Rapport POO

Membre du groupe

Ulysse Perret

Lorenzo DEL ROSSI

Theo Sctrick MeMo

# Sommaire

[**Sommaire**](#_tfx3my8qztze) **1**

[**Organisation**](#_ga8bpxve17kc) **1**

[**Problématique de Coordination**](#_7s5lukeceet4) **2**

[**Point Technique**](#_y21yqv3gqgou) **2**

[Analyse du sujet](#_yefnora6f2gb) 2

[Format](#_xur34ry6t9eq) 3

[**Point Graphique**](#_hpqjqqn4tyq2) **4**

# Organisation

Initialement pour l’organisation nous avons respecté les consignes et Lorenzo a pris la tête du projet POO en tant que chef de projet.

Cependant le projet est partie très rapidement dans une complication d’organisation et nous ne comprenons pas le sujet.

Nous avons tenté de relire chacun de notre côté le projet, essayer de faire des réunions Discord ensemble.

L’objectif premier du projet est de faire avancer le projet autant que possible et de réaliser le site avec le cahier des charges fourni.

Nous sommes partis sur une organisation via Discord et Google Doc au vu de la simplicité que cela offre.

Nous avons choisi Discord car il permet de mieux communiquer ensemble.

De plus cela permet une collaboration étroite et dans la mesure de nos compétences tout en respectant nos calendriers personnelle respectifs.

En effet Ulysse Perret était en alternance ce qui fait qu’il n’était pas présent tout le temps. Il avait un rythme de 2 semaine en entreprise et 2 semaine de cours

Nous avons essayé de communiquer dès que possible dans la mesure du possible et notamment dans les horaires de cours . Malheureusement, nous avons eu des problèmes de coordination entre nous notamment dû au fait de nos calendriers différent respective, et des problèmes personnels de chacun. Je me suis permis de mettre en avant ce point plus en détail après.

Nous avons choisi d'écrire sur Google doc pour permettre la mise en commun ensemble.

Nous avons établi un Gantt et voici l’organisation des tâches qu’on avait prévu de départ, que vous retrouvez en Annexe 1. Dans cette annexe nous avons écrit aussi les changements éventuellement par rapport au gant établie du 5 mars et les éventuelles évolutions.

Cependant, nous avons eu des difficultés à faire le projet notamment suite à un problème d'organisation et nous avons remarqué un manque de compétence quand nous avons voulu faire le projet.

Au départ nous étions parties pour faire des échanges vocales, des réunions hebdomadaire pour suivre l'évolution de chacun

Cependant cela ne nous a pas empêchés de partager des compétences entre nous.

Pour nous aider en termes de communication nous avons pu bénéficier de l’aide du professeur Didier notamment.

Il est à noter que suite à la réunion du 22 mai avec Jean Luc Bourdon, nous avons pu constater que nos tâches n'étaient pas toutes faites et que nous n'avons pas pu présenter un rendu final. De plus, au vue du retard sur l’autre projet, nous avons décidé avec l’accord de Jean Luc Bourdon de nous concentrer sur le projet web.

Le projet POO a été donc laissé de côté et nous nous sommes concentrés sur le projet Web.

# Problématique de Coordination

Il est clair que nous n’avons pas assez eu suffisamment de moyens ou de méthodes pour nous coordonner. Ce manque de coordination s’est fait ressentir notamment dans le partage des tâches, dans la cohésion de groupe ou dans le manque de résultat.

Malgré tout, nous sommes partis initialement sur un partage en commun sur le projet.

Finalement, cela c’est résulter sur le partage de connaissances et de tutoriels pour apprendre plus.

Même si le Gantt a été répartie dans le temps, le fait du manque de compétences, de motivation et de coordination nous a tous impactés.

C’est pourquoi nous n’avons pas respecter le GANTT finalement et fait de notre mieux chacun de notre côté, tout en essayant de prendre en compte les remarques des uns des autres.

# 

# Point Technique

## Analyse du sujet

Le gros point que nous avons passé presque 2 mois à comprendre le sujet et créer l’UML présent.

En effet au début nous avons étudié les mockups POO mis en place.

Pour le logiciel nous avons décidé d'utiliser Eclipse et notamment la version JAVA 11.

Nous avons aussi tenté de faire notre propre

Nous avons réfléchi à la façon de procéder, noter et prendre en compte en générale des point vu.

L’analyse et faire l’UML a été un travail commun fait avec Lorenzo et Ulysse.

La recherche de l’UML a été fait en Annexe 1

## Format

Un impératif du cahier des charges à été d'étudier le format MP3 et XSPF, en sachant que le format XSPF a été imposé.

Le format MP3 permet d’écouter de la musique.

Le format XSPF est un format particulier pour la musique et qui compile dans un format objet, c'est-à- dire un format qui permet de lister des paramètres avec ces clefs.

Nous avons nommé une complication dans l’extraction des données des fichiers transmis.

Avec L’analyse de fichier, nous avons pu notamment mieux déterminer la classe Donnée brute Morceau.

Le gros point problématique a été de créer un héritage et de tester les objets MP3.

Nous avons essayé d’appliquer les cours , essayer de reprendre l’exercice répertoire sans succès.

Nous n’avons pu que déterminer les données que nous utiliserons pour les morceaux.

les voici :

Classe DonnéeBruteMorceau

//String

**public** String ouverture;//Ouverture du morceau

**public** String titre;//titre

**public** String auteurs; //auteur

**public** String canaux\_audio; //caunaux audio

**public** String taux\_echantillonnage; //méthode de musique

**public** String album; //album

**public** String genre; //genre de musique

**public** String compositeur; //compositeur

**public** String taille; //taille du fichier

//Donnee int

**public** **int** annee; //Annee

**public** **int** duree; //duree du morceau

**public** **int** datecreation; //date de creation

**public** **int** datemodifier; //date de la modificaiton

## CLI

Nous avons essayé de réfléchir sur le côté CLI, ou de l’interface de commande du projet. Cependant on n’est pas arrivé, et nous avons dû nous contenter d’un test sur Eclipse directement et avec la main.

De plus nous n’avons pas pu fournir un Jar et un GUi

Pour rappelle

Un JAR pour le fonctionnement en mode console cli.jar et un autre pour le fonctionnement en mode graphique gui.jar

## API

Nous avons essayer les apis suivant :

<https://sourceforge.net/projects/jd3lib/>

<https://github.com/mpatric/mp3agic>

De plus en plus des api, nous avons aussi tenter de faire un projet Maven en suivant <http://blog.paumard.org/tutoriaux/eclipse-maven/>

# Point Graphique

Initialement, le cahier des charges devait faire une interface graphique. Cependant car on manquait de temps et du fait des points de blocage précédemment, nous ne l'avons pas fait. Initialement on pensait le faire en mai, mais au vue de la réunion avec Jean-Luc Bourdon nous somme arrêtés ici et de plus nous étions bloquées sur une tâche plus simple qui était l'héritage et la coordination entre les classes.

# Point de blocage

Le gros point de blocage qui fait que nous n’avons pas réussi a avancer a été l'Héritage et la coordination entre les classes.

De plus nous avons aussi eu un point de blocage pour l’extraction de la musique malgré nos recherches respectives.

# Ressources utilisé

Travaille en commun :

jamboard

<https://jamboard.google.com/d/1GmnC6Xz5TV7fXT_H6cvokWz2OFh89Qu0_nSbDUeWve4/viewer?f=3>

Trello

<https://play.google.com/store/apps/details?id=com.trello>

Google doc

<https://docs.google.com/document/d/1btO8_oXZQXqYmRCiVXyQlslasogtGF6RVm2HkWUccN4/edit>

Musique

<https://www.auboutdufil.com/>

Recherche de bibliothèque pour extraire musique

<https://sourceforge.net/projects/jd3lib/>

<https://github.com/mpatric/mp3agic>

Dossier drive

<https://drive.google.com/drive/folders/1zaxqFY15ImfylaULvwQaD4wp-SaouSiq?usp=sharing>

Eclipse

<https://openclassrooms.com/fr/courses/6106191-installez-votre-environnement-de-developpement-java-avec-eclipse>

Maven

<http://blog.paumard.org/tutoriaux/eclipse-maven/>

Tuto

<https://youtu.be/cfxJOZ-dbMc>

Wiki

MP3 : <https://fr.wikipedia.org/wiki/MP3>

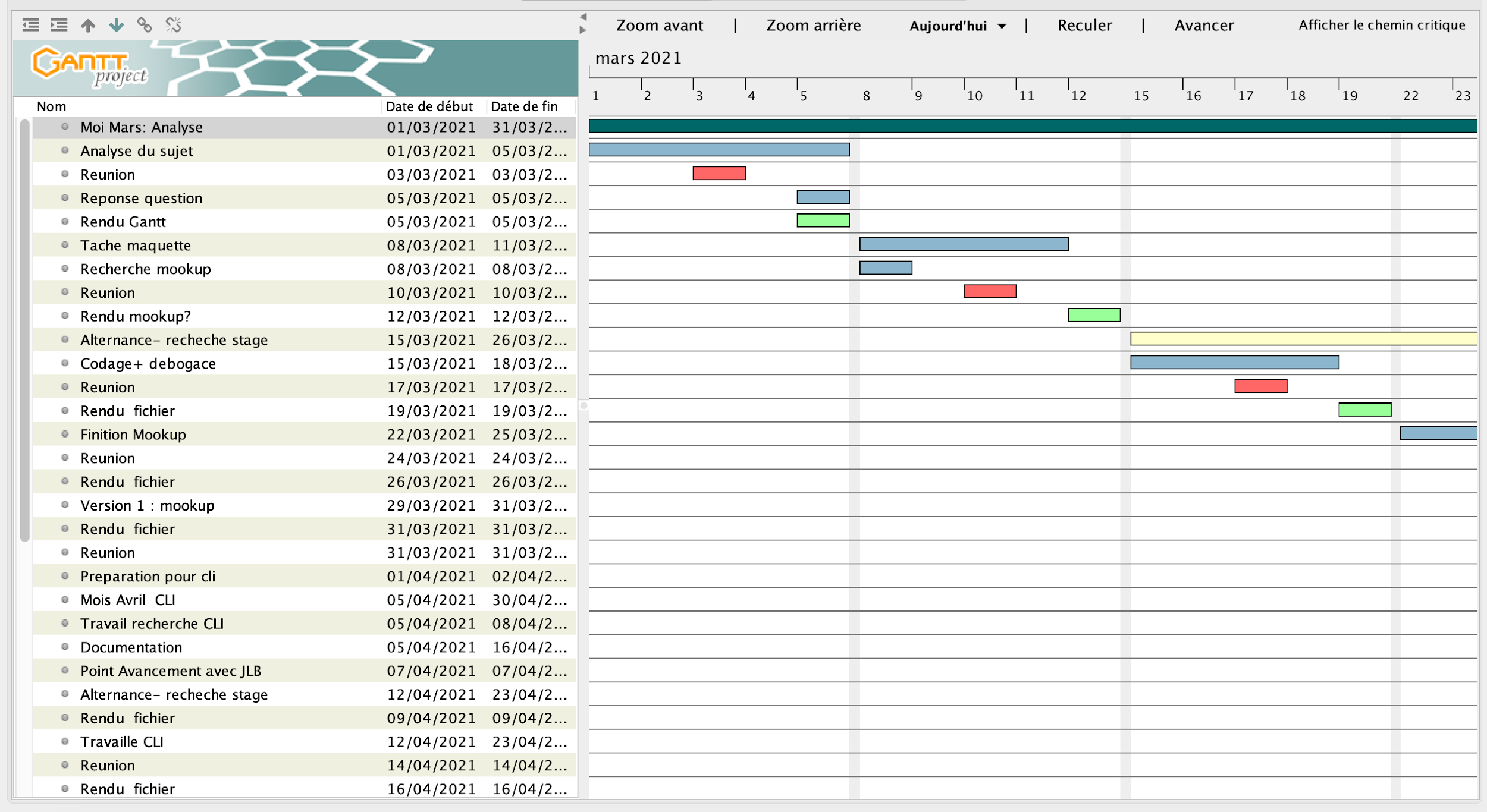
XSPF : <https://fr.wikipedia.org/wiki/XSPF>

<https://www.online-convert.com/fr/format-fichier/xspf>

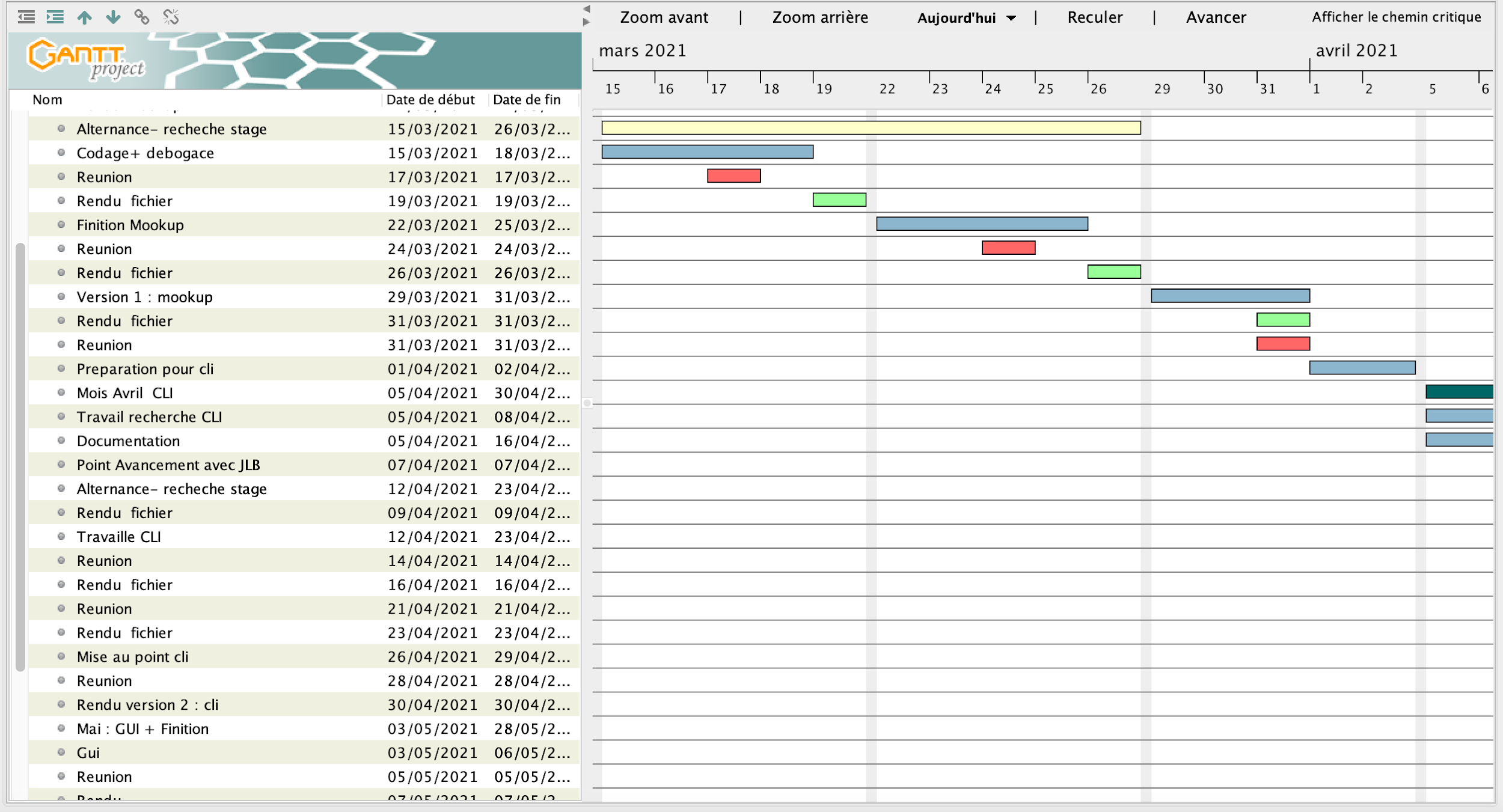
# 

# Annexe 1: GANTT

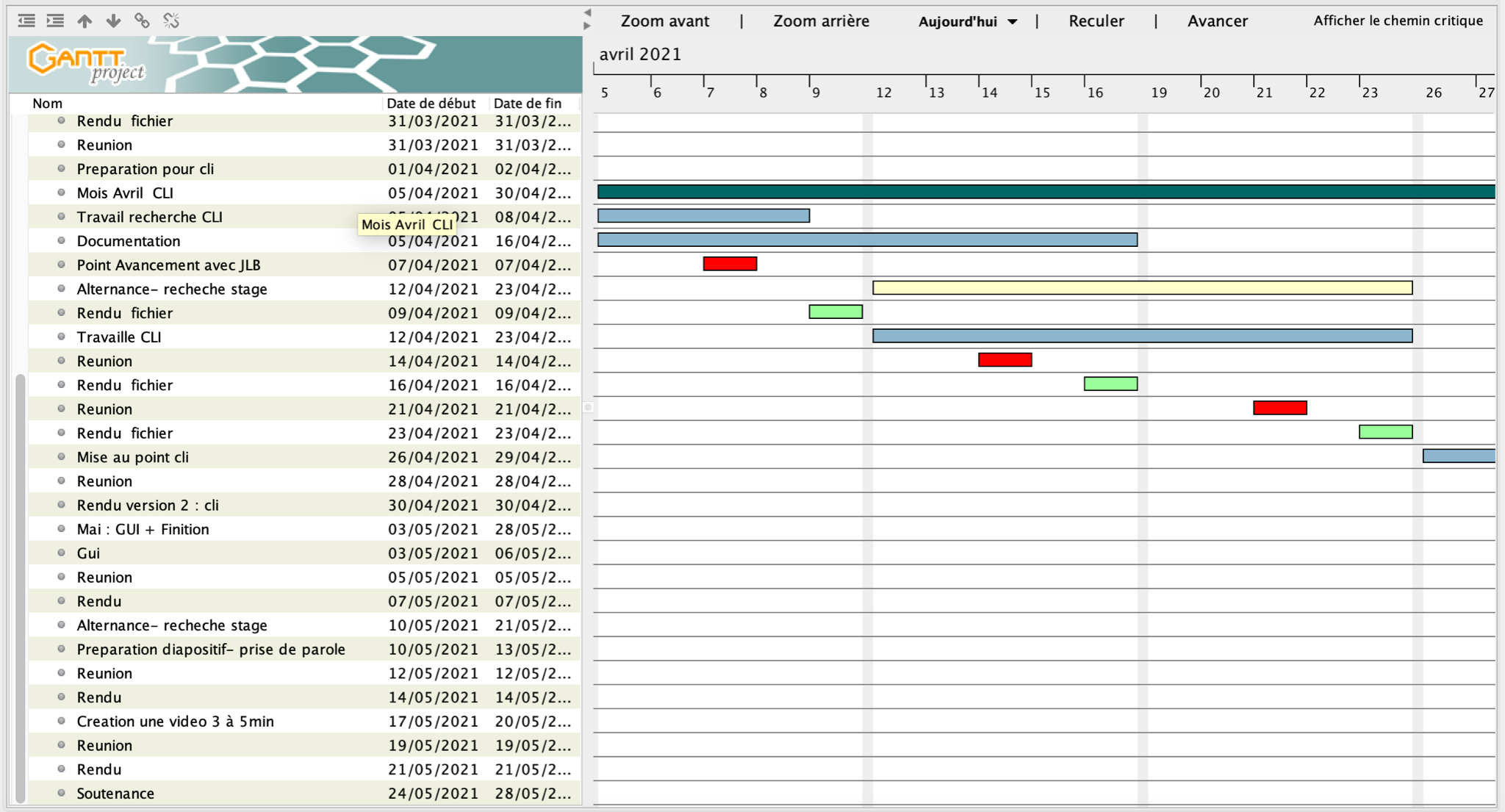
1 mars - 23mars

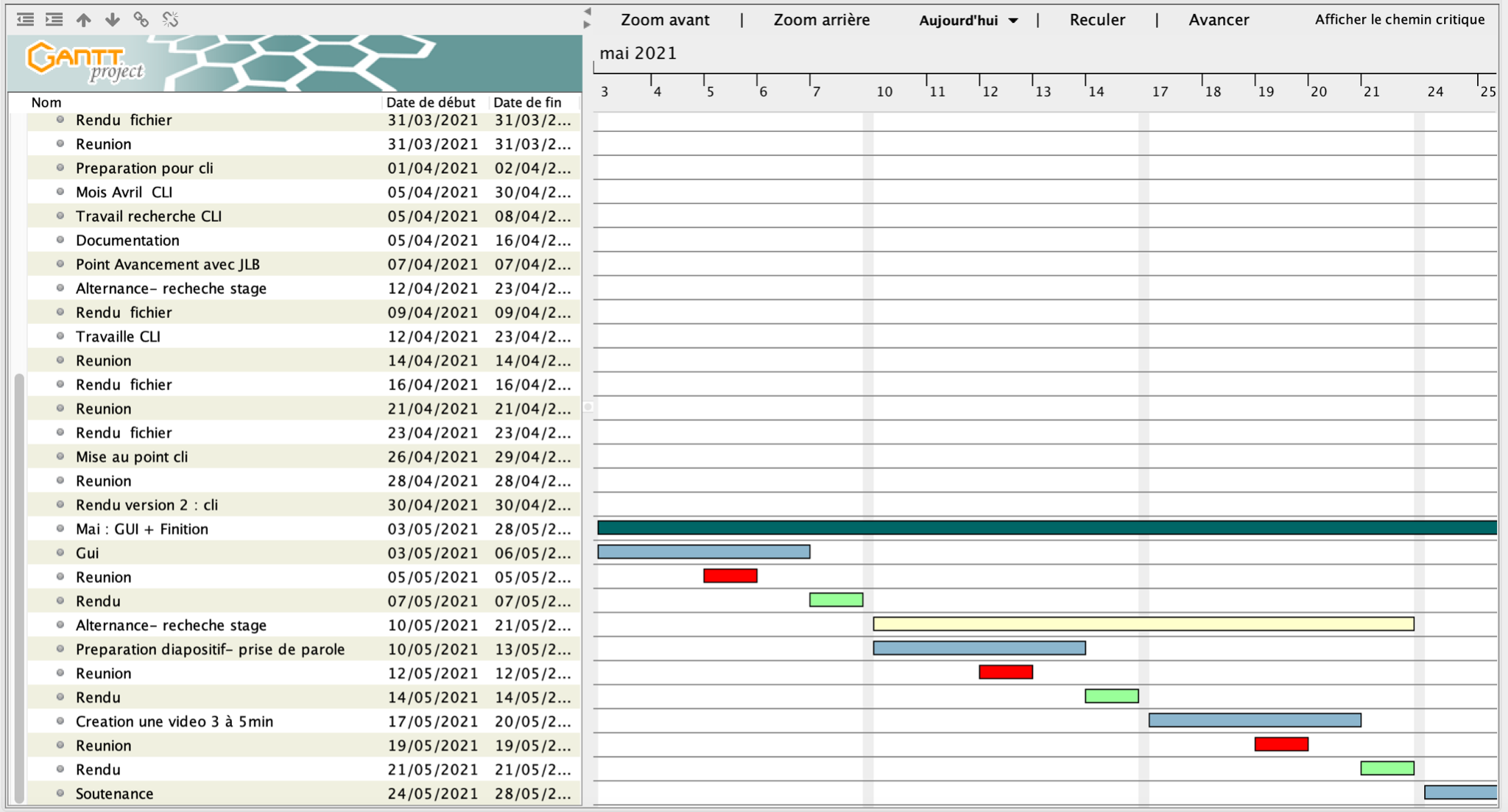


15mars -5avril



5 avril au 26 avril

3 mai au 24 mai

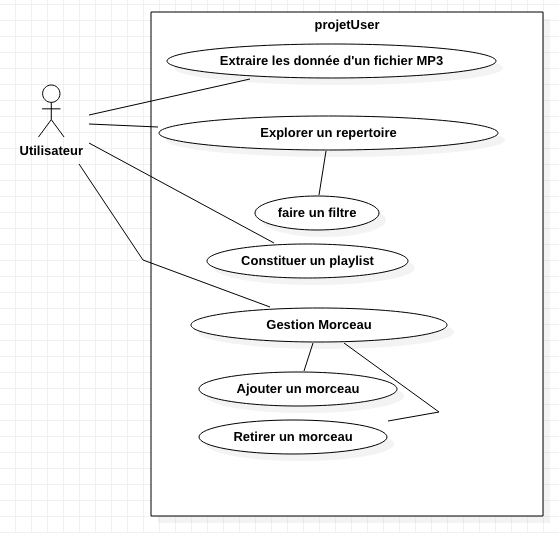


Il est noté qu'à partir du 22 mai suite à la réunion de Jean Luc Bourdon nous avons abandonnée le projet POO au profit du projet WEB

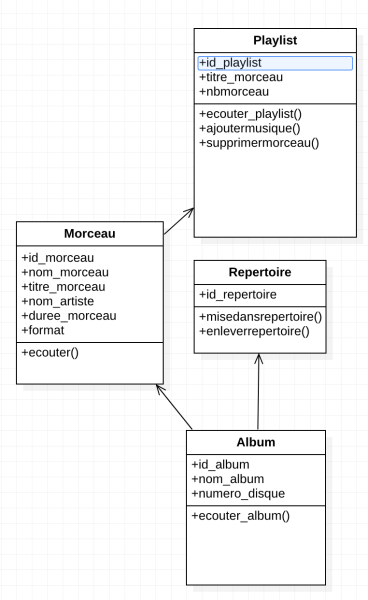
# Annexe 2: Analyse fonctionnel - UML

Projet User

Diagramme utilisation



Proposition 1.1: initiale et pour la recherche UML



Explication :

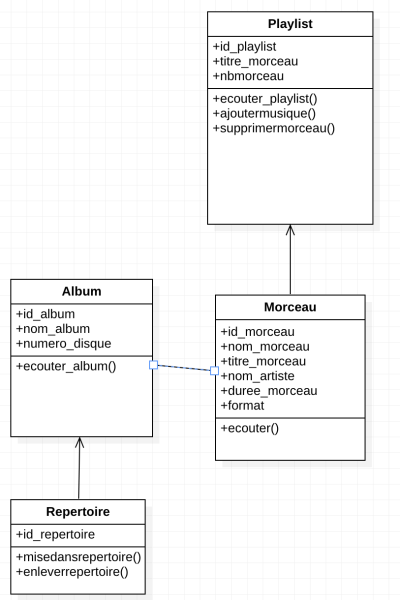
Initialement on a défini 4 classes :

La classe Repertoire, ou on devra stocker les albums ici

La classe Album qui permet d’avoir les informations de l’album et de l’écouter. Il sera stocker dans un répertoire et comportes des morceaux

La classe Morceau , ou il a les informations concernant le morceau approprié

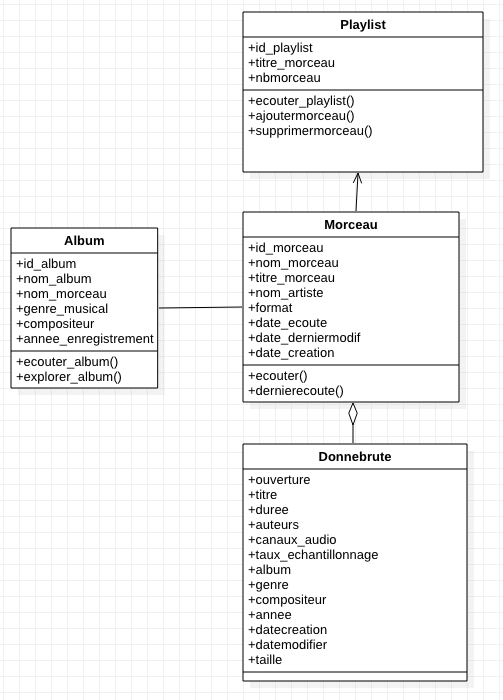
La classe Playlist qui permet d’avoir la playlist des morceaux.

Proposition 1.2: initiale et pour la recherche UML

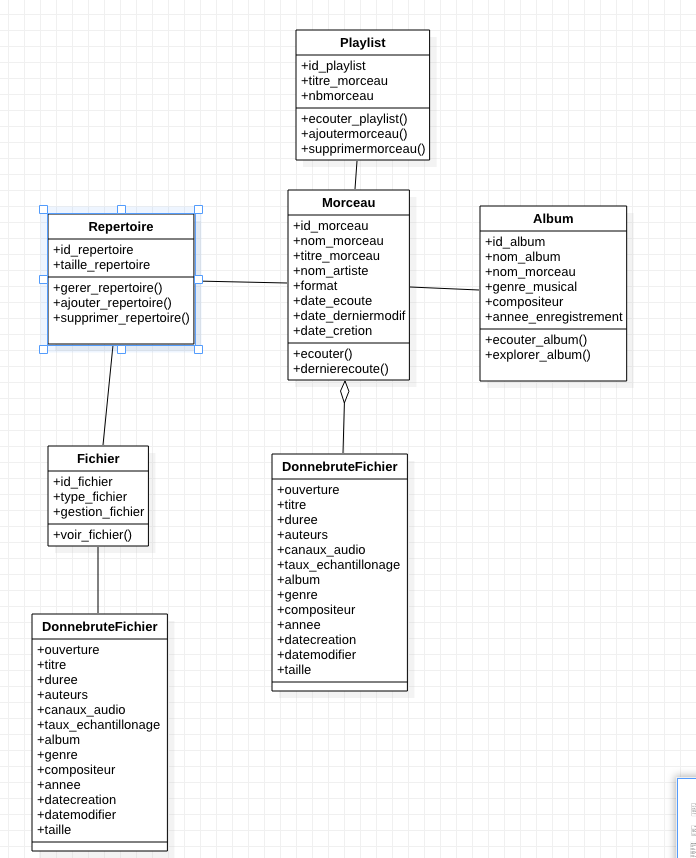
Cette proposition est plus en cascade et permet une meilleure visibilité. cependant il convient pas au besoin du cahier des charges

Suite à ces deux premier essai de proposition nous avons établie la proposition suivante, qui est plus structuré

Proposition 2: : proposition mieux structurer



Proposition 3 : finale



· la classe Fichier soit hérité de la classe Donne Fichier

· la classe Répertoire soit hérité de la classe Fichier

· la classe Morceau soit hérité de DonnebruteMorceau

· la classe Morceau soit hérité de la classe Playlist

· la classe album soit associé à la classe morceau

# Annexe 3 : Code

<dependencies>

<dependency>

<groupId>com.mpatric</groupId>

<artifactId>mp3agic</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

-----

import java.io.IOException;

import com.mpatric.mp3agic.InvalidDataException;

import com.mpatric.mp3agic.Mp3File;

import com.mpatric.mp3agic.UnsupportedTagException;

public class Main {

public static void main(String[] args) throws UnsupportedTagException, InvalidDataException, IOException {

Mp3File mp3file = new Mp3File("C://Users/loren/Music/Karma - Back 2 Back.mp3");

System.out.println("Length of this mp3 is: " + mp3file.getLengthInSeconds() + " seconds");

System.out.println("Bitrate: " + mp3file.getBitrate() + " kbps " + (mp3file.isVbr() ? "(VBR)" : "(CBR)"));

System.out.println("Sample rate: " + mp3file.getSampleRate() + " Hz");

System.out.println("Has ID3v1 tag?: " + (mp3file.hasId3v1Tag() ? "YES" : "NO"));

System.out.println("Has ID3v2 tag?: " + (mp3file.hasId3v2Tag() ? "YES" : "NO"));

System.out.println("Has custom tag?: " + (mp3file.hasCustomTag() ? "YES" : "NO"));

}

}

chemin absolu : "C://Users/loren/Music/Karma - Back 2 Back.mp3"

--------

public class DonnebruteMorceau{

String ouverture;

String titre;

String auteurs;

String canaux\_audio;

String taux\_echantillonnage;

String album;

String genre;

String compositeur;

String taille;

int annee;

int duree;

int datecreation;

int datemodifier;

//constructeur

public DonnebruteMorceau(String ouverture,String titre, String auteurs, String canaux\_audio, String taux\_echantillonnage, String album, String genre, String compositeur, String taille;) {

this.ouverture = ouverture;

this.titre = titre;

this.auteurs = auteurs;

this.canaux\_audio = canaux\_audio;

this.taux\_echantillonnage= taux\_echantillonnage;

this.album = album;

this.genre= genre;

}

//getteur

public String getOuverture() {

return ouverture;

}

public String getTitre() {

return titre;

}

public String getAuteurs() {

return auteurs;

}

public void setOuverture(String ouverture) {

this.ouverture = ouverture;

}

public String getCanaux\_audio() {

return canaux\_audio;

}

public String getAlbum() {

return album;

}

public String getTaux\_echantillonnage() {

return taux\_echantillonnage;

}

public String getGenre() {

return genre;

}

public String getCompositeur() {

return compositeur;

}

public String getTaille() {

return taille;

}

public int getAnnee() {

return annee;

}

public int getDuree() {

return duree;

}

public int getDatecreation() {

return datecreation;

}

public int getDatemodifier() {

return datemodifier;

}

//setteur

public void setTitre(String titre) {

this.titre = titre;

}

public void setAuteurs(String auteurs) {

this.auteurs = auteurs;

}

public void setCanaux\_audio(String canaux\_audio) {

this.canaux\_audio = canaux\_audio;

}

public void setTaux\_echantillonnage(String taux\_echantillonnage) {

this.taux\_echantillonnage = taux\_echantillonnage;

}

public void setAlbum(String album) {

this.album = album;

}

public void setGenre(String genre) {

this.genre = genre;

}

public void setCompositeur(String compositeur) {

this.compositeur = compositeur;

}

public void setTaille(String taille) {

this.taille = taille;

}

public void setAnnee(int annee) {

this.annee = annee;

}

public void setDuree(int duree) {

this.duree = duree;

}

public void setDatecreation(int datecreation) {

this.datecreation = datecreation;

}

public void setDatemodifier(int datemodifier) {

this.datemodifier = datemodifier;

}

/\*

Redéfinition de toString() pour l'affichage

\*/

public String toString() {

String msg = "Donnee [value=" + this.ouverture + "]";

return msg;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

//creation d'une donne brute d'un morceau

DonnebruteMorceau monMorceau = new DonnebruteMorceau();

}

}

# Annexe 4 : Guide pour Eclipse API selon Lorenzo

J'ai enfin réussi à trouver, faire fonctionner une librairie qui extrait les métadonnées (id3) des fichiers mp3 !

alors ducoup les étapes :

Aller sur Eclipse : Créer un Projet Maven => aller dans pom.xml

Ajoutez

<dependencies>

<dependency>

<groupId>com.mpatric</groupId>

<artifactId>mp3agic</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

Veillez à bien l'ajouter entre les balises project

Créer une classe bidon dans src/main/java style Main

Et entrez précisément ce code pour tester déjà

import java.io.IOException;

import com.mpatric.mp3agic.InvalidDataException;

import com.mpatric.mp3agic.Mp3File;

import com.mpatric.mp3agic.UnsupportedTagException;

public class Main {

public static void main(String[] args) throws UnsupportedTagException, InvalidDataException, IOException {

Mp3File mp3file = new Mp3File("C://Users/loren/Music/Karma - Back 2 Back.mp3");

System.out.println("Length of this mp3 is: " + mp3file.getLengthInSeconds() + " seconds");

System.out.println("Bitrate: " + mp3file.getBitrate() + " kbps " + (mp3file.isVbr() ? "(VBR)" : "(CBR)"));

System.out.println("Sample rate: " + mp3file.getSampleRate() + " Hz");

System.out.println("Has ID3v1 tag?: " + (mp3file.hasId3v1Tag() ? "YES" : "NO"));

System.out.println("Has ID3v2 tag?: " + (mp3file.hasId3v2Tag() ? "YES" : "NO"));

System.out.println("Has custom tag?: " + (mp3file.hasCustomTag() ? "YES" : "NO"));

}

}

en remplaçant :

"C://Users/loren/Music/Karma - Back 2 Back.mp3"

par votre chemin vers votre MP3 à vous

Preuve que ça marche bien pour moi :

Length of this mp3 is: 194 seconds

Bitrate: 241 kbps (VBR)

Sample rate: 48000 Hz

Has ID3v1 tag?: NO

Has ID3v2 tag?: YES

Has custom tag?: NO