans la section Music ou la section Ad) appel la même méthode d'événement. Pour différencier quel bouton appel la méthode, la valeur « Music » ou « Ad » est

inscrite dans la propriété « Tag » des boutons. De ce fait, il est facile de détecter si quel type de morceau l'utilisateur veut ajouter à quel type de playlist.

En premier temps, la vue va générer une liste de type Dictionary<int,string>. La clé (int) correspond à l'identifiant de la musique sélectionnée et la valeur (string) correspond à son filename. Ce dictionnaire va permettre au modèle d'ajouter les différents éléments dans le modèle et la base de données. Il est généré à l'aide de la propriété « SelectedRows » du composant dataGridView. Cette propriété donne la liste des lignes sélectionnées par l'utilisateur.

Par la suite, le dictionnaire est envoyé au modèle via le contrôleur. Il est parcouru par la méthode « AddToPlaylist » qui prend en paramètre un objet Playlist (correspondant à la playlist sélectionnée) et le dictionnaire. Pour chacun des éléments, il est ajouté à la base de données via la méthode de la classe Bdd « AddToPlaylist » qui prend en paramètre : la clé de l'événement (la valeur int qui correspond à l'id du morceau) et l'id de l'objet Playlist. Si cette méthode retourne une erreur, la boucle est arrêtée et le modèle retourne false. Sinon, la boucle se finit et la playlist génère sa configuration (GenerateConfigFile sur l'objet playlist).

5.11.6 Retirer d'une playlist

La suppression d'éléments d'une playlist se déroule exactement comme l'ajout (dictionnaire avec les morceaux sélectionnés par l'utilisateur qui est parcouru par le modèle) à l'exception que les éléments seront supprimés de la base de données (suppression du lien dans la table « tplaylist_has_music ») et du modèle. A la fin, la nouvelle configuration est générée.

5.11.7 Affichage du contenu

Lorsque l'utilisateur choisit un élément dans un des ListBox (section Music ou Ad), l'événement « SelectedIndexChanged » est appelé. Comme pour les autres éléments, chacun des ListBox a son type dans sa propriété Tag afin de différencier dans la méthode de l'événement en question.

La méthode privée de la vue « GetPlaylistContent » qui prend un objet de type Playlist en paramètre, permet de vider le DataGridView servant d'affichage au contenu des playlists, puis, de le remplir avec le contenu de la playlist voulue. Pour se faire, elle récupère une liste d'objets de type AudioFile en provenance du modèle via son contrôleur. Le modèle va simplement parcourir sa bibliothèque et prendre les AudioFile dont le nom de fichier (filename) correspond à un des noms de fichier présent dans la playlist :

```
List<AudioFile> audioFiles = new List<AudioFile>();
foreach(string filename in playlist.AudioFileList)
{
    foreach(AudioFile af in this.Library)
    {
        if (af.Filename == filename)
            audioFiles.Add(af);
    }
}
return audioFiles;
```

Ensuite, la vue va parcourir cette liste afin de remplir le DataGridView d'affichage.

Les classes Playlist disposent d'une méthode « GetInfosArray » qui retourne un tableau avec les informations de leurs champs. Cela permet de donner un tableau à la méthode d'ajout de ligne au


```
dataGridView : dgvPlaylistContent.Rows.Add(af.GetInfosArray());
```

Par la même occasion, la durée de chaque morceau ajouté est additionné à une variable de type TimeSpan afin de calculer la durée totale de la playlist et de l'affichée.

5.11.8 Suppression d'une playlist

La suppression de playlist va supprimer l'enregistrement correspondant dans la base de données, la variable dans le modèle ainsi que le fichier de configuration enregistré sur le disque. Les vues sont ensuite mise à jour via UpdateObservers.

5.12 Grille horaire

5.12.1 Outil utilisé

Pour l’affichage et la gestion de la timetable⁹, un composant tiers est utilisé :

<http://calendar.codeplex.com/>

Il s’agit de Day View Calendar. Ce composant a été choisi car il propose une vue sous forme de jour et il est entièrement personnalisable (nombre de jours affichés, découpage des heures etc.). Voici un exemple d’utilisation :

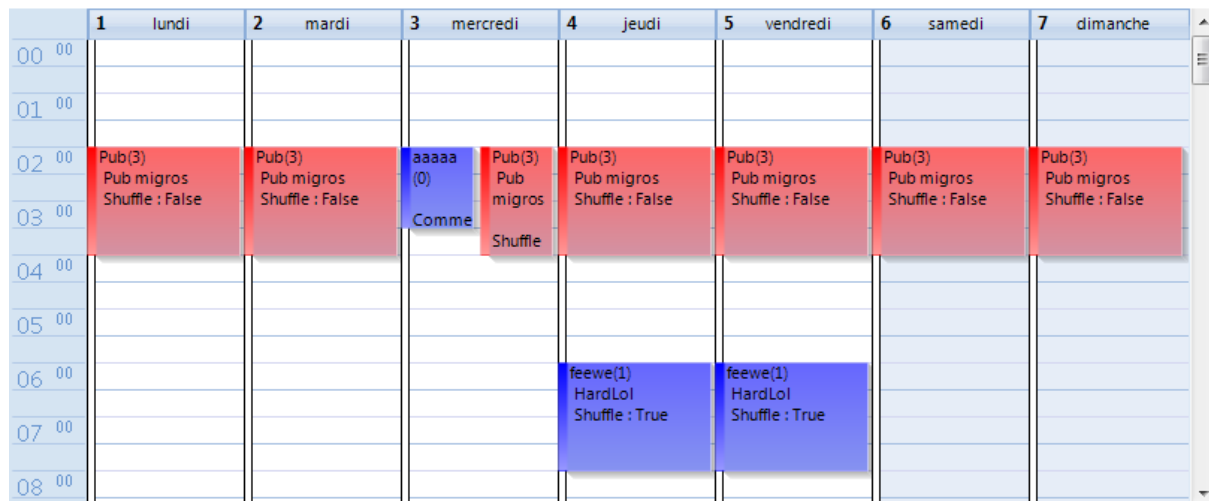


Figure 30 - Day View Calendar

5.12.2 Gestion du calendrier par ShoutCAST

C’est le transcodeur fourni par ShoutCAST qui se charge de prendre en compte le calendrier et jouer les playlist aux bons moments. Ce calendrier est sous forme XML et il est composé d’événements (event). Chaque événement est composé des propriétés suivantes (celles présentées sont celles utilisées dans l’application. Pour une description complète, rendez-vous sur le site officiel :

http://wiki.winamp.com/wiki/SHOUTcast_Calendar_Event_XML_File_Specification) :

- Un type (playlist ou DJ) : Dans le cas de l’application, le type playlist sera toujours utilisé
- Une playlist : La lecture de la playlist dispose de plusieurs paramètres :
 - « loopatend » est un booléen qui définit si la playlist doit être rejouée quand tous les morceaux qui s’y trouvent ont été joués. Dans le cas de l’application, cette valeur est toujours « true ».
 - « shuffle » est un booléen qui définit si les morceaux de la playlist doivent être joués de façon aléatoire.
 - « priority » est une valeur entière non-signée qui définit la hauteur de la priorité de l’événement. C’est-à-dire, si 2 événements sont prévus au même moment, celui avec la plus haute priorité est joué.
 - La nom de la playlist jouée (nom du fichier .lst)
- Un horaire : L’horaire définit les paramètres suivants :
 - « starttime » est l’heure (format : hh:mm:ss) de début de l’événement.
 - « duration » est la durée (format : hh:mm:ss) de lecture de la playlist.

⁹ Table du temps ou grille horaire en anglais.

- « repeat » est une valeur entière non-signée servant au transcoder pour savoir quel jour doit être lancé cet événement aux horaires données. Chaque jour de la semaine a une valeur et l'addition des valeurs des jours sélectionnés donne la valeur de « repeat ». Voici les valeurs définies par ShoutCAST :
 - 1 - Every Sunday
 - 2 - Every Monday
 - 4 - Every Tuesday
 - 8 - Every Wednesday
 - 16 - Every Thursday
 - 32 - Every Friday
 - 64 - Every Saturday
 - 128 - Time periodic : Cette dernière n'est pas encore implémentée. Pour plus d'information, rendez-vous au lien ci-dessus (lien vers le site officiel).

http://wiki.winamp.com/wiki/SHOUTcast_Calendar_Event_XML_File_Specification#Calendar_Tag

c'est dans repeat que l'on va définir les jours, chaque jour = une valeur donc il faudra un tableau de valeur

afficher les valeurs utilisés

TODO : event lié à calendrier et a une playlist

TODO : explication avec les fichiers transcodeurs

<http://calendar.codeplex.com/> utilisation du composant externe

5.12.3 Classes utilisées

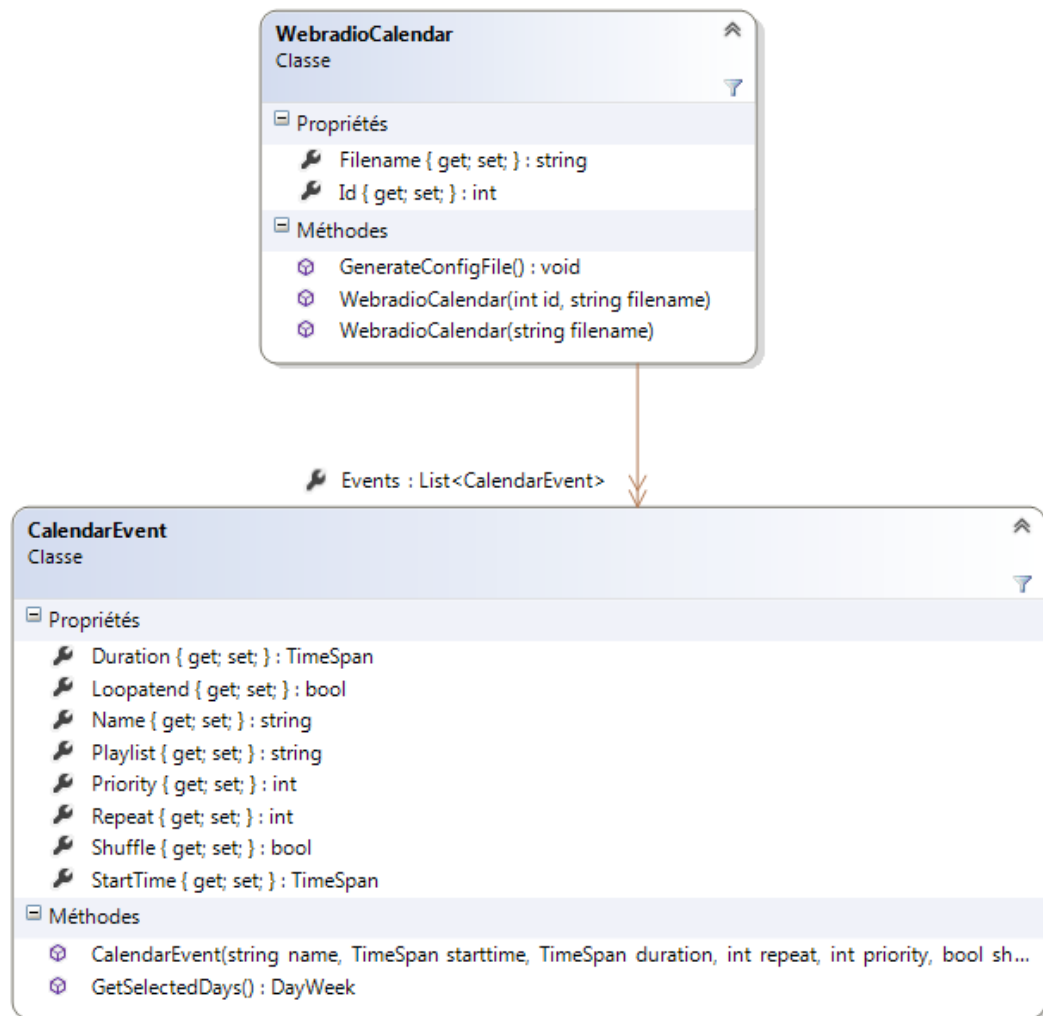


Figure 31 - Classes grille horaire

5.12.4 Génération de configuration

ShoutCAST (transcoder) utilise un fichier XML pour la gestion de son calendrier. La génération d'un fichier XML sous C# se fait simplement avec un objet XmlDocument et des XmlElement. Voici un exemple de création de fichier de configuration XML pour un calendrier :

```

XmlDocument document = new XmlDocument();
XmlElement root = document.CreateElement("eventlist");
foreach(CalendarEvent ev in this.Events)
{
    XmlElement eventelement = document.CreateElement("event");
    eventelement.SetAttribute("type", "playlist");
    XmlElement playlist = document.CreateElement("playlist");
    playlist.SetAttribute("loopatend", (ev.Loopatend)? "1": "0");
    playlist.SetAttribute("shuffle", (ev.Shuffle) ? "1" : "0");
    playlist.SetAttribute("priority", ev.Priority.ToString());
    playlist.InnerText = ev.Playlist;
    eventelement.AppendChild(playlist);

    XmlElement calendar = document.CreateElement("calendar");
    calendar.SetAttribute("starttime", ev.StartTime.ToString("hh:mm:ss"));
    calendar.SetAttribute("duration", ev.Duration.ToString("hh:mm:ss"));
    calendar.SetAttribute("repeat", ev.Repeat.ToString());
}
  
```

```

    eventelement.AppendChild(calendar);
    root.AppendChild(eventelement);
}

document.AppendChild(root);
document.Save(this.FileName);

```

Voici le XML une fois généré :

```

<eventlist>
  <event type="playlist">
    <playlist loopatend="1" shuffle="1" priority="1">HardLol</playlist>
    <calendar starttime="06:00:00" duration="02:00:00" repeat="48" />
  </event>
  <event type="playlist">
    <playlist loopatend="0" shuffle="0" priority="0">Pub migros</playlist>
    <calendar starttime="01:00:00" duration="00:30:00" repeat="0" />
  </event>
  <event type="playlist">
    <playlist loopatend="0" shuffle="0" priority="0">Commercial</playlist>
    <calendar starttime="02:00:00" duration="01:30:00" repeat="8" />
  </event>
</eventlist>

```

Figure 32 - Exemple XML calendrier

5.12.5 Affichage du calendrier

TODO : avec le composant

TODO : bleu = music et rouge = pub – Nom event (priority) \n playlistname etc

TODO : création classe qui derive de Appointment pour ajouter un champ « Playist »

TODO : masque de bit comme pour le chmod pour la valeur repeat afin de savoir quel jour est concerné par l'événement

TODO : attention au fait que ce soit des date time et que la calendrier commence le 01.01.0001

TODO : expliquer l'histoire de la structure dayweek, avec les masque, le tableau et l'affichage tout ça est géré par la classe eventcalendar

5.12.6 Sélection depuis le calendrier

TODO : explication du fait que la selection sur le tableau s'affiche dans les champs en haut

5.12.7 Création d'un événement

TODO : via la selection sur le calendrier et evenement SelectionChanged

TODO : pas 2 fois le meme nom d'événement pour la meme webradio

TODO : maskedtextbox

TODO : durée minimum + tryparse des maskedtextbox pour voir si format bon

TODO : génération de la valeur repeat : avec un enum DayValue

TODO : loop at end = toujours true

TODO : ajout comme les autre onglet (ajout,k recuperation id etc)

TODO : regeneration du fichier de config a la fin

5.12.8 Modification d'un événement

TODO : drag and drop (mouse up event)

TODO : seulement duration et starttime pour le moment

TODO : pour recalculer le repeat après un déplacement :

- Récupérer tous les événements (case dans la calendrier) concernés (lié à l'événement de celui qui a été déplacé)
- Calculer le repeat avec les dayofweek de chaque Appointment.startdate

TODO : regeneration du fichier de config a la fin

TODO : pas 2 fois le meme event le meme jour ! vérifier lors de la modification (tout ca est défini par shoutcast et son système de calendrier, vérification avec methode CheckMovePossible)

5.12.9 Suppression d'un événement

TODO Cliquez droit, supprime tous l'événement

5.12.10 Génération automatique

5.13 Transcoder

TODO : schema : tous les transcodeurs utilisent le même exécutable (pourquoi ? car il faut juste mettre à jour UN exécutable) mais juste des fichiers de config différents. Ils utilisent aussi tous le même calendrier

TODO explication du système de fichiers xml pour gérer le transcodeur etc (détails dans des nouveaux chapitres sur les playlists etc) et qu'il gère tout seul le calendrier etc

5.13.1 Définition des bitrates, taux d'échantillonnage et type d'encoder

TODO : expliquer que pour les types c'est un énorme, les bitrates c'est un tableau fixe et statique dans la classe WebradioTranscoder et de même pour les taux

5.13.2 Fichier de configuration

A chaque génération, suppression du fichier déjà présent

TODO : explication des différents éléments du fichier de config ainsi que les éléments modifiables par l'utilisateur

Chaque transcodeur = un fichier de config

5.13.3 Licence MP3

TODO

5.13.4 Exécution

5.13.5 Statut et logs

TODO : se référer au processus ou alors requête ajax dessus pour avoir toutes les informations

http://wiki.winamp.com/wiki/SHOUTcast_Transcoder AJAX_api_Specification#LogData

Ou afficher depuis fichier de log (blinder fichier au listBox)

5.13.6 Historique

TODO : historique dans bdd etc

Historique dans transcoder car coter serveur = pas possible (infomaniak par exemple)

5.13.7 Gestion des processus

TODO : avoir une liste de processus où les transcoders sont lancés afin de pouvoir récupérer leur log et statut facilement

TODO : lancement de transcoder, etc

5.14 Serveur de diffusion interne

http://wiki.winamp.com/wiki/SHOUTcast_DNAS_Server_2

Shoutcast est avant tout un serveur de diffusion de flux audio ou vidéo (Shoutcast DANS server 2).

L'outil propose un serveur en ligne de commande qui sera utilisé dans ce projet. Le serveur fonctionne avec un fichier de configuration qui lui est donné en paramètre lorsque l'on l'exécute :

```
sc_serv.exe myconfig.config
```

La documentation concernant la configuration du serveur via le fichier est expliquée ici :

http://wiki.winamp.com/wiki/SHOUTcast_DNAS_Server_2. Dans le cadre de ce projet, seules quelques options seront utilisées et configurées. Une partie est configurable par l'utilisateur via l'interface de l'onglet «[server](#)» et l'autre partie est configurée par défaut par l'application :

- L'emplacement du fichier de log
- L'emplacement du fichier de configuration lui-même

Pour plus de détails sur les emplacements des différents fichiers, rendez-vous au chapitre concernant [la structure des dossiers/fichiers](#).

TODO fonctionnement classe WebradioServer

5.14.1 Configuration

TODO : explication fichier de configuration

5.14.2 Exécution

5.14.3 Log

TODO : affichage depuis fichier log (filename dans bdd)

6 Tests

7 Plannings

7.1 Prévu

7.2 Final

8 Apports personnels

9 Conclusion

10 Améliorations possibles

- Modification des informations de la bibliothèque
- Modification de la « timetable » avec des drag and drop etc.
- Multi-serveur de diffusion par transcoder
- Capture live
- DJ

11 Références

- <https://cacoo.com> : Création de diagrammes en ligne
- <http://balsamiq.com/> : Création de « mokuup »
- <http://calendar.codeplex.com/> : Composant C# pour l'affichage du calendrier sur une semaine
- <http://sqlitestudio.pl/> : Logiciel de gestion de base de données SQLite

12 Annexes