|  |
| --- |
| CFPT-I |
| WebradioManager |
| Travail de diplôme 2014 |
|  |
| **Simon Menetrey – T.IN E2A** |
| **31/03/2014** |

|  |
| --- |
| [Tapez le résumé du document ici. Il s’agit généralement d’une courte synthèse du document. Tapez le résumé du document ici. Il s’agit généralement d’une courte synthèse du document.] |

# Résumé

## Français

## English

Contenu

[1 Résumé 1](#_Toc384303791)

[1.1 Français 1](#_Toc384303792)

[1.2 English 1](#_Toc384303793)

[2 Introduction 5](#_Toc384303794)

[2.1 Mise en situation 5](#_Toc384303795)

[2.2 Sujet 5](#_Toc384303796)

[2.3 Qu’est-ce qu’une webradio ? 5](#_Toc384303797)

[2.4 Pourquoi ce sujet ? 5](#_Toc384303798)

[2.5 Termes 6](#_Toc384303799)

[3 Cahier des charges 7](#_Toc384303800)

[4 Analyse fonctionnelle 8](#_Toc384303801)

[4.1 Schéma de l’application 8](#_Toc384303802)

[4.2 Interface principale 9](#_Toc384303803)

[4.2.1 Fenêtre d’administration 9](#_Toc384303804)

[4.2.2 Menu principale 9](#_Toc384303805)

[4.3 Gestion de plusieurs webradio 10](#_Toc384303806)

[4.3.1 Création d’une webradio 11](#_Toc384303807)

[4.3.2 Sélection d’une webradio 11](#_Toc384303808)

[4.4 Serveur de diffusion interne et externe 11](#_Toc384303809)

[4.5 Gestion des musiques/pubs 12](#_Toc384303810)

[4.5.1 Affichage 12](#_Toc384303811)

[4.5.2 Importer et indexer 12](#_Toc384303812)

[4.5.3 Ajout à une playlist 13](#_Toc384303813)

[4.6 Gestion des listes de lecture 14](#_Toc384303814)

[4.6.1 Listes de lecture musicales 14](#_Toc384303815)

[4.6.2 Listes de lecture publicitaires 14](#_Toc384303816)

[4.6.3 Création 14](#_Toc384303817)

[4.6.4 Génération automatique 14](#_Toc384303818)

[4.6.5 Affichage du contenu d’une playlist 14](#_Toc384303819)

[4.7 Gestion des horaires 15](#_Toc384303820)

[4.7.1 Événement 15](#_Toc384303821)

[4.7.2 Événement périodique 16](#_Toc384303822)

[4.7.3 Remplissage manuel 16](#_Toc384303823)

[4.7.4 Remplissage automatique 16](#_Toc384303824)

[4.7.5 Suppression d’un événement 16](#_Toc384303825)

[4.8 Gestion des transcoders 17](#_Toc384303826)

[4.8.1 Création 17](#_Toc384303827)

[4.8.2 Affichage 18](#_Toc384303828)

[4.8.3 Historique de diffusion 18](#_Toc384303829)

[4.9 Gestion du serveur 18](#_Toc384303830)

[4.9.1 Contrôles 18](#_Toc384303831)

[4.9.2 Configuration 18](#_Toc384303832)

[4.9.3 Log 19](#_Toc384303833)

[4.9.4 Interface web 19](#_Toc384303834)

[4.9.5 Administration web 19](#_Toc384303835)

[5 Analyse organique 21](#_Toc384303836)

[5.1 Environnement 21](#_Toc384303837)

[5.2 Diagramme de classes 21](#_Toc384303838)

[5.3 Base de données 21](#_Toc384303839)

[5.3.1 SQLite 21](#_Toc384303840)

[5.3.2 Schéma 21](#_Toc384303841)

[5.4 Schéma de diffusion 22](#_Toc384303842)

[5.4.1 Principe de base 22](#_Toc384303843)

[5.4.2 Infomaniak 22](#_Toc384303844)

[5.5 ShoutCast 23](#_Toc384303845)

[5.5.1 Présentation 23](#_Toc384303846)

[5.5.2 Pourquoi cet outil ? 23](#_Toc384303847)

[5.5.3 Serveur 24](#_Toc384303848)

[5.5.4 Transcoder 24](#_Toc384303849)

[5.5.5 Schéma de fonctionnement résumé 24](#_Toc384303850)

[5.6 Structures des dossiers/fichiers 24](#_Toc384303851)

[5.7 Webradio 25](#_Toc384303852)

[5.7.1 Création 25](#_Toc384303853)

[5.8 Transcoder 25](#_Toc384303854)

[5.8.1 Fichier de configuration 25](#_Toc384303855)

[5.8.2 Exécution 25](#_Toc384303856)

[5.8.3 Statut et logs 25](#_Toc384303857)

[5.8.4 Historique 25](#_Toc384303858)

[5.8.5 Gestion des processus 25](#_Toc384303859)

[5.9 Listes de lecture 25](#_Toc384303860)

[5.9.1 Musicale 25](#_Toc384303861)

[5.9.2 Publicitaires 25](#_Toc384303862)

[5.9.3 Ajout à une playlist 25](#_Toc384303863)

[5.9.4 Génération automatique 26](#_Toc384303864)

[5.9.5 Suppression 26](#_Toc384303865)

[5.10 Musiques 26](#_Toc384303866)

[5.10.1 Ajout et indexation 26](#_Toc384303867)

[5.10.2 Tags ID3 26](#_Toc384303868)

[5.10.3 Analyse des tags 26](#_Toc384303869)

[5.10.4 Suppression 26](#_Toc384303870)

[5.11 Grille horaire 26](#_Toc384303871)

[5.11.1 Affichage du calendrier 26](#_Toc384303872)

[5.11.2 Création d’événement 26](#_Toc384303873)

[5.11.3 Génération automatique 26](#_Toc384303874)

[5.12 Serveur de diffusion interne 26](#_Toc384303875)

[5.12.1 Configuration 27](#_Toc384303876)

[5.12.2 Exécution 27](#_Toc384303877)

[5.12.3 Log 27](#_Toc384303878)

[6 Tests 28](#_Toc384303879)

[7 Plannings 29](#_Toc384303880)

[7.1 Prévu 29](#_Toc384303881)

[7.2 Final 29](#_Toc384303882)

[8 Apports personnels 30](#_Toc384303883)

[9 Conclusion 31](#_Toc384303884)

[10 Améliorations possibles 32](#_Toc384303885)

[11 Références 32](#_Toc384303886)

[12 Annexes 32](#_Toc384303887)

# Introduction

## 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 à 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 Suisa[[1]](#footnote-1).  
J’ai donc réalisé le cahier des charges avec KTFM afin de répondre au mieux à leurs besoins.



## 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

## 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.

## 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.

## Termes

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

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

# 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 facilité 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

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

Analyses

# Analyse fonctionnelle

## Schéma de l’application



Figure - 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.

## Interface principale

### Fenêtre d’administration



Figure - 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 décrient dans les chapitre suivants.

### Menu principale

TODO : complétement en fonction du menu choisi

## Gestion de plusieurs webradio



Figure - 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 clique pour ouvrir une webradio.  
Le bouton « Duplicate » crée une copie exacte de la webradio sélectionnée avec un suffix « 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 fraichement 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](#_Schéma_de_diffusion).



Figure - Schéma webradios

### 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 fraichement créé. Le nom d’une webradio ne peut pas dépasser 255 caractères.

### 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.

## 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 provider[[2]](#footnote-2) (exemple : infomaniak[[3]](#footnote-3) 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.

## Onglet « Status »

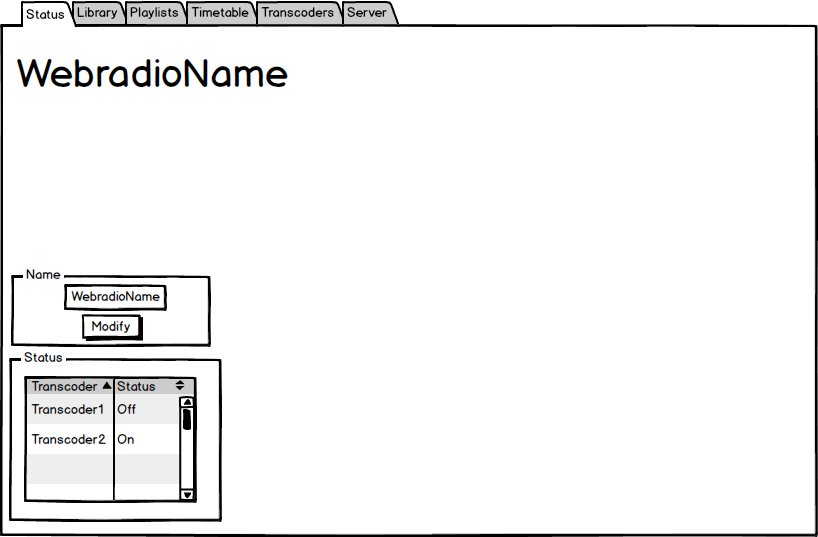


Figure - Onglet "status"

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 trancoders de la webradio actuelle.

## Gestion des musiques/pubs



Figure - 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.

### 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.

### Importer et indexer

L’utilisateur peut importer des fichiers musicaux dans sa bibliothèque. Le bouton « Import from folder… » ouvre une boite 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 boite 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 boite 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.

### 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.

## Gestion des listes de lecture

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

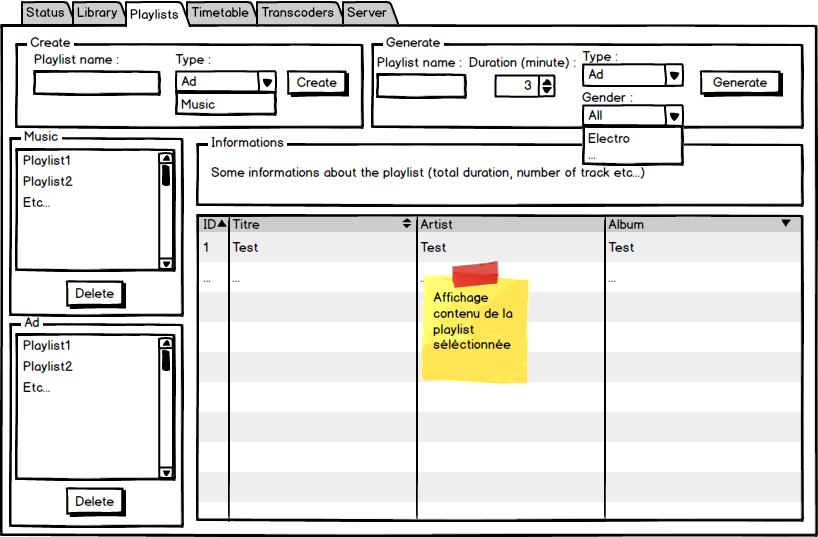


Figure - Onglet "playlists"

### Listes de lecture musicales

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

### Listes de lecture publicitaires

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

### 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 fraichement créé est ensuite ajouté à l’une des 2 listes situées à gauche en fonction de son type.

### Génération automatique

L’utilisateur a la possibilité de générer automatiquement une playlist en définissant une durée, un nom, un genre 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.

### 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. 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.

## Gestion des horaires

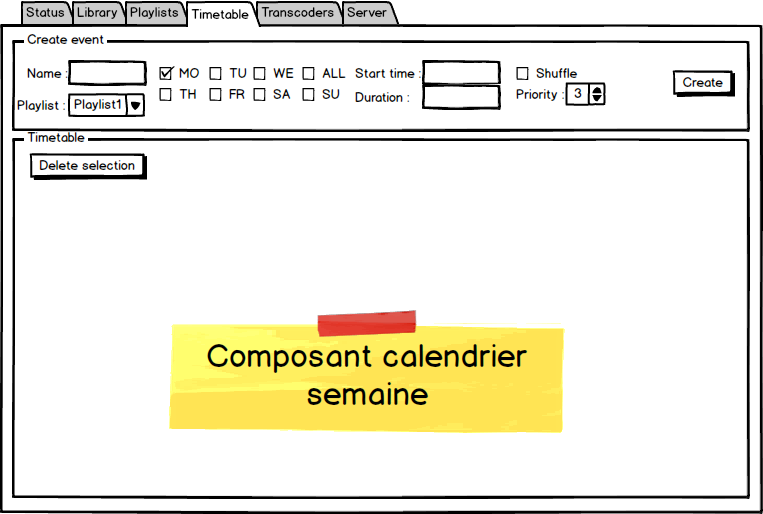


Figure - Onglet "timetable"



Figure - Calendrier

### 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.

### Événement périodique

Un événement périodique se produit à intervalles réguliers.

TODO

### 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 crée un événement.



Figure - 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.

### Remplissage automatique

TODO

### 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](#_Remplissage_manuel)).

## Gestion des transcoders

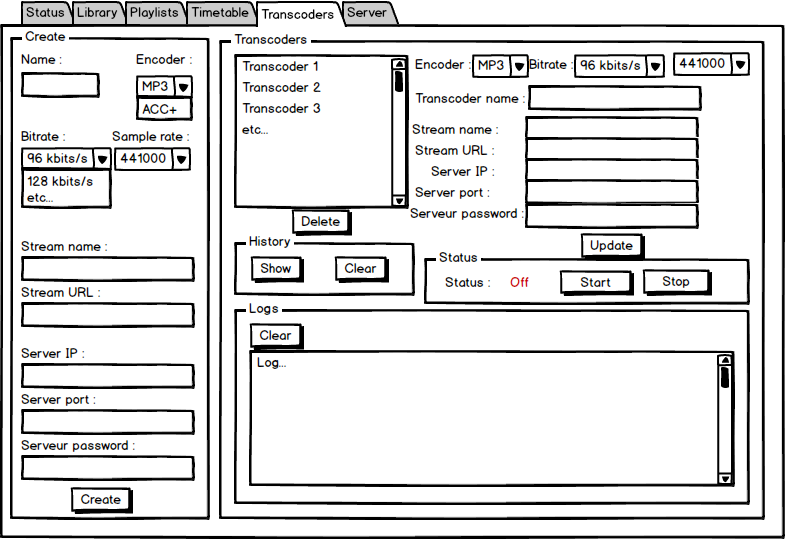


Figure - Onglet "Transcoders"

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.

### 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 bitrate[[4]](#footnote-4) du flux
* Le sample rate[[5]](#footnote-5)
* 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

### 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 ».

### 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.

### 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 ».

## Gestion du serveur

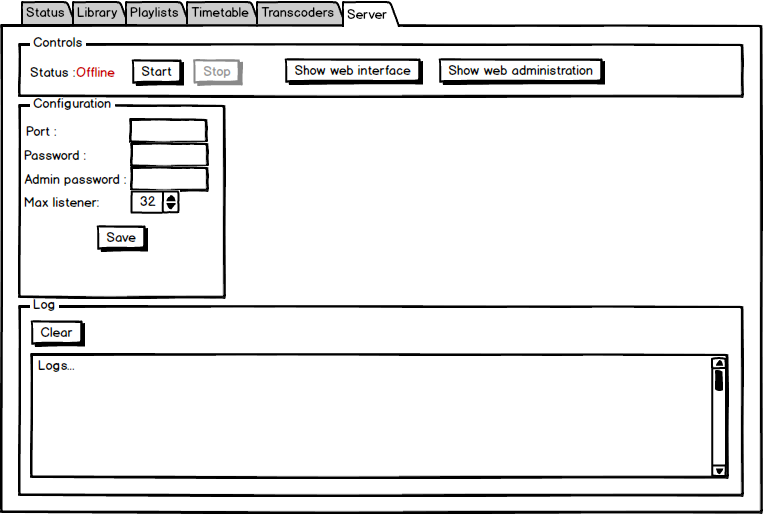


Figure - Onglet "server"

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.

### 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.

### 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.

### Log

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

### 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.



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

### 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.



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.

# Analyse organique

## Environnement

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

## Diagramme de classes

### Diagramme



Figure - Diagramme de classes

### Observateurs/Sujet

TODO : excplicastiondu principe

### Classes abstraites

TODO : explication du principe

### Stockage des webradios

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

## 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.

### SQLite



Figure - 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/>

### Utilisation

Une bibliothèque sous forme d’une DLL[[6]](#footnote-6) 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.

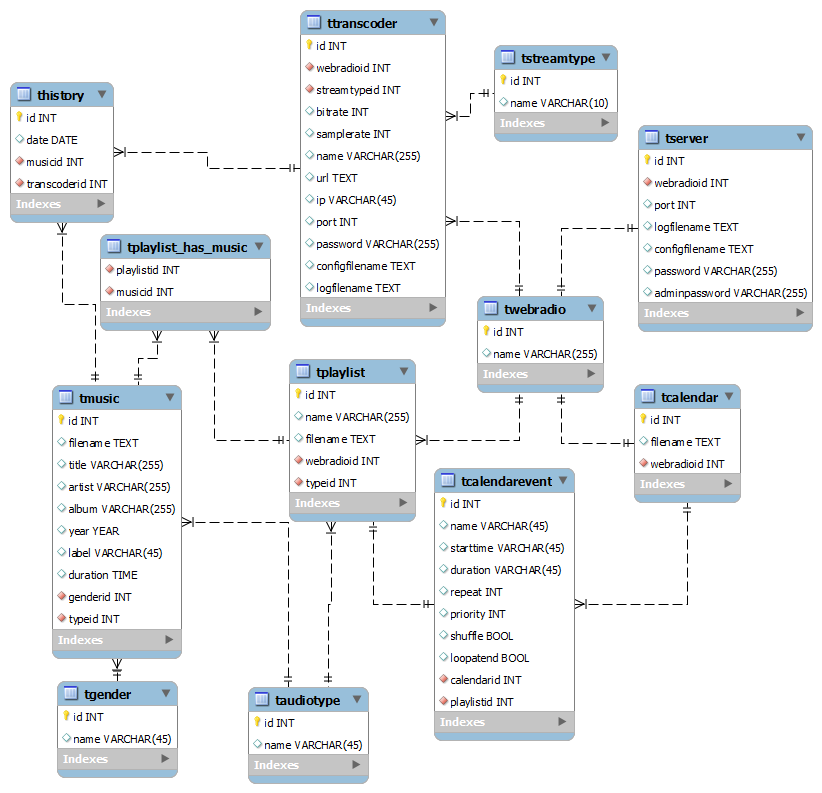
La classe Bdd utilise BddControls afin d’effectuer des requêtes sur la base de données. Bdd propose des méthodes simples comme par exemple « AddWebradio ». Ainsi la partie traitement des données se fait dans Bdd et la partie exécution dans BddControls.

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

### 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 supprimée automatiquement.

### Schéma



### Twebradio

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

### 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 |

### 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) |

### 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](#_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) |

### 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) |

### Taudiotype

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

### 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) |

### Tgender

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

### Tplaylist\_has\_music

|  |  |  |
| --- | --- | --- |
| Nom | Type | Description |
| Playlistid | Int | Clé étrangère (Playlist concernée) |
| Musicid | Int | Clé étrangère (Musique concernée, qui fait partit de la playlist ayant l’identifiant Playlistid) |

### Thistory

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

### 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 |

## Schéma de diffusion

### Principe de base



Figure - 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).

### Infomaniak



Figure - Schéma de diffusion



Figure - 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.

## ShoutCast



Figure - Logo Schoutcast

### 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 Winamp[[7]](#footnote-7). Ce sont donc ces derniers qui seront utilisé pour le projet.

### Pourquoi cet outil ?

J’ai choisi Shoutcast car il propose des outils en ligne de commande ainsi qu’une gestion facilitée via des fichiers XML[[8]](#footnote-8) ou texte basiques. Cela permet de pouvoir interagir plus facilement avec via des applications externes telle que la mienne.

### 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](#_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.](#_Interface_web)

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

### 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.

### Schéma de fonctionnement résumé



Figure - Schéma shoutcast

## Structures des dossiers/fichiers

### Schéma

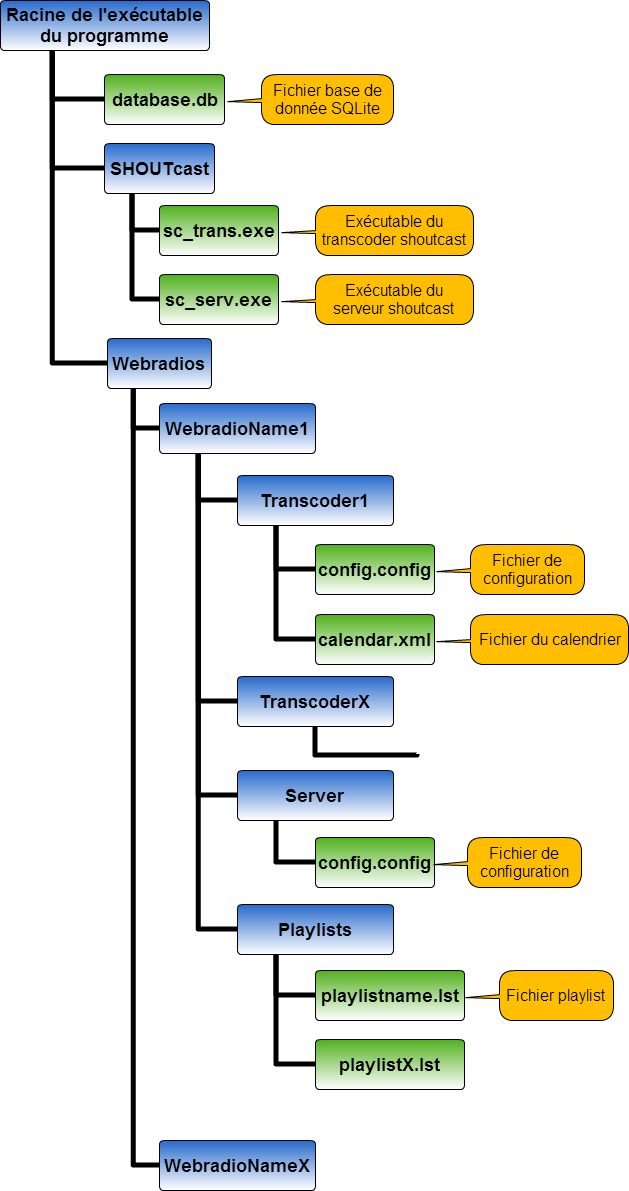


Figure - Schéma structure des fichiers/dossiers

Ce schéma décrit l’organisation des fichiers dans le dossier de l’application. La base de données contient les informations pour les transcoders 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.

### Exécutables Shoutcast

Attention : Il n’y a pas un exécutable de transcodeur par transcoder de webradio ni un exécutable de serveur par webradio. Il existe seulement un seul et unique exécutable transcoder 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 transcoder 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 transcoder 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 :

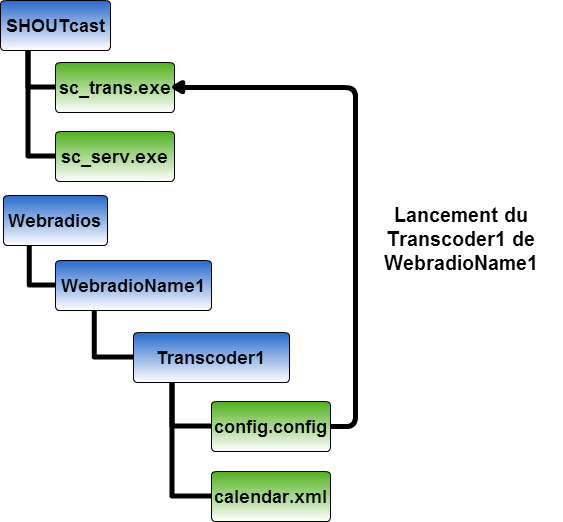


Figure - Exemple lancement transcoder

## 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.



Figure - 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 2ème charge les musiques présentes dans la bibliothèque ainsi que les playlist qui leur sont associées.

## Gestion des processus

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

TODO : POUR CHAQUE CHAPITRE, EXPLIQUER LA CLASSE ASSOCIEE

## Webradio

### Classes associée



Figure - Classe "Webradio"

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

### Affichage des webradios disponibles

Les webradios disponibles sont affiché dans la ListBox centrale de la fenêtre SelectionView. Pour la remplir, cella 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. Le composant ListBox prend n’importe quel type d’élément mais il appel la méthode « ToString() » de ces derniers. C’est pour cela que la classe Webradio implémente cette méthode qui est « override » de celle proposée par sa classe parente « Object ».

### Création

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



Figure - 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 nouvelle 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’objet 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](#_Génération_des_configurations).

### 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 :



Figure - Diagramme de séquence instanciation 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.

TODO : dire qie ée fpomnctionnement du composant calendrier est exploiqué dans le chapitre prévu

TODO : les combobox etc sont remplis avec des items de type de la classe cocnernée (exemple : playlist) et ces classes ont un tostring() override

### Duplication

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

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

### 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 Dictionnary qui prend l’identifiant de la webradio à supprimer.

### Génération des configurations

La classe Webradio dispose d’une méthode « GenerateConfigFiles() » qui appel 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 classes en question. En fonction de la classe, le type de fichier de configuration sera différent (simple fichier texte ou fichier XML).

Par exemple, le fichier de configuration d’un calendrier est sous fourme XML. 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);

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.

## Transcoder

TODO : schema : tous les transcoder utilise le même executable (pourquoi ? car il faut juste mnettre a jour UN executable) mais juste des fichier de config différent. Ils utilisent aussi tous les meme calendrier

TODO explication du system de fichier xml pour gerer le transcoder etc ( détails dans des nouveau chapitre sur les playlist etc ) et qu’il gère tout seul els calendrier etc

### Définition des bitrates, taux d’échantillonnage et type d’encoder

TODO : expliquer que pour les type c’est un enorme, les bitrate c’est un tableau fixe et static dans la classe WebradioTranscoder et de même pour les taux

### Fichier de configuration

A chaque generation, suppression du fichier déjà présent

TODO : explication des différents elements du fichier de conf ainsi que les element modifiable par l’utilisateur

Chaque transcoder = un fichier de config

### Licence MP3

TODO

### Exécution

### Statut et logs

TODO : se référé au processus ou alors requete 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)

### Historique

TODO : historique dans bdd etc

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

### Gestion des processus

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

TODO : lancement de transcoder, etc

## Listes de lecture

TODO : explication avec les fichiers transcodeur

TODO : explication génération xml et génération automatique

### Musicale

### Publicitaires

### Ajout à une playlist

TODO : depuis l’onglet library

TODO : refraichissement de la vue

### Génération automatique

TODO : genre et durée

### Suppression

## Musiques

### Listes de lecture associées

TODO : explication de la liste de playloistz associé a un objet music

### Ajout et indexation

TODO : recursivité ou pas

<http://www.novell.com/products/linuxpackages/opensuse11.1/taglib-sharp.html>

### Tags ID3

### Analyse des tags

TODO : genre unique dans la db

### Suppression

TODO : supprimer des playlist

## Grille horaire

TODO : expliquer qu’il faudra jouer sur les duration pour générer les playlist et le calendrier

<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 joiur = une valeur donc il faudra un tableau de valeur

afficher les valeurs utilisé

TODO : event lié à calandrier et a une playlist

TODO : explication avec les fichier transcodeur

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

<http://www.devcomponents.com/kb2/?p=404>

### Affichage du calendrier

TODO : avec le composant

TODO : masque de bit comme pour le chmod pour la valeur repeat afin de savoir quel jour est cooncernépar l’event

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 ca ets géré apr la classe eventcalendar

### Création d’événement

TODO : via la selection sur le calendrier et evenement SelectionChanged

### Génération automatique

## 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](#_Gestion_du_serveur)» 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](#_Structures_des_dossiers/fichiers).

TODO fonctionnamenrt classe WebradioServer

### Configuration

TODO : explication fichier de configuration

### Exécution

### Log

TODO : affichage depuis fichier log (filename dans bdd)

# Tests

# Plannings

## Prévu

## Final

# Apports personnels

# Conclusion

# Améliorations possibles

* Multi-serveur de diffusion par transcoder
* Capture live
* DJ

# Références

* <https://cacoo.com> : Création de diagrammes en ligne
* <http://balsamiq.com/> : Création de « mokup »
* <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

# Annexes

1. SUISA est la coopérative des auteurs et éditeurs de musique <http://www.suisa.ch/fr/> [↑](#footnote-ref-1)
2. Fournisseur de services internet [↑](#footnote-ref-2)
3. <http://www.infomaniak.com/> [↑](#footnote-ref-3)
4. Débit binaire : une mesure de la quantité de données numériques transmises par unité de temps. [↑](#footnote-ref-4)
5. Taux d’échantillonnage [↑](#footnote-ref-5)
6. Dynamic Link Library [↑](#footnote-ref-6)
7. Lecteur multimédia développé par Nullsoft [↑](#footnote-ref-7)
8. eXtensible Markup Language : Langage de balisage extensible [↑](#footnote-ref-8)