ÉCOLE CENTRALE LYON

Projet d'étude promotion 2014

Projet d'études N° 87

Noms des élèves :
Adrien AUTRICQUE
Clément COCQUEMPOT
Thomas GAUDELET
Marie GAUTHIER
Ghislain GANDOLFI

Commanditaire :
Département Mathématiques et Informatique
De l'Ecole Centrale de Lyon

Tuteur(s) scientifique(s) :
Emmanuel DELLANDREA

Conseiller en communication :
Ariane PIERROU

Conseiller en gestion de projet :
Vincent FRIDRICI

Département d'accueil :
Département Mathématiques et Informatique (MI)

Date du rapport :
19/05/2012

Création d'un logiciel intelligent pour la gestion de la musique, des images et des vidéos

TABLE DES MATIERES

INTRODUCTION :

Nous assistons depuis plusieurs années à un développement très intense des technologies de l'information. Les technologies multimédias ne sont pas en reste. Il existe maintenant sur le marché de nombreux logiciels de lecture et d'édition de médias. Nombre d'entre eux sont capables de traiter les trois principaux types de médias que l'on rencontre dans le milieu informatique, à savoir : la musique, les images et les vidéos. Ils font également intervenir la notion de tag, c'est-à-dire un mot attaché à un fichier qui le caractérise. Ces tags permettent d'effectuer des recherches plus facilement dans la bibliothèque d'un logiciel multimédia.

Ce projet d'étude se propose de créer un logiciel multimédia, baptisé Tag'n'Link, comportant des fonctionnalités nouvelles. Il a débuté en septembre 2011 pour prendre fin en juin 2012. Le commanditaire de ce projet d'étude est le laboratoire LIRIS (laboratoire d'informatique en image et systèmes d'information) et son tuteur scientifique est Emmanuel DELLANDREA. L'équipe d'élèves étudiant en première année à l'Ecole Centrale de Lyon en charge du projet est composée de Marie GAUTHIER, Clément COCQUEMPOT, Ghislain GANDOLFI, Thomas GAUDELET et Adrien AUTRICQUE.

Le laboratoire LIRIS développe à l'Ecole Centrale de Lyon des algorithmes capables de saisir des émotions et des concepts dans des fichiers de type musique ou image. Une émotion est caractérisée par son degré de positivité (joie ou tristesse) et d'activité (émotion active ou passive). Un concept, quant à lui, est un mot qui peut être considéré comme un tag. Ainsi, les algorithmes du laboratoire LIRIS sont capables de reconnaitre une voiture sur une photo en représentant une. La notion d'émotion est très complexe à programmer. Le laboratoire LIRIS nous propose d'intégrer ses algorithmes au sein de notre logiciel afin de pouvoir créer des liens entre des fichiers multimédias émotionnellement proches.

L'objectif principal de ce projet réside en la création d'un logiciel capable de créer de façon intelligente des liens entre différents fichiers de type différents. Le but est bien-sûr de pouvoir regrouper des fichiers émotionnellement proches dans la bibliothèque du logiciel en vue d'en faciliter la lecture sous forme d'un diaporama musical par exemple. Ces différents liens devront pouvoir se faire à l'aide des algorithmes fournis pas le laboratoire LIRIS, ainsi que par l'intermédiaire des tags attachés à chaque fichier multimédia. Ce logiciel, accompagné de sa notice d'utilisation, constitue le livrable final de ce projet d'étude.

La création d'un logiciel se déroule en trois grandes phases : la conception, la réalisation et les tests. La dernière partie peut paraître moins importante que les deux autres, mais il serait illusoire de penser que notre logiciel fonctionnera parfaitement dès son premier lancement. C'est pourquoi nous comptons effectuer de nombreux tests tout au long de la programmation afin d'obtenir le moins de bugs possible lors de la compilation.

Les détails des fonctionnalités de ce logiciel seront exposés dans une première partie, où le projet sera présenté de façon générale ; les parties de conception et de réalisation occupant les deux parties suivantes.

I- Présentation générale

1) Liens et gestion de médias

À l'ère de l'informatique, nos ordinateurs sont surchargés de fichiers multimédias de tous types affublés de métadonnées (tags) diverses. Bien qu'il existe de nombreux logiciels sur le marché permettant la lecture, le traitement ou la modification de ces fichiers multimédias, il est rare qu'un logiciel remplisse toutes ces fonctionnalités simultanément.

Ce projet d'étude a consisté en la création entière d'un tel logiciel à partir de l'idée suivante : dans une bibliothèque multimédia, pourquoi ne pas classer ses différents fichiers multimédias par émotions plutôt que par ordre alphabétique ou par année ?

Le commanditaire de ce projet d'étude, le laboratoire LIRIS, souhaite intégrer les algorithmes qu'il développe au sein du logiciel créé. Ces algorithmes permettent de représenter un fichier multimédia de type image ou musique par un point sur un graphique bidimensionnel baptisé *Graphe des Emotions* :

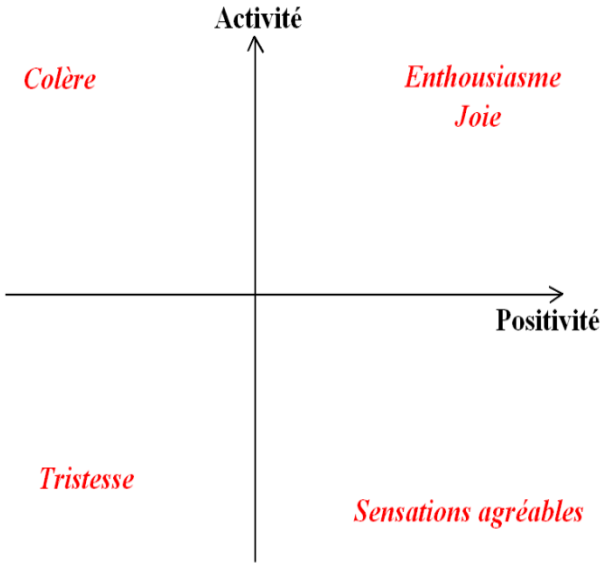

Figure 1.1 : Graphe des Emotions

Le graphique ci-contre est une représentation sommaire du *Graphe des Emotions*. Les deux dimensions correspondent à la positivité (également appelée appréciation) et l'activité, et les termes en rouge désignent les principales émotions que l'on peut caractériser à l'aide de ce modèle. Tout le panel des émotions humaines ne peut être atteint par celui-ci, mais la programmation d'un algorithme capable de placer un titre musical sur ce graphe est déjà très complexe.

En outre, les algorithmes produits par le LIRIS sont capables de saisir des concepts dans des fichiers de type image : ils peuvent par exemple reconnaître une voiture sur une photo. La liste complète des concepts que sont capables de reconnaitre les algorithmes est présente en *annexe 2*.

Grâce à ces algorithmes, il est alors possible de capter des informations très intéressantes sur un fichier multimédia. Le but de notre logiciel est de pouvoir traiter ces informations le plus intelligemment possible. Les concepts fournis par les algorithmes du laboratoire LIRIS attachés à un fichier multimédia ainsi que les coordonnées de ce fichier sur le Graphe des Emotions devront être stockés dans un (ou plusieurs) fichiers.

La possibilité de lier ainsi différents fichiers multimédias par affinité émotionnelle permet d'en faire une utilisation nouvelle : l'utilisateur pourra par exemple réaliser des diaporamas musicaux, comme le permet le logiciel Windows Movie Maker, en regardant des images tout en écoutant de la musique évoquant des émotions similaires. Cela permettra également à l'utilisateur d'organiser sa bibliothèque multimédia d'une manière inédite jusqu'ici : grâce à une fonction de recherche, il pourra trouver très facilement un fichier évoquant l'émotion recherchée.

2) Demandes du maître d'ouvrage

Depuis plusieurs années, les utilisateurs de logiciels multimédias doivent faire face à des problèmes de gestion et d'utilisation de collections de plus en plus volumineuses concernant aussi bien les photographies que les musiques.

Ce projet d'étude se positionne dans ce contexte et a pour objectif de développer un logiciel innovant permettant l'organisation, la recherche la visualisation et l'association des images et musiques. Cet outil proposera à l'utilisateur une interface intuitive et efficace permettant notamment d'enrichir les informations associées aux données multimédias (annotations ou tags par exemple) afin d'améliorer leur organisation et leur utilisation. Il devrait ainsi permettre d'écouter de la musique, de regarder des photos à partir de critères avancés correspondant aux choix de l'utilisateur (photos des dernières vacances, photos s'accordant bien avec telle ou telle musique,…). Le terme *tag* désigne une annotation ou un mot associé à un fichier multimédia. Ce peut être n'importe quel mot, et un grand nombre de tags peuvent être associés à un même fichier. Par exemple, le nom du fichier, le titre et l'artiste d'une musique sont autant de tags associés à cette musique. Des algorithmes d'analyse de données audio ou visuelles développés au LIRIS pourront être intégrés pour enrichir le logiciel.

À l'origine, le logiciel devait pouvoir également traiter les fichiers multimédias de type vidéo. Cependant, pendant la phase de conception du logiciel, il est apparu que les vidéos et films n'ont pas lieu d'être liés entre eux ou avec d'autres types de fichiers multimédias. En effet, on peut écouter de la musique tout en regardant des photographies, mais toute lecture simultanée faisant intervenir des vidéos est impossible. C'est également pour une question de temps qu'il a été décidé de se concentrer exclusivement sur les images et musiques. L'intégration du traitement des fichiers de type vidéo constitue une amélioration éventuelle qui pourrait être envisagée si ce projet d'étude venait à être repris l'année prochaine.

Une fonction de recherche doit être intégrée au logiciel. Elle devra permettre d'effectuer des recherches à la fois dans la bibliothèque d'images et dans celle de musiques. Elle aura également pour fonctionnalité de pouvoir effectuer une recherche dans les listes de tags associées aux fichiers multimédias. Par exemple, si l'on tape « voiture » dans la barre de recherche, toutes les images possédant le concept *voiture* dans leur liste de tags devront apparaître.

Les objectifs de production de ce projet d'étude sont le logiciel Tag'n'Link accompagné de sa notice d'installation et d'utilisation, ainsi que le présent rapport.

3) Etat de l'art des logiciels multimédias

Après avoir étudié les logiciels les plus usuels, tel que iTunes d'Apple ou encore Winamp et le Lecteur Windows Media Player de Microsoft, force est de constater qu'aucun logiciel à ce jour ne donnait à l'utilisateur la possibilité de créer des mots-clés ou encore de lier les différents fichiers multimédias entre eux. Plus précisément, il n'en existe aucun qui permette de créer des liens entre médias dans le but de faciliter l'immersion de l'utilisateur dans une ambiance cohérente. C'est donc l'une des fonctions principales que le logiciel doit remplir.

De plus, la plupart des logiciels ne sont pas capables de gérer les trois types de fichiers (c'est-à-dire la musique, les images et les vidéos), ils sont parfois complets mais très peu intuitif (Winamp par exemple). Il faut parfois plusieurs heures pour maîtriser parfaitement un logiciel. C'est aussi ce caractère d'intuitivité qui est un critère important du logiciel.

Voilà un tableau regroupant les informations recueillies :

Logiciel		Musique	Image	Vidéo	Intuitivité	Complet	Mobilité	Lien	Payant
iTunes		X		X	X		X		X
Winamp		X	X	X		X	X		
Media Player		X		X	X				
QuickTime		X		X			X		X
VLC		X	X	X	X		X		
Real Player		X		X					
Audacity		X				X			
Zune		X	X	X	X				

Tableau 1 : Etat de l'art des logiciels de traitement de médias

La mobilité désigne la possibilité pour le logiciel de s'installer sur différentes plateformes. On observe qu'aucun des logiciels étudiés ne font intervenir la notion de lien émotionnel entre fichiers multimédias. On remarque également que le caractère intuitif d'un logiciel est d'autant plus grand qu'il n'est pas complet. C'est un des grands défis de ce projet : réaliser un logiciel suffisamment intuitif pour que l'utilisateur ne soit pas dissuadé par une difficulté trop importante, mais aussi suffisamment complet pour offrir des fonctionnalités variées.

Enfin, il est intéressant de voir que les logiciels ne lisent pas tous les formats des fichiers multimédias. Si Tag'n'Link doit pouvoir faire face aux logiciels présents sur le marché, il doit être capable de lire la plupart des formats présents.

4) Les outils informatiques à notre service

a) Conduite d'un projet informatique

Lors de la réalisation d'un projet informatique, il ne faut pas tomber dans le piège de commencer à coder dès le début. En effet, la phase de conception a une importance primordiale. Lors de cette phase, le groupe de projet doit discuter fréquemment avec le commanditaire afin de bien cerner les fonctionnalités attendues pour le logiciel à développer.

L'équipe du projet d'étude doit commencer par discuter du sujet avec le tuteur, afin de bien cerner les fonctionnalités auxquelles le logiciel demandé doit répondre. Ces discussions mènent alors à la rédaction d'un premier cahier des charges qui évolue peu au cours du projet. Ce cahier des charges contient de nombreuses contraintes et fonctions à réaliser, car comme le souligne la sous-partie précédente, un tel logiciel n'existe pas

encore sur le marché et il y a donc peu d'éléments de référence sur lesquels se baser. Enfin, dans la phase de conception, il faut prévoir l'apparence finale du logiciel créé : l'interface doit être réalisée sous forme de croquis. « Où placer tel bouton ? », « Que doit-il se passer lorsque l'on clique ici ? » sont des exemples de questions qu'il faut s'être posées avant de se lancer dans la programmation.

La phase de programmation pure n'arrive que lorsque cette première phase de conception est terminée. Si la première phase a été accomplie avec soin, la seconde n'en sera que plus facile et rapide. Aussi, beaucoup moins de bugs et d'incohérences sont rencontrés.

b) Programmation orientée objet

L'un des intérêts principaux de travailler en C++ réside dans la Programmation Orientée Objet ou POO. Cet outil ou plutôt cette manière de penser objet est d'une utilité très précieuse dans un tel projet informatique. Il va permettre d'organiser tout le projet beaucoup plus aisément. Ce mode de programmation est en effet une modélisation directe du projet étudié par l'intermédiaire de classes et des relations entre ces classes. Une classe est un type abstrait qui servira à instancier des objets, ainsi par exemple la classe Morceau représente le modèle d'un fichier audio en général à partir duquel on instancie un objet Morceau qui correspondra à un fichier audio en particulier. Dans chaque classe, on définit des attributs et des méthodes. Pour reprendre l'exemple de la classe Morceau, les attributs seront, entre autres, les informations d'une musique, c'est-à-dire son titre, sa durée, son artiste, etc. Tandis que les méthodes permettront, entre autre fonction, de modifier les attributs.

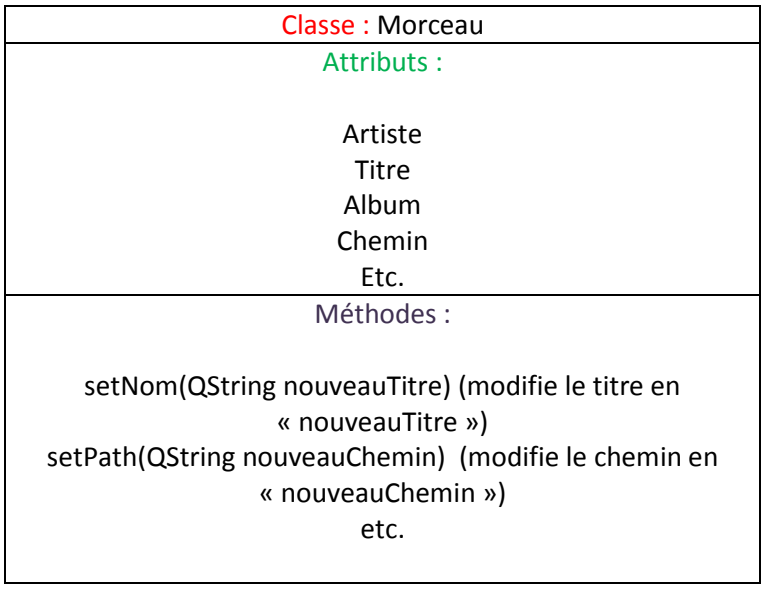

Figure 1.2. Classe Morceau

A partir de cela, l'une des étapes fondamentales d'un projet informatique en C++ consiste à identifier, puis à définir en détail toutes les classes nécessaires à la bonne conduite du projet. Par définir, il faut comprendre non seulement définir la classe en elle-même avec ses attributs et méthodes propres, mais aussi réfléchir à toutes les relations entre les classes et les interactions entre elles. Pour réaliser ceci un outil précieux est le langage UML ou « langage de modélisation unifié » qui permet de définir et de visualiser toutes ces relations à l'aide de diagrammes divers, ce qui permet de gagner en efficacité et donc en temps lors de la phase de programmation. A titre d'exemple, le diagramme de classe figure 1.3. donne, de manière non exhaustive, les relations de la classe Morceau avec certaines des classes implémentées.

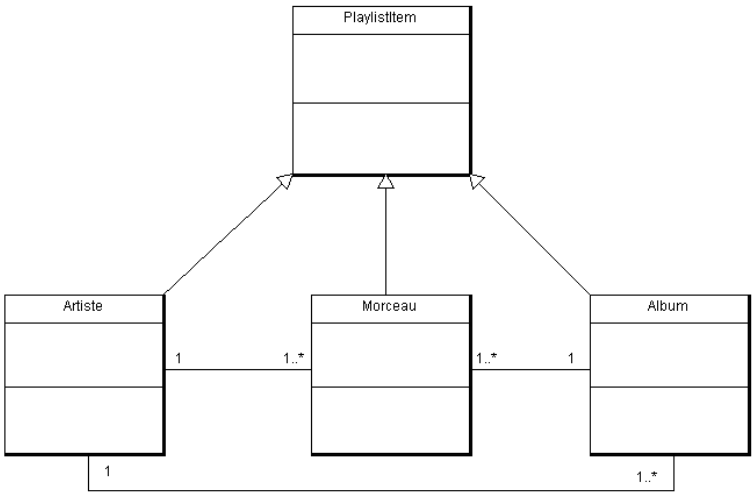

Figure 1.3. Diagramme de classes partiel

Les classes sont laissées volontairement vides pour plus de clarté. On observe alors que les trois classes Morceau, Artiste et Album héritent de la classe PlaylistItem, c'est-à-dire que ces trois classes sont construites à partir de la classe PlaylistItem qui leur sert de base de départ. Cela évite les redondances de codes. Ensuite, les liens entre Artiste, Morceau et Album sont naturellement de simples associations : à un Artiste donné correspond plusieurs Morceaux et plusieurs Albums.

Le but de toute cette réflexion préliminaire à la programmation est de clarifier toutes les relations entre les classes de sorte que lors de la programmation, la seule réflexion nécessaire soit celle liée directement au codage en lui-même et non sur toutes les interactions des différents objets.

Conclusion intermédiaire :

Ainsi, grâce à de nombreuses réunions de groupe et avec le tuteur scientifique du projet, nous avons pu rapidement cerner les propriétés attendues par le commanditaire. Une hiérarchie de ces fonctionnalités attendues a été créée afin de pouvoir faire plus facilement des choix quant à quelle fonction intégrer si le temps vient à nous manquer. Un cahier des charges a été rédigé rapidement après cette première phase. Une partie en est présentée dans la partie suivante.

Les outils très performants apportés par l'utilisation de la programmation orientée objet permettront de programmer de façon claire et efficace en découpant le code en classes hiérarchisées. L'intégration des algorithmes fournis par le laboratoire LIRIS n'en sera que plus facile.

II- Conception

1) Spécifications du cahier des charges

À l'aide de la fiche de présentation du projet, et après de nombreuses réunions avec l'équipe de projet et le tuteur scientifique Emmanuel Dellandréa, un cahier des charges a pu être rédigé assez rapidement. En voici un extrait (la version complète étant disponible en *annexe 1*) :

- **Objectifs :**

Ce logiciel a pour but de simplifier l'organisation des fichiers multimédias. Mais il permettra aussi de visualiser ces fichiers, de pouvoir faire des recherches intelligentes ou encore de les associer entre eux. Le tout dans une certaine simplicité pour l'utilisateur.

- **Contraintes :**

a. Définition fonctionnelle

Les données d'entrée du logiciel sont :
- Fichier de type musique, de différents formats ;
- Fichier de type image, de différents formats ;
- Des algorithmes fournis par le laboratoire LIRIS.

Les données de sorties du logiciel sont :
- Lecture des fichiers ;
- Sorties des algorithmes.

b. Condition d'utilisation

Le logiciel peut être utilisé par tout utilisateur désirant avoir un logiciel de lecture de fichiers multimédias. Il sera dans un premier temps utilisé au service du commanditaire, c'est-à-dire le laboratoire LIRIS.

c. Définition de l'environnement.

L'interface homme-machine doit respecter des facteurs de confidentialité (les algorithmes ne peuvent être lus par l'utilisateur, les algorithmes étant la propriété du laboratoire LIRIS), de maniabilité (usage intuitif du logiciel, exploitabilité…) et de communicabilité. Le logiciel doit être réutilisable d'année en année, plusieurs fois par jours si nécessaire. Le logiciel pourrait être accessible via une connexion internet depuis l'Ecole Centrale de Lyon et cela de n'importe quel navigateur ou OS. Le logiciel doit être rapide pour assurer le confort des utilisateurs (efficacité mémoire et de temps d'exécution).

d. Définition

Le logiciel devra être utilisable le plus tôt possible. Il pourra au préalable être testé avec une petite base de données de fichiers multimédias.
Une notice d'utilisation devra être fournie.

- **Cahier des charges fonctionnel (non exhaustif) :**

a. Organiser

Le logiciel doit être capable d'organiser de manière autonome et efficace les données importées.

- *Pour les images :* classer principalement par nom, date. L'organisation sera enrichie par les algorithmes qui fourniront d'amples informations. Le logiciel doit pouvoir les prendre en compte.
- *Pour la musique :* utilisation du logiciel sous forme d'un Juke-box, c'est-à-dire un classement par genre, artiste, album, …

b. Rechercher

La recherche est une caractéristique importante du logiciel. La recherche musicale doit fonctionner comme un lecteur classique de musique, mais le logiciel doit pouvoir rechercher les fichiers qui sont susceptibles de nous

intéresser. Cette recherche sera faite grâce aux algorithmes. L'accent est porté sur ce nouveau système de recherche intelligent.

c. Intégrer les algorithmes

Le logiciel doit pouvoir intégrer les algorithmes du laboratoire LIRIS sans difficulté. Ils sont le cœur du logiciel.

d. Lier les fichiers multimédias

Lier les fichiers se fera selon deux procédés :

- *Automatique :* le logiciel liera les fichiers par les algorithmes.
- *Manuel :* l'utilisateur choisira lui-même de lier les fichiers entre eux.

Le taux de confiance sera plus fort pour les procédés manuels. Il faut différencier ces deux méthodes pour que l'utilisateur confirme ou non les liens faits par le logiciel.

L'utilisateur peut lier les fichiers de différentes manières :

- *Lier une image à une musique*

Comme une pochette d'album pour une musique. L'image n'a pas de lien sémantique avec la musique. Elle fait partie de ses métadonnées.

- *Lier des images à une musique selon un thème*

Ces images auront dans leurs données un pointeur vers la musique. Quand la musique est demandée et que l'on souhaite faire une visualisation de photos, ces photos seront automatiquement appelées. Cette association se fait autant automatiquement que manuellement.

- *Lier les images à une musique temporairement*

Possibilité à prendre. On peut donc prévoir la création d'une fenêtre spéciale.

e. Recevoir et donner des flux

f. Pouvoir « tagguer » les fichiers

L'utilisateur ainsi que le logiciel peuvent ajouter des données supplémentaires aux fichiers.

g. Lire tous les types de formats musique

h. Lire tous les types de formats image

i. Avoir une interface intuitive

Dans sa version finale, le logiciel devra être fluide, propre, et accueillant pour l'utilisateur. Une interface « avancée » pourra être exploitée. Un chapitre sera consacré à son interface.

j. Minimiser la mémoire

Pas de fuite de mémoire.

2) Phase de créativité : l'esquisse des interfaces

Une interface ergonomique et intuitive est l'un des points essentiels des demandes faites par le commanditaire. Avant de construire concrètement une interface à l'aide d'un support informatique, il est nécessaire d'en établir des croquis. Cela permet d'une part de laisser libre cours à la créativité et d'autre part d'avoir un modèle sur lequel s'appuyer.

On dispose de multiples outils pour élaborer ces esquisses.

La première méthode utilisée est en général la plus rudimentaire : le dessin sur papier. Elle a pour avantages d'être un moyen d'expression rapide et facile d'accès en vue de partager ses idées dans un cadre restreint, ici au sein des membres du groupe de projet. Elle permet également de mettre en forme toutes les idées rassemblées lors d'un brainstorming et d'en faire un premier tri. Dans un premier croquis de l'interface, on ne représente pas les barres de menu ou barres d'outils habituellement présentes dans une fenêtre principale mais simplement la structure de la

zone centrale. Elle se divise en trois parties : à gauche, un espace dédié à un type de médias, à droite, celui dédié à un autre et au centre toutes les actions que l'on peut effectuer en fonction des sélections des parties latérales. Le choix du type de médias se fait grâce un système d'onglets, par exemple un onglet Musique, Images, Vidéos.

Le dessin ne constitue cependant pas un support suffisamment clair et souple, ce dernier pouvant rapidement devenir illisible après de multiples modifications. La seconde méthode est donc de construire ces esquisses sur ordinateur. En organisant des captures d'écran d'éléments déjà existants, on obtient des rendus plus réalistes.

Deux versions d'interface sont ainsi conçues. La première reprend la structure de l'esquisse faite avec papier et crayon (voir figure 2.1.).

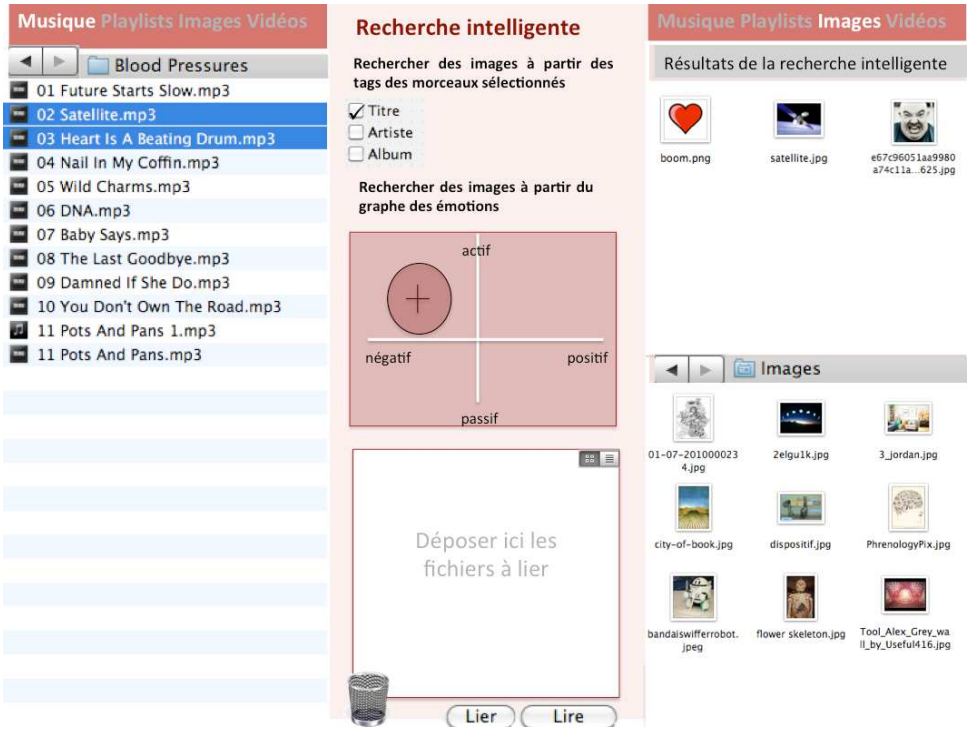

Figure 2.1. Première version de l'interface

Les outils dont on dispose dans la barre centrale sont précisés. C'est un « chaudron magique » dans lequel on met nos médias. Elle matérialise les liens que l'on peut effectuer entre les différents médias. On recense deux types de liens : les liens intelligents se basant sur les étiquettes ou tags ou sur les émotions et les liens dont le choix des médias liés est laissé libre à l'utilisateur. La recherche intelligente permet d'obtenir à partir d'une sélection de médias dans la barre latérale gauche des médias associés dans la barre latérale droite. En qui concerne le second type de lien, l'utilisateur dispose d'un espace dans lequel il peut déposer les fichiers souhaités.

Une seconde version de l'interface fait intervenir deux fenêtres principales, l'une consacrée au traitement d'un unique type de médias (lecture, ajout et modification de tags) que l'on désigne par mode « simple », l'autre étant une fenêtre de liens ou mode « lier ». La figure 2.2. montre une esquisse du premier mode. On retrouve l'utilisation d'onglets pour différencier le traitement des types de médias. Un simple clic sur le bouton « associer » situé en bas à droite enclenche le passage du mode simple au mode *lier*.

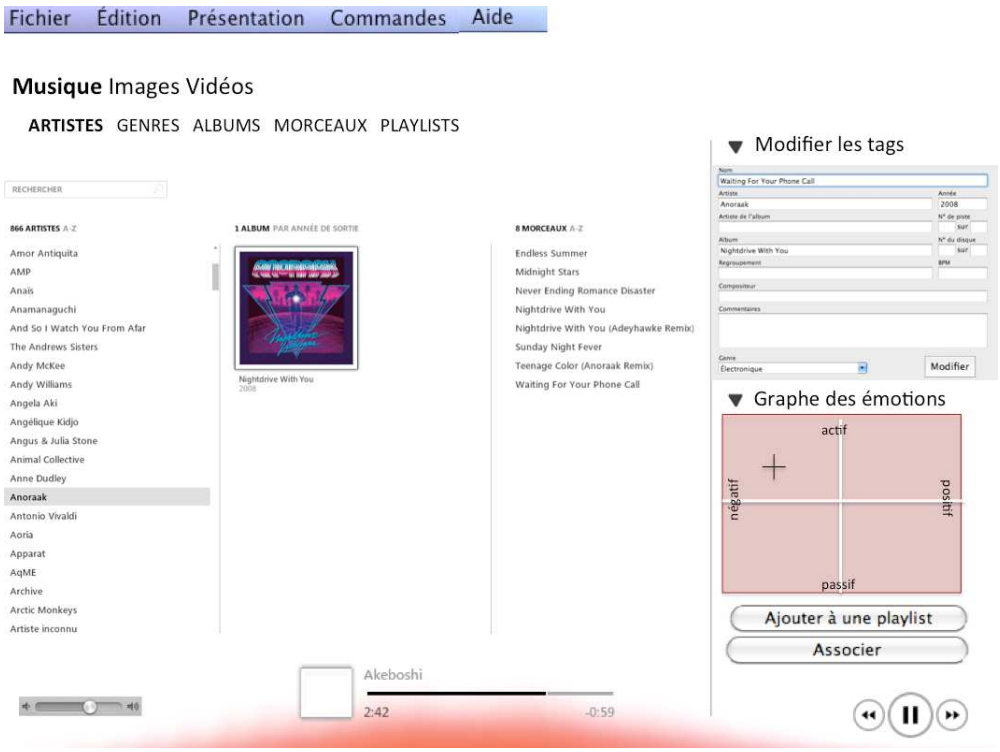

Figure 2.2. Deuxième version de l'interface en mode "simple"

La figure 2.3. résume le fonctionnement du mode *lier*. On retrouve une interface similaire à celle du mode précédent avec le même système d'onglets. Mais cette fois-ci, le choix de l'onglet sélectionné permet de déterminer avec quel type de médias on veut faire l'association. Le bouton « lier » permet d'enregistrer les liens effectués.

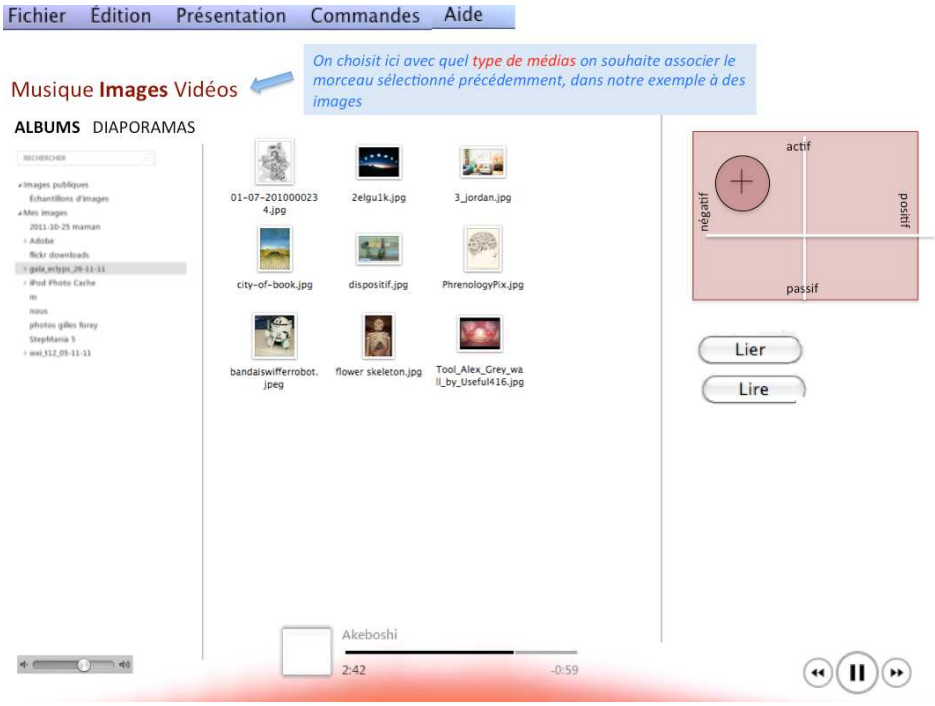

Figure 2.3. Deuxième version de l'interface en mode "lier"

La ressemblance des interfaces pour les deux modes a l'avantage d'accompagner le lecteur en conservant un même environnement. Mais elle présente deux faiblesses. D'abord, les liens entre médias ne sont pas assez explicites, la distinction entre les deux types de médias à lier est trop forte car deux fenêtres indépendantes sont utilisées. Ensuite, elle est sans doute trop complexe et pas suffisamment rapide d'utilisation.

De plus, le cadre d'utilisation du logiciel est restreint à la musique et aux images. La première version constituée de trois zones est donc plus adaptée dans la mesure où on dispose à gauche d'une barre dédiée à la musique, à droite celle dédiée aux images. Aussi, cela supprime le système d'onglets qui complexifiait l'utilisation du logiciel. Enfin, la présence des deux types de médias sur une même page reliés par la barre centrale, surnommée « marmite magique » met l'accent sur le fait qu'on puisse établir des liens entre eux.

3) Spécifications UML

Afin de concevoir la partie programmation, nous avons utilisé un outil reconnu tant par les chercheurs que les industriels pour la conception de projets informatiques. Il s'agit de l'outil UML (Unified Modeling Language) qui permet de concevoir l'ensemble des fonctionnalités du projet, de la liste des classes à l'architecture du logiciel. L'étude avec UML est présentée ci-dessous.

La modélisation UML permet de concevoir les aspects statiques et dynamiques du logiciel.

L'apport principal de la partie statique est le diagramme des classes. Celui-ci permet de lister les différentes classes à programmer ainsi que les liens entre elles. Cette liste est obtenue en recherchant l'ensemble des fonctions que le logiciel doit effectuer en considérant les besoins de l'utilisateur. Par exemple, un logiciel de lecture de musique doit pouvoir gérer l'affichage des données de la musique sélectionnée. Dans Tag'n'Link, il s'agit de la classe *MusiqueDetails*. L'ensemble des classes répertoriées lors de la phase de conception sont regroupées sous forme de diagrammes des classes dont celui de la fenêtre principal est présenté ci-dessous. Les autres sont disponibles dans la documentation qui sera fournie en même temps que le logiciel.

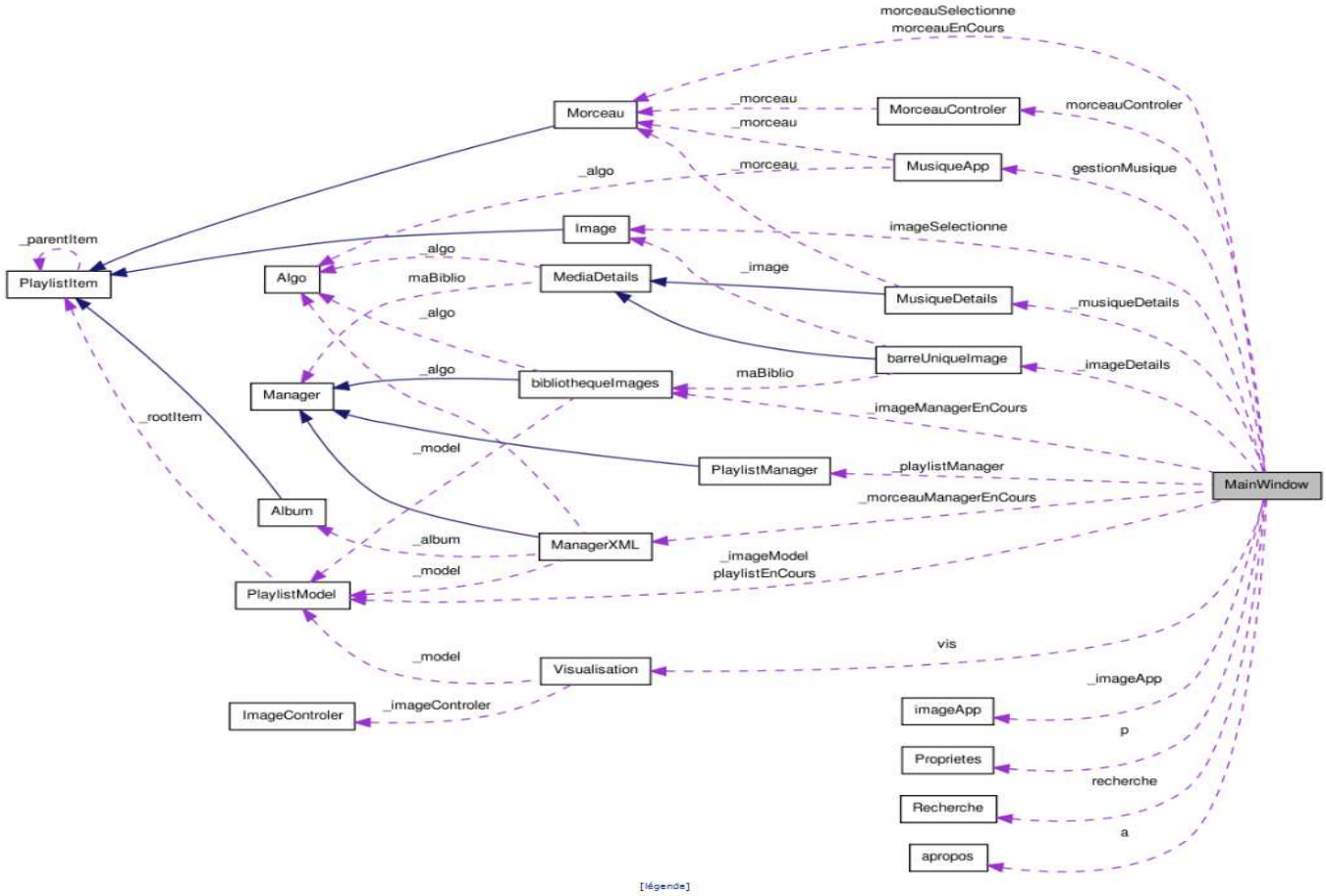

Figure 2.4. Diagramme des classes

Les flèches bleues désignent les relations d'héritage, les violettes en pointillés désignent les relations d'appartenance.

Pour la partie dynamique, on peut s'intéresser à d'autres diagrammes : les diagrammes états-transitions. Ces diagrammes ont pour objectif de présenter la suite logique des événements qui doivent s'effectuer pour chaque cas d'utilisation. Par exemple l'importation d'un dossier musical nécessite de récupérer le chemin du dossier, de récupérer chaque musique, de l'ouvrir, récupérer les tags, enregistrer la musique dans le fichier xml, le tout en affichant une barre de défilement pur indiquer l'avancement de l'importation à l'utilisateur. Ces études de cas sont essentielles pour n'oublier aucuns détails lors de la programmation. L'ensemble de ces diagrammes est disponible en *annexe 3*.

Il existe d'autres types de diagrammes plus ou moins utilisés pour concevoir un logiciel. Cependant, tous n'ont pas été présentés car ils ne présentent de l'intérêt que pour le travail de programmation.

Conclusion intermédiaire :

La phase de conception se décompose en trois étapes. Les spécifications du cahier des charges formulent de manière formelle les demandes du commanditaire. Ensuite, l'esquisse des interfaces permet de construire un premier squelette de l'interface. Puis, définissant les classes du programme et les relations entre elles, les spécifications UML sont des bases indispensables pour débuter la phase de programmation.

La réalisation du logiciel s'organise alors en trois axes. D'abord, il convient de préciser les moyens utilisés car la création d'un logiciel de cette complexité nécessite des outils additionnels au langage C++. Les différentes fonctionnalités du programme, telles que la gestion des données, leur lecture et leur mise en relation, sont codées de manière indépendante pour être finalement rassemblées au sein d'une interface ergonomique et intuitive.

III- Réalisation

1) Précision des moyens utilisés

La réalisation du logiciel peut se diviser en trois grandes parties plus ou moins indépendantes :

- l'écriture du code réalisant les fonctions du logiciel ;
- la réalisation de l'interface graphique ;
- la gestion des données de l'utilisateur.

Les deux derniers points nécessitent des outils plus riches que le langage C++ pour le premier et des outils extérieurs pour le second, mais qui s'adaptent au langage de programmation utilisé.

a) Création d'une interface avec Qt

En ce qui concerne l'écriture du code brut et la réalisation de l'interface graphique, l'utilisation du logiciel Qt s'impose de manière évidente. Il s'agit en effet d'un espace de travail (aussi appelé *framework*) orienté objet développé en C++. Qt rassemble un ensemble de fonctions utilitaires auxquelles on peut avoir accès sans avoir à les réécrire, en d'autres termes, c'est une bibliothèque logicielle. Elle offre entre autres des composantes d'interface graphique ou *widgets*. Ces derniers sont les éléments qui constituent généralement une interface de logiciel, comme les fenêtres, les boutons, les afficheurs de texte ou d'image ou encore les champs permettant à l'utilisateur d'entrer des données. QtCreator désigne le programme regroupant l'ensemble des outils fournis par la bibliothèque Qt, il s'agit d'un environnement de développement intégré (IDE en anglais).

Pour l'écriture du code en C++, on utilise le mode « Éditer » de QtCreator qui présente les modules habituels d'un IDE avec un éditeur de texte, un compilateur, un débogueur (voir figure 3.1.). C'est avec ce mode qu'on écrit les en-têtes et les fichiers sources du programme.

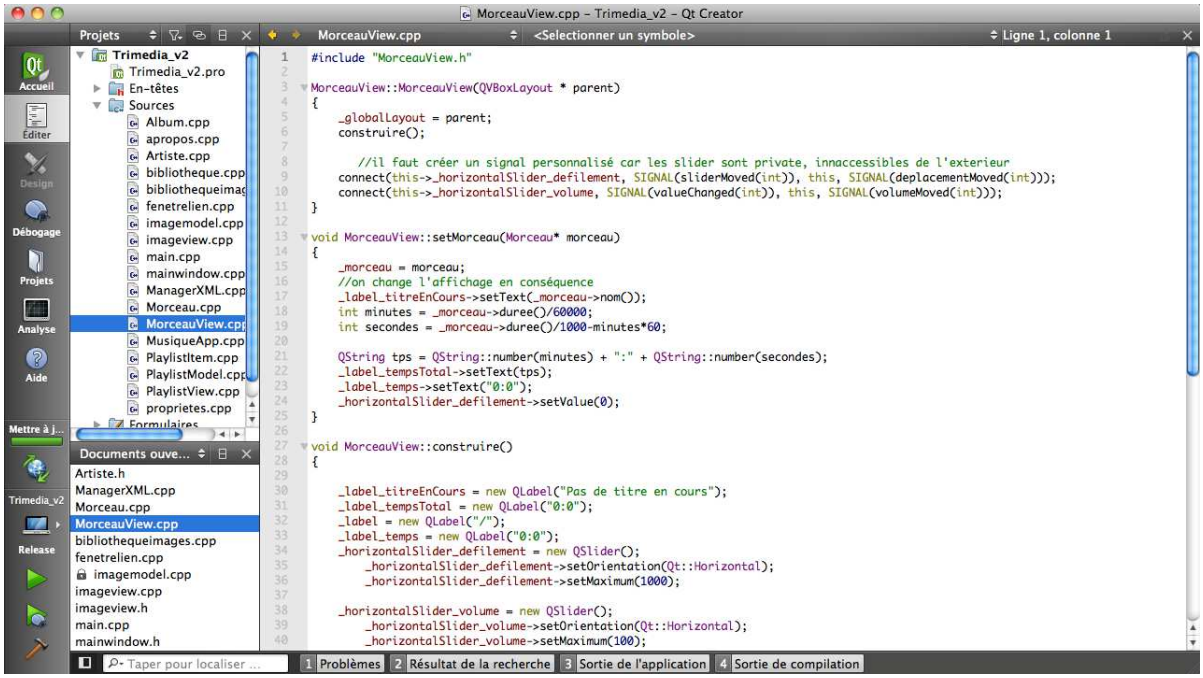

Figure 3.1. Capture d'écran du mode Editer de Qt Creator

Le mode « Design » fait la particularité de Qt par rapport aux IDE traditionnels dans le sens où il permet de placer rapidement et facilement les principaux éléments d'une fenêtre. La figure 3.2. montre comment fonctionne ce mode : à gauche, l'ensemble des widgets à notre disposition. Il suffit d'un simple glissé-déposé dans la partie

centrale pour les incorporer à l'interface. À droite figure la liste des objets, leurs propriétés et leur organisation les uns par rapport aux autres dans la fenêtre. On construit ainsi le squelette du logiciel qui doit accueillir les classes créées avec le mode « Éditer », rendant finalement le tout fonctionnel.

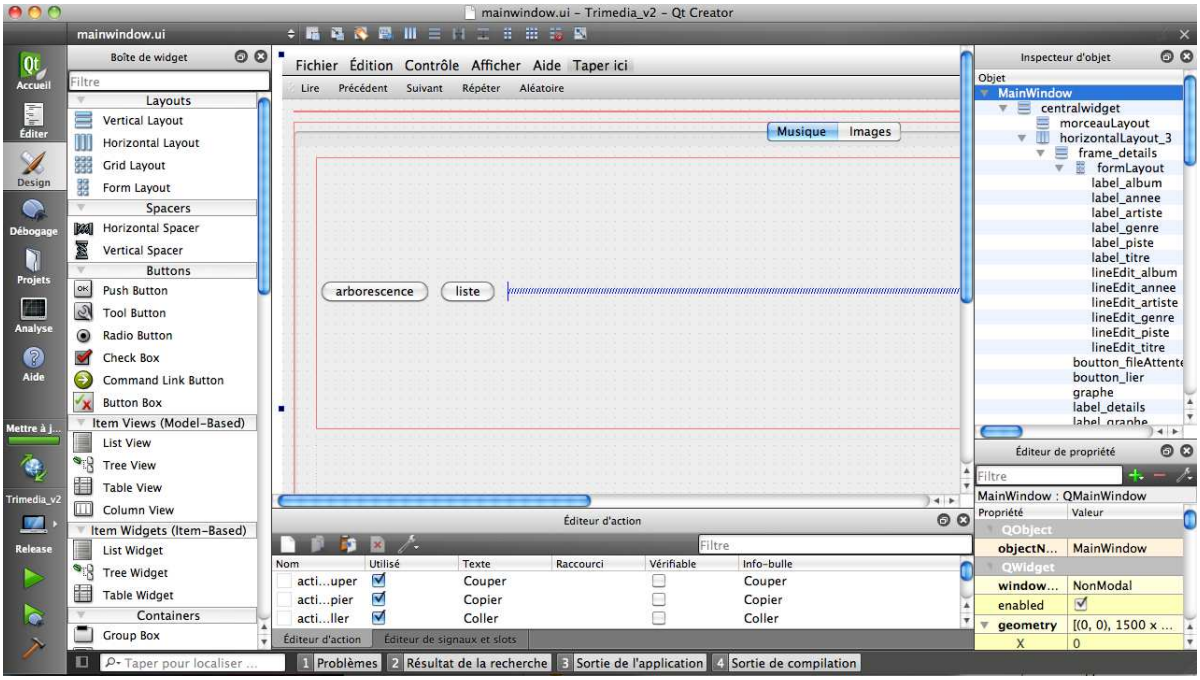

Figure 3.2. Capture d'écran du mode Design de Qt Creator

Qt présente de nombreux avantages. Tout d'abord, il permet une visualisation instantanée de l'interface que l'on construit, ce qui n'est pas le cas si on code directement. Ce point est très important dans le cadre de ce projet d'étude car l'accent est mis sur l'ergonomie de l'interface, il est dont primordial de pouvoir avoir un aperçu de l'interface créée sans perdre de temps à écrire des lignes de code pour enfin compiler et obtenir un résultat pas forcément convainquant. De plus, Qt est une bibliothèque multiplateforme. Cela rend les programmes conçus avec utilisables sur les trois systèmes d'exploitation les plus répandus aujourd'hui, à savoir Windows, Mac OS et Linux, qui est un aspect figurant dans le cahier des charges. Enfin, Qt met à disposition un très grand nombre d'objets dont le nom commence toujours par la lettre Q, et gère les signaux qu'ils peuvent émettre, par exemple un clic de la souris sur un bouton. Ceci permet de ne pas avoir un code linéaire et de gérer plusieurs actions à la fois. Par exemple, l'utilisateur peut regarder un diaporama d'images et écouter de la musique en même temps.

Ces multiples fonctions font que, pour un même rendu final, on peut utiliser des objets différents et mettre en place différentes stratégies. Une démarche continue d'autoformation est donc nécessaire ainsi qu'une bonne maîtrise de la documentation fournie avec la bibliothèque afin d'exploiter pleinement tous les outils proposés par Qt.

b) Gestion des données avec le langage XML

Le logiciel doit gérer les données de l'utilisateur et plus particulièrement des données modifiées car il peut ajouter des tags aux médias et créer des liens entre eux. Dupliquer tous les fichiers et y incorporer les ajouts faits par l'utilisateur est une méthode bien trop lourde et peu rapide. Un moyen de stocker toutes ces informations en occupant peu d'espace et de manière organisée est d'utiliser le langage XML. C'est un langage structuré autour d'unités sémantiques qui délimitent un ensemble, les balises, que l'on place à l'intérieur d'un fichier texte. Ces balises permettent de structurer le document et l'information qui y est relative est écrite à l'intérieur. Pour la gestion de la musique et des images, plusieurs fichiers XML sont nécessaires :

- un pour la bibliothèque de musiques ;
- un pour la bibliothèque d'images ;
- un gérant la liste des playlists existantes ;
- un pour chaque playlist créée par l'utilisateur ;

À titre d'exemple, les lignes suivantes présentent une partie de la structure du fichier XML utilisé pour la gestion des images :

```
<Dossier name="dossier">
        <Dossier name="sousDossier">
                <Image ac="-0,9" path="cheminImage.jpg" name="nom1" ap="-0,2"/>
                <Image ac="-0,3" path="cheminImage.jpg" name="nom2" ap="-0,1"/>
                <Image ac="-0,5" path="cheminImage.jpg" name="nom3" ap="-0,2"/>
                <Image ac="-0,1" path="cheminImage.jpg" name="nom4" ap="-0,1"/>
        </Dossier>
</Dossier>
```

Ces lignes sont facilement compréhensibles : la bibliothèque d'images représentée ici par le fichier XML contient deux images, la première « cityofbook » avec les tags « ville » et « livre » et la deuxième « dispositif » avec le tag « experience ». Chaque section du document est désignée sous le terme de « nœud » ou plus précisément « élément » qui est le type de nœud le plus utilisé ici. Par exemple, <Image> est le nœud père de <chemin> qui est son nœud fils. On peut ajouter autant de nœuds fils souhaités à un nœud père. Cette syntaxe a donc pour atout la possibilité d'ajout illimité d'images et de tags associés à chaque image.

Le fichier XML associé à la musique prend quant à lui la forme suivante :

```
<artist name="Artiste1">
      <album image="cheminPochette.jpg" name="nomMorceau" year="1998">
            <title ac="-0,5" ap="0,1"  tag="tag1;tag2;tag3" path="cheminMorceau.mp3"
      name="nomMorceau" track="3" duration="220604" />
            <title ac="-0,5" ap="0,1"  tag="tag1;tag2;tag3" path="cheminMorceau.mp3"
      name="nomMorceau" track="3" duration="220604" />
            <title ac="-0,5" ap="0,1"  tag="tag1;tag2;tag3" path="cheminMorceau.mp3"
      name="nomMorceau" track="3" duration="220604" />

      </album>
```

```
    <album>
        <title ac="-0,5" ap="0,1" tag="tag1;tag2;tag3" path="cheminMorceau.mp3"
    name="nomMorceau" track="3" duration="220604" />
    </album
</artiste>
```

Le fichier s'organise en trois balises: "artist", "album" et "title" contenant chacune des attributs. Par exemple, la balise "album" contient les attributs "name", "image" et "year" qui contiennent respectivement le nom, le chemin de la pochette de l'album et la date de parution de l'album.

Avec l'utilisation du langage XML, on stocke ainsi de manière transparente pour l'utilisateur autant d'informations qu'il le souhaite, tout en minimisant l'espace occupée sur le disque puisqu'il s'agit de simples fichiers texte. Ces informations sont modifiables à partir du logiciel et sont conservées entre chaque utilisation.

2) Codage des différentes fonctionnalités

a) Importation, gestion de données

Les documents XML permettent de stocker les informations concernant les bibliothèques de l'utilisateur à long terme. Toutefois, il est nécessaire d'utiliser un élément intermédiaire pour pouvoir manipuler le contenu des fichiers XML durant l'utilisation du logiciel. L'outil de programmation alors utilisé s'appelle un parseur. Il vérifie la cohérence du document XML en termes de syntaxe et de structure et transmet à l'application les informations utiles au traitement du document [1]. Différents types de parseurs existent. En particulier, DOM (*Document Object Model*) est une représentation objet d'un document XML, donc particulièrement adaptée au type de programmation choisie et disponible avec Qt au moyen de la classe QDomElement.

La classe Manager gère tous les accès aux fichiers XML et au disque dur. Son attribut principal est donc un objet QDomDocument. Cette classe possède notamment une méthode permettant d'ouvrir le fichier XML désiré et le transcrire sous la forme d'un QDomDocument. Une fois cette tâche préliminaire achevée, différentes méthodes permettent de manipuler les données du XML au moyen du DOM :

- ajouterTagXML(QString, QString) prenant en paramètre deux chaînes de caractères, le média et le tag à ajouter à ce dernier ;
- recupTags(QString) récupérant les tags associés au média indiqué en paramètre.

Les bibliothèques d'images et les bibliothèques de musique possèdent des structures différentes. En effet, les musiques s'organisent par artiste, puis par album alors que les images peuvent avoir des hiérarchisations multiples. Par exemple, les regroupements peuvent se faire par date, par lieu, par thème, etc. Pour cette raison, l'organisation des deux fichiers XML dédiés à la musique et aux images est différente. La classe Manager a donc deux classes filles : ManagerMusique et ManagerImage (voir figure 3.3.) dans lesquelles les fonctions citées précédemment sont implémentées en fonction de la structure des documents XML correspondants.

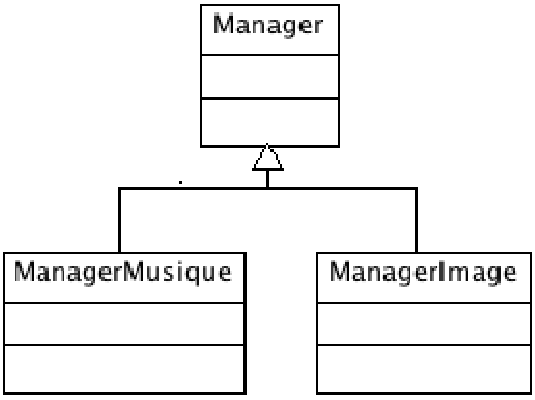

Figure 3.3. L'héritage de la classe Manager pour les classes ManagerImage et ManagerMusique

Les autres méthodes des classes ManagerMusique et ManagerImage sont spécifiques au traitement des médias concernés. Le ManagerMusique manipule des artistes, des albums et des morceaux représentés sous forme de classes contenant chacune leurs données relatives sous forme d'énumération. Par exemple, la classe Artiste contient simplement le nom de l'artiste alors que Morceau a un nom, une durée, un numéro de piste et une adresse sur le disque dur. Le ManagerImage, lui, fonctionne par dossiers préalablement définis par l'utilisateur. Ainsi, pour les images, les classes DossierImage et Image sont le pendant des classes Artiste, Album et Morceau pour la musique.

Au sein du programme, les classes ManagerMusique et ManagerImage sont en fait nommées respectivement ManagerXML et bibliothequeImages. Ces nouveaux noms sont employés afin d'être plus clair et de souligner la relation d'héritage.

Le logiciel propose d'importer un seul fichier à la fois ou un dossier contenant plusieurs fichiers. La procédure d'importation d'un dossier distingue celle d'un dossier d'images de celle d'un dossier de musique comme le montre la figure 3.4. Au cours de cette procédure, l'ensemble des fichiers présents dans le dossier à importer sont listés au moyen de la classe QDirIterator, qui navigue dans des dossiers. Lors de ce listage, un filtre est appliqué afin de conserver uniquement les fichiers dont le format est valide. Par exemple, lors de l'importation d'un dossier de musique, les formats valides sont .mp3, .wma, .wav, etc. et lors de l'importation d'un dossier d'images, .jpeg, .jpg, .png, .gif, etc.

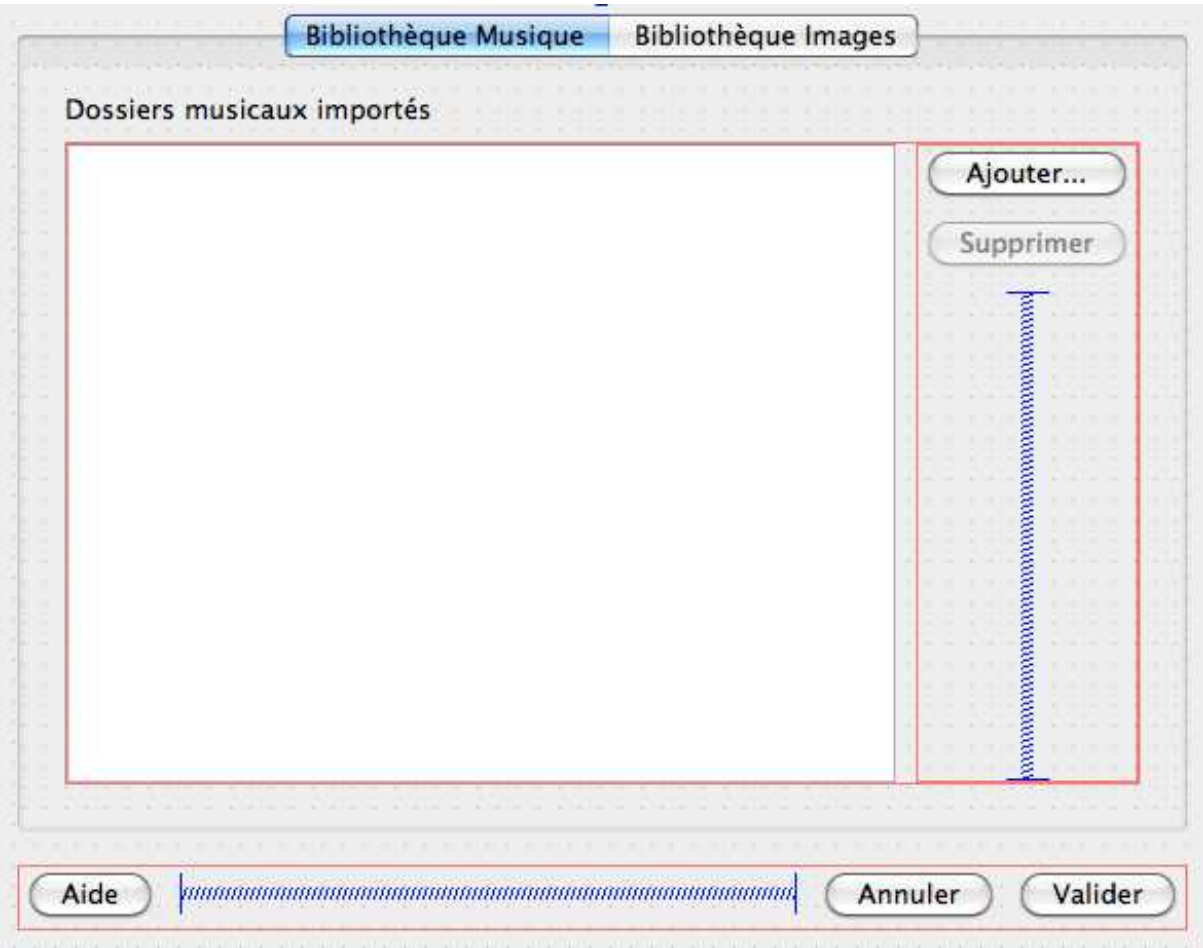

Figure 3.4. Boîte de dialogue d'importation de dossier dans QtDesigner

b) Lecture de musique

L'utilisation d'une bibliothèque supplémentaire s'avère indispensable pour créer un lecteur de musique comprenant les fonctionnalités basiques play, pause, suivant, précédent, répéter et la gestion du volume.

FMOD, récemment rebaptisée FMOD Ex, est la bibliothèque employée pour la gestion du son. Outre le fait d'être réputée, elle est multiplateforme et comprend des outils puissants avec une prise en main relativement simple et rapide. Elle supporte également un large panel de types de fichiers audio.

Au sein du programme, deux classes commandent la lecture de musique :

- ImageApp manipule des objets fournis par FMOD, les fonctions de base d'un lecteur y sont implémentées. Ainsi, un morceau est joué à partir de son adresse sur le disque dur. ImageApp gère également les événements, c'est-à-dire le temps écoulé du morceau joué, par l'intermédiaire d'un QTimer actualisant l'écoulement du temps à intervalle régulier.

- MorceauControler fournit l'interface pour la lecture de musique conçue à partir du module Design de Qt (voir figure 3.5.).

Figure 3.5. Contrôle de la musique jouée

Le titre du morceau, l'artiste ainsi que les temps total et écoulé s'affichent. Plusieurs boutons permettent de mettre en play/pause, gèrent les fonctions précédent/suivant et répéter. Comme dans la majorité des lecteurs de musique actuels, une barre de défilement interactive donne accès au temps écoulé grâce à un curseur pouvant être déplacé par l'utilisateur pour évoluer dans le morceau joué. La barre de volume permet le contrôle aisé de ce dernier.

Enfin, FMOD permet de récupérer les informations associées à un morceau donné. Cette récupération se fait au moment de l'importation et fait appel à trois classes énoncées précédemment : MusiqueApp, Morceau et ManagerMusique.

Dans un premier temps, la méthode bool recupererTags() de la classe MusiqueApp récupère les tags classiques contenus dans le fichier audio traité : le nom de l'artiste, le nom de l'album, le titre, la durée, l'année de parution, le numéro de piste et l'adresse sur le disque dur. Cette méthode est basée sur les outils disponibles avec FMOD et renvoie "vrai" si la récupération des informations s'est passée sans encombre, "faux" sinon. Les sources d'erreur possibles sont liées principalement au fait que le format d'un fichier à importer n'est pas supporté par FMOD, soit parce que ce n'est pas un fichier audio, soit parce que c'est un format spécifique à un logiciel donné, par exemple le format .m4a de iTunes.

De plus, il est possible qu'un fichier ne contienne pas tout ou partie des informations requises. Dans ces cas-ci, des valeurs des bases sont données aux différents tags qui nous intéressent. Ainsi, si l'année ou le numéro de piste sont absents, ils sont fixés à -1, un album dont le nom est inconnu est classiquement nommé « Album Inconnu », et les artistes inconnus sont numérotés. L'entier en paramètre de la fonction add correspond justement à cette numérotation. Le même principe de numérotation est utilisé pour les pistes inconnues d'un album donné.

Cette procédure décrit l'importation d'un seul morceau. Pour importer un dossier, ses éléments sont d'abord listés. Ils peuvent alors être des fichiers ou des sous-dossiers. Dans le premier cas, la fonction d'importation de morceau décrite plus haut est simplement appelée. Dans le second cas, le processus d'importation d'un dossier est appelé récursivement. La fonction consacrée à l'importation d'un dossier a pour prototype int importationDossier(QString, Progressbar*, int), le premier argument correspond au chemin sur le disque du dossier à importer, les deux suivants ont pour rôle la mise en place d'une barre de progression lors de l'importation.

La classe Progressbar réalise cette fonction en gérant l'affichage et la mise à jour d'une barre de progression lors de l'importation. L'entier en entrée de la méthode importationDossier correspond ainsi au nombre de fichiers importés et c'est aussi celui renvoyé par cette méthode pour que les appels récursifs prennent en compte l'avancement total.

c) Visualisation d'images

Du point de vue de la structuration du code, la visualisation des images se calque sur la lecture de la musique. Encore une fois, deux classes interviennent, la première permettant la lecture et la seconde l'interaction avec l'utilisateur. Ainsi, tout comme la classe MusiqueApp permet la lecture de la musique, la classe ImageApp permet la visualisation d'une image, avec la possibilité d'agrandir ou de réduire la taille de l'image visualisée.

ImageApp comprend deux boutons correspondant aux fonctions de zoom in et out. L'espace de visualisation s'appuie sur les classes suivantes fournies par : Qt QGraphicsScene, le contenant et QGraphicsView, le contenu. La classe QGraphicsScene fournit une surface pour gérer un grand nombre d'éléments graphiques 2D [2]. Dans le cas présent, il suffit de pouvoir gérer un seul élément 2D, à savoir une image. QGraphicsView affiche le contenu d'une scène. Cette classe hérite de QAbstractScrollArea (zones de défilement), ce qui facilite grandement la gestion des barres de défilement lors des opérations de zoom. L'utilisation couplée d'une scène et d'une vue permet d'intégrer ImageApp à la fois pour un simple aperçu des images et pour une visualisation plein écran.

Au sein de la classe ImageApp, l'image visualisée est stockée dans deux objets de type QPixmap et QImage qui sont deux classes traitant les images. QPixmap est optimisée pour montrer des images à l'écran, c'est donc un QPixmap qui est ajouté à la scène pour être ensuite visualisée dans la vue. Quant à QImage, c'est une classe conçue pour la manipulation et l'accès directs aux pixels d'une image. C'est donc une instance de QImage qui est utilisée pour modifier la taille d'affichage, évitant ainsi la dégradation de l'image lors des zooms.

De même que MorceauControler, ImageControler bénéficie d'une interface avec des boutons suivant, précédent et répéter. La combinaison de la scène d'ImageApp avec ImageControler donne naissance à une troisième classe, Visualisation utilisée pour créer des diaporamas d'images. Le curseur de la souris est caché à moins que l'utilisateur déplace ce dernier. De plus, un QTimer modifie l'image affichée toutes les 2 secondes. La figure 3.6. illustre cette fenêtre de visualisation.

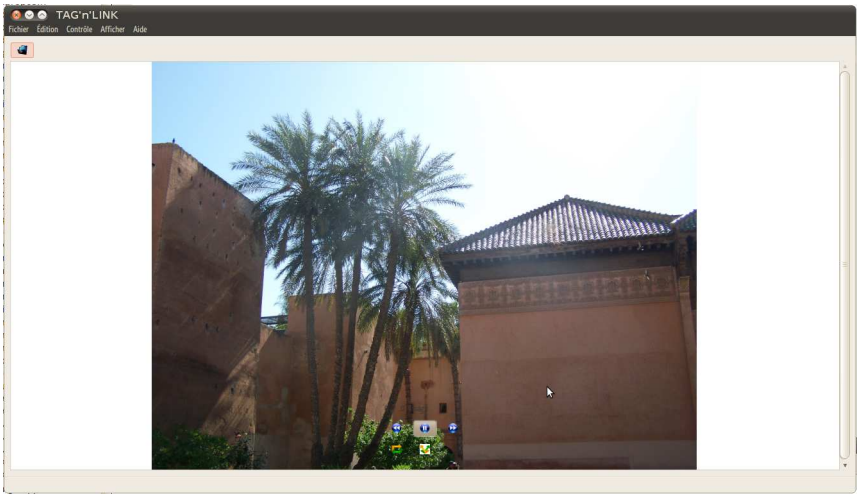

Figure 3.6. Fenêtre de visualisation

d) Graphe des émotions

À un ensemble de médias donné correspond un ensemble de points dans le plan des émotions. Pour rappel, l'axe des abscisses représente l'appréciation et l'axe des ordonnées l'activité.

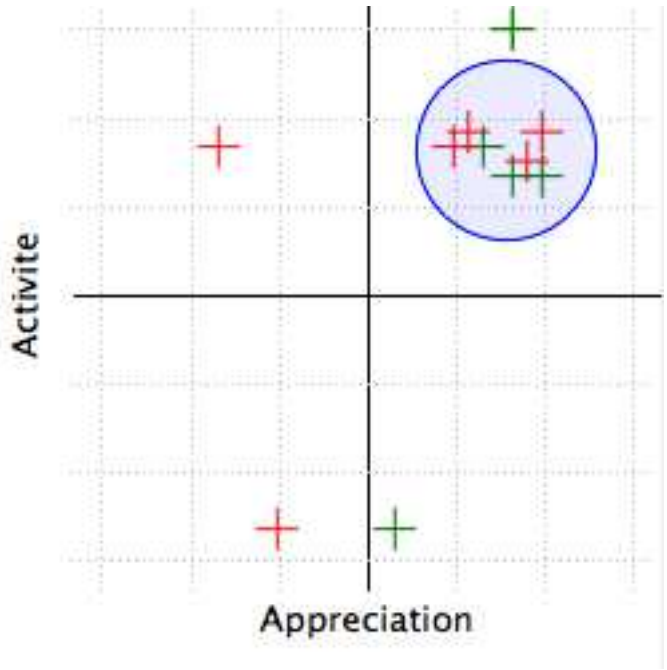

Figure 3.7. Aperçu du graphe des émotions

D'une part, le graphe des émotions permet de représenter tous ces points pour les deux types de médias intervenant dans le logiciel. On les différencie par des couleurs, en vert la musique et en rouge les images. Cette vision permet d'avoir un premier aperçu des médias proches sur le plan émotionnel. Par exemple, sur la figure 3.7., certains morceaux et images sont regroupés dans le quart de plan supérieur droit, indiquant qu'ils procurent de la joie.

D'autre part, l'utilisateur peut interagir avec le graphe. Lorsqu'il presse le bouton droit de la souris puis fait glisser le curseur, il dessine un cercle pour désigner les médias qu'il souhaite sélectionner. Cette sélection permet ainsi de créer un diaporama musical, et dans l'exemple de la figure 3.7., un diaporama joyeux.

D'un point de vue programmation, la classe GrapheEmotions gère ces fonctionnalités. Son attribut principal est un QCustomPlot qui permet entre autres de tracer des courbes dans un repère. Cette classe n'est pas fournie par la bibliothèque Qt, elle est disponible gratuitement sur le site Works Like Clock Work qui fournit des outils simples à manier permettant aux programmeurs de gagner beaucoup de temps [3]. La méthode placerPoint(double appreciation, double activite, int media) permet de placer un point de coordonnées (appreciation, activite) dans le plan des émotions, associé à un média. Le dernier paramètre entier est nul lorsqu'il s'agit d'une musique, et égal à 1 pour une image. La méthode setCercle(double centrex, double centrey, double rayon) trace un cercle dont les paramètres sont détectés à partir de la position du curseur de la souris au moment où le bouton droit est appuyé puis relâché.

e) Intégration des algorithmes du laboratoire LIRIS

L'algorithme développé par le LIRIS détecte des concepts dans un média donné et lui attribue une position dans le plan des émotions. Concrètement, il s'agit d'un exécutable qui prend en paramètre l'adresse d'un fichier de type son ou image et qui renvoie un fichier texte comprenant la liste des concepts détectés suivie des coordonnées dans le graphe des émotions.

Les tâches successives effectuées par le logiciel pour intégrer l'algorithme sont résumées dans la figure ci-dessous :

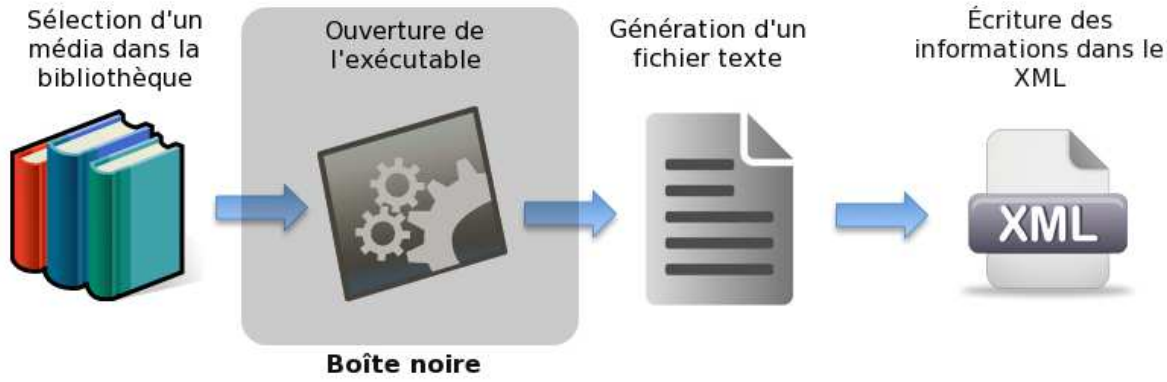

Figure 3.8. Opérations successives effectuées pour l'intégration de l'algorithme

La classe Algo prend en charge ces diverses opérations. Tout d'abord, l'ouverture de l'exécutable est réalisée par l'intermédiaire d'un QProcess, classe utilisée pour lancer des programmes externes et communiquer avec eux. En fournissant le nom de l'exécutable et la liste des arguments du programme, un processus est lancé. Une fois le fichier texte généré, le programme l'ouvre par l'intermédiaire d'un QFile et le lit grâce à un QTextStream qui fournit une interface simple pour la lecture et l'écriture de fichiers texte. Enfin, il convient de transférer les informations récupérées dans les fichiers XML afin de permettre l'affichage de la liste des concepts et le point correspondant dans le graphe des émotions pour un média donné d'une part, et éviter de devoir relancer l'exécutable pour un même fichier entre deux utilisations du logiciel d'autre part.

L'exécutable n'étant pas fourni au cours de la réalisation du projet, les méthodes de la classe Algo sont implémentées de manière factice afin de pouvoir effectuer des tests. Ainsi, une liste de tags est prédéfinie dans la classe Algo et les points dans le graphe des émotions sont générés aléatoirement.

3) Incorporation des fonctionnalités dans une interface intuitive

a) Affichage et recherche des données avec l'architecture modèle-vue

L'ensemble des fonctionnalités exposées précédemment manipule un nombre important de données : les différents médias et leurs métadonnées. Par soucis de clarté et d'organisation du code, il convient donc de différencier ces données de leur affichage. L'architecture Modèle Vue Contrôleur (MVC) est une technique de programmation basée sur cette distinction.

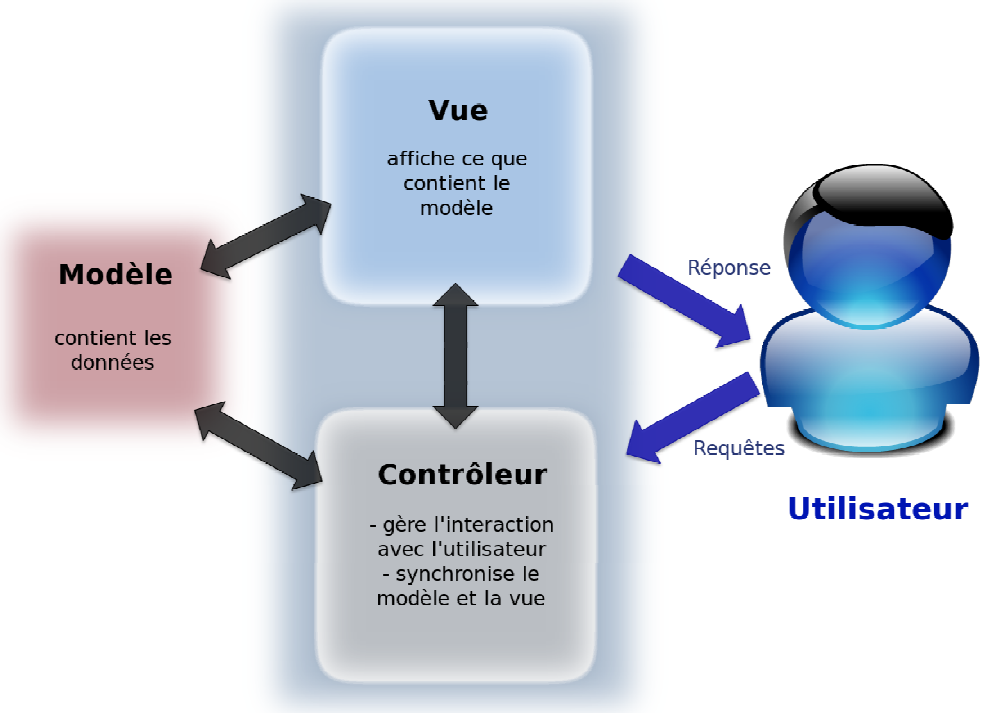

Figure 3.9. L'architecture MVC et son interface avec l'utilisateur

Comme représenté sur la figure 3.9., elle propose de séparer les éléments d'un programme en trois parties interconnectées :

- le **modèle**, contenant les données. Ici, les modèles sont des listes d'éléments (morceaux, albums et artistes pour la musique ; dossiers et images pour les images) organisées sous forme d'arbre (figure 3.10.).

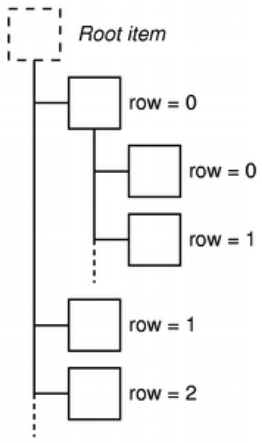

Figure 3.10. Modèle organisé sous forme d'arbre [2]

- La **vue**, affichant le modèle. Les modèles ayant une structure d'arbre, les vues associées sont aussi sous forme d'arbre.

- Le **contrôleur**, partie « intelligente », définissant comment l'interface réagit aux entrées de l'utilisateur. Qt combine le contrôleur et la vue. Ainsi, la séparation entre la façon dont sont stockées les données et la façon dont elles sont affichées est conservée mais l'architecte modèle/vue fournit des composants logiciels plus simples.

Affichage des bibliothèques et des détails d'un média :

La particularité de la structure des données manipulées impose de créer un modèle personnalisé. Ce dernier prend donc la forme d'une classe nommée PlaylistModel héritant d'une classe de modèle déjà existant sous Qt. Conceptuellement, la classe PlaylistModel représente une liste de médias hiérarchisés. Ces listes peuvent être la bibliothèque entière ou une playlist créée par l'utilisateur. Un objet PlaylistModel peut donc être utilisée autant pour les musiques que pour les images. Dans cette classe figurent des méthodes d'accès aux différents éléments de la playlist considérée. Ces éléments sont modélisés par la classe PlaylistItem, qui est la « brique » de base de PlaylistModel. Un objet PlaylistItem peut être une instance des classes Morceau, Artiste, Album, Image ou DossierImage citées dans la sous-partie précédente.

La figure 3.11. résume les relations entre les différentes classes détaillées ci-dessus. Ainsi, PlaylistModel est un agrégat de PlaylistItem et Morceau, Artiste, Album, Image et DossierImage héritent de PlaylistItem.

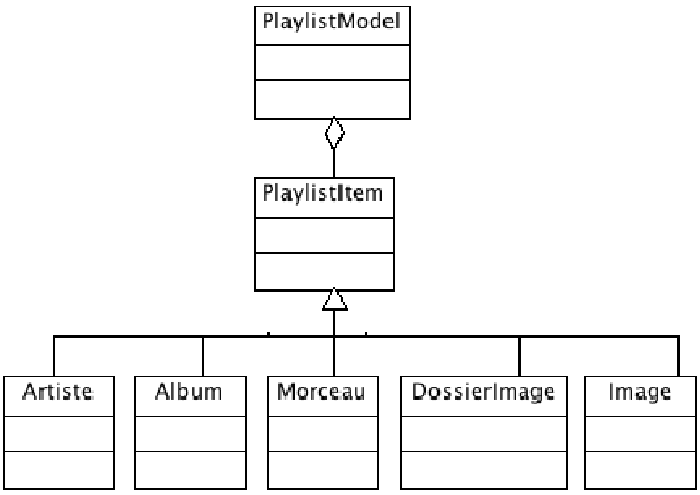

Figure 3.11. Diagramme des classes lié au modèle

Les modèles étant correctement établis, des QTreeView sont les vues permettant d'afficher le contenu de ces modèles sous forme d'arbre. L'utilisateur visualise l'ensemble de ces bibliothèques et un simple clic sur un élément morceau ou image de ce widget vue donne accès aux détails de l'élément sélectionné.

La classe MediaDetails réalise cette fonction. Un objet de cette classe est construit à partir d'un objet Manager (Images ou Musique), ce qui permet la lecture et l'écriture dans le fichier XML correspondant. MediaDetails possède des classes filles MusiqueDetails et ImageDetails (appelée dans le code barreUniqueImage) qui gèrent en plus l'affichage propre à chaque média. Comme visible sur les figures 3.12. et 3.13., l'espace où figure les détails d'un media possède plusieurs zones :

- une zone pour l'affichage des informations. Par exemple, pour un morceau, il s'agit du nom du morceau, de l'album, de l'artiste, la pochette de l'album, etc.
- une zone pour l'affichage des tags intelligents, c'est-à-dire ceux fournis par l'algorithme du LIRIS
- une zone pour l'affichage et l'édition des tags personnalisés avec notamment une zone de texte pour la saisie de nouveaux tags et deux boutons pour l'ajout et la suppression de tags.

Figure 3.12. Détails d'un morceau

Figure 3.13. Détails d'une image

Recherche des données :

Un champ de texte permet à l'utilisateur de faire une recherche globale sur l'ensemble des musiques et des images en même temps. Une recherche peut porter sur toutes les informations liées à un média. Par exemple, si l'utilisateur recherche un album de musique en particulier, il lui suffit d'entrer une partie du nom de l'album et l'ensemble des morceaux à l'intérieur de cet album s'affiche. Il peut aussi effectuer une recherche à partir des tags personnalisés et intelligents. La fonction de recherche s'appuie donc sur la lecture des fichiers XML gérant les bibliothèques car c'est là que sont stockées toutes les informations.

D'un point de vue codage, c'est la raison pour laquelle un objet Manager (Image ou Musique) est utilisé avec sa méthode getModel. Cette dernière prend en paramètre la chaîne de caractères entrée par l'utilisateur dans le champ de recherche et renvoie un playlistModel filtré qui contient le résultat de la recherche.

Au sein de la méthode getModel, un algorithme compact appelant tour à tour différentes méthodes permet un scannage rapide des bibliothèques. En effet, la solution de facilité qui consiste à accumuler des boucles parcourant les fichiers XML à la recherche du mot souhaité est peu optimisée et trop longue.

Dans la suite, l'algorithme de recherche employé est explicité pour la recherche dans la bibliothèque de musique, celle dans la bibliothèque d'images est basée sur le même principe. L'algorithme va scanner l'ensemble des artistes, albums et morceaux. Si un artiste vérifie la recherche, on conserve l'ensemble des albums et morceaux associés. Si un album vérifie la recherche, de même on conserve l'ensemble des musiques associées. Enfin si ni l'album et l'artiste ne vérifient la recherche, on ne conserve que les morceaux dont les tags vérifient la recherche.

Un algorithme simple a été utilisé pour créer le model filtré. Cependant, cette simplicité peut présenter des incohérences comme la présence d'un artiste avec un album vide. C'est pourquoi une méthode récursive (qui s'appelle elle-même) permet le parcours du modèle pour supprimer les artistes et les albums ne contenant pas de morceaux.

b) Création de liens entre médias

Le logiciel Tag'n'Link se démarque des logiciels de musique classiques par sa possibilité de créer des liens entre les différents médias. Cet atout lui confère la possibilité d'avoir une cohérence entre la musique écoutée et le diaporama visualisé.

Les liens avec Tag'n'Link sont de deux types : Les liens entre média de même nature et les liens entre médias de nature différentes.

Liens entre des médias de natures identiques : Il s'agit de la possibilité de créer des playlists et des diaporamas. L'objectif est d'offrir à l'utilisateur la possibilité de rassembler des musiques ou des images selon certains critères. Ces critères peuvent être définis grâce à plusieurs outils :
La barre de recherche qui permet de sélectionner les média qui contiennent un mot donné dans un de leurs tags ;
- Le système des tags qui permet de mettre de définir, manuellement ou automatiquement, des caractéristiques à un média ;
- Le système de sélection de musiques ou d'images pour créer une nouvelle liste ;
- Enfin, le graphe des émotions qui permet de sélectionner les médias suivant des critères d'activité/passivité et de positivité/négativité

Liens entre médias de natures différentes : il s'agit de créer un lien entre une playlist et un diaporama. Ainsi l'utilisateur a la possibilité de lire un diaporama en écoutant une musique qui y est liée et vice versa.
Un des moyens qui à été mis en place pour pouvoir lier les playlists et diaporamas entre eux réside dans la mise en place des outils de création de playlist et de diaporamas. En effet, ceux -ci ont été conçus pour agir en même temps

sur les musiques et les images. Ainsi l'utilisateur créé naturellement des liens entre les musiques et les images, même s'il n'en a pas l'intention au premier abord.
L'utilisateur a aussi la possibilité de lier manuellement ou automatiquement une playlist et un diaporama.
Il est enfin toujours proposé à l'utilisateur lors du lancement d'une playlist s'il veut lancer en même temps un diaporama automatique ou déjà lié manuellement et vice versa.

Grâce à l'ensemble de ces moyens, l'utilisateur est donc à tout moment capable de manière intuitive, de lier les médias entre eux. De plus, la forme du logiciel le pousse à utiliser ces moyens de liaison grâce à une interface intuitive et des outils s'appliquant aux deux types de médias à la fois. Ces caractéristiques, à l'origine du nom de Tag'n'Link, en font un lecteur proche de l'utilisateur en répondant à ses besoins de mélanger les médias.

c) Fenêtre principale

La fenêtre Principale est la fenêtre de gestion des différents fichiers multimédias. Cette fenêtre contient de nombreuses informations, cependant, dans une nécessité d'intuitivité pour l'utilisateur, cette fenêtre est conçue de manière intelligente. Une capture d'écran est disponible figure 3.14.

Figure 3.14. Fenêtre principale sous Linux

La fenêtre principale peut être découpée en plusieurs parties détaillées dans la suite.

Musiques :

Cette partie, située à gauche, permet d'afficher la liste des musiques importées depuis l'ordinateur, de visualiser les listes de lecture et de pouvoir sélectionner un morceau en particulier.

La visualisation des morceaux est sous forme d'arborescence pour plus de clarté pour les utilisateurs. Ainsi, les musiques sont triées d'abord par artiste puis par album et enfin par nom comme le montre la figure 3.15. Lors d'un double-clic sur un morceau sélectionné, la musique se met en route automatiquement. Il est également possible de naviguer dans la bibliothèque des musiques tout en écoutant un morceau.

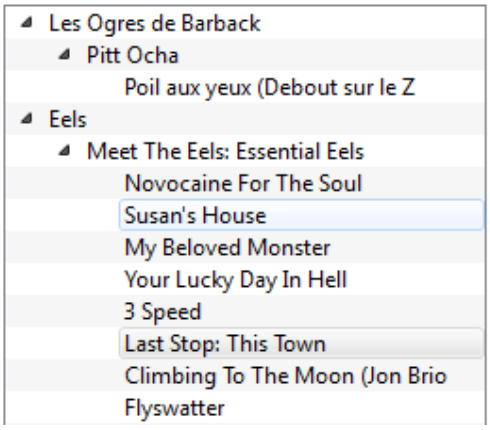

Figure 3.15. Exemple d'arborescence pour la musique

Images :

Cette partie se situe dans la partie droite de la fenêtre. Tout comme la partie musique, elle est construite sous forme d'arborescence (figure 3.16). Ainsi, l'utilisateur peut retrouver la même forme pour les deux formats multimédias, rendant le logiciel intuitif et son utilisation cohérente. Lors d'un double-clic, cette fois ci, c'est un diaporama d'images qui apparait.

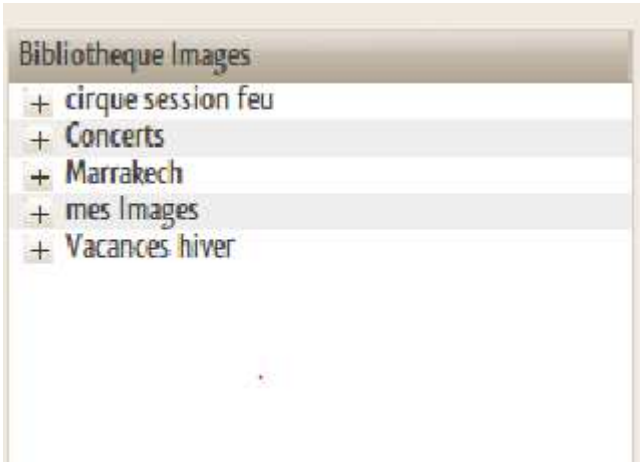

Figure 3.16. Exemple d'arborescence pour les images

Lecture :

La lecture d'une musique et la lecture d'une image est effectuée de manière distincte. C'est pourquoi il est possible de faire défiler les musiques grâce à un mini lecteur intégré à la fenêtre, au milieu en bas, et de faire défiler les images dans un onglet prévu à cet effet.

Détails :

Cette dernière partie se trouve au centre de la fenêtre. La figure 3.17. montre une capture d'écran de cette partie. Elle donne accès aux informations d'un média. Une partie est réservée aux tags donnés par l'utilisateur et aux tags automatiques créés à partir des algorithmes du LIRIS. Une zone est réservée au graphe des émotions. Les points en vert représentent des musiques et ceux en rouge des images.

Enfin, la zone en bas à droite permet de créer des liens entre médias.

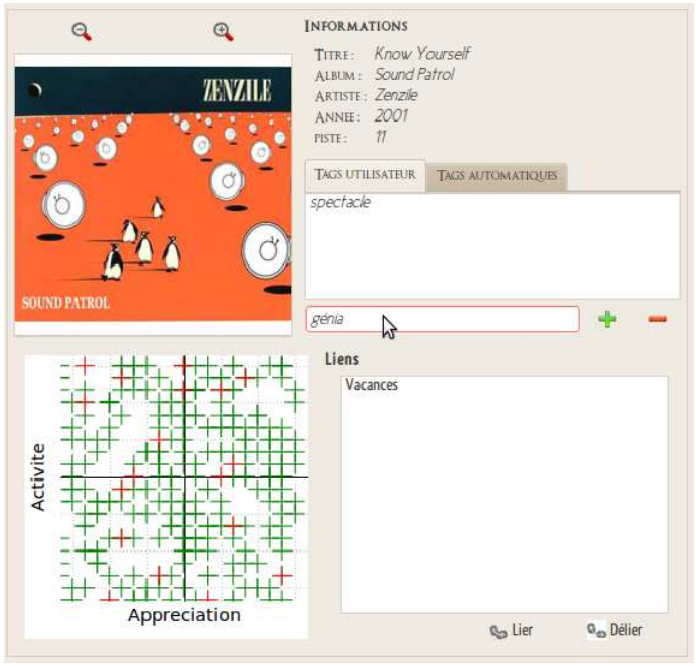

Figure 3.17. Zone dédiée aux détails, au graphe des émotions, à l'ajout de tags et de liens

Notons que sur cette partie il est possible d'interagir avec le graphe. En effet, il est possible de créer un cercle autour d'une musique, ayant pour effet de sélectionner toutes les musiques et images proche de cette musique « émotionnellement » parlant, plus au moins éloigné, selon la taille du cercle.

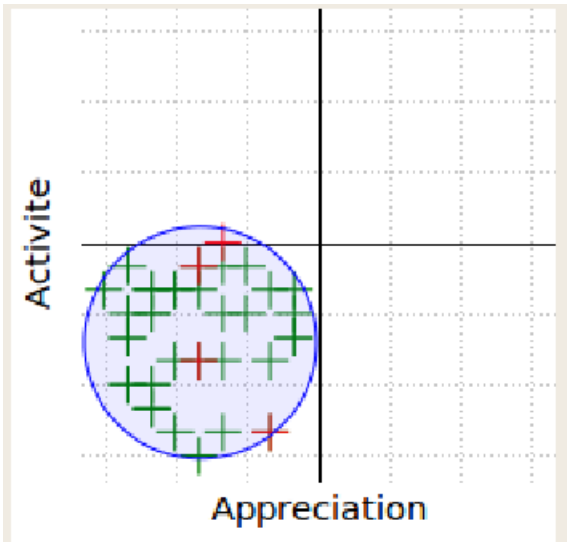

Figure 3.18. Sélection de médias sur le graphe des émotions

Conclusion intermédiaire :

Le logiciel conçu manipule un nombre important de médias accompagnés de leurs métadonnées. XML est un langage simple et économique pour stocker et éditer toutes ces données. Le logiciel Qt fournit des outils puissants pour d'abord coder de façon fractionnée chaque fonctionnalité du logiciel, à savoir l'importation, la gestion et la lecture des médias ainsi que l'intégration des émotions et des liens entre médias.

Ces différentes fonctions sont incorporées à la fenêtre principale affichant les bibliothèques de musique et d'images, les détails d'un média et les actions que l'utilisateur peut effectuer sur les médias. Des icônes sont définies pour chaque bouton, rendant ainsi l'interface plus esthétique et moins surchargée par du texte. Finalement, le logiciel présente une interface claire et organisée.

CONCLUSION :

L'objectif du projet d'étude n°87 était de créer entièrement un logiciel de traitement de médias, par le biais de l'émotion qu'ils dégagent et des tags qui leurs sont associés, et intégrant les algorithmes fournis par notre tuteur scientifique.

Ce projet d'étude est très intéressant au sens où il permet d'appréhender la création d'un logiciel dans son intégralité, de la phase de conception jusqu'aux tests finaux en passant par la phase de réalisation, en ayant toujours pour objectif de respecter un cahier des charges que nous avons nous-mêmes établi à partir des attentes du commanditaire, d'un état de l'art de l'existant et des nos propres attentes en tant qu'utilisateurs de logiciels de traitement de médias et internautes.

Cependant, notre logiciel n'est pas parfait et il reste de nombreux points à améliorer. À titre d'exemple, on pourrait imaginer améliorer de design de l'interface du logiciel, ajouter un lecteur de vidéos, récupérer des paroles de chanson sur internet… Le fait que ces fonctionnalités n'aient pas pu être intégrées à Tag'n'Link est dû à un manque de temps. En effet, lors du lancement du projet, notre idée de la quantité de travail nécessaire à la programmation d'un logiciel restait assez vague à cause de notre manque d'expérience. C'est pourquoi notre diagramme GANTT ainsi que notre cahier des charges ont été quelque peu optimistes.

Nous sommes toutefois parvenus à remplir les principales fonctions attendues afin d'obtenir un logiciel fonctionnel, intuitif et suffisamment complet pour répondre aux attentes du commanditaire du projet. Bien que Tag'n'Link puisse nécessiter certaines améliorations qui pourraient éventuellement faire l'objet d'un prochain projet d'étude, il est déjà à ce jour plus que satisfaisant.

APPENDICE

1) Organisation de l'équipe

Trombinoscope de l'équipe :


Adrien Autricque


Clément Cocquempot


Ghislain Gandolfi


Thomas Gaudelet


Marie Gauthier

Organisation de l'équipe

La réalisation du projet d'étude a représenté une expérience enrichissante dans bien des domaines. Mais cela nous a surtout permis de nous familiariser avec le travail en groupe de taille relativement importante comparativement à ce que nous avons pu rencontrer jusqu'à présent. Et naturellement avec l'augmentation de la taille du groupe de travail et aux vues de la complexité de la tâche à réaliser, il a été nécessaire de s'organiser, de répartir les différentes tâches à réaliser. Pour cela il a fallu que chacun renonce à avoir le contrôle complet du projet, apprendre à faire confiance au travail des autres ce qui n'est pas toujours évident lorsqu'on avait l'habitude de prendre part à toutes les parties des projets entrepris jusqu'à maintenant. En effet le projet étant assez conséquent, il est impossible pour une personne de rentrer dans les détails de chacune des parties de la réalisation.

Les premières réunions du groupe ont donc eu pour objectifs non seulement de cerner le problème posé mais aussi de commencer l'organisation du groupe avec notamment l'élection du chef de projet. Une fois l'étape de décomposition du projet en plusieurs tâches il a fallu se les répartir plus ou moins selon nos connaissances et capacités respectives, les diagrammes ODT/ODR ci-dessous récapitulent cette répartition des tâches :

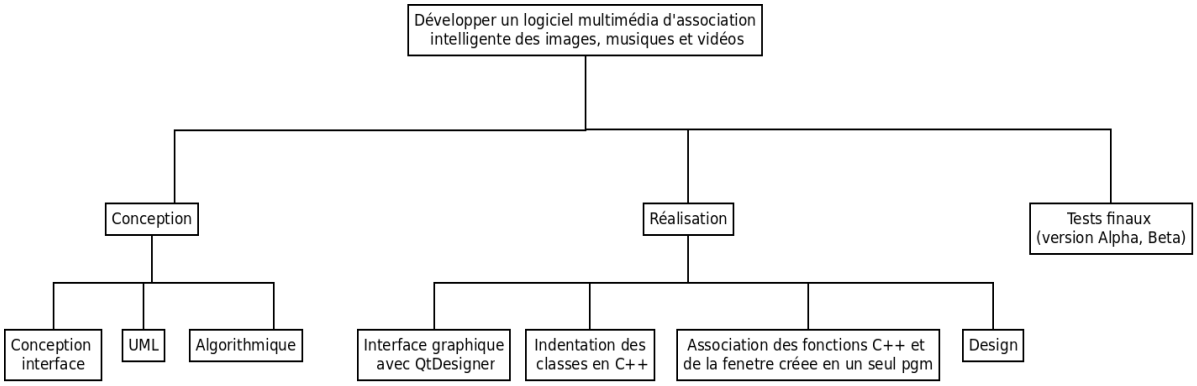

Figure 4.19. Organigramme des Tâches

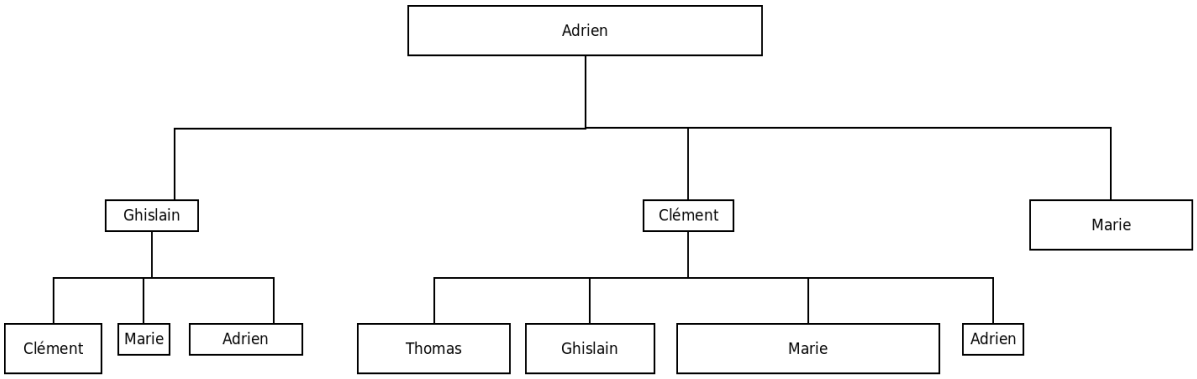

Figure 4.20. Organigramme des Responsabilités

Les deux arbres sont à regarder en parallèle, la personne dont le nom apparait dans une case est le responsable de la tâche correspondante dans l'ODT. Il nous a paru intéressant que le responsable d'une tâche donnée ne participe pas à la réalisation de celle-ci pour qu'il puisse garder une certaine distance et un œil critique vis-à-vis du travail réalisé. Il faut tout de même signaler que lors de la réalisation du projet l'organigramme n'a pas toujours été respecté à la lettre. La principale raison de cela réside dans la différence de connaissances selon les personnes, certain se sont avérés plus à l'aise que d'autre dans, par exemple, la programmation brute, ce qu'il aurait été difficile de prévoir en début de projet puisque, et c'était l'une des difficultés de notre projet, la plupart des membres du groupe n'avait jamais programmé auparavant et deux d'entre nous ne débutaient les cours d'informatique qu'au S6.

Ce dernier point mis à part, l'organisation et le travail du groupe s'est avérée efficace notamment lorsqu'il a fallu redoubler d'effort pour rattraper le léger retard qui s'était accumulé.

2) Gestion du projet

Cette section présente la manière dont ce projet d'étude a été organisé. Dans le cadre de notre projet, il n'y a pas eu besoin de gérer de budget. La seule dépense faite a été l'obtention d'un manuel d'explication du logiciel Qt et de programmation en C++.

Pour mener à terme un projet de cette envergure, des outils efficaces de gestion de projet ont été utilisés. Ces outils ont été proposés lors des séances de travaux dirigés de gestion de projet, avec notre conseiller en gestion de projet : M.Fridrici. Ces outils sont les diagrammes de GANTT et de PERT.

N'ayant jamais programmé de logiciel aussi complexe, la principale difficulté était que ne pouvions qu'estimer de façon grossière le temps que prendrait toutes les tâches de la programmation. Les délais que nous nous étions imposés n'ont pas toujours été respectés, comme le montrent la figure 4.21. Ceci nous a donc contraints

à devoir revoir le cahier des charges au mois de mars. Nous avons été obligés de laisser de côté la partie vidéo dans notre logiciel.

Diagramme PERT

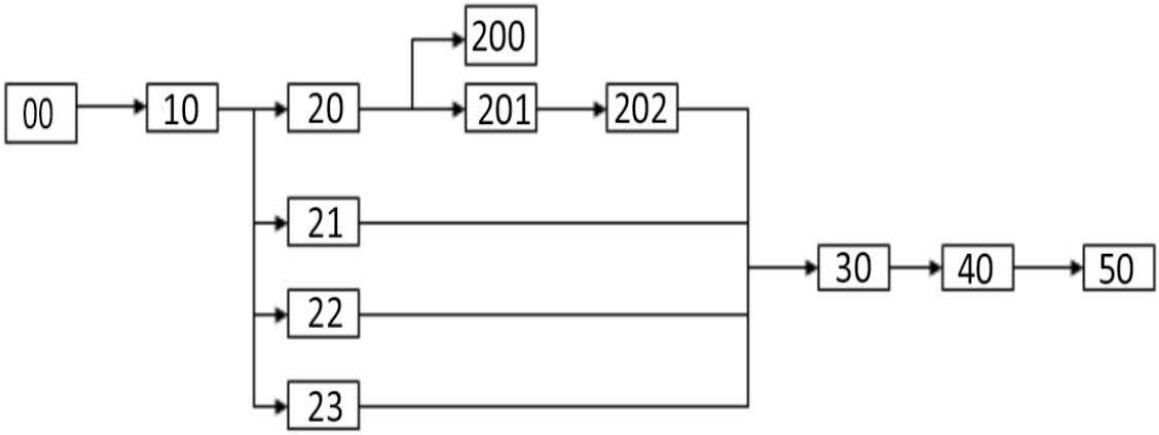

Figure 4.21. Diagramme PERT

Tâches	**Début →Fin**	**Tâches nécessaires**
00 Compréhension du sujet	Sept →2/10	
10 Cahier des charges	05/11→12/11	00
20 Conception Interface	25/11→13/12	10
22 Réalisation des fonctions de base du logiciel	02/01→16/04	10
23 Apprentissage programmation C++	Sept → 18/12	10
200 Conception des fonctions utilisant les algorithmes	5/01→03/03	20
201 Conception avec UML	12/11→18/12	20
202 Réalisation de l'interface graphique	21/11→28/01	201
30 Association des fonctions et de l'interface	4/02→08/05	202, 21, 22,23
40 Tests finaux	3/04→05/06	30
50 Finalisation du rapport et préparation soutenance	08/05→22/06	40

Diagramme GANTT

Le diagramme GANTT permet la visualisation et le temps imparti pour les différentes tâches liées au projet. Avec le GANTT construit au début du mois de novembre 2011, l'interface graphique devait se finir au mois de janvier, ainsi que l'apprentissage et la conception UML.

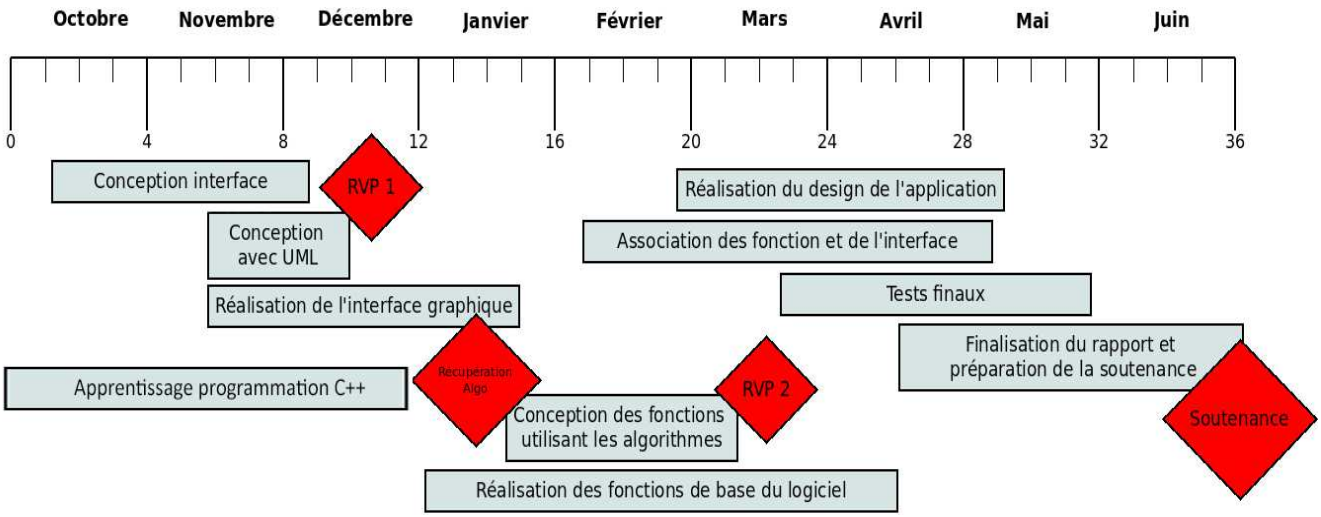

Figure 4.22. Diagramme GANTT

Cependant un certain retard s'est accumulé, principalement durant la phase d'apprentissage du langage C++ qui s'est avérée plus chronophage que prévu initialement. Il faut ajouter à cela qu'il a fallu maîtriser l'IDE retenu par notre groupe dans le but de réaliser le logiciel, à savoir Qt. Ce logiciel offre de multiples options lorsqu'il s'agit de créer une interface, mais la multitude des fonctions proposées par Qt fait qu'il est impossible de parvenir à le maîtriser en si peu de temps. Il a donc fallu s'adapter, et se résoudre à découvrir au fur et à mesure des nécessités les outils que Qt offre pour continuer à avancer.

La seconde tâche qui s'est révélée plus longue que prévue est la conception avec UML qui n'a pas toujours été évidente pour des novices tels que nous.

En revanche, et c'est là où nous avons rattrapé notre retard et constaté l'efficacité de la conception UML, la phase de codage en elle-même nous a demandé moins de temps que ce que nous avions imaginé. Alors que l'on prévoyait que ce serait la tâche qui nous occuperait le plus longtemps et que la conception UML serait beaucoup plus rapide, on se rend compte a posteriori que c'est l'inverse et qu'une bonne conception UML permet une programmation plus efficace.

3) Evolution des objectifs du projet

Lors de la phase préliminaire de définition du sujet, les objectifs que nous nous étions fixés se sont avérés un peu ambitieux sur certains points. Il a fallu notamment abandonner, en accord avec notre tuteur scientifique, Monsieur Dellandréa, la gestion des vidéos. Ce choix d'abandonner le traitement des vidéos a été motivé par le fait que ce type de fichier se prête moins à la liaison avec d'autres médias, la liaison d'un film avec une playlist ou un diaporama n'a pas vraiment d'utilité pratique. Les autres fonctionnalités qui ont dû être abandonnées, faute de temps, sont des applications qui nous semblent intéressantes et originales dans un tel logiciel, tel que par exemple la création d'une connexion avec l'internet pour aller chercher des informations tel que les paroles des chansons ou les couvertures d'albums.

Malgré ces révisions des objectifs initiaux, le logiciel est opérationnel et offre à l'utilisateur les fonctions classiques d'un lecteur de musiques/images et la capacité de lier les médias entre eux, fonction au cœur de ce projet.

4) Apport du projet à chacun des membres

GAUTHIER Marie :

Au cours de ma licence en mathématiques, j'ai eu l'occasion de suivre des cours d'informatique. Je possédais donc déjà quelques bases en programmation et notamment en langage C/C++. En plus d'être un projet informatique, ce projet d'études fait intervenir une démarche créative d'une part, par la création d'une interface ergonomique, d'autre part, par la manipulation d'images et de musiques. C'est la réunion de ces deux aspects, scientifique et artistique, qui m'a orientée vers le choix de ce projet en début d'année.

C'est pour cette raison que je suis intervenue au niveau du design du logiciel durant la phase de conception avec l'élaboration de croquis, puis dans la programmation au cours de la phase de réalisation.

Ce projet a été à la fois formateur sur le plan scientifique et sur le plan humain. D'abord, j'ai appris à « penser objet », ce qui m'a convaincue de la supériorité de ce modèle de programmation. De plus, j'ai eu l'occasion de manipuler de nouveaux outils tels que Qt. Cette bibliothèque fournit d'innombrables possibilités et bien qu'elle soit claire et intelligemment conçue, cela implique beaucoup d'implication pour commencer à la maîtriser. Par exemple, pour réaliser une même fonction, on dispose de plusieurs méthodes différentes. Il convient donc avant de se lancer dans l'écriture du code, d'évaluer laquelle est la plus adaptée à notre situation, laquelle est la plus efficace et la plus simple à mettre en œuvre. Cela est l'une des raisons pour lesquelles la programmation est chronophage et le temps estimé à réaliser chaque fonctionnalité du logiciel est d'autant plus difficile à évaluer. Ensuite, j'ai été confrontée aux problématiques de la gestion de projet. En effet, le travail en groupe implique une capacité d'organisation quant à la répartition des tâches mais aussi une capacité de communication quand il s'agit de trouver des consensus ou encore d'exprimer ses idées clairement pour se faire comprendre par tous.

Bien qu'il n'ait pas permis de définir définitivement mon projet professionnel, ce projet d'études a été très enrichissant dans la mesure où il a combiné des concepts à première vue diamétralement opposés : sciences/art, travail collectif/autonomie.

AUTRICQUE Adrien :

Contrairement aux autres membres de l'équipe de projet, je n'avais aucune notion en informatique lors du lancement du projet, en septembre. En outre, je n'ai pu assister aux cours d'informatique dispensés à l'Ecole Centrale de Lyon que lors du S6 (débutant en janvier). J'ai donc dû acquérir des compétences en informatique de manière autodidacte, principalement grâce au Site du Zéro, afin de rester au niveau de mes camarades. Grâce à ce projet, j'ai ainsi pu acquérir des connaissances en programmation parfois redondantes avec celles enseignées à l'école, mais aussi complétant ces dernières. La création d'un logiciel, l'utilisation des bibliothèques de Qt ou Fmod ou encore la gestion des données avec XML ne sont pas abordées en cours. Ce projet m'a donc permis de découvrir une facette de l'ingénieur que je ne connaissais absolument pas et qui m'a pourtant très intéressé : l'informatique.

Il faut également savoir que le projet d'étude de première année à centrale est le premier projet important qui nous est donné de mener à bien. C'est donc pour nous une expérience nouvelle et délicate faisant intervenir de nombreux acteurs occupant des places différentes. La gestion de projet y occupe une place importante est a été pour moi une discipline inédite, mais qui me parait aujourd'hui indispensable à tout projet d'envergure conséquente.

En tant que chef de projet, j'ai du gérer l'ensemble de l'équipe en planifiant les réunions (très fréquentes) ainsi que leur ordre du jour. J'ai également été l'intermédiaire entre l'équipe de projet et les acteurs extérieurs (tuteur scientifique, conseiller en gestion de projet, conseiller en expression et organisation). Cette position privilégiée m'a permis d'appréhender de la façon la plus complète possible le déroulement d'un tel projet.

Enfin, ce projet m'a permis de réapprendre à travailler en équipe après avoir passé deux ans dans le système élitiste et individualiste des concours d'entrée aux grandes écoles.

GAUDELET Thomas :

Pour ma part j'avais choisi l'option informatique durant mon cursus préparatoire, je suis donc arrivé avec une certaine familiarité avec la programmation. Cependant le codage effectué en classe préparatoire est très théorique, tourné vers la résolution de problèmes donnés plutôt que vers la création d'une application concrète. C'est l'une des raisons qui m'a poussé à choisir ce projet d'étude, j'avais la volonté de participer à la création d'un logiciel, de découvrir quelles étaient les étapes de développement et comment une application de ce type, que nous utilisons tous plus ou moins régulièrement, s'articule derrière l'interface. L'autre raison étant qu'il me semble utile et important d'avoir une connaissance de base et une maitrise correcte de la programmation.

Ce que m'a apporté ce projet, en dehors de la fierté du travail réalisé et des compétences développées, c'est l'apprentissage du travail en groupe. Pour la première fois durant mes études il a fallu vraiment travailler en équipe sur un projet conséquent à réaliser sur une période relativement courte. Il a donc fallut se répartir les tâches, apprendre à se reposer sur les autres pour certaine chose. C'est cet aspect qui m'a le plus marqué, même si il est vrai que la plupart du temps le travail effectif, et je pense particulièrement au codage, se faisait par soi-même il y avait toute l'organisation du groupe derrière visant à optimiser le travail.

En conclusion, même si ce projet d'étude ne m'a pas aidé à me fixer les idées quant à mon projet personnel, il a représenté une expérience enrichissante tant du point de vue de l'apprentissage de connaissances que de l'organisation du travail qui sera naturellement nécessaire dans n'importe lequel des métiers de l'ingénierie.

COCQUEMPOT Clément :

Un projet tel que le projet d'étude est une source d'enrichissements tant au niveau des connaissances que du travail en équipe.

Ceci est possible notamment grâce à son ampleur. En effet, un projet qui dure une année n'est pas évident à prendre en main et gagne en complexité par rapport aux projets auquel je suis plus habitué. Il faut réussir à prendre en compte l'ensemble des paramètres qui permettent la réussite du projet. Ce projet m'a ainsi fait comprendre l'avantage de concevoir correctement un projet avant de le réaliser.

Ce projet m'a bien sûr aussi apporté de nombreuses connaissances techniques en programmation. J'ai pu en effet apprendre à utiliser la bibliothèque Qt et d'améliorer mon aisance en programmation en C++. A cette occasion, j'ai aussi compris qu'un projet ne se résume pas à de la programmation, mais doit tenir compte de préoccupation de design, de portabilité et d'aisance à l'utilisation.

Enfin, le travail et l'organisation au sein d'une équipe fut pour moi le plus grand apport de ce projet d'étude. Je me suis rendu compte que le plus dur n'est pas le travail en lui même mais la prise en compte que nos réalisations seront utilisées et doivent donc être facilement comprises. Il a donc fallu faire des efforts au niveau documentation et rigueur de programmation afin d'éviter les pertes de temps dues à la compréhension.

Pour conclure, ce projet représente pour moi un premier projet important qui fut pour moi à la hauteur de mes attentes et qui m'a aidé à comprendre l'intérêt mais aussi les défis que posent le travail en équipe.

GANDOLFI Ghislain :

Ayant toujours été attiré par l'informatique, je n'avais cependant jamais réellement travaillé sur ce domaine dans mon parcours scolaire. L'acoustique étant aussi un domaine qui m'est sensible, ce projet d'étude, liant ces deux matières, m'a de suite intéressé. C'était pour moi une occasion de voir ce qu'était réellement l'informatique,

d’apprendre à programmer en C++, pour acquérir une culture informatique, et de travailler dessus en groupe. Car au-delà des connaissances techniques, ce projet m’a également appris à travailler en équipe et à pouvoir déléguer le travail en fonction des affinités de chacun et chacune.

La partie design du logiciel était important pour moi, et comme nous étions assez libre dessus, nous devions savoir ce qu’il était possible de faire en 9 mois, sachant que nous ne connaissions pas encore le logiciel Qt, utilisé pour la création d’interface.

Sur le plan professionnel, ce projet m’a montré ce qu’était un travail d’équipe, et j’ai compris l’intérêt de la programmation modulaire, qui nous a permis de pouvoir coder séparément sans empiéter sur le code des autres. Ce projet m’a également conforté sur le fait que l’informatique, de plus en plus présent dans notre quotidien, est une voie qui m’intéresse particulièrement et pourrait être le domaine dans lequel j’aimerai construire mon parcours pédagogique.

ANNEXE 1 : Cahier des charges

I- Point de vue du Client

A. Description du besoin

Présentation générale

Nous assistons depuis plusieurs années à un développement extrêmement rapide des moyens de création, diffusion et stockage de données numériques. Nous sommes passés en très peu de temps à des bases de données de petites tailles, de l'ordre du kilo octets, à des tailles gigantesques. En ce qui concerne les données multimédias, plus que présentes sur nos ordinateurs, leurs tailles ne cessent de s'accroitre. Il faut donc être capable de gérer cette masse d'information. A ce jour, il n'existe aucun logiciel informatique permettant de stocker, d'organiser, de visualiser et d'associer des musiques, des images et des vidéos, le tout de manière simple.

B. Le Cahier des Charges

I- Présentation

Dans le cadre du projet d'étude proposé aux élèves de première année à l'Ecole Centrale de Lyon, le laboratoire d'Informatique en Image et Systèmes d'information veut améliorer l'utilisation de collections volumineuses de données multimédias et offre aux étudiants de travailler en un groupe organisé sur un projet informatique et de développer leurs compétences en terme de programmation de logiciels en utilisant des librairies graphiques et audio.

L'objectif de ce projet est donc de mettre en place un environnement informatique supportant tous les fichiers multimédias, à savoir la musique, les images et les vidéos.

II-Présentations des objectifs et des contraintes

1. **Objectifs**
   Ce logiciel a pour but de simplifier l'organisation des fichiers multimédias. Mais il permettra aussi de visualiser ces fichiers, de pouvoir faire des recherches intelligentes ou encore de les associer entre eux. Le tout dans une certaine simplicité pour l'utilisateur.
2. **Contraintes**
   a. Définition fonctionnelle
   Les données d'entrée du logiciel sont :
   - Fichier de type musique, de différents formats
   - Fichier de type image, de différents formats
   - Des algorithmes fournis par le laboratoire LIRIS.

   Les données de sorties du logiciel sont :
   - Lecture des fichiers
   - Sorties des algorithmes

   b. Condition d'utilisation
   Le logiciel peut être utilisé par tout utilisateur désirant avoir un logiciel de lecture de fichiers multimédias. Il sera dans un premier temps utilisé au service du commanditaire, c'est-à-dire le laboratoire LIRIS.
   c. Définition de l'environnement.

L'interface homme-machine doit respecter des facteurs de confidentialité (les algorithmes ne peuvent être lus par l'utilisateur, les algorithmes étant la propriété du laboratoire LIRIS), de maniabilité (usage intuitif du logiciel, exploitabilité…) et de communicabilité. Le logiciel doit être réutilisable d'année en année, plusieurs fois par jours si nécessaire. Le logiciel pourrait être accessible via une connexion internet depuis l'Ecole Centrale de Lyon et cela de n'importe quel navigateur ou OS. Le logiciel doit être rapide pour assurer le confort des utilisateurs (efficacité mémoire et de temps d'exécution).

d. Définition

Le logiciel devra être utilisable le plus tôt possible. Il pourra au préalable être testé avec une petite base de données de fichiers multimédias.

Une notice d'utilisation devra être fournie.

III- Condition de réalisation

1. Engagement du maître d'ouvrage (client)

Le maître d'ouvrage n'a aucune responsabilité vis-à-vis du maître d'œuvre.

Il s'engage néanmoins à fournir au maître d'œuvre des algorithmes pour réaliser des tests au cours de la programmation.

2. Engagement du maître d'œuvre (développeur)

Le maître d'ouvrage devra utiliser ses propres machines et personnel pour réaliser ce projet. Il ne devra cependant pas fournir les machines au client, mais pourra apporter ses conseils pour l'achat de machines le plus compatible possible avec le logiciel.

Le maître d'œuvre s'engage à fournir le logiciel complet, un manuel d'utilisation au client et à former une personne choisie par le client.

3. Suivi de la réalisation

En dehors de la phase de lancement du projet, et de la réception du livrable, le maître d'ouvrage n'est pas tenu de suivre la réalisation du logiciel. Il aura cependant fourni des algorithmes que le développeur devra utiliser lors de la compilation totale.

4. Condition de réception

Le logiciel sera installé par le développeur sur les ordinateurs du laboratoire LIRIS. La tenue d'un procès-verbal de réception est obligatoire, un document devra être signé par le développeur, ainsi que le laboratoire LIRIS lors de la réception du système.

Le CRI aura le droit de demander l'assistance du développeur pour la maintenance du logiciel pendant l'année scolaire de réception du livrable. Des améliorations pourront être exigées si elles rentrent dans le cadre des exigences formulées au début du projet par le client.

IV- Les Facteurs : Caractéristiques d'utilisation

1. Les facteurs liés à l'environnement d'exploitation

Facteurs d'exploitation	**Sans intérêt**	**Peu important**	**Moyennement important**	**Très important**
Confidentialité : **aptitude d'un logiciel à être protégé contre tout accès par des personnes**			**X**	

non autorisées.				
Couplabilité : **aptitude d'un logiciel à être couplé à un autre logiciel.**		**X**		
Efficacité : **aptitude d'un logiciel à minimiser l'utilisation des ressources disponibles.**			**X**	
Maniabilité : **aptitude d'un logiciel à être convivial et facile d'emploi pour l'utilisateur auquel il est destiné.**				**X**
Robustesse : **aptitude d'un logiciel à conserver un comportement conforme aux besoins exprimés en présence d'événements non souhaités ou non prévus.**		**X**		
Maintenabilité : **aptitude d'un logiciel à faciliter la localisation et correction d'erreurs résiduelles.**		**X**		
Adaptabilité : **aptitude d'un logiciel à faciliter la supp ou l'évolution de fonctionnalités existantes ou l'adjonction de nouvelles fonctionnalités.**		**X**		
Portabilité: **aptitude d'un logiciel à minimiser les répercussions d'un changement d'environnement logiciel et matériel.**				**X**
Testabilité : **aptitude d'un logiciel à permettre la vérification de son fonctionnement.**			**X**	
Observabilité : **aptitude d'un logiciel à fournir des informations sur son fonctionnement.**		**X**		
Réutilisabilité : **aptitude d'un**		**X**		

logiciel à être intégré en partie ou en entier dans d'autres logiciels.				

2. Les facteurs liés à l'environnement de maintenance et de suivi

Facteurs de maintenance	Sans intérêt	Peu important	Moyennement important	Très important
Maintenabilité : aptitude d'un logiciel à faciliter la localisation et correction d'erreurs résiduelles.		**X**		
Adaptabilité : aptitude d'un logiciel à faciliter la suppression ou l'évolution de fonctionnalités existantes ou l'adjonction de nouvelles fonctionnalités.		**X**		
Portabilité : aptitude d'un logiciel à minimiser les répercussions d'un changement d'environnement logiciel et matériel.				X
Testabilité : aptitude d'un logiciel à permettre la vérification de son fonctionnement.			**X**	
Observabilité : aptitude d'un logiciel à fournir des informations sur son fonctionnement.			**X**	
Réutilisabilité : aptitude d'un logiciel à être intégré en partie ou en entier dans d'autres logiciels.				**X**

3. Cahier des charges fonctionnel (non exhaustif)

- Organiser

  Le logiciel doit être capable d'organiser de manière autonome et efficace les données importées.

Pour les images : principalement classer par nom, date. L'organisation sera enrichie par les algorithmes qui fourniront d'amples informations. Le logiciel doit pouvoir les prendre en compte.

Pour la musique : utilisation du logiciel sous forme d'un Juke-box, c'est-à-dire un classement par genre, artiste, album, …

- Rechercher

  La recherche est une caractéristique importante du logiciel. La recherche musicale doit fonctionner comme un lecteur de musique classique, mais le logiciel doit pouvoir rechercher les fichiers qui sont susceptible de nous intéresser. Cette recherche sera faite grâce aux algorithmes. **L'accent est porté sur ce nouveau système de recherche intelligent.**

- Intégrer les algorithmes

Le logiciel doit pouvoir intégrer les algorithmes du laboratoire LIRIS sans difficulté. Ils sont le cœur du logiciel.

- Lier les fichiers multimédias

  Lier les fichiers se fera selon deux procédés :

Automatique : le logiciel liera les fichiers par les algorithmes.

Manuel : l'utilisateur choisira lui-même de lier les fichiers entre eux.

Le taux de confiance sera plus fort pour les procédés manuels. Il faut différencier ces deux méthodes pour que l'utilisateur confirme ou non les liens faits par le logiciel.

L'utilisateur peut lier les fichiers de différentes manières :

1. Lier une image à une musique
   Comme une pochette d'album pour une musique. L'image n'a pas de lien sémantique avec la musique. Elle fait partie de ses métadonnées.
2. Lier des images à une musique selon un thème
   Ces images auront dans leurs données un pointeur vers la musique. Quand la musique est demandée et que l'on souhaite faire une visualisation de photos, ces photos seront automatiquement appelées. Cette association se fait autant automatiquement que manuellement.
3. Lier les images à une musique temporairement
   Possibilité à prendre. On peut donc prévoir la création d'une fenêtre spéciale.

- Recevoir et donner des flux
- Pouvoir « tagguer » les fichiers

  L'utilisateur ainsi que le logiciel peuvent ajouter des données supplémentaires aux fichiers.
- Lire tous les types de formats musique
- Lire tous les types de formats image
- Avoir une interface intuitive

  Dans sa version finale, le logiciel devra être fluide, propre, et accueillant pour l'utilisateur. Une interface « avancée » pourra être exploitée. Un chapitre sera consacré à son interface.
- Minimiser la mémoire

  Pas de fuite de mémoire.

ANNEXE 2 : Liste des concepts détectables dans un média par l'algorithme du LIRIS

0	Partylife	33	Underexposed	67	technical
		34	Neutral_Illumination	68	abstract
1	Family_Friends	35	Motion_Blur	69	boring
2	Beach_Holidays	36	Out_of_focus	70	cute
3	Building_Sights	37	Partly_Blurred	71	dog
4	Snow	38	No_Blur	72	cat
5	Citylife	39	Single_Person	73	bird
6	Landscape_Nature	40	Small_Group	74	horse
7	Sports	41	Big_Group	75	fish
8	Desert	42	No_Persons	76	insect
9	Spring	43	Animals	77	car
10	Summer	44	Food	78	bicycle
11	Autumn	45	Vehicle	79	ship
12	Winter	46	Aesthetic_Impression	80	train
13	Indoor	47	Overall_Quality	81	airplane
14	Outdoor	48	Fancy	82	skateboard
15	Plants	49	Architecture	83	female
16	Flowers	50	Street	84	male
17	Trees	51	Church	85	Baby
18	Sky	52	Bridge	86	Child
19	Clouds	53	Park_Garden	87	Teenager
20	Water	54	Rain	88	Adult
21	Lake	55	Toy	89	old_person
22	River	56	MusicalInstrument	90	happy
23	Sea	57	Shadow	91	funny
24	Mountains	58	bodypart	92	euphoric
25	Day	59	Travel	93	active
26	Night	60	Work	94	scary
27	Sunny	61	Birthday	95	unpleasant
28	Sunset_Sunrise	62	Visual_Arts	96	melancholic
29	Still_Life	63	Graffiti	97	inactive
30	Macro	64	Painting	98	calm
31	Portrait	65	artificial		
32	Overexposed	66	natural		

ANNEXE 3 : Diagrammes état-transition

Diagramme 1 – Associer :

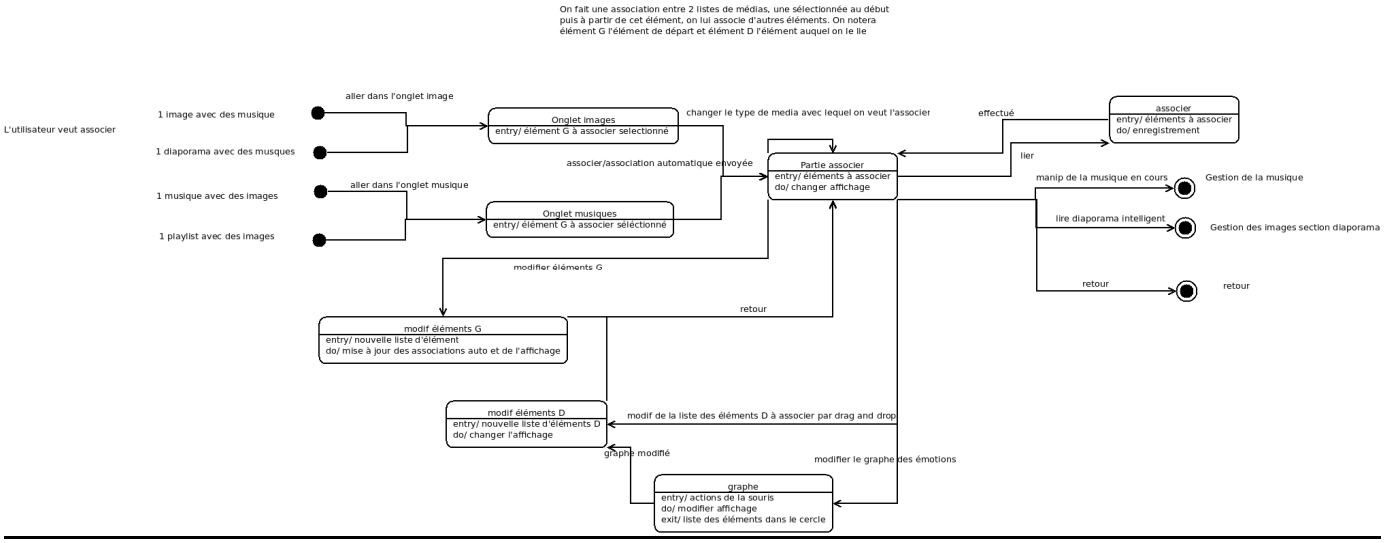

Diagramme 2 – Images :

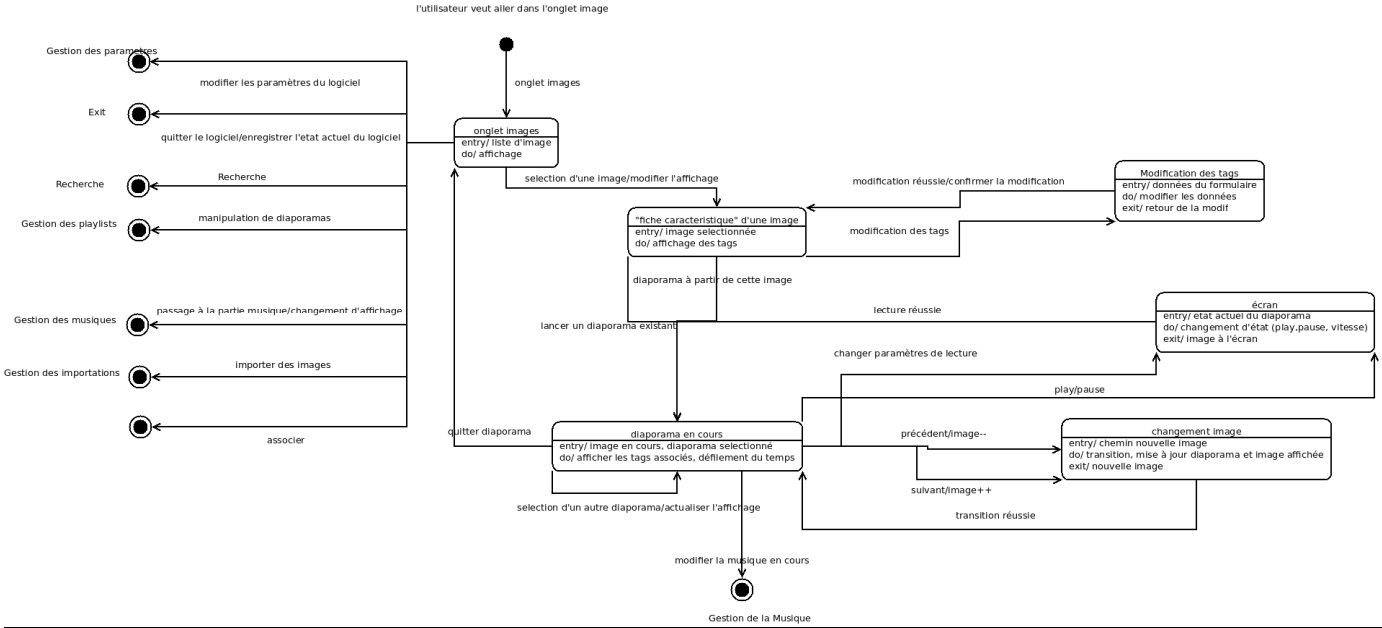

Diagramme 3 – Importation :

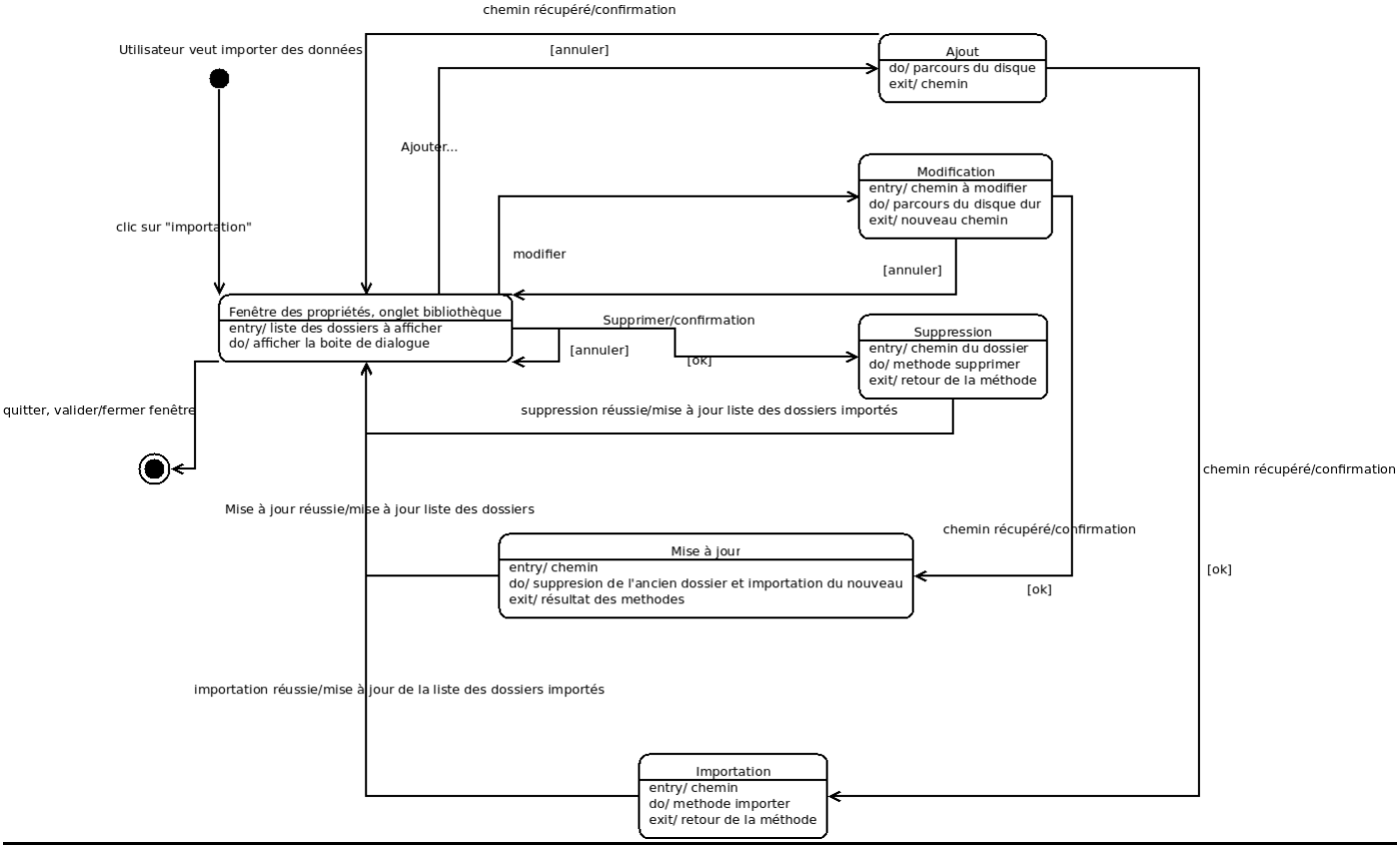

Diagramme 4 - ManipMusique :

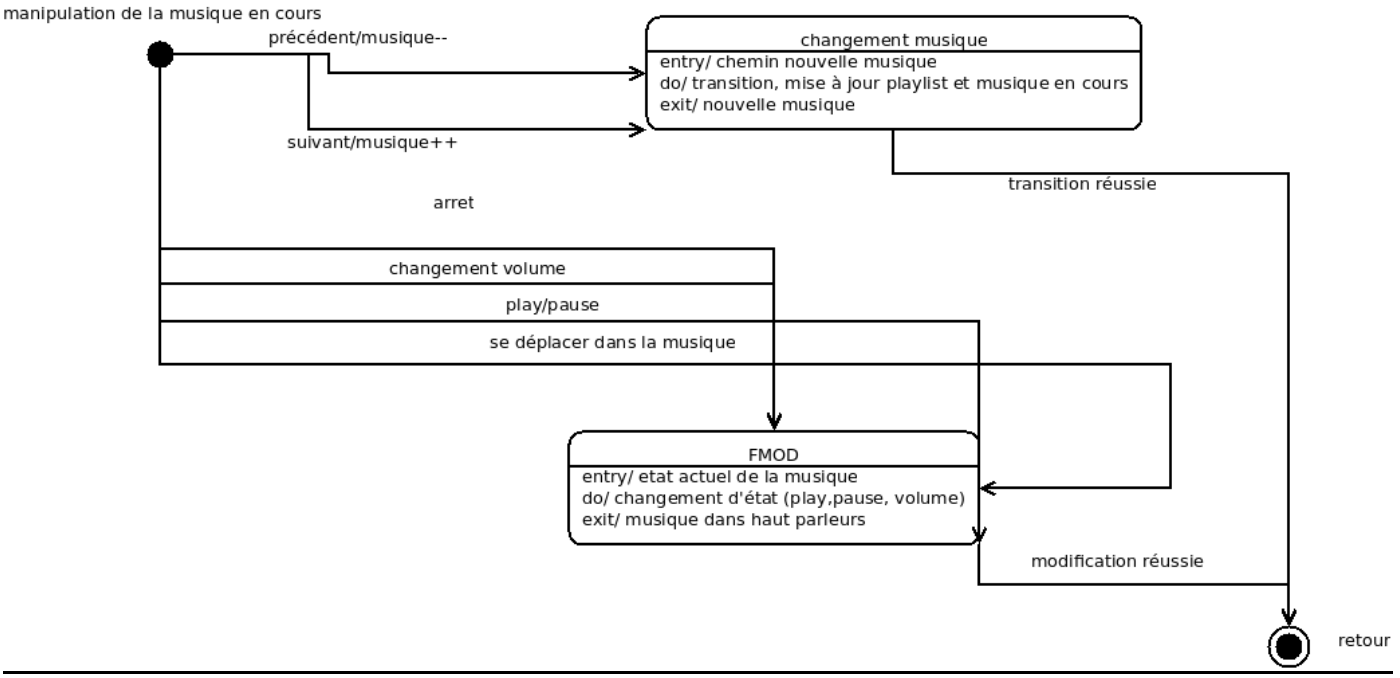

Diagramme 5 – Musique :

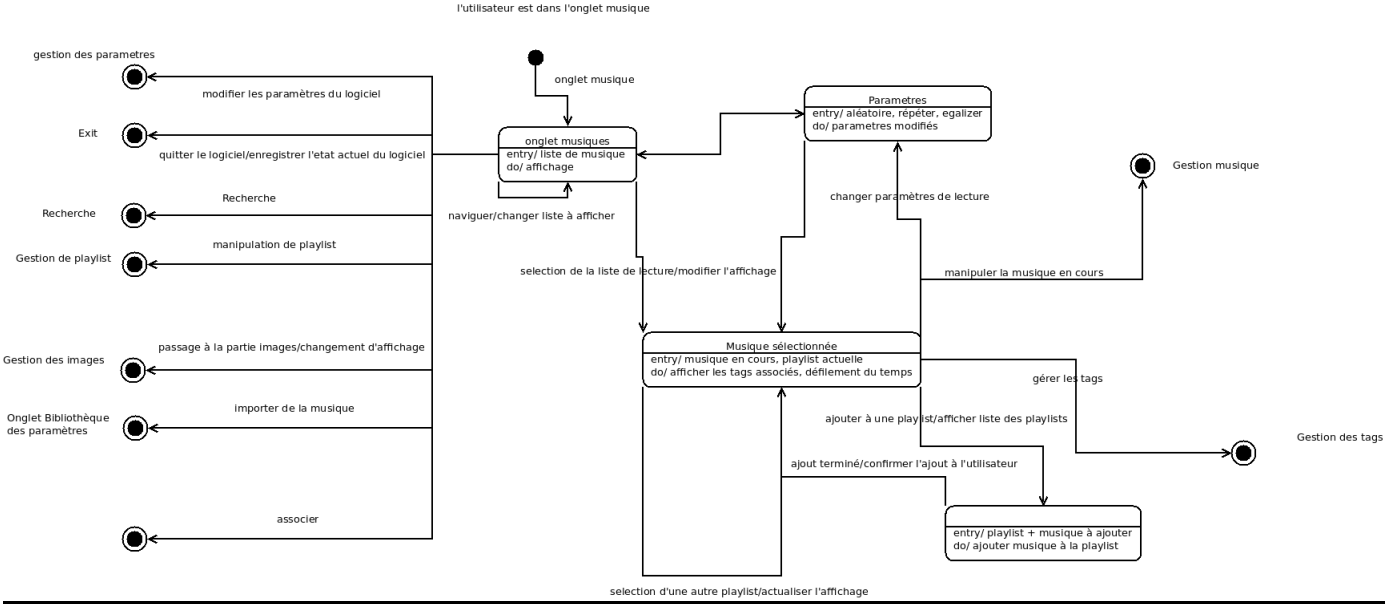

Diagramme 6 – Playlist :

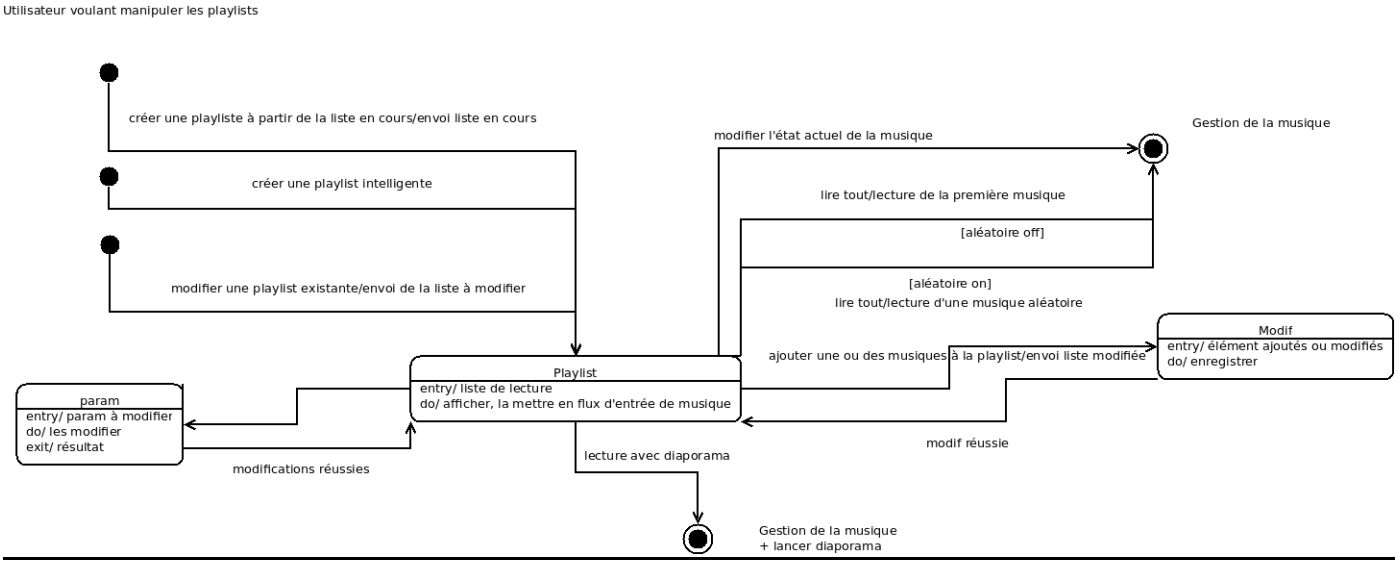

Diagramme 7 – Recherche :

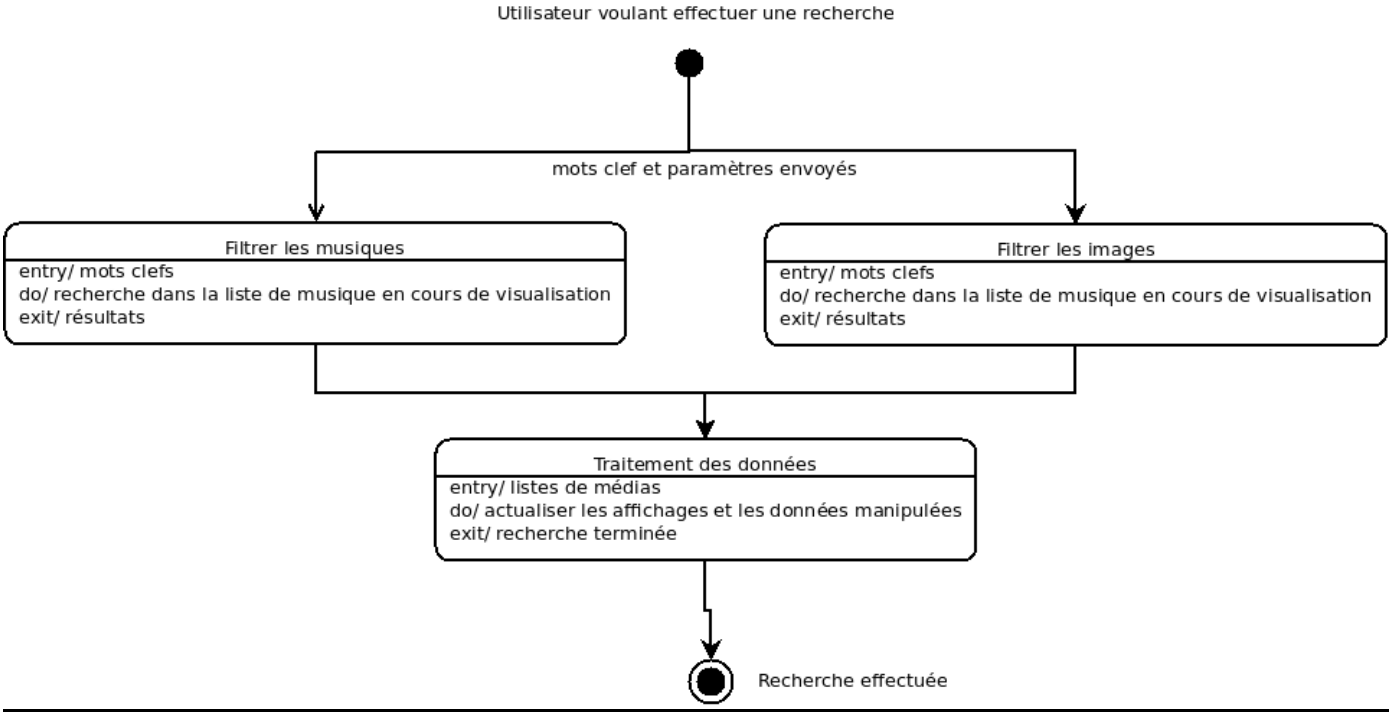

Diagramme 8 – Tags :

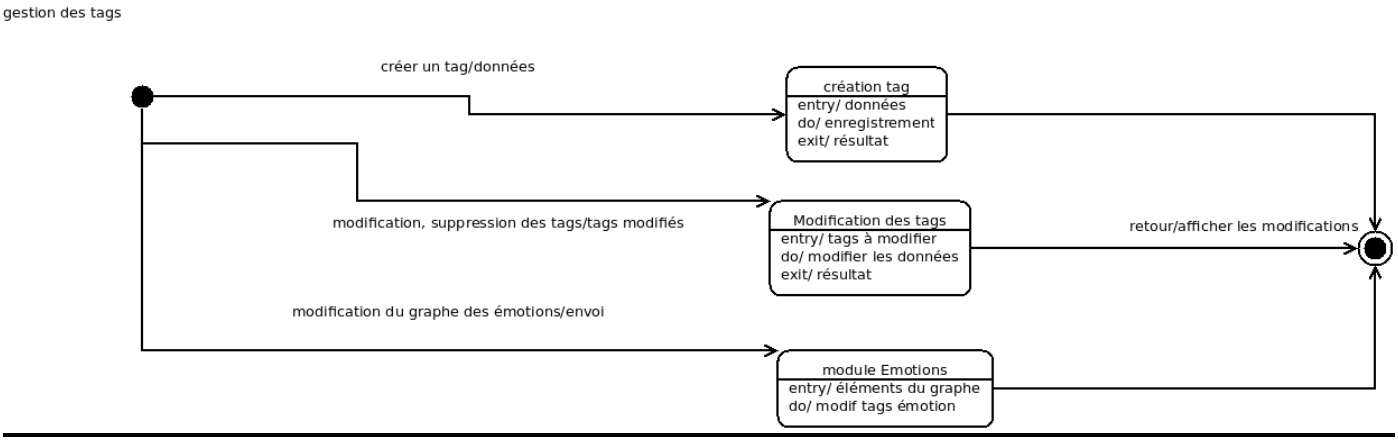

TABLE DES FIGURES ET TABLEAUX

BIBLIOGRAPHIE

- [1] BRILLANT A., *XML Cours et exercices*. Paris : Eyrolles, 2007. 284 p. ISBN 978-2-212-12151-3

- [2] Documentation de la bibliothèque Qt. Disponible sur http://doc.qt.nokia.com (consulté de janvier à juin 2012).

- [3] Works Like Clock Work, *Qt Plotting Widget* [En ligne]. Disponible sur http://www.workslikeclockwork.com/index.php/components/qt-plotting-widget/ (consulté en mai 2012).

- DELLANDREA E. *Programmation Orientée Objet – Mise en œuvre en C++*. École Centrale de Lyon : Année 2011-2012.

- DELLANDREA E. *Langage C avec extensions syntaxiques C++ - Démarrage rapide*. École Centrale de Lyon : Année 2011-2012.

- DAVID B., Génie logiciel et conduite de projet informatique [en ligne]. École Centrale de Lyon : Année 2011-2012. Disponible sur : https://pedagogie.ec-lyon.fr (Consultation au cours de l'année scolaire 2011-2012).

- NEBRA M., Le Site du Zéro, site communautaire de tutoriels gratuits pour débutants, *Programmez avec le langage C++* [en ligne]. Disponible sur : http://www.siteduzero.com/tutoriel-3-11406-programmez-avec-le-langage-c.html - part 11407 (consulté en novembre, décembre 2011 et janvier 2012).

- NEBRA M., SCHALLER M., *Programmez avec le langage C++.* Paris: Simple IT, 2011. 704 p. **ISBN** 978-2-9535278-5-8.

- BLANCHETTE J., SUMMERFIELD M., *Qt4 et C++ : Programmation d'interfaces GUI* (EBERHARDT C., KOLB C., SITTLER D. trad.). Paris : Pearson, 2007. 569 p. ISBN 978-2-7440-4092-4.

- Documentation de la bibliothèque FMOD. Livrée avec la bibliothèque. Disponible sur http://www.fmod.org/index.php/download#FMODExProgrammersAPI (consulté de février à avril 2012).

- BOUILLOT T., LECLERC J., NOGIER J.-F., *Ergonomie des interfaces : Guide pratique pour la conception des applications web, logicielles, mobiles et tactiles*. Paris: Dunod, 2011. 320 p. ISBN 978-2100557929.

CHECK-LIST

	Vérification présence	Vérification qualité
Contenu		
Table des matières	Clément	Adrien
Introduction	Marie	Clément
Conclusion générale	Thomas	Marie
Bibliographie	Adrien	Thomas
Résumé	Ghislain	Adrien
Table des figures	Clément	Adrien
Forme		
Vérification orthographe	Marie	Ghislain
Pagination	Adrien	Thomas
Homogénéité de la mise en page	Marie	Clément

RESUME

Voici le rapport du projet d'étude numéro 87. Très centré autour de l'informatique, ce projet a pour livrable final un logiciel de traitement de médias aux fonctionnalités inédites. Les médias traités sont les trois principaux médias rencontrés sur n'importe quel ordinateur : la musique, les images (incluant les photos), et les vidéos (incluant les films). Outre la lecture de ces médias, notre logiciel baptisé Tag'n'Link devra pouvoir créer des liens entre différents médias de différents types. De plus, Tag'n'Link devra intégrer des algorithmes fournis par le laboratoire LIRIS qui permettront de saisir des concepts de type émotion ou objet (sur des photos).

Le principal objectif est de réaliser un logiciel fonctionnel, facile d'utilisation et le plus rapide possible réalisant toutes les fonctions demandées par le commanditaire. Le principal problème que nous avons rencontré est la complexité nécessaire à la programmation d'un logiciel complet, d'autant plus que certains membres de l'équipe de PE n'avaient jamais eu de cours d'informatique avant le semestre S6. Il a donc fallu apprendre par nous-mêmes et nous répartir les tâches à accomplir en fonction de leur complexité.

Nous avons programmé le logiciel à l'aide du langage C++, qui est d'un niveau suffisamment bas et qui a l'avantage d'être enseigné à l'école centrale, et avons réalisé l'interface graphique à l'aide du logiciel Qt, gratuit et disponible sur internet.