UNIVERSITE DE CERGY-PONTOISE

UFR Sciences et Techniques Saint-Martin

DEPARTEMENT D'INFORMATIQUE

RAPPORT DU PROJET

Sujet : Images & métadonnées avec Java

Présenté par :

- **HAMLAT MOHAMED ARSLANE**
- **REBUFIE PABLO**

Groupe : E1

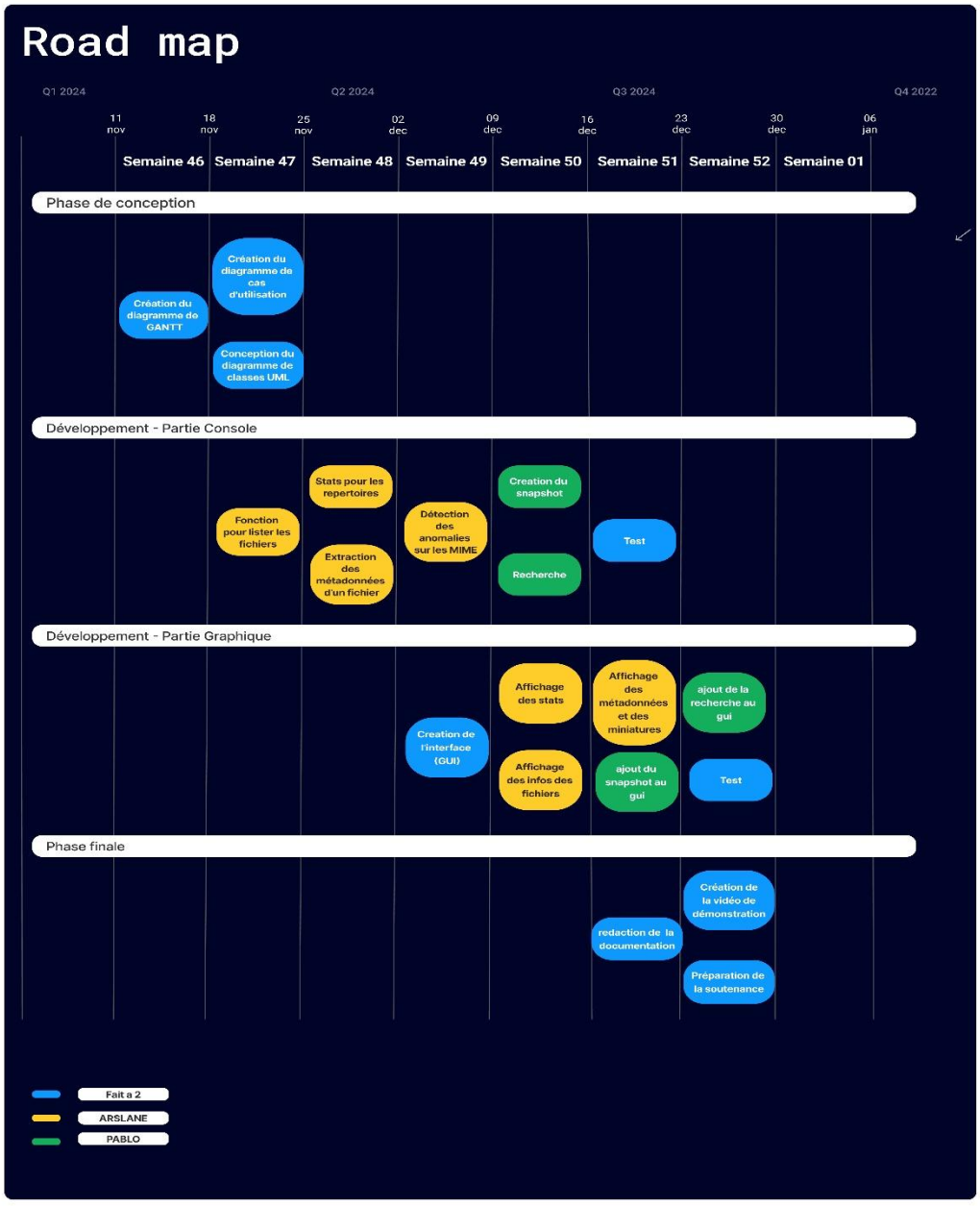
Année Universitaire : 2024/2025

LA TABLE DES MATIERES :

I. Les diagrammes :

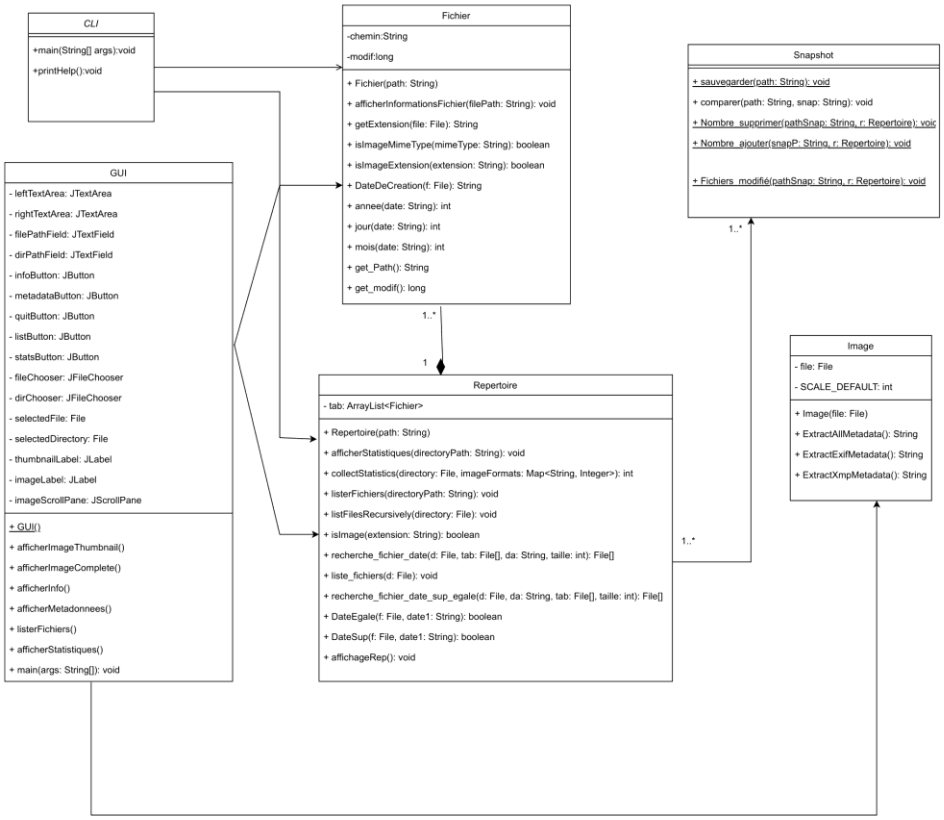
1. Diagramme de Gantt :

Voici le diagramme de Gantt avec toutes les taches faites durant le projet et la répartition du travail entre les membre du binôme suivant les couleurs ci-dessous :



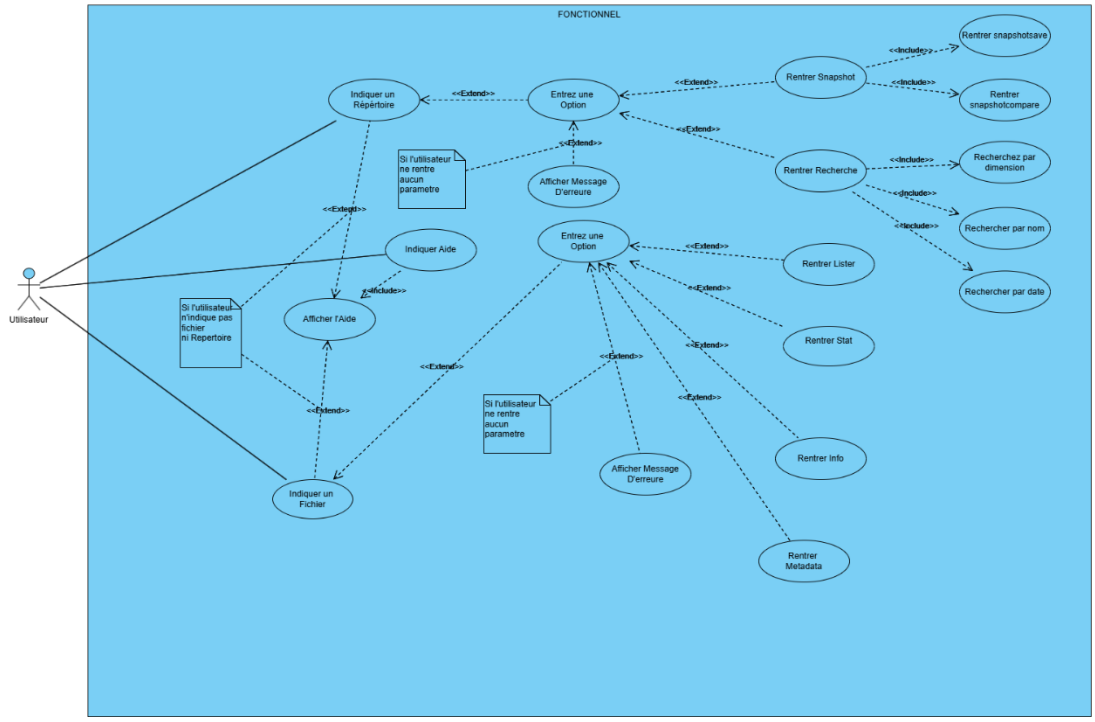
2. Diagramme de classes :

Voici le diagramme de classe contenant toute les classes utiliser durant ce projet et aussi toute les relation entre ces dernière :



3. Diagramme de cas d'utilisation :

Voici le diagramme de cas d'utilisation qui montre tous les acteurs qui interagissent avec le projet et les différents cas d'utilisation :



II. Aspect fonctionnel :

1. La classe Fichier :

L'objectif principale de cette classe est de représenter un fichier et fournir des fonctionnalités utiles pour récupérer, afficher et manipuler les métadonnées du fichier. Voici une analyse détaillée de ses fonctionnalités et de ses objectifs :

- Des méthodes pour afficher les informations générales d'un fichier (nom, chemin, taille, type MIME, extensions, dates de création et modification).
- Des outils pour vérifier la cohérence entre l'extension du fichier et son type MIME, en particulier pour les fichiers d'images.
- Des utilitaires pour extraire des informations spécifiques sur les dates et les extensions.

Voici une capture d'écran d'un exemple de fonctionnalité de cette classe tester grâce a la classe CLI que nous verrons un peu plus tard :

```
PS C:\Users\arsla> java -jar Cli.jar -f "C:\Users\arsla\OneDrive\Bureau\imagepoo\Metadata_test_file_-_includes_data_in_IIM,_XMP,_and_Exif.jpg" --info

======= Informations sur le fichier =======
Nom : Metadata_test_file_-_includes_data_in_IIM,_XMP,_and_Exif.jpg
Chemin absolu : C:\Users\arsla\OneDrive\Bureau\imagepoo\Metadata_test_file_-_includes_data_in_IIM,_XMP,_and_Exif.jpg
Taille : 138810 octets
Type MIME : image/jpeg
Extension : jpg
Date de création : 2024-12-22 14:19:19 CET
Date de dernière modification : 2024-12-22 14:19:19 CET
```

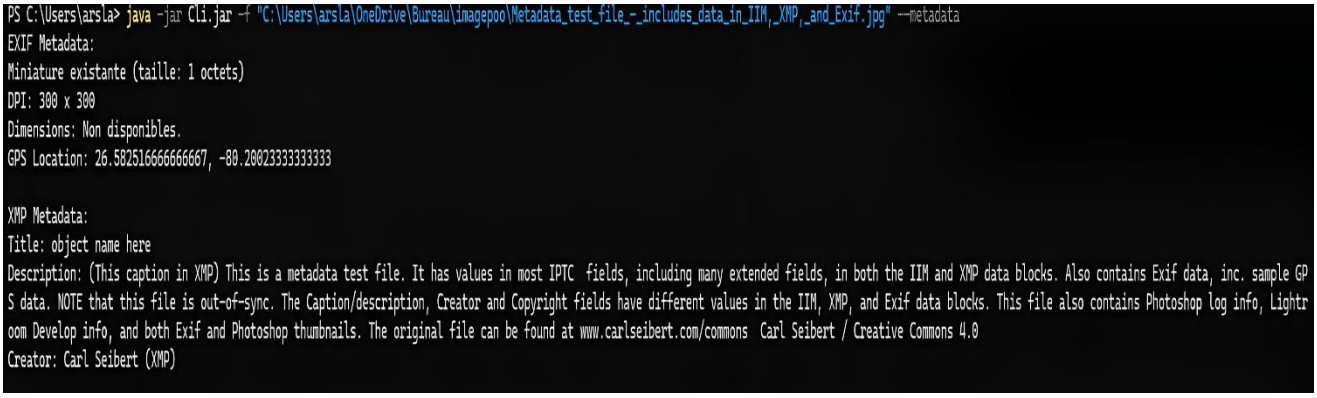
En donnant le chemin d'accès vers un fichier quelconque on récupère toutes les informations relatives à ce fichier grâce à l'option –info.

2. La classe Image :

Cette est classe est conçue pour extraire et manipuler les métadonnées d'une image en utilisant la bibliothèque externe **`metadata-extractor`**. Cette bibliothèque est puissante et permet d'accéder facilement aux informations EXIF et XMP des fichiers image grâce au tag. Voici une analyse détaillée de cette classe :

- **D'extraire les métadonnées EXIF** : Inclut les informations sur les dimensions, la résolution, les miniatures, et les coordonnées GPS.
- **D'extraire les métadonnées XMP** : Inclut les informations liées aux titres, descriptions, créateurs, et autres propriétés.

Voici un un exemple de test de cette classe grâce a une image qui contient toute les métadonnée nécessaire :



En donnant le chemin d'accès vers une image quelconque on récupère toute les métadonnées relative a ce fichier grâce a l'option –metadata et on utilise l'application **PhotoMe** pour s'assurer de l'exactitude des métadonnées extraites .

3. La classe Répertoire :

La classe Répertoire offre un ensemble de fonctionnalités pour manipuler les fichiers contenus dans un répertoire donné. Elle permet d'effectuer diverses opérations telles que :

- Collecter des statistiques sur les fichiers présents dans le répertoire et ses sous-répertoires.
- Lister tous les fichiers récursivement
- Rechercher des fichiers basés sur des critères tels que la date de création.

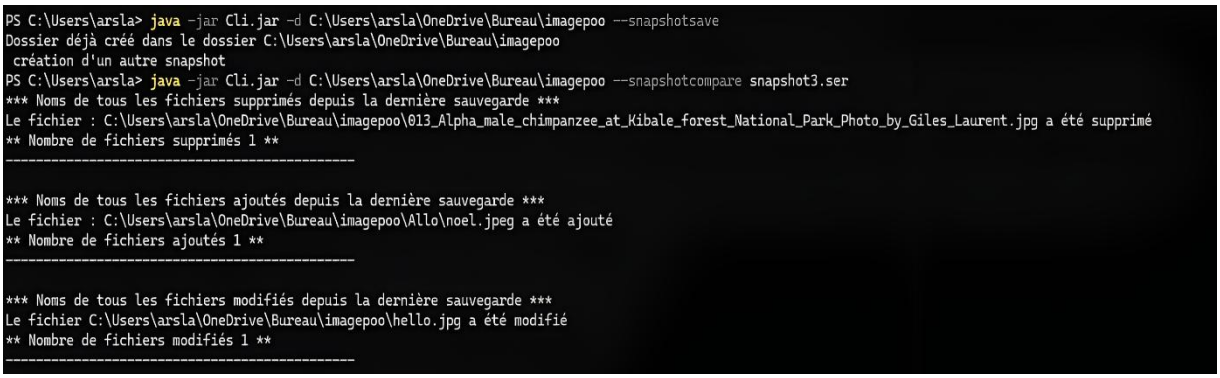
Cette classe est également **sérialisable**, permettant ainsi de sauvegarder son état (la collection de fichiers) pour une utilisation ultérieure (Snapshot).

4. La classe Snapshot :

La classe `Snapshot` est une implémentation pour sauvegarder et comparer l'état des répertoires en utilisant la sérialisation. Elle permet de :

- **Sauvegarder** un instantané (snapshot) d'un répertoire, en capturant son contenu et ses métadonnées grâce a l'option –snapshotsave
- **Comparer** un répertoire actuel avec un snapshot précédemment sauvegardé pour identifier les fichiers ajoutés, supprimés, ou modifiés grâce à l'option –snapshotcompare

Nous avons aussi choisi de stocker tout les fichier générer lors du snapshotsave dans un seule et même dossier appeler snapshots et aussi nous les avons numérote par exemple snapshot1,snapshot 2 … etc. pour plus d'organisation . Voici des captures d'écran sur les test effectuer pour la classe snapshot :



III. Aspect Console :

Nous avons utilisé une classe nommer CLI qui représente la classe main pour utiliser les différentes fonctionnalités citer lors de la présentation de chaque classe et grâce à leur option dans un mode console comme :

a) Traitement des fichier (-f) :

- `--info` : Affiche les informations détaillées sur un fichier.
- `--metadata` : Extrait et affiche les métadonnées d'une image.
- `-eq` : Affiche la date de création d'un fichier au format "jour/mois/année".

b) Traitement des dossiers (-d) :

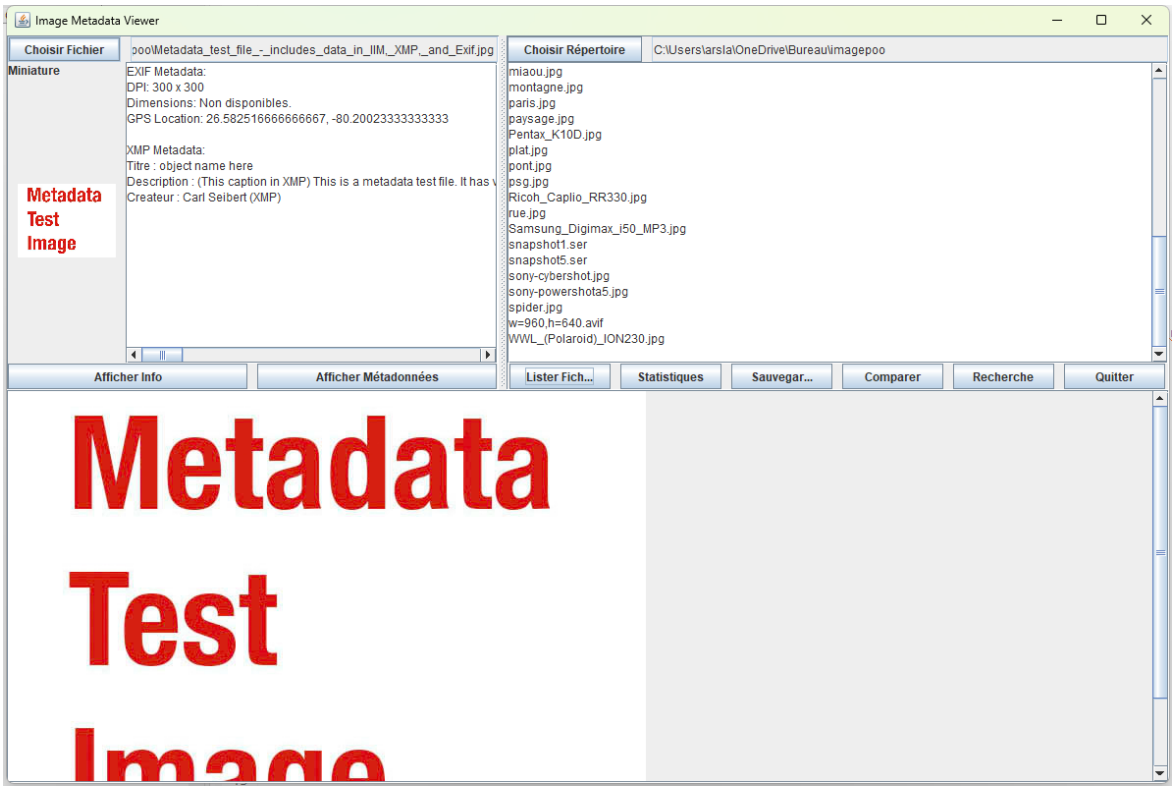
- `--stat` : Affiche les statistiques des fichiers et images dans le répertoire spécifié.
- `--list` : Liste tous les fichiers présents dans le répertoire.
- `--snapshotsave` : Sauvegarde un snapshot de l'état actuel du répertoire.
- `--snapshotcompare` : Compare le répertoire actuel avec un snapshot donné.

c) Aide et documentation :

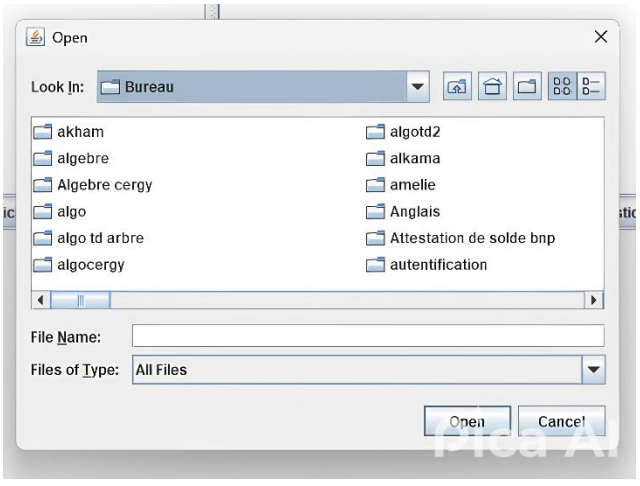
- `-h` ou `--help` : Affiche une liste détaillée des commandes et des exemples d'utilisation

IV. Aspect Graphique :

Comme pour l'aspect console nous avons une classe nommer GUI qui représente la classe main pour utiliser les différente fonctionnalité citer lors de la présentation de chaque classe et grâce a leur option et ce en fournissant une interface graphique avec laquelle on pourra interagir directement .Voici un aperçu :



Et voici l'interface pour choisir l'image ou le répertoire que vous voulez :



V. Conclusion :

Au terme de ce projet, nous avons pu mettre en œuvre un ensemble d'outils orientés objet dédiés à la gestion et à l'exploitation des métadonnées d'images (EXIF et XMP) ainsi qu'à l'organisation des fichiers et dossiers. Ce projet a permis d'aborder des problématiques clés dans la manipulation de données multimédias, telles que :

- L'extraction, la modification et la validation des métadonnées, garantissant la cohérence et la pertinence des informations associées aux images.
- L'automatisation de l'organisation des fichiers et dossiers selon des critères définis par les métadonnées, simplifiant ainsi leur gestion à grande échelle.
- La structuration et la modularité du code, en respectant les principes de la programmation orientée objet (encapsulation, héritage et polymorphisme), offrant une base évolutive pour d'éventuelles extensions futures.

En conclusion, ce projet nous a permis d'acquérir une expérience précieuse et d'approfondir nos connaissances dans le domaine de la gestion des métadonnées et de la programmation orientée objet. Nous avons beaucoup appris sur les techniques d'extraction et de manipulation des données multimédias ainsi que sur les bonnes pratiques de structuration et de modularité du code.

Une des parties les plus dures était de gérer les cas de bords, en effet le CLI qui paraissait plutôt "simple" a longtemps été une problématique avec la gestion des indices comme gérer la possibilité de rentrer plusieurs options. Pour cela nous avons décider d'utiliser des tableaux ArrayList ou des HashMap de files qui ont joué un role clé dans la gestion des contenus des répertoires.

Il y aussi l'aspect graphique où la recherche des bons packages pour rendre l'interface utilisable était compliqué .Tout comme la mise en forme du projet avec l'ajout de dépendances et la création d'un projet Maven . Je pense notamment au cas du menu déroulant qui s'avère très utile pour l'affichage des miniatures et des options de recherche mais difficile à mettre en place .

Cependant une fois en confiance avec le projet (appropriation de l'énoncé des méthodes et des packages à importer) le réel problème était la gestion du temps et l'organisation du groupe . Comme le partage du code et la fusion des codes de chaque personne (très chronophage) . Nous avons donc utilisé guithub pour pallier a ce problème qui c'est avéré utile .