

Figure 1 - Image titre

|  |
| --- |
| **MusicSort**  Tri, renommage et création d’une playlist de musique |

Auteur : Dévaud Aurélien

Chef de projet : Lymberis Dimitrios

Expert 1 : Montemayor Ernesto

Expert 2 : Viret Loic

Du lundi 29 avril 2024 au mercredi 29 mai 2024

ETML - TPI

Table des matières

[1 Analyse préliminaire 4](#_Toc167870816)

[1.1 Introduction 4](#_Toc167870817)

[1.2 Objectifs 4](#_Toc167870818)

[1.3 Planification 4](#_Toc167870819)

[1.3.1 Méthode des six pas IPDRCE 4](#_Toc167870820)

[1.3.2 Planification détaillée 6](#_Toc167870821)

[2 Analyse 6](#_Toc167870822)

[2.1 Concept 7](#_Toc167870823)

[2.1.1 Environnement utilisé 7](#_Toc167870824)

[2.1.2 Stratégie de conception 7](#_Toc167870825)

[2.1.3 Structure du code 7](#_Toc167870826)

[2.1.4 Installer 7](#_Toc167870827)

[2.2 Stratégie de test 7](#_Toc167870828)

[2.3 Risques 8](#_Toc167870829)

[2.3.1 Risques techniques 8](#_Toc167870830)

[2.3.2 Manque de compétences 8](#_Toc167870831)

[3 Conception 9](#_Toc167870832)

[3.1 Modèles de cas d’utilisation 9](#_Toc167870833)

[3.2 Maquettes 12](#_Toc167870834)

[3.2.1 Maquette de base 12](#_Toc167870835)

[3.2.2 Maquette des menus 13](#_Toc167870836)

[3.3 Diagramme d’activité 13](#_Toc167870837)

[3.3.1 Gestion des fichiers FLAC 14](#_Toc167870838)

[3.3.2 Changement du mode d’application 14](#_Toc167870839)

[3.3.3 Affichage des changements 15](#_Toc167870840)

[3.3.4 Application des changements 16](#_Toc167870841)

[3.4 Diagramme de classe 17](#_Toc167870842)

[3.4.1 Models 17](#_Toc167870843)

[3.4.2 Views 18](#_Toc167870844)

[3.4.3 Controllers 18](#_Toc167870845)

[3.5 Liste des tests 19](#_Toc167870846)

[4 Réalisation 23](#_Toc167870847)

[4.1 Environnement 23](#_Toc167870848)

[4.2 Interface 24](#_Toc167870849)

[4.3 Structure 25](#_Toc167870850)

[4.3.1 Changements par rapport à la conception 25](#_Toc167870851)

[4.4 Comportement du programme 26](#_Toc167870852)

[4.4.1 Navigateur de dossiers 26](#_Toc167870853)

[4.4.2 Gérance de la playlist 26](#_Toc167870854)

[4.4.3 Changements généraux 27](#_Toc167870855)

[4.4.4 Application des changements 28](#_Toc167870856)

[4.4.5 Music Player 29](#_Toc167870857)

[4.5 Installeur 30](#_Toc167870858)

[4.6 Tests des fonctionnalités 32](#_Toc167870859)

[4.7 Liste des documents fournis 34](#_Toc167870860)

[5 Conclusions 35](#_Toc167870861)

[5.1 Bilan des fonctionnalités 35](#_Toc167870862)

[5.2 Bilan de la planification 36](#_Toc167870863)

[5.3 Analyse post-projet 36](#_Toc167870864)

[5.4 Améliorations possibles 37](#_Toc167870865)

[5.5 Bilan personnel 37](#_Toc167870866)

[6 Glossaire 37](#_Toc167870867)

[7 Table des illustrations 39](#_Toc167870868)

[8 Manuels 40](#_Toc167870869)

[8.1 Manuel d'Installation 40](#_Toc167870870)

[8.2 Manuel d'Utilisation 41](#_Toc167870871)

[9 Annexes 43](#_Toc167870872)

[9.1 Résumé du rapport du TPI / version succincte de la documentation 43](#_Toc167870873)

[9.2 Sources – Bibliographie 44](#_Toc167870874)

[9.3 Archives du projet 44](#_Toc167870875)

# Analyse préliminaire

Dans cette partie, nous allons décrire les objectifs et détails du projet ainsi que les buts, tests et les moyens à disposition pour l’effectuer.

## Introduction

L’objectif de ce projet est de développer une application en C# avec une interface utilisateur graphique (GUI) permettant à l’utilisateur d’explorer et de gérer des fichiers musicaux de différents formats tels que LFAC, WMA, MP3, etc.

L’application offrira des fonctionnalités telles que la navigation à travers les dossiers, la création de listes de lecture personnalisées par copie dans un dossier spécifique et le renommage de fichiers. Il est demandé également au candidat de mettre en place un package d’installation pour le programme avec son protocole d’utilisation.

Le projet commence par une planification initiale qui, selon les objectifs donnés dans le cahier des charges et la méthode des six étapes, essaie d’organiser en différentes tâches le déroulement du projet et leur charge de travail relationnels (pourcentage).

La méthode des 6 étapes a été favorisée car les autres méthodes, comme la méthode agile et Scrum, sont plus adaptées au travail en équipe.

Les tests sont effectués à l’aide d’un tableau contenant plusieurs colonnes qui permettent d’identifier le nom du test, la fonctionnalité testée, de décrire le test et de décrire le résultat demandé.

Pour la sauvegarde du projet, elle sera effectuée en enregistrant les fichiers dans Github de manière journalière. Les ‘commit’ pourrons servir pour garder plusieurs versions.

## Objectifs

L'application doit permettre à l'utilisateur de :

1. Parcourir et explorer les fichiers musicaux stockés dans différents dossiers sur son système.
2. Copier des fichiers musicaux sélectionnés dans d’autres dossiers pour créer des listes de fichiers de lecture personnalisées.
3. Renommer les fichiers musicaux selon ses préférences.
4. Prendre en charge plusieurs formats de fichiers musicaux courants, tels que FLAC, MP3, WMA avec possibilité d’écouter le morceau.
5. Fournir une interface utilisateur conviviale et intuitive pour faciliter la navigation et l'utilisation de l'application à travers une seule fenêtre Windows.

## Planification

La planification initiale est la partie où une première structure est donnée au déroulement du projet. Elle sera décrite ci-dessous.

### Méthode des six pas IPDRCE

Pour la planification du projet, la méthode des six pas sera utilisée pour sa simplicité et son efficacité. Les estimations de temps incluent le temps passé sur la documentation.

#### Informer

Cette partie de la méthode des six pas est le moment où l’on s’informe sur les sujets impliqués à la mise en place du projet. Donc, les tâches suivantes seront effectuées :

1. Analyse du cahier des charges.
2. Recherche des thèmes impliqués et manques de compétences à combler
3. Recherche des risques et des solutions

Due aux informations déjà possédée, elle représente ~2% du projet (~1h45).

#### Planifier

Cette partie de la méthode des six pas se centre sur la planification du projet et donc la répartition des tâches sur le temps. Donc, les tâches suivantes seront effectuées :

1. Sélection de la méthode de gestion de projet (IPDRCE).
2. Mise en place des tâches à effectuer durant le projet.
3. Répartition des tâches sur les jours

La planification prenant moins d’un jour, elle représente ~6% du projet (~5h16).

#### Décider

Cette partie est le moment où ce qui doit être réalisé est décidé. C’est la conception des modèles et maquettes.

1. Création du diagramme de cas d’utilisations.
2. Mise en place de la stratégie de test et création des tests,
3. Création de la maquette.
4. Création du diagramme d’activité.
5. Création du diagramme de classe.

Cette partie étant le moment où toute l’application est conceptualisée représente ~30% du projet (~26h24).

#### Réaliser

Cette partie est le moment où les éléments conceptualisés dans la partie ‘Décider’ sont réalisés.

1. Mise en place de l’environnement.
2. Création de l’interface principale.
3. Création de la structure du projet.
4. Remplissage des classes.
5. Débogage.
6. Création de l’installer.

Cette partie étant le moment où tout le projet, déjà bien préparé, est mis en place représente ~50% du projet (~44h).

#### Tester

Ceci est la partie où les fonctionnalités sont testées.

1. Test des fonctionnalités et commentaires des résultat si erreurs.
2. Description des erreurs, s’il y en a, et suggestions pour les résoudre.

Cette partie étant une simple utilisation de l’application et description des résultats représente ~4% du projet (~3h31).

#### Évaluer

Dans cette partie, le travail fait est évalué.

1. Comparaison du CDC avec le travail réalisé.
2. Corrections au projet si nécessaire ou si le temps le permet.
3. Discussion sur les résultats et améliorations possibles.

Cette partie représentant un grand temps de rédactions prendra donc le reste, ~8% du projet (~7h02).

### Planification détaillée

La planification détaillée a été créée à l’aide des tâches et ratios présenté ci-dessus. Un fichier Excel a été utilisé qui décrit plus ou moins précisément les dates et durées des tâches sur les jours. Le fichier peut se trouver dans l’annexe du document. ￼

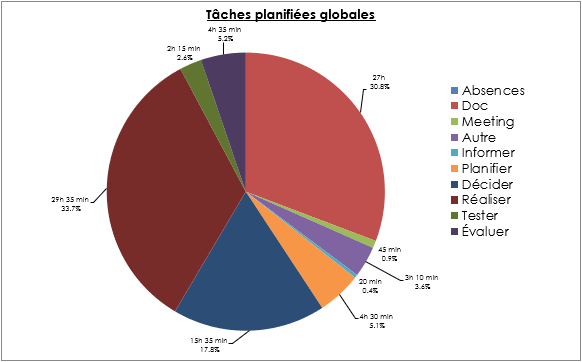


Figure 2 - Répartition des tâches planifiées

# Analyse

Le concept de ce projet est de créer une application permettant de sélectionner des fichiers de musique depuis des emplacements sur le PC et de les placer dans une playlist. Cette playlist doit pouvoir être triée et il doit être possible de renommer les fichiers et/ou de les déplacer ou copier dans un autre dossier.

Pour y arriver, différentes technologies seront utilisées, telles que draw.io pour créer les modèles, et Windows Forms pour réaliser l’application. Étant l’une des versions les plus récentes, donc qui contient plus du contenu, ni la dernière, donc ayant moins de chance d’avoir des problèmes, la version 4.7.2 de .NET Framework sera utilisée.

Dans l’optique d’optimiser le temps, une aide sous la forme de ChatGPT sera utilisée pour débloquer des problèmes ou donner de l’inspiration.

## Concept

Ici est analysé tous les concepts utilisés pour arriver à fin du projet tel que l’environnement dans lequel l’application serait codée, la stratégie utilisée pour concevoir le projet, la structure du code et l’installer.

### Environnement utilisé

Le projet demandant la navigation dans des fichiers et leur manipulation, il a été décidé d’utiliser des librairies capables d’afficher des formulaires. Pour ce cas, Windows Forms sera utilisé avec comme aide VisualStudio 2022 qui propose beaucoup d’options pour faciliter son utilisation.

Pour lire les fichiers de musique et pouvoir les écouter, le ‘control’ Windows Media Player sera employé en addition d’une librairie, NAudio.Wave, pour convertir les fichiers FLAC, illisibles par le ‘control’, en MP3.

### Stratégie de conception

Pour concevoir l’application, un premier diagramme de cas d’utilisation pour identifier les fonctionnalités à implémenter et une maquette qui devra présenter l’apparence de l’interface qui devrait donner à l’utilisateur la possibilité d’exécuter toutes les actions décrites dans le premier diagramme seront créés.

Ensuite les processus seront pensés et présentés dans le diagramme d’activité et le programme structuré dans un diagramme de classe.

### Structure du code

Dans le but de pouvoir bien s’organiser et se retrouver dans le code malgré les différences entre ce qui est affiché et ce qui est réel, on utilisera le model MVC pour séparer effectivement l’interface des données.

### Installer

Pour que l’utilisateur puisse mettre en place l’application facilement et rapidement, un installer est nécessaire. L’installer de l’application ne sera pas celui de VisualStudio car de précédents cas ont prouvé qu’il était enclin à ne pas fonctionner correctement. Une alternative est l’application Inno setup.

## Stratégie de test

La stratégie de test utilisée sera une suite de test créée en fonction des actions identifiées dans le diagramme de cas d’utilisation. Chaque action possédera au moins un test dans le but de vérifier le bon fonctionnement de chaque fonctionnalité de l’application. Il y aura additionnellement un test pour l’installation.

Dû à la taille limitée de l’application, il a été décidé de ne pas utiliser des tests unitaires qui serait potentiellement trop volumineux pour des actions assez simples.

Un premier tableau utilisant 4 colonnes qui sert à lister les tests à effectuer a donc été utilisé avec la première colonne servant de nom et d’identifiant, la deuxième désignant quelle action elle représente, la troisième décrivant le test et la dernière montrant le résultat attendu.

Un nouveau tableau sera créé pour décrire les résultats des tests. Il possède 5 colonnes. La première l’identifie avec son par rapport au premier tableau des tests. La deuxième donne la date à laquelle le test a été effectué. Le troisième dit si le test a été réussi. La quatrième donne les conditions exactes dans lesquelles le test a été effectué. Et la dernière permet d’écrire des commentaires sur le résultat. Comme, par exemple, comment il a raté. Plusieurs tableaux pourraient être créés pour plusieurs séries de tests.

L’environnement des tests sera idéalement différent de celui utilisé durant le développement dans le but d’éviter de manquer des erreurs et les tests seront effectués au moins une fois par un camarade de classe avec des dossier remplis de données divers incluant des fichiers musiques.

## Risques

Ce paragraphe décrit les différents risques techniques qui pourraient être rencontrés durant le projet ainsi que les potentiels manques en compétences possibles.

### Risques techniques

Ici, les risques techniques qui viennent des technologies utilisées sont décrits.

* Absence de Framework lors de l’installation.
  + Solution : ajout du processus d’installation du Framework dans l’installer.
* Le ‘control’ Windows Forms utilisé ne supporte pas les fichiers FLAC.
  + Solution : créer une copie du fichier convertie en MP3 pour être lu en utilisant la librairie Naudio.Wave (librairie proposée par ChatGPT).

### Manque de compétences

* Manque de connaissance sur la lecture de fichier son.
  + Solution : lecture de la documentation Windows Forms pour trouver un ’control’ utilisable. ‘Control’ Windows Media Player trouvé.

# Conception

Dans cette partie, le processus de conceptualisation du projet, les décisions effectuées pour la création du projet et la création des modèles pour la réalisation du produit final, sont expliqués.

## Modèles de cas d’utilisation

Ici, le modèle de cas d’utilisation qui affiche les différentes actions que l’utilisateur peut entreprendre est montré et expliqué.

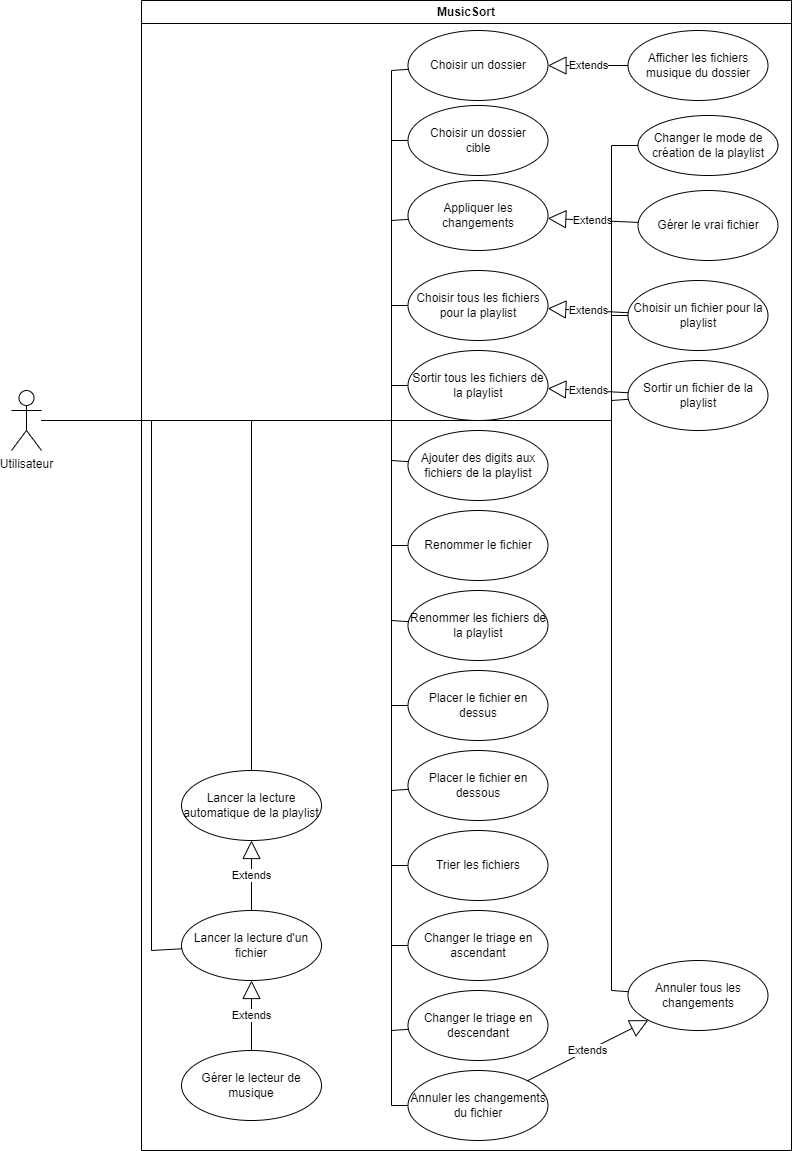


Figure 3 - Diagramme de cas d'utilisation

#### Choisir un dossier

Cette action permet à l’utilisateur de choisir un dossier dans lequel il pourra choisir des fichiers de musique à mettre dans la playlist

#### Afficher les fichiers musiques du dossier

Cette action affiche les fichiers musiques (FLAC, MP3, WMA) dans une fenêtre dans laquelle il sera possible de sélectionner ses fichiers et de les lire.

#### Choisir un fichier pour la playlist

Cette action permet à l’utilisateur de choisir un fichier d’un dossier et de le déplacer vers la playlist

#### Sortir un fichier de la playlist

Cette action permet de sortir un fichier de la playlist.

#### Choisir tous les fichiers pour la playlist

Cette action permet d’envoyer tous les fichiers du dossier sélectionné dans la playlist.

#### Sortir tous les fichiers de la playlist

Cette action permet de sortir tous les fichiers de la playlist.

#### Renommer le fichier

Cette action permet de renommer un fichier se trouvant dans la playlist.

#### Ajout de digit aux fichiers de la playlist

Cette action permet à l’utilisateur d’ajouter des chiffres comme préfixe ou suffixe au nom dans le but de garder l’ordre quand les fichiers seront sauvegardés dans le dossier. Elle donne la possibilité de placer un nombre de départ et de garder le nombre de digit donné dans le nombre de départ.

#### Renommer les fichiers de la playlist

Cette action permet à l’utilisateur de renommer les fichiers de la playlist avec un nom uniforme. Dans ce cas-là, il faudra obligatoirement attribuer un numéro pour ne pas avoir de doublon.

#### Placer le fichier en dessus

Cette action permet de placer le fichier sélectionné dans la playlist un cran au-dessus dans liste.

#### Placer le fichier en dessous

Cette action permet de placer le fichier sélectionné dans la playlist un cran en-dessous dans liste.

#### Trier les fichiers

Cette partie va trier les fichiers de la playlist dans l’ordre alphabétique.

#### Changer le triage en ascendant

Cette action va changer le triage en mode ascendant par rapport à l’ordre donné.

#### Changer le triage en descendant

Cette action va changer le triage en mode descendant par rapport à l’ordre donné.

#### Annuler les changements du fichier

Cette action va annuler les changements donnés à ce fichier tel que son ordre personnalisé dans la playlist ainsi qu’un éventuel renommage.

#### Annuler tous les changements

Cette action va annuler tous les changements dans la playlist. Cela inclut le renommage général de la liste, les digits et les changements individuels des fichiers.

#### Changer le mode de création de la playlist

Cette action change le mode de création de playlist dans les données réelles. Par défaut, il est en renommage, il renomme tous les fichiers renommés sans changer leurs emplacements originaux. Le deuxième mode et ‘Renommer et copier’. Dans ce cas, les fichiers originaux seront renommés et copiés dans le dossier cible. Le troisième mode est ‘Renommer et déplacer’ qui déplace le fichier de son emplacement original au dossier cible tout en le renommant.

#### Gérer le lecteur de musique

Cette action gère le lecteur de musique. Elle fait en sorte que le fichier donné est lu sans avoir de double lecture.

#### Lancer la lecture d’un fichier

Cette action lance la lecture individuelle d’un fichier.

#### Lancer la lecture automatique de la playlist

Cette action lance la lecture automatique ordrée de la playlist depuis le début.

#### Choisir un dossier de destination

Cette action permet de choisir un dossier qui sert de destination dans le cas où l’utilisateur déciderait de copier ou déplacer les fichiers de la playlist.

#### Appliquer les changements

Cette action applique les changements effectués aux éléments de la playlist.

#### Gérer le vrai fichier

Cette action renomme, copie et déplace le vrai fichier par rapport à son état dans la playlist.

#### Changer le mode de création de la playlist

Cette action change le mode de ‘création’ de la playlist entre ‘Renommer’, ‘Renommer et copier’ et ‘Renommer et déplacer’.

Pour compléter les objectifs du CDC, il faut que l’utilisateur ait l’option d’interagir avec l’application de la manière décrite ci-dessus.  
En premier temps, l’utilisateur doit pouvoir choisir un dossier pour chercher les fichiers qu’il doit pouvoir mettre dans la playlist individuellement ou en groupe et faire la même chose avec les fichiers de la playlist vers leur dossier d’origine. Il doit pouvoir être possible de renommer les fichiers individuellement et de faire un renommage automatique de tous les membres de la playlist avec l’option d’ajouter un numéro. L’utilisateur peut aussi trier les fichiers et changer le mode de tri comme ascendant ou descendant. Les fichiers peuvent être placés en dessus ou en dessous des autres. Les changements des fichiers sont aussi annulables individuellement ou totalement. L’utilisateur peut appliquer les changements de la playlist sur les fichiers réels et peut décider quel mode d’application est utilisé entre le renommage, renommage et copie et renommage et déplacement. Finalement, l’utilisateur peut lancer la lecture automatique ou individuelle de la playlist ou des fichiers.

## Maquettes

Pour que l’utilisateur puisse interagir avec l’application, il a besoin d’une interface. Ce paragraphe montre et explique les maquettes utilisées qui décrivent comment l’interface devra être faite. Elles ne représentent pas la manière exacte dans laquelle l’application se présentera mais serviront de guides pour la création du programme.

### Maquette de base



Figure 4 - Maquette de l'application

La maquette ci-dessus et la maquette suivante permettent toutes les actions notées dans le diagramme de cas d’utilisation.

Tout à gauche se trouve le navigateur de dossiers. Il permet à l’utilisateur de naviguer dans les dossiers du PC et d’en sélectionner un. En dessus de lui est noté le nom du dossier dans une boîte de texte. Il devrait être possible de donner son propre chemin.

Au milieu s’affichent les fichiers du dossier sélectionné. Au-dessus, le nom du fichier sélectionné et, au-dessous, la flèche unique pour envoyer la sélection dans la playlist et la double flèche pour envoyer tout le dossier dans la playlist.

La fenêtre à droite représente les fichiers de la playlist avec le chemin du fichier sélectionné au-dessus. La flèche unique pour sortir la sélection. La flèche double pour tout sortir. Le bouton trier pour trier la liste alphabétiquement. Le bouton réinitialiser pour réinitialiser toutes les modifications apportées à la liste. Le bouton Play pour lancer la lecture automatique de la liste et le bouton Stop pour la stopper.

Sous les trois fenêtres, le lecteur de musique se présente et au-dessous de lui, le panel de gestion de la playlist.

Dans le panel, à gauche, la boîte de texte ‘Nom général’ permet de donner un nom à tous les fichiers de la playlist. ‘Chiffre de départ’ contient le numéro de commencement dans la numérotation. La boîte avec un vu, dessous, donne la possibilité d’activer ou de désactiver la numérotation et les trois petits points permettent de changer le mode l’application des changements.

À droite, on peut choisir le dossier de destination de la playlist. On peut choisir d’afficher les changements et de les appliquer dans les fichiers réels.

### Maquette des menus

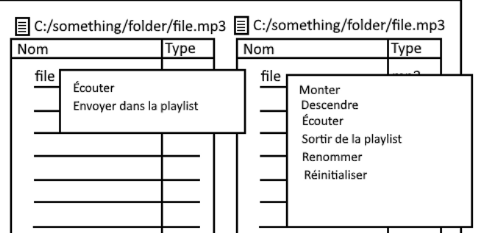


Figure 5 - Maquette des menus

Le menu à gauche est activé lorsque l’on fait un clic droit sur un élément des fichiers du dossier sélectionné. Il donne l’option à l’utilisateur d’écouter le fichier individuellement ou de l’envoyer dans la playlist.

Le menu à droite est activé lorsque l’on fait un clic droit sur un élément de la playlist. Il donne l’option à l’utilisateur de monter et descendre le fichier dans la liste, de l’écouter, de le sortir de la playlist, de le renommer et de réinitialiser les changements appliqués sur lui.

## Diagramme d’activité

Le diagramme d’activité décrit conceptuellement comment les processus du programme devraient se dérouler. Le diagramme étant trop volumineux, seules des parties limitées seront affichées et expliquées. Pour observer le diagramme en son intégralité, allez dans les annexes.

### Gestion des fichiers FLAC

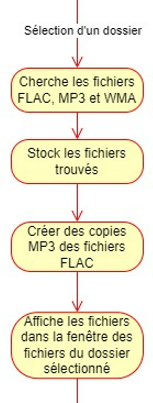


Figure 6 - Sélection d'un dossier

Lorsqu’un dossier est choisi, les fichiers MP3, WAV et FLAC sont sélectionnés et affichés pour pouvoir être lus. Malheureusement, le moyen de lecture utilisé n’est pas compatible avec les fichiers de type FLAC. Alors, une copie temporaire convertie en MP3 de ces fichiers dans un dossier en effectuée. Le lecteur utilisera ces copies durant l’écoute.



Figure 7 - Changement du mode d'application

### Changement du mode d’application

Lorsque l’on active le mode ‘Renommer’, une vérification de l’unicité des noms est entreprise. Quand le mode ‘Renommer et déplacer’ ou ‘Renommer et copier’ sont eux activés, il y a en plus une vérification que l’on a bien sélectionné un dossier et sinon, la fenêtre de navigation des dossiers s’affiche.

### Affichage des changements

L’affichage des changements est l’action qui affiche le résultat du renommage général et de la numérotation.

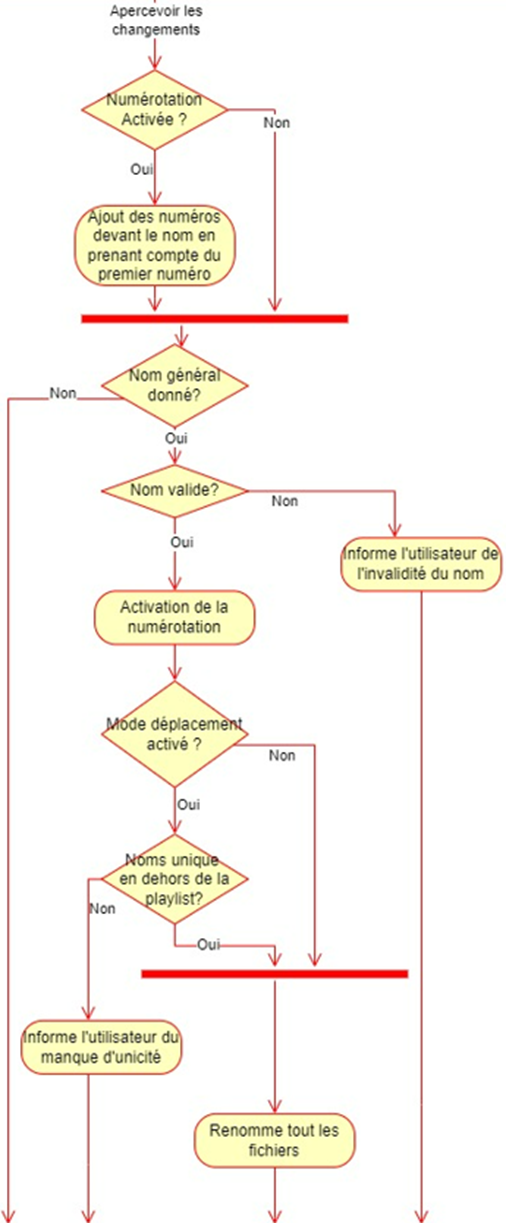


Figure 8 - Affichage des changements

Si un nom a été donné, il sera vérifié s’il est valide pour des fichiers. Puis, la numérotation sera activée pour éviter les doublons. Dans le cas où les fichiers devraient se trouver en dehors de la playlist même avec l’application des changements, alors il sera testé si les noms seront uniques même dans ces conditions. Finalement, s’il n’y a aucune erreur détectée, les noms s’afficheront.

### Application des changements

L’application des changements dans les fichiers et dossiers réels se fait de la façon suivante.

En premier temps, la validité des noms est testée. Puis, le processus se divise en trois.

Dans le renommage, on teste l’unicité des fichiers. On les renomme, puis on informe l’utilisateur du succès.

Dans ‘Renommer et copier’, on vérifie encore une fois l’unicité dans les dossiers originaux et dans la playlist. On renomme les fichiers et on copie les fichiers renommés dans le dossier cible. Finalement, on informe l’utilisateur du succès.  
Dans ‘Renommer et déplacer’, on vérifie seulement l’unicité dans la playlist et on déplace tout en renommant les fichiers dans le dossier cible, puis on informe l’utilisateur.

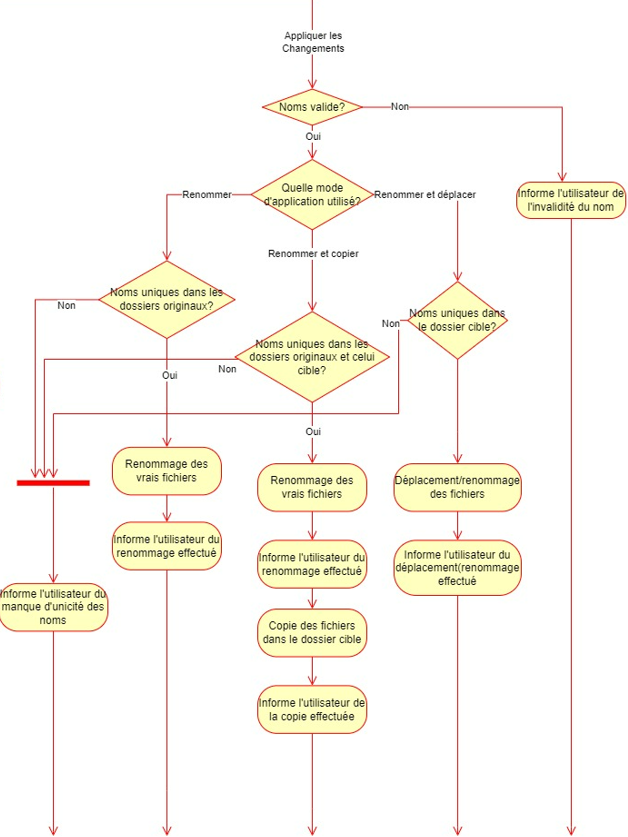


Figure 9 - Application des changements

## Diagramme de classe

Dans ce diagramme de classe, on décrit les méthodes et propriétés nécessaires au fonctionnement de l’application. Bien sûr, des imprévus peuvent faire que le résultat final soit bien différent. Pour voir le diagramme en entier, allez dans les annexes.

### Models

Le ‘model’ est la partie de l’application qui stocke les informations et qui interagit avec les données extérieures pour les rendre utilisables dans le programme.

Dans ce ‘model’, il y a deux classes.

La première, Model, contient les informations générales de l’application tels que la playlist, les chemins des dossiers, le nom général, le nombre et le mode d’application. Il y a aussi les méthodes de vérification et de recherche dans les dossiers.

La deuxième classe est ‘File’. File permet de sauvegarder les changements et les appliquer dans les vrais fichiers.

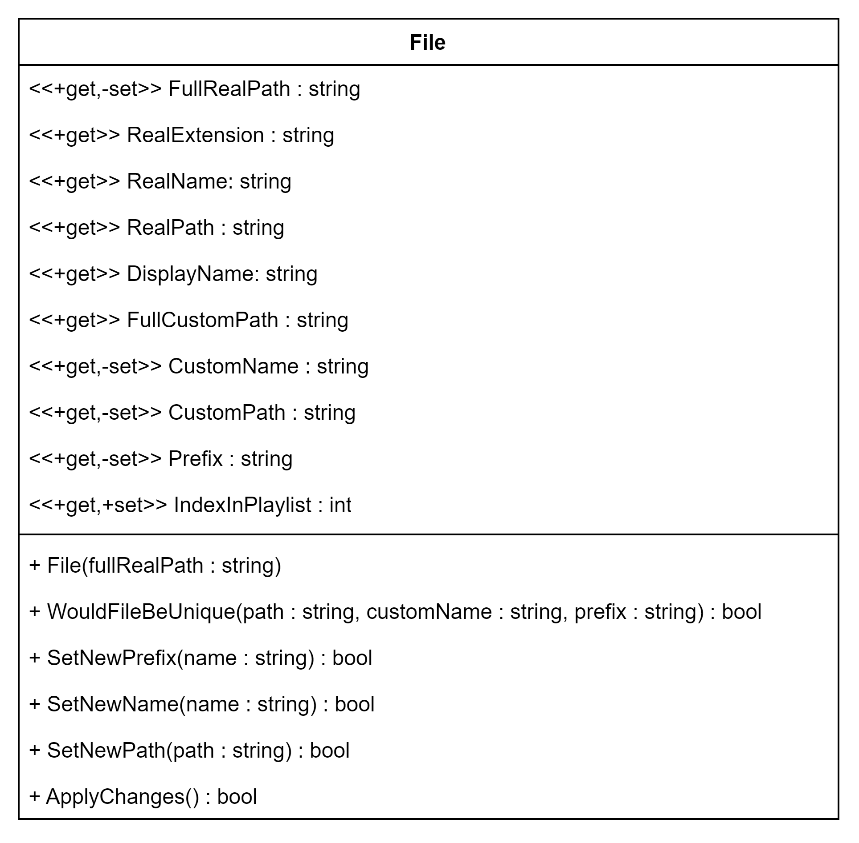


Figure 10 - File (MVC)

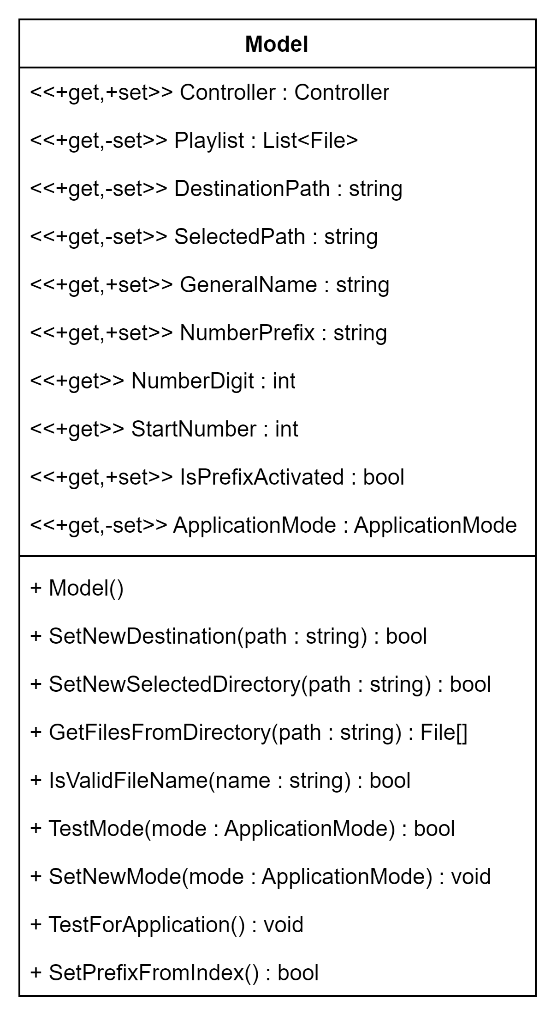


Figure 11 - Model (MVC)

### Views

Les classes ‘Views’ servent comme interface entre l’utilisateur et l’application. Ce sont elles qui représentent et gèrent les formulaires et leurs contrôles.

Le formulaire principal est représenté par la classe ViewForm qui sert aussi à envoyer des pop-ups à l’utilisateur pour l’informer de l’état de l’application ou pour lui demander des confirmations ou des dossiers. La classe contient des instances d’autres classes comme le lecteur de musique, le browser et les listes.

La classe FileItem représente les fichiers sous forme ajoutable dans les listes. Ils contiennent des menus et sont inclus dans des ListView, comme la classe PlaylistView qui montre la playlist et tous ses fichiers.

### Controllers

Le ‘controller’ contrôle les processus du programme et sert d’intermédiaire et de liaison entre les données et l’interface.

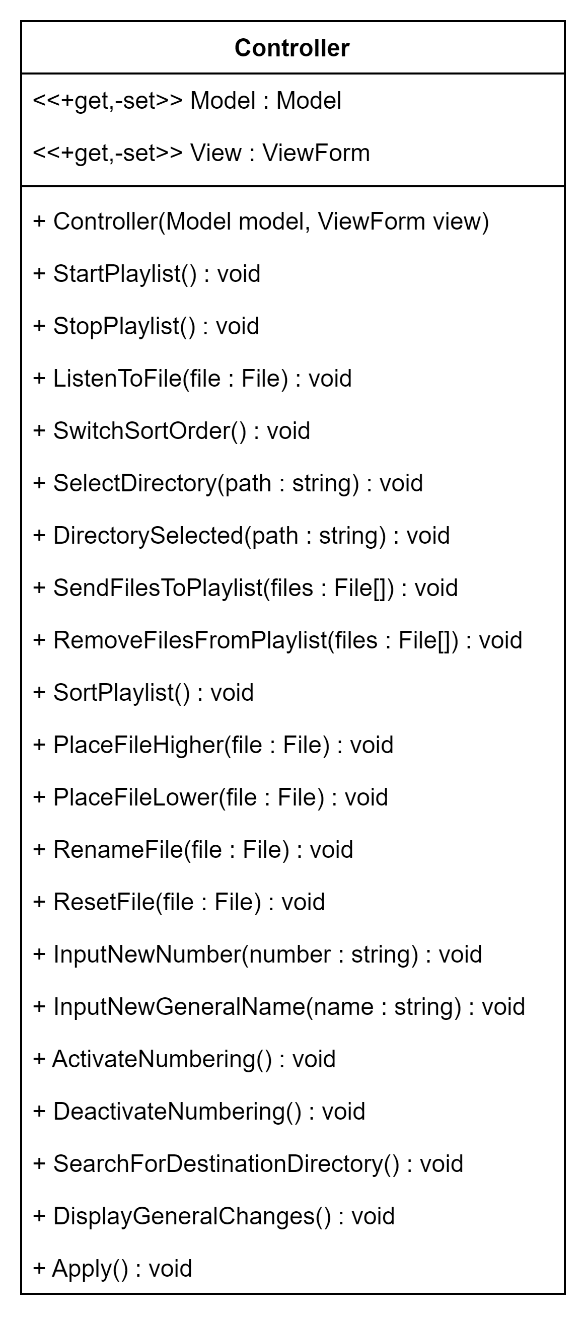


Figure 12 - Controller (MVC)

Toutes les fonctions importantes et les opérations sont guidées par le contrôleur. Il gère la manière dont les fichiers sont ordrés, comment les envoyer d’une liste à l’autre, comment afficher les changements ou encore comment écouter les fichiers.

## Liste des tests

Ici se trouve le tableau des tests pour vérifier que le programme fonctionne correctement.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom du test | Fonctionnalité testée | Description | Condition de réussite |
| Choix du dossier | Choisir un dossier | Tester la possibilité de parcourir les dossiers de la machine et de choisir un dossier. | Le dossier est bien sélectionné. |
| Affichage du contenu du dossier | Afficher les fichiers musiques du dossier | Tester que les fichiers du dossier sélectionné s’affichent dans une fenêtre. | Tous les fichiers du dossier sélectionné de type FLAC, MP3 et WMA devront être affichés. |
| Déplacement d’un fichier dans la playlist | Choisir un fichier pour la playlist | Tester la possibilité d’envoyer un fichier individuel du dossier sélectionné dans la playlist. | Le fichier sélectionné doit se trouver dans la playlist. |
| Sortie d’un fichier de la playlist | Sortir un fichier de la playlist | Tester la possibilité de sortir un fichier individuel de la playlist. | Le fichier ne se trouve plus dans la playlist. |
| Déplacement de tous les fichiers dans la playlist | Choisir tous les fichiers pour la playlist | Tester la possibilité de déplacer tous les fichiers trouvés dans le dossier dans la playlist. | Tous les fichiers du dossier type FLAC, MP3 et WMA se trouvent dans la playlist. |
| Sortie de tous les fichiers | Sortir tous les fichiers de la playlist | Tester la possibilité de sortir tous les fichiers se trouvant dans la playlist. | La playlist est vide. |
| Renommage individuel | Renommer le fichier | Tester la possibilité de renommer un fichier de la playlist | Le fichier est renommé dans la playlist |
| Renommage de playlist | Renommer les fichiers de la playlist | Tester la possibilité de donner un nom général à tous les fichiers de la playlist. | Les fichiers de la playlist ont tous le même nom donné sans compter l’extension et préfixe/suffixe. |
| Numérotation des fichiers | Ajout de digit aux fichiers de la playlist | Tester que les fichiers de la playlist sont numérotés selon l’ordre donné et le numéro de départ donné. | Les fichiers ont comme préfixe/suffixe leur position dans la playlist, le nombre de digit du nombre de départ est gardé. |
| Remontée du fichier | Placer le fichier en dessus | Tester la possibilité de placer le fichier au-dessus des autres dans la playlist. | Le fichier se trouve un cran au-dessus dans la playlist. |
| Descente du fichier | Placer le fichier en dessous | Tester la possibilité de placer le fichier au-dessous des autres dans la playlist. | Le fichier se trouve un cran au-dessous dans la playlist. |
| Triage des fichiers | Trier les fichiers | Tester la possibilité de donner un ordre alphabétique aux fichiers de la playlist. | Les fichiers sont triés de manière alphabétique. |
| Triage ascendant | Changer le triage en ascendant | Tester la possibilité de trier les fichiers avec les premiers en haut et les derniers en bas. | Les premières lettres de l’alphabet et plus petits nombres sont au début de la liste. |
| Triage descendant | Changer le triage en descendant | Tester la possibilité de trier les fichiers avec les derniers en haut et les premiers en bas. | Les dernières lettres de l’alphabet et plus grands nombres sont au début de la liste. |
| Annulation des changements d’un fichier | Annuler les changements du fichier | Tester la possibilité d’annuler le renommage d’un fichier ainsi que sa position personnalisée dans la liste. | Le fichier recouvre son nom d’origine si le changement de nom de la playlist n’est pas actif. Le digit reste s’il est activé. |
| Annulation des changements de tous les fichiers | Annuler tous les changements | Tester la possibilité d’annuler tous les changements effectués tels que les noms personnalisés, le nom général de la playlist et le digit. | Les fichiers recouvrent leur nom d’origine. |
| Modes d’application : Renommer | Changer le mode de création de la playlist | Tester la possibilité de renommer les fichiers originaux. | Dans le mode ‘Renommer’, tous les fichiers ayant des noms qui diffèrent changent de nom dans leurs dossiers originaux. |
| Modes d’application : Renommer et Copier | Changer le mode de création de la playlist | Tester la possibilité de renommer les fichiers originaux et de les copier dans le dossier cible. | Dans le mode ‘Renommer et copier’, tous les fichiers ayant des noms qui diffèrent changent de nom dans leurs dossiers originaux et sont copiés dans le dossier cible. |
| Modes d’application : Renommer et Déplacer | Changer le mode de création de la playlist | Tester la possibilité de déplacer les fichiers dans le dossier cible et de les renommer. | Dans le mode ‘Renommer et déplacer’, tous les fichiers originaux sont déplacés dans le dossier cible et renommés. |
| Lecture de musique | Gérer le lecteur de musique | Tester que les fichiers donnés sont bien lus et peuvent être écoutés. | Des fichiers des types FLAC, MP3 et WMA peuvent tous être lus. |
| Lecture d’un fichier | Lancer la lecture d’un fichier | Tester la possibilité de lire les fichiers de la playlist et du dossier sélectionné. | Quand un fichier est double-cliqué, cela lance sa lecture. |
| Lecture automatique | Lancer la lecture automatique de la playlist | Tester la possibilité de lancer la lecture automatique de la playlist. | Tous les fichiers de la playlist sont lus un par un de haut en bas. |
| Dossier de destination | Choisir un dossier de destination | Tester la possibilité de choisir un dossier de destination. | Lorsque l’on applique le changement des fichiers avec l’option ‘Renommer et copier’, les fichiers sont copiés dans le dossier cible choisi. |
| Application des changements | Appliquer les changements | Tester la possibilité d’appliquer les changements effectués dans l’application. | Les changements sont appliqués de la manière choisie. |
| Modification de fichier | Gérer le vrai fichier | Tester que les fichiers sont bien changés en fonction de ce qui est fait dans l’application. | Ne change rien tant que rien n’est appliqué.  Change le vrai fichier quand les changements sont appliqués. |

# Réalisation

Bien que beaucoup de pensées ont été mises dans la conception, la réalisation pourra toujours différer quand il sera temps de mettre les modèles en pratique. Cette partie va expliquer comment l’environnement a été créé. Comment l’interface et la structure ont été achevées et comment l’application a été programmée.

## Environnement

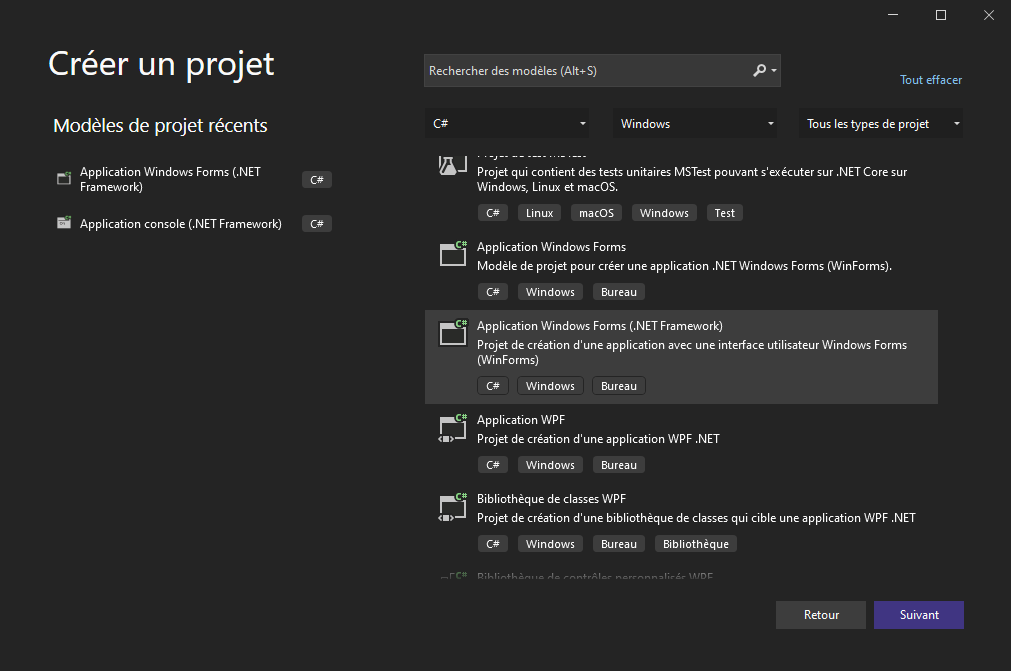


Figure 13 - Choix du modèle de projet

La première chose à faire est de préparer l’environnement dans lequel l’application sera créée. Alors, dans Visual Studio 2022, j’ai créé un projet en utilisant le modèle « Application Windows Forms (.NET Framework) » qui est un modèle permettant l’utilisation de formulaires dans un milieu .NET Framework.

Le projet a reçu le nom Music Sort comme le titre donné dans le CDC et la version du Framework a bien été choisie comme 4.7.2.

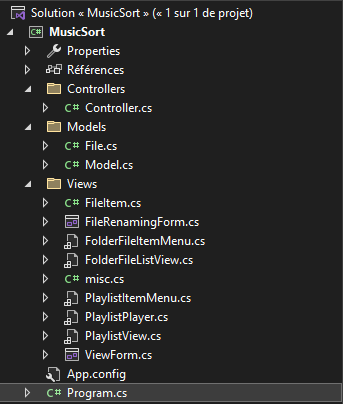


Figure 14 - Fichiers

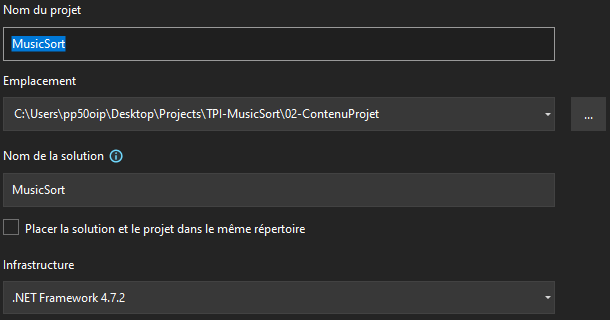


Figure 15 - Création du projet

Un dossier a été créé pour chaque partie du MVC ; Models, Views, Controllers. Pour chaque classe, excepté le ‘Comparer’, un fichier a été dédié. Les autres éléments Views ont été placés dans le fichier misc et l’énumérateur ‘ApplicationMode’ dans la classe Model.

## Interface

L’interface permet à l’utilisateur d’interagir avec l’application. La première chose faite a été de préparer les ‘Controls’ personnalisés à ajouter dans le formulaire, puis les ‘Controls‘ de base ont été ajoutés sur le formulaire principal.

L’interface générale a été créée en prenant comme modèle la maquette faite précédemment. La sélection du dossier de sélection se fait avec le FolderBrowser mentionné plus tard tandis que les fichiers du dossier de sélection et les fichiers de la Playlist sont affichés dans une ListView qui permettent de les lister effectivement. Ces ListView ont deux colonnes qui décrivent le nom et l’extension de chaque fichier.

L’outil qui permet de lire les musiques, quant à lui ,est un Control nommé Windows Media Player qui est accessible en cochant « Windows Media Player » dans l’onglet « Composants COM » de l’option « Choisir les éléments » de la « boîte à outils » de Windows Forms.

Les modes d’application sont choisis à l’aide de controls « RadioButton » qui sont regroupés dans un control GroupBox.

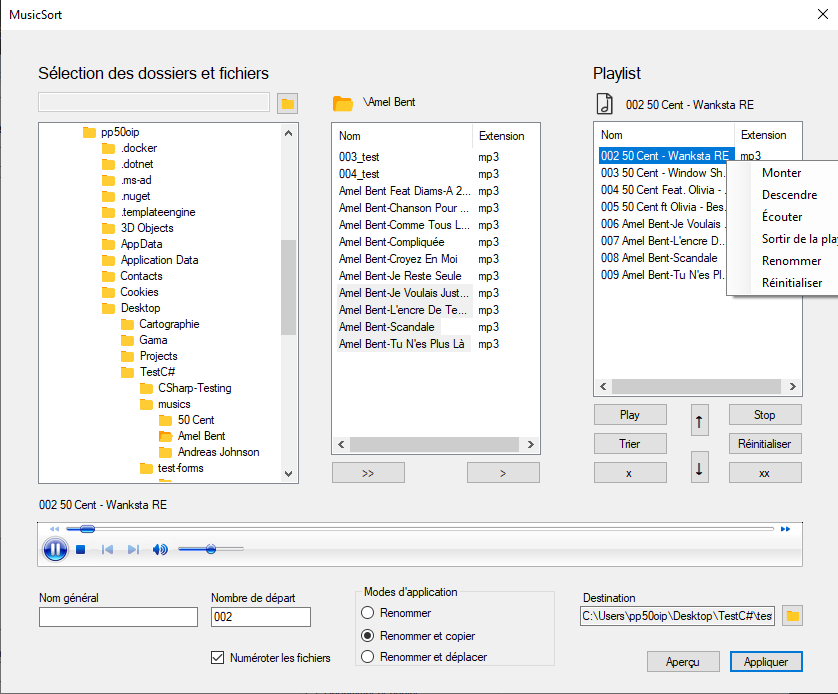


Figure 16 - Interface

Pendant l’ajout, j’ai rencontré mon premier problème. Je suis parti du principe qu’un explorateur de dossier sans ‘Dialog’ existait déjà. Ce qui n’est pas le cas. Il a donc été décidé d’utiliser un ‘TreeView’ à la place pour naviguer dans les dossiers.

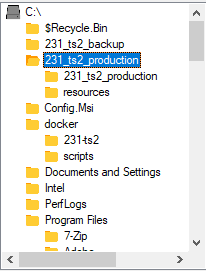


Figure 17 - Navigateur de dossier

La classe créée pour remédier à ça s’appelle FolderBrowser et permet de mettre en place un dossier de base avec ‘SetBaseDirectory()’ dans lequel il est possible de naviguer dans ses enfants et d’en sélectionner un. La détection de la sélection se fait avec un event personnalisé ; ‘FolderSelectedEvent’.

## Structure

La structure du code suit le diagramme de classe UML créé dans la conception ainsi que quelques conventions de nommage et des commentaires.

En premier temps, chaque page directement créée ou modifiée est munie d’un entête décrivant l’auteur, la date de création et la description brève du code.

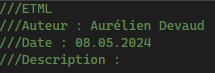


Figure 18 - Résumé du fichier

En second, chaque méthode, champ et propriété sont commentés et suivent la convention de nommage suivante ;

* Les champs sont en « LowerCamelCase » avec un « \_ » au début.
* Les propriétés sont en « UpperCamelCase ».
* Les méthodes sont en « UpperCamelCase » avec un verbe au début du nom.
* Les méthodes qui répondent à un événement sont en « UpperCamelCase » avec le nom de l’objet qui lance l’événement au début et le nom de l’événement à la fin séparés par un « \_ ».
* Les méthodes qui répondent à un délégué sont en « UpperCamelCase » avec le nom de l’action.

### Changements par rapport à la conception

Pour la copie du diagramme, quelques changements ont été effectués ;

La méthode « SelectDirectory » n’accepte plus de chemin mais lance le processus pour en trouver un.

« NumberPrefix » perd aussi son « setter » et son « getter » crée le préfixe par lui-même. « SetPrefixFromIndex » lui rend une liste de chaque fichier et de ses messages d’erreurs.

Pour la classe « File », le constructeur contient maintenant aussi la possibilité de donner un index directement au fichier. Un délégué « FileInfoChangedEventHandler » a aussi été ajouté pour gérer les événements d’informations changées sur le fichier « FileInfoChangedEvent ».

Le constructeur de « FolderFileItemMenu » a été corrigé pour accepter le fichier et les autres délégués.

Dans la classe « FolderFileListView », il a été décidé de ne pas passer les délégués directement mais de les donner après l’initialisation de l’objet. Ce qui explique les « « getters ».

Dans « ViewForm », la méthode « SendFileItemsToPlaylist » n’est pas ajoutée car une méthode de « PlaylistView » se charge déjà de cette tâche.

La méthode de triage alphabétique a été placée dans la classe Model pour ne pas la confondre avec la méthode de triage du « ListView » qui trie en fonction de l’index.

Finalement, tous les événements des interactions avec l’interface ont été ajoutés.

## Comportement du programme

Ici sera décrit comment les différentes fonctions du programme se comportent.

### Navigateur de dossiers

Pour choisir un dossier, il faut utiliser le navigateur de dossier (FolderBrowser). Par défaut, le navigateur affiche les différents ‘drives’.

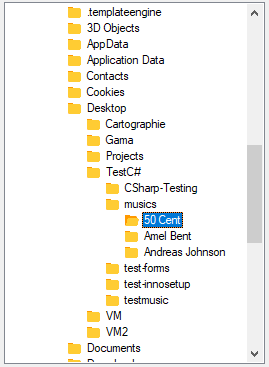


Figure 19 - Navigateur de dossier

Depuis là, en appuyant sur ces répertoires, les répertoires enfants sont affichés et tous les fichiers des types autorisés (‘mp3’,’wma’,’flac’,’aac’,’wav’,’midi’,’asf’) sont récupérés en tant que File et affichés dans des objets ‘FileItem’s dans la fenêtre à gauche, le ‘FolderFileListView’. Le choix du dossier est récupéré grâce à l’événement FolderSelectedEvent.

### Gérance de la playlist

La playlist permet de créer une liste de fichiers se trouvant dans plusieurs dossiers différents et de les écouter dans l’ordre donné.

Pour ajouter des fichiers dans la liste, il y a 3 manières. La première est d’appuyer sur le bouton des doubles flèches pour tout envoyer et la deuxième est le bouton de la flèche unique pour envoyer la sélection. Finalement, il est possible d’envoyer individuellement un fichier en utilisant le menu contextuel. Ces derniers utilisent SendFilesToPlaylist qui récupère les Files des FileItems du FolderFileListView et les ajoute à Playlist et PlaylistView en tant que nouveau FileItems.



Figure 20 - Envoyer

Pour sortir les fichiers de la playlist, les mêmes fonctions sont disponibles mais avec des croix. Cela utilise RemoveFilesFromPlaylist qui supprime les File de Playlist et les FileItems de PlaylistView.



Figure 21 - Suppression

Le tri, quant à lui, se fait de la manière suivante : d’abord, les fichiers sont ordrés selon leur ordre d’ajout, puis lorsque l’utilisateur appuie sur le bouton du tri qui enclenche SortPlaylist(), tous les éléments de la playlist sont ordrés de manière alphabétique et les IndexInPlaylist des Files sont mis à jour pour que les fichiers soient visuellement ordrés par le \_comparer de PlaylistView. Il est alors possible de faire monter ou descendre la sélection avec les flèches ascendantes et descendantes avec PlaceFilesHigher() et PlaceFilesLower() qui se chargent de changer le IndexInPlaylist des Files et de rafraichir le tri de PlaylistView. Additionnellement, il est possible d’inverser l’ordre en appuyant sur l’entête du nom. Les changements de position des FileItems se fait par le \_comparer de PlaylistView qui les ordre selon les IndexInPlaylist des Files qu’ils représentent.



Figure 22 - Changement de position



Figure 23 - Trier

Le renommage des fichiers se fait en allant dans l’option ‘Renommer’ du menu contextuel des fichiers. Il est effectué à l’aide d’un formulaire secondaire qui va vérifier si le nom est juste et informer l’utilisateur de potentielles erreurs. Il est ouvert par la méthode OpenFolderBrowserDialog qui affiche le FileRenamingForm.

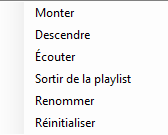


Figure 24 - Menu Contextuel

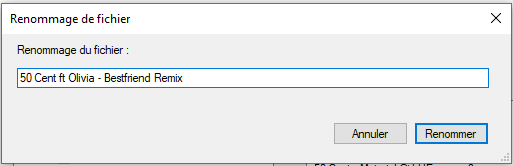


Figure 25 - Formulaire de renommage

Chaque fichier peut être réinitialisé individuellement via le menu qui enclenche ResetFile(). Cela veut dire que son nom re-devient celui d’origine. Sinon, il est possible, avec le bouton ‘Réinitialiser’, de remettre à zéro le nom, le chemin, le préfixe, le mode d’application et l’ordre de tri de tous les fichiers avec ResetAll().



Figure 26 - Tout réinitialiser

Quand une action qui change le nom de l’objet du fichier est entreprise, un événement ,FileInfoChangedEvent, est déclenché qui met à jour tous les objet FileItems liés servant à afficher l’objet.

### Changements généraux

Une fonctionnalité du programme est la numérotation des objets de la playlist et la possibilité de leur donner un nom général.

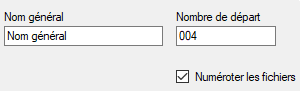


Figure 27 - Changements généraux

Quand la numérotation est activée lorsque l’utilisateur coche l’option, avec ActivateNumbering(), la possibilité de donner un nombre de départ est proposée en réactivant les boîtes de texte du nombre de départ et du nom général. Le programme prendra en compte le premier nombre donné par InputNewNumber(), qui active SetNewStartingNumber() qui lui met en place le StartingNumber et NumberDigit, et commencera la numérotation en utilisant comme repère la propriété IndexInPlaylist des Files. De plus, le nombre total de chiffres, NumberDigit, sera respecté temps que le nombre de fichiers ne dépasse pas le nombre de chiffres donnés.

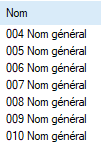


Figure 28 - Renommage général

Pour donner un nom général uniforme dans la playlist, il est impératif de donner un nombre pour garder l’unicité des noms des éléments de la playlist. Donner un nom général lance InputNewGeneralName() activant SetNewGeneralName() qui met en place le nouveau nom général.

Finalement, aucun de ces changements n’est mis en place avant d’avoir appuyé sur le bouton ‘Aperçu’ lançant la méthode DisplayGeneralChanges() qui applique les préfixes et le nom général sur les Files si un nombre est donné.



Figure 29 - Bouton aperçu

### Application des changements

Par défaut, tous les changements amenés aux éléments de la playlist ne sont pas appliqués sur les fichiers qu’ils représentent. Pour les appliquer, il suffit d’appuyer sur le bouton ‘Appliquer’ qui lance Apply().



Figure 30 - Appliquer

Mais d’abord, il faut choisir un mode d’application et il y a en a trois ;

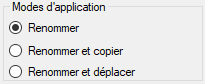


Figure 31- Modes d'application

Le premier mode est celui du renommage qui renomme les fichiers et ne prend pas en compte le dossier de destination. Donc, lorsqu’il est choisi avec ChangeApplicationMode(), les objets des fichiers sont renommés par rapport à leurs noms personalisés tandis que leurs chemins reste conforme à la place des fichiers réels (File.CustomName File.CustomPath). Quand l’utilisateur demande l’application des changements avec Apply(), le programme vérifie, avec TestMode(), si les nouveaux noms pourraient causer des conflitsavec des fichiers du même nom déjà existant et si c’est le cas, le programme demandera à l’utilisateur s’il veut écraser ces doublons.

Le deuxième mode est le mode ‘Renommer et Copier’ qui copie le fichier dans un dossier de destination avec son nom personnalisé. Donc, lorsqu’il est choisi avec ChangeApplicationMode(), on renomme tous les objets des fichiers avec leurs noms personalisés et on leur donne le chemin du dossier de destination. Quand l’utilisateur demande l’application des changements avec Apply(), le programme vérifie, avec TestMode(), si les nouveaux emplacements des fichiers possède déjà des fichiers du même non et si c’est le cas, le programme demandera à l’utilisateur s’il veut les écraser.



Figure 32 - Destination

Le troisième mode est le mode ‘Renommer et Déplacer’ qui se comporte exactement de la même manière que ‘Renommer et Copier’ à l’exception que dans ce mode, le fichier originel est supprimé.

Si des problèmes d’autorisation se produisent, les actions concernées sont annulées.

L’application se fait dans l’ordre suivant ; En premier temps, la présence du dossier de destination est testée si elle est nécessaire, puis, si des changements généraux ont été faits, avec DisplayGeneralChanges(), ils sont affichés. Ensuite, le programme demande à l’utilisateur de confirmer l’action avec AskForConfirmation(). Si l’application des changements est confirmée, les changements seront testés avec TestMode() et les résultats enregistrés. Une nouvelle confirmation sera demandée dans le cas où des complications telles qu’un écrasement de fichier ou un manque de permission auront été détectées.

Après ces multiples premières questions à l’utilisateur, les changements sont appliqués fichier par fichier avec File.ApplyChanges(). Si les fichier demande un écrasement, alors une confirmation est demandée à l’utilisateur. Si des permissions sont manquantes, alors l’application des changements du fichier est annulée.

Finalement, le dossier de destination est réinitialisé et la liste des fichiers du dossier sélectionné, FolderFileListView, est mise à jour.

### Music Player

L’objet du lecteur de musique (MusicPlayer de la classe PlaylistPlayer) est utilisé pour lire les musiques de la playlist. Il hérite du ‘control’ AxWindowsMediaPlayer (ou MediaPlayer) de la librairie AxWMPLib.



Figure 33 - PlaylistPlayer Librairie

Pour accéder au ‘Media Player’, il faut : aller dans la boîte à outils, puis ouvrir le menu contextuel et appuyer sur l’option ‘Choisir les éléments’. Dans l’onglet ‘Composants COM’ de la fenêtre qui s’ouvre il faut finalement cocher l’option ‘Windows Media Player’ pour pouvoir utiliser le composant.



Figure 34 - Windows Media Player

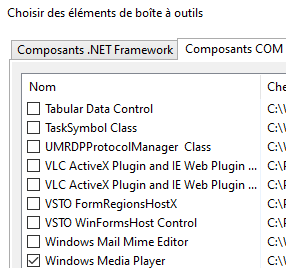


Figure 35 - Composants COM

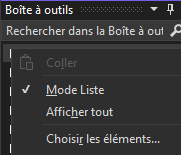


Figure 36 - Choisir les éléments

Par défaut, l’objet ne peut lire que les fichiers musiques de type ‘mp3’, ’wma’, ’aac’, ’wav’, ’midi’, ’asf’ mais dans le Cahier des Charges, il est demandé de pouvoir aussi lire les fichiers de type ‘flac’. Pour régler ce problème, j’ai décidé de créer une copie temporaire du fichier en MP3 quand sa lecture est demandée. Pour parvenir à cela, il a fallu en premier mettre en place un dossier ‘temp\_files’ pour ces fichiers qui se crée automatiquement au lancement de l’application. Ce dossier est nettoyé au lancement et à la fermeture de l’application.

Pour convertir les fichiers Flac en MP3, j’ai suivi les suggestions de ChatGPT et de StackOverflow et utilisé la librairie NAudio.Lame et NAudio.Wave qui, avec le lecteur de données audios AudioFileReader et le rédacteur LameMP3FileWriter, donnent la possibilité de lire le fichier FLAC et de l’écrire sous format MP3 dans un fichier temporaire.

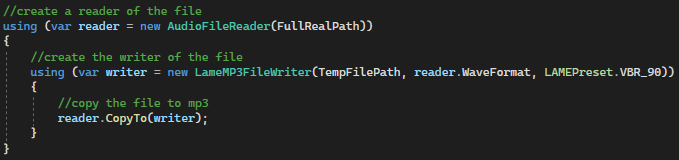


Figure 37 - Conversion FLAC -> MP3

Il y a deux façons de lire des fichiers. La première est de double-clicker le fichier ou d’utiliser le menu contextuel pour lire un fichier unique. Cela va vider la liste des fichiers en attente et lancera la lecture du fichier sélectionné. La deuxième façon est d’appuyer sur le bouton ‘Play’, qui lance StartPlaylist() qui lui lance PlayFiles(), qui mettra en attente en ordre tous les fichiers de la playlist dans le lecteur pour les écouter. Pour arrêter la lecture et vider la liste d’attente il suffit d’appuyer sur le bouton ‘Stop’ qui lance StopPlaylist() et lui StopPlaying(). Un label sur le ‘control’ qui décrit le fichier en cours de lecture se met à jour à chaque nouvelle musique jouée.



Figure 38 - Play Stop

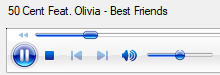


Figure 39 - PlaylistPlayer

## Installeur

Le générateur d’installeur utilisé, InnoSetup, permet de créer un scripte qui, après une configuration, peut générer le fichier d’installation de l’application.

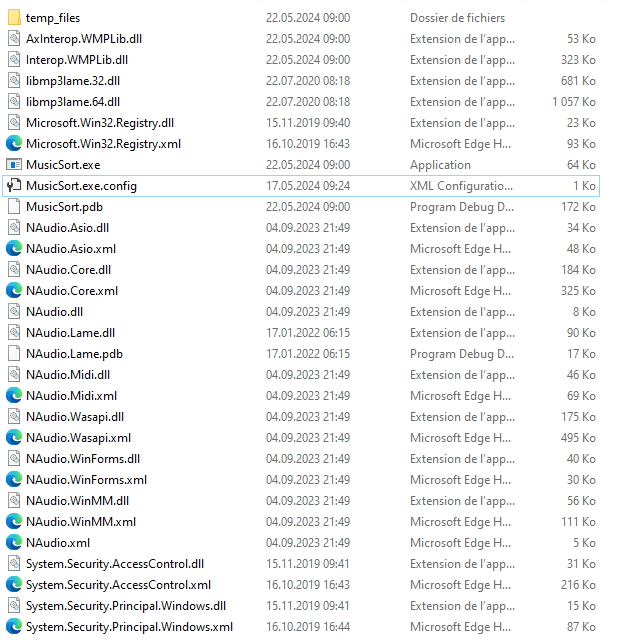


Figure 40 - Fichiers à installer

Pour commencer, il a fallu choisir l’environnement à installer. C’est-à-dire, tous les fichiers du dossier de release de la solution dans le but d’inclure les diverses librairies comme NAudio et autres.

Ensuite, avec l’aide du GUI de InnoSetup, j’ai créé un scripte ‘MusicSort-Setup.iss’ générant un installeur ‘MusicSort-Setup.exe’ pour l’application.

Le script prend donc tous les fichiers choisis dans son onglet ‘[Files]’ et les déploient dans le dossier des programmes de l’utilisateur courant.

Additionnellement, le scripte prend en compte si le Framework .NET 4.7.2 est présent au moment de l’installation de l’application avec la fonction NeedsDotNetFramework. S’il ne l’est pas, il est installé à l’aide de l’installeur ‘ndp472-devpack-fra.exe’ se trouvant dans le même dossier que le scripte.

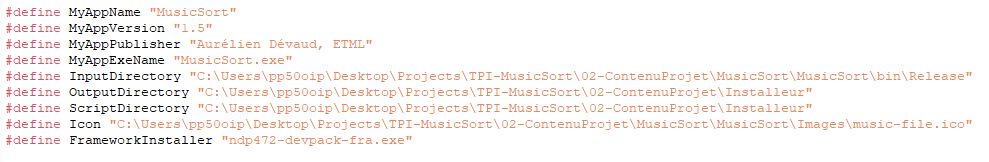


Figure 41 - Scripte, define

Dans le but de pouvoir facilement utiliser le scripte même si des modifications sont apportées à ses éléments, des variables ‘#define’ ont été utilisées.

* InputDirectory est le dossier où l’application et ses dépendances se trouvent.
* OutputDirectory est le dossier où l’installeur est placé après la compilation du scripte
* ScriptDirectory est le dossier où le script se trouve (utilisé pour trouver l’installeur du Framework).
* Icon est l’icône de l’application.
* FrameworkInstaller est l’installeur du Framework.

Finalement, après avoir configuré les variables ci-dessus, il suffit de compiler le scripte pour générer l’installeur.

## Tests des fonctionnalités

Ici, les fonctionnalités du projet sont testées par rapport au tests créé dans la liste de tests de la conception. Ce premier tableau consiste aux tests effectués par moi-même, Aurélien Devaud.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom du test | Date de passation | Test passé ? | Conditions | Commentaire |
| Choix du dossier | 22.05.2024 |  | PC, Windows 10 |  |
| Affichage du contenu du dossier | 22.05.2024 |  | Dossiers avec fichier des types demandés et sans. | Tous les fichiers des types demandés sont affichés. |
| Déplacement d’un fichier dans la playlist | 22.05.2024 |  | Fichiers présents dans le dossier. |  |
| Sortie d’un fichier de la playlist | 22.05.2024 |  | Fichiers présents dans la playlist. |  |
| Déplacement de tous les fichiers dans la playlist | 22.05.2024 |  | Fichiers présents dans le dossier. |  |
| Sortie de tous les fichiers | 22.05.2024 |  | Fichiers présents dans la playlist. |  |
| Renommage individuel | 22.05.2024 |  | Fichier dans la playlist. |  |
| Renommage de playlist | 22.05.2024 |  | Plusieurs fichiers dans la playlist. |  |
| Numérotation des fichiers | 22.05.2024 |  | Plusieurs fichiers dans la playlist. | Il n’y a pas d’option suffixe. |
| Remontée du fichier | 22.05.2024 |  | Plusieurs fichiers dans la playlist. | Le déplacement des fichiers supporte le déplacement de groupe non-liés d’éléments. |
| Décente du fichier | 22.05.2024 |  | Plusieurs fichiers dans la playlist. | Le déplacement des fichiers supporte le déplacement de groupe non-liés d’éléments. |
| Triage des fichiers | 22.05.2024 |  | Plusieurs fichiers dans la playlist. |  |
| Triage ascendant | 22.05.2024 |  | Plusieurs fichiers dans la playlist. | Il est possible d’inverser l’ordre de la playlist. Qu’il soit trié ou non. |
| Triage descendant | 22.05.2024 |  | Plusieurs fichiers dans la playlist. | Il est possible d’inverser l’ordre de la playlist. Qu’il soit trié ou non. |
| Annulation des changements d’un fichier | 22.05.2024 |  | Un fichier renommé avec un préfixe. |  |
| Annulation des changements de tous les fichiers | 22.05.2024 |  | Plusieurs fichiers renommés avec le renommage général et le préfixe. |  |
| Modes d’application : Renommer | 22.05.2024 |  | Des fichiers de plusieurs dossiers dans la playlist renommés. |  |
| Modes d’application : Renommer et Copier | 22.05.2024 |  | Des fichiers de plusieurs dossiers dans la playlist renommés et un dossier de destination choisi. |  |
| Modes d’application : Renommer et Déplacer | 22.05.2024 |  | Des fichiers de plusieurs dossiers dans la playlist renommés et un dossier de destination choisi. |  |
| Lecture de musique | 22.05.2024 |  | Des fichiers des types FLAC, MP3 et WMA sont disponibles à la lecture. |  |
| Lecture d’un fichier | 22.05.2024 |  | Des fichiers dans le dossier et la playlist. |  |
| Lecture automatique | 22.05.2024 |  | La playlist possède plusieurs éléments. |  |
| Dossier de destination | 22.05.2024 |  | Un dossier cible est choisi. |  |
| Application des changements | 22.05.2024 |  | Des fichiers sont ajoutés dans la playlist et renommés. |  |
| Modification de fichier | 22.05.2024 |  | Des fichiers sont ajoutés dans la playlist et renommés. |  |

En addition à ces tests exécutés par moi-même, Émilien Charpié s’est aussi chargé de faire une bêta test et n’a pas trouvé de problème particulier avec l’application.

## Liste des documents fournis

* Fichiers initiaux :  
   TPI-MusicSort\01-Fichiers-Initiaux
  + Cahier des charges :   
    D-tpi-Cdc-DLS-Devaud\_Aurelien-MusicSort.docx
  + Installeur d’InnoSetup :   
    innosetup-6.2.2.exe
* Contenu du projet :
* TPI-MusicSort\02-ContenuProjet
  + Installeur :  
     02-ContenuProjet\Installeur
    - Dossiers des versions :   
      Installeur\Versions
    - Scripte :   
      MusicSort-Setup.iss
  + Projet :   
    02-ContenuProjet\MusicSort
    - Solution :   
      MusicSort.sln
* Documentation :   
  TPI-MusicSort\03-Documentation
  + Planification/Journal de travail sous format Excel :   
    T-TPI-MusicSort-planification-AurelienDevaud.xlsm
  + Rapport :   
    TPI-MusicSort-AurelienDevaud-Documentation.docx
  + Maquette :   
    TPI-MusicSort-AurelienDevaud-Maquette.png
  + Maquette avec menu :   
    TPI-MusicSort-AurelienDevaud-Maquette-Menu.png
  + Diagrammes : 03-Documentation\UML-Diagrams
    - Diagramme UML de cas d’utilisation :   
      D-TPI-MusicSort-AurelienDevaud-UseCaseDiagram.drawio  
      D-TPI-MusicSort-AurelienDevaud-UseCaseDiagram.jpg
    - Diagramme UML d’activité :   
      D-TPI-MusicSort-AurelienDevaud-ActivityDiagram.drawio  
      D-TPI-MusicSort-AurelienDevaud-ActivityDiagram.jpg
    - Diagramme UML de classe initial :  
      D-TPI-MusicSort-AurelienDevaud-ClassDiagram.drawio  
      D-TPI-MusicSort-AurelienDevaud-ClassDiagram.jpg

# Conclusions

Ici, les fonctionnalités sont résumées. La planification et le journal de travail comparés. Les différences entre la conception et la réalisation expliquées et mon avis personnel donné.

## Bilan des fonctionnalités

Pour ce projet toutes les fonctionnalités demandées dans les points techniques évalués ont été terminées.

1. Il est possible de naviguer dans des dossiers et de donner une base pour la navigation. Les fichiers des dossiers disponibles pour la lecture sont tous affichés. Ils peuvent être ajouté et supprimé de la liste de lecture ainsi qu’être réorganisé et trié avec l’option de déplacer les fichiers dans la liste en groupe ou seul de haut en bas.
2. Les fichiers musiques peuvent être lu avec un double clic ou avec le menu contextuel (clic droit). Les formats MP3, WMA, AAC, WAV, MIDI, ASF sont supporté et il est possible de lancer la lecture de liste entière. Dans ce cas la navigation entre les musiques est possible. Un bouton permet de stopper la lecture.
3. Il est possible de renommer individuellement et de donner un nom général aux fichiers de la liste. Il est possible d’activer la numérotation dans tel cas elle respectera le nombre de chiffre et le nombre de départ donnés. Toutes ces changements peuvent être réinitialisés.
4. Il est possible de renommer les fichiers d’origines. L’application cause une validation et chaque écrasement de fichier avec un nom en conflit causent aussi une validation. La fin de l’application des nouveaux noms lance un message d’avertissement.
5. Il est possible de copier et de renommer les fichiers originaux. Les écrasements causent des validations. Les fichiers originaux existent toujours et la fin du processus lance un message d’avertissement.
6. Il est possible de déplacer et renommer les fichiers originaux. Les écrasements causent des validations et la fin du processus cause un avertissement.
7. Finalement, un installeur est généré à l’aide d’un scripte qui permet d’installer l’application sur Windows 10 et 11. L’application fonctionne avec 64 bits et le nom est pertinent.

## Bilan de la planification

La planification a été assez bien faite car le projet s’est déroulé d’une manière assez similaire à elle. Les différences sont assez marginales et compensées par d’autre tâches.

La plus grande différence est le temps supplémentaire qu’a pris la création de l’interface. Cela provient du navigateur sans ‘Dialog’ de dossier que j’ai dû faire moi-même.

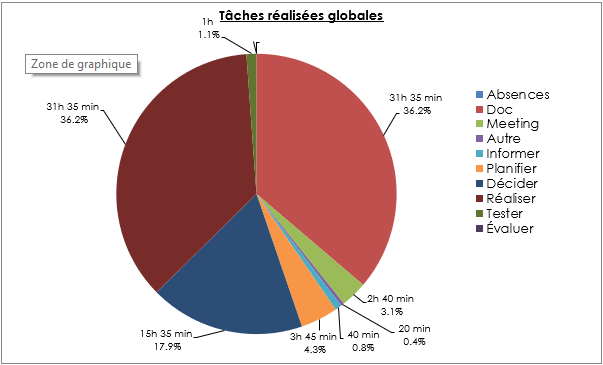


Figure 42 - Temps des tâches réalisées

## Analyse post-projet

Ici sont expliquées quelques différences entre la conception et la réalisation ainsi que quelques améliorations possibles.

En premier temps l’interface diffère de la maquette, des titres ont été ajouté pour clarifier les listes et des positions ont été réajustées pour aligner les controls.

En deuxième temps, les FileItems sont créé dans la classe ViewForm contrairement au modèle de classe où ils sont créés dans FolderFileListView pour être plus tard transféré dans PlaylistView. Cela a été changé car les FileItems ne peuvent pas être dans plusieurs listes à la fois.

Un dernier changement fu comment les éléments de la playlist sont trié et gérer presque entièrement par le Model contrairement à la conception où c’est le View qui s’en charge. Le changement a été appliqué car l’ordre permet de donner les bons numéros aux éléments et c’est plus pratique si le model s’en charge.

Sinon, comme améliorations seraient de donner à l’utilisateur l’option de trier les fichiers avec leur propre filtre à la place d’avoir tous les types lisibles affichés. Une barre de recherche ou autre serait aussi une bonne idée.

## Améliorations possibles

Quelques améliorations possibles pour l’application seraient par exemple le rafraichissement des dossiers dans le navigateur après réouverture, la clarification des noms des modes d’application pour mieux comprendre leurs fonctions ainsi que l’affichage de l’icône dans le menu de démarrage.

## Bilan personnel

Ce fu un projet intéressant pour ma part. J’ai pu enrichir mes connaissances dans les fichiers musiques ainsi que la façon dont les lires et les convertir.

Je suis assez satisfait du résultat de mon travail.

# Glossaire

Ci-dessous sont expliqués les termes utilisés pouvant porter à confusion. Les définitions sont pour la majeur partie générée par ChatGPT.

|  |  |
| --- | --- |
| Sujet | Définition |
| #define | Un outil qui permet de créer des variables dans un scripte ISS qui s’applique pour la compilation. |
| Boîte à outils | Une fenêtre dans un environnement de développement intégré (IDE) qui contient des outils et des contrôles que les développeurs peuvent glisser-déposer dans leur interface utilisateur. |
| Commit | Une opération dans le système de contrôle de version Git qui enregistre les modifications apportées au code dans le dépôt, créant ainsi un instantané de l'état actuel du projet. Chaque commit inclut un message descriptif et un identifiant unique pour permettre de suivre l'historique des modifications. |
| Control | Un élément graphique dans une interface utilisateur, tel qu'un bouton, une zone de texte, une liste déroulante, etc., qui permet à l'utilisateur d'interagir avec le programme. |
| Délégué | Un type de données qui représente une référence à une méthode avec une liste de paramètres spécifiée et un type de retour. |
| Dialog | Une fenêtre modale affichée à l'utilisateur pour obtenir des informations ou des instructions supplémentaires, ou pour lui demander de prendre une décision. |
| Drives | Les périphériques de stockage sur un ordinateur, tels que les disques durs, les lecteurs flash et les lecteurs optiques. |
| Événement | Une action ou un changement qui se produit dans un programme, auquel le programme peut répondre par l'exécution de code spécifique. |
| Famework | Un ensemble de bibliothèques, de composants et d'outils logiciels qui fournissent une structure de développement pour créer et gérer des applications logicielles. |
| Getter | Une méthode utilisée pour récupérer la valeur d'une propriété d'un objet. |
| GroupBox | Un conteneur d'interface utilisateur qui regroupe des contrôles associés et les affiche ensemble avec une bordure. |
| Innosetup | Un logiciel d'installation pour Windows qui permet de créer des programmes d'installation pour les applications développées sous Windows. |
| Installer | Un programme qui installe des logiciels sur un ordinateur en copiant les fichiers nécessaires, en configurant les paramètres, et en effectuant d'autres tâches nécessaires pour préparer le logiciel à l'exécution. |
| ListView | Un contrôle d'interface utilisateur qui affiche une liste d'éléments avec des colonnes pouvant être triées et personnalisées. |
| LowerCamelCase | Une convention de nommage où chaque mot dans un identifiant commence par une lettre minuscule, sauf le premier mot. |
| Objet | Une instance d'une classe dans la programmation orientée objet, contenant des données et des méthodes pour manipuler ces données. |
| RadioButton | Un type de contrôle d'interface utilisateur qui permet à l'utilisateur de sélectionner une option parmi un ensemble d'options mutuellement exclusives. |
| Setter | Une méthode utilisée pour attribuer une valeur à une propriété d'un objet. |
| SubItem | Un élément individuel dans un contrôle ListView qui correspond à une colonne spécifique pour un élément donné. |
| TreeView | Un contrôle d'interface utilisateur qui affiche une hiérarchie de données sous forme d'arbre, généralement utilisé pour représenter des structures hiérarchiques telles que les répertoires de fichiers. |
| UpperCamelCase | Une convention de nommage où chaque mot dans un identifiant commence par une lettre majuscule, y compris le premier mot. |
| Windows Forms | Une technologie de développement d'interface utilisateur pour les applications Windows utilisant le langage de programmation C#. |

# Table des illustrations

[Figure 1 - Image titre 1](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870772)

[Figure 2 - Répartition des tâches planifiées 6](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870773)

[Figure 3 - Diagramme de cas d'utilisation 9](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870774)

[Figure 4 - Maquette de l'application 12](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870775)

[Figure 5 - Maquette des menus 13](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870776)

[Figure 6 - Sélection d'un dossier 14](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870777)

[Figure 7 - Changement du mode d'application 14](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870778)

[Figure 8 - Affichage des changements 15](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870779)

[Figure 9 - Application des changements 16](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870780)

[Figure 10 - File (MVC) 17](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870781)

[Figure 11 - Model (MVC) 17](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870782)

[Figure 12 - Controller (MVC) 18](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870783)

[Figure 13 - Choix du modèle de projet 23](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870784)

[Figure 14 - Fichiers 23](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870785)

[Figure 15 - Création du projet 23](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870786)

[Figure 16 - Interface 24](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870787)

[Figure 17 - Navigateur de dossier 25](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870788)

[Figure 18 - Résumé du fichier 25](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870789)

[Figure 19 - Navigateur de dossier 26](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870790)

[Figure 20 - Envoyer 26](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870791)

[Figure 21 - Suppression 26](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870792)

[Figure 22 - Changement de position 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870793)

[Figure 23 - Trier 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870794)

[Figure 24 - Menu Contextuel 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870795)

[Figure 25 - Formulaire de renommage 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870796)

[Figure 26 - Tout réinitialiser 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870797)

[Figure 27 - Changements généraux 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870798)

[Figure 28 - Renommage général 27](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870799)

[Figure 29 - Bouton aperçu 28](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870800)

[Figure 30 - Appliquer 28](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870801)

[Figure 31- Modes d'application 28](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870802)

[Figure 32 - Destination 28](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870803)

[Figure 33 - PlaylistPlayer Librairie 29](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870804)

[Figure 34 - Windows Media Player 29](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870805)

[Figure 35 - Composants COM 29](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870806)

[Figure 36 - Choisir les éléments 29](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870807)

[Figure 37 - Conversion FLAC -> MP3 29](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870808)

[Figure 38 - Play Stop 30](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870809)

[Figure 39 - PlaylistPlayer 30](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870810)

[Figure 40 - Fichiers à installer 30](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870811)

[Figure 41 - Scripte, define 30](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870812)

[Figure 42 - Temps des tâches réalisées 36](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870813)

[Figure 43 - Interface numérotée 41](file:///C:\Users\pp50oip\Downloads\TPI-MusicSort-AurelienDevaud-Documentation-2-1-2.docx#_Toc167870814)

# Manuels

Ce chapitre explique comment utiliser l’application.

## Manuel d'Installation

Pour installer l’application, allez dans le dossier ‘TPI-MusicSort\02-ContenuProjet\Installeur\Versions’ et choisissez la version qui vous convient et lancez l’installeur. Il est conseillé d’utiliser la dernière version proposée car elle aura moins de chance d’avoir des bugs.

## Manuel d'Utilisation

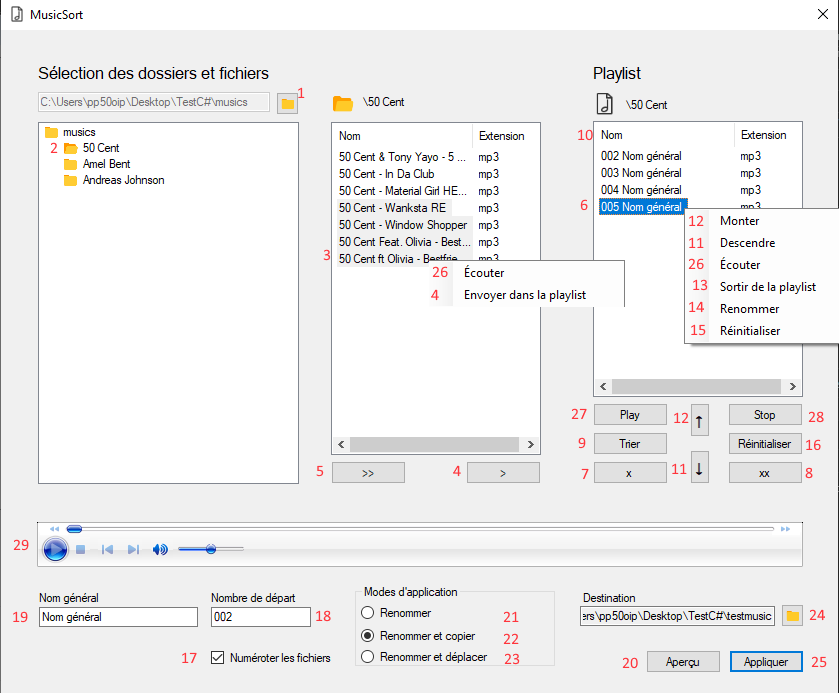


Figure 43 - Interface numérotée

Le manuel d’utilisation

1. Choix du dossier ‘racine’ du navigateur de dossier
2. En cliquant sur ces dossiers, leurs enfants sont affichés et les fichiers compatibles à la lecture sont affichés dans la liste à droite.
3. Fichier du dossier sélectionné. Un double clic démarre l’écoute du fichier.
4. Envoie les fichiers sélectionnés dans la playlist.
5. Envoie tous les fichiers dans la playlist.
6. Élément de la playlist. Un clic droit ouvre son menu contextuel. Un double clic lance sa lecture.
7. Sort les éléments sélectionnés de la playlist.
8. Sort tous les éléments de la playlist.
9. Trie les éléments de la playlist par ordre alphabétique
10. En cliquant sur nom, l’ordre est inversé.
11. Descend les fichiers sélectionnés d’un cran.
12. Monte les fichiers sélectionnés d’un cran.
13. Sort le fichier de la playlist.
14. Ouvre la fenêtre de renommage du fichier.
15. Réinitialise le nom du fichier.
16. Réinitialise tous les noms des fichiers ainsi que le renommage général.
17. Permet d’activer la numérotation
18. Nombre de départ utilisé pour donner un préfixe aux noms. Le nombre de chiffre total sera respecté.
19. Nom général à donner au fichier. Il est impératif de numéroter les fichiers si un nom général est donné.
20. Applique les changements généraux sur les éléments (pas les fichiers !).
21. Mode d’application des changements qui renomme les fichiers originaux en fonction du nom des éléments.
22. Mode d’application des changements qui copie et renomme les fichiers en fonction du noms des éléments et du dossier de destination choisi.
23. Mode d’application des changements qui déplace les fichiers dans le dossier de destination choisi et les renomme en fonction de leur nom.
24. Permet de choisir un dossier de destination.
25. Applique les changements. Chaque écrasement demandera une confirmation individuelle.
26. Lance la lecture individuelle du fichier.
27. Lance la lecture de la playlist dans l’ordre donné.
28. Stop la lecture de la playlist et vide la file d’attente d’écoute.
29. Controls pour la lecture de musique.

# Annexes

Ici, je résume le rapport, cite mes sources et donne l’emplacement du projet.

## Résumé du rapport du TPI / version succincte de la documentation

Pour commencer le projet, j’ai reçu un cahier des charges spécifiant plusieurs objectifs et demandes. Une maquette d’exemple a aussi été fournie dans le but de donner une meilleure idée sur le projet. Le but est de programmer une application capable de créer une playlist en naviguant dans des dossiers et en sélectionnant des fichiers pour les ajouter dedans, les trier, les renommer et d’enregistrer ces changements. Bien sûre, l’application devait être capable de lire la playlist.

Pour m’y prendre, j’ai d’abord choisi la méthode des six pas. Je me suis informé sur les controls Windows Forms à utiliser ainsi que la lecture des types de fichiers demandés (FLAC, WMA, MP3). J’ai planifié le projet en prenant en compte ses six parties et estimé les heures selon mes expériences passées. Pour la conceptualisation, j’ai créé un diagramme de cas d’utilisation pour mettre au clair les fonctionnalités à programmer. J’ai réalisé un diagramme d’activité pour déterminer le comportement du programme et j’ai conçu un diagramme de classe pour définir la structure du code qui suit le model MVC. La dernière partie de la conceptualisation fut la liste de tests à performer.

Le résultat de ce projet fut une application permettant de naviguer les dossiers et d’afficher leurs fichiers compatibles à la lecture (MP3, WMA, FLAC, AAC, WAV, MIDI, ASF). Ses fichiers sont écoutables à l’aide du control Windows Media Player et peuvent être placé dans la playlist. Les objets de la playlist sont classables dans l’ordre alphabétique mais il est aussi possible de les déplacer dans la liste. Les noms des éléments peuvent être renommé soit individuellement ou généralement, dans ce cas les fichiers seront numérotés selon un nombre donner. Finalement, l’application supporte l’application des changements sur les fichiers sous la forme d’un renommage des fichiers originaux, d’une copie et renommage ou d’un déplacement et renommage. La lecture des éléments FLAC s’est effectuée en faisant une conversion des fichiers en MP3 dans un dossier de fichiers temporaire à l’aide de la librairie NAudio qui a lus le fichier en FLAC et réécris en MP3. Quant au navigateur de dossier, il a été créé avec un control TreeView.

## Sources – Bibliographie

Ici sont listées les sources des ressources utilisées durant le projet.

[Folder Icons, Logos, Symbols – Free Download PNG, SVG (icons8.com)](https://icons8.com/icons/set/folder)

[Drive Icons, Logos, Symbols – Free Download PNG, SVG (icons8.com)](https://icons8.com/icons/set/drive)

[Open folder Icons, Logos, Symbols – Free Download PNG, SVG (icons8.com)](https://icons8.com/icons/set/open-folder)

[Music file Icons, Logos, Symbols – Free Download PNG, SVG (icons8.com)](https://icons8.com/icons/set/music-file)

## Archives du projet

Le projet est stocké dans GitHub en privé sous ce lien : [AureDeva/TPI-MusicSort (github.com)](https://github.com/AureDeva/TPI-MusicSort)