# Table des matières

[Table des matières 1](#_Toc188833382)

[Table des figures 3](#_Toc188833383)

[INTRODUCTION 4](#_Toc188833384)

[1 Etude De L’existant 5](#_Toc188833385)

[1.1 Logiciels existants pour la lecture de bandes dessinées 5](#_Toc188833386)

[1.1.1 Présentation des outils populaires 5](#_Toc188833387)

[1.1.2 Points forts et limites des solutions existantes 5](#_Toc188833388)

[1.2 Formats de fichiers utilisés 6](#_Toc188833389)

[1.2.1 Formats d’archives 6](#_Toc188833390)

[1.2.2 Formats d’images 6](#_Toc188833391)

[1.2.3 Limites des formats actuels 6](#_Toc188833392)

[1.3 Technologies et algorithmes pertinents 6](#_Toc188833393)

[1.3.1 Algorithmes de redimensionnement d’image 6](#_Toc188833394)

[1.3.2 Gestion de la mémoire et mise en cache 7](#_Toc188833395)

[1.3.3 Outils et bibliothèques existants 7](#_Toc188833396)

[1.4 Synthèse et opportunités pour le projet 7](#_Toc188833397)

[2 ANALYSE DES BESOINS 7](#_Toc188833398)

[2.1 Exigences fonctionnelles 8](#_Toc188833399)

[2.2 Exigences non fonctionnelles 10](#_Toc188833400)

[3 CONCEPTION GLOBALE 11](#_Toc188833401)

[3.1 Une Architecture en 4 couches 11](#_Toc188833402)

[3.1.1 Couche présentation (UI Layer) 12](#_Toc188833403)

[3.1.2 Couche métier : 12](#_Toc188833404)

[3.1.3 Couche infrastructure (Infrastructure Layer) 13](#_Toc188833405)

[3.1.4 Couche d’accès aux données (Data Access Layer) 14](#_Toc188833406)

[3.2 Architecture logicielle 14](#_Toc188833407)

[3.2.1 Description des modules principaux 14](#_Toc188833408)

[3.3 Diagramme de contexte 15](#_Toc188833409)

[3.4 Diagramme de Paquetage 15](#_Toc188833410)

[3.5 Diagrammes de cas d’utilisation 15](#_Toc188833411)

[3.6 Diagramme de classes global 16](#_Toc188833412)

[3.7 Diagramme de déploiement 16](#_Toc188833413)

[4 CONCEPTION DETAILLEE 16](#_Toc188833414)

[4.1 Gestion des livres et fichiers 17](#_Toc188833415)

[4.2 Diagrammes de séquence 17](#_Toc188833416)

[4.3 Traitement des images 18](#_Toc188833417)

[4.4 Gestion des bibliothèques 18](#_Toc188833418)

[5 CAHIER DE CONCEPTION 18](#_Toc188833419)

[5.1 Technologies utilisées : 18](#_Toc188833420)

[5.1.1 C++ : Le Langage Principal 18](#_Toc188833421)

[5.1.2 Qt : Framework de Développement d'Interface Graphique 18](#_Toc188833422)

[5.1.3 CMake : Outil de Construction et de Gestion de Projet 19](#_Toc188833423)

[5.1.4 ImageMagick : Bibliothèque de Manipulation d'Images 19](#_Toc188833424)

[5.1.5 Structure de l’Interface Utilisateur avec Qt 19](#_Toc188833425)

[CONCLUSION 23](#_Toc188833426)

# INTRODUCTION

Les bandes dessinées, occupent une place privilégiée dans le cœur des amateurs de littérature visuelle. Avec l’évolution des formats numériques, la nécessité d’outils dédiés pour leur lecture et leur création s’est accentuée. Ce rapport présente le développement d’un projet intitulé **Comic Book Reader/Writer**, une application visant à offrir une expérience optimale pour lire, gérer, et créer des bandes dessinées numériques.

Le projet repose sur des objectifs ambitieux : une prise en charge étendue des formats d’archives et d’images courants tels que .CBR, .CBZ, .JPEG, et .PNG, tout en garantissant une interface intuitive et performante. En outre, il intègre des fonctionnalités avancées comme le redimensionnement asynchrone des pages, l’application de filtres pour améliorer la lisibilité des textes, et la gestion d’une bibliothèque numérique enrichie.

L’approche adoptée pour ce projet se concentre sur l’évolution et la modularité, avec une architecture logicielle permettant d’ajouter facilement de nouvelles fonctionnalités. Ce rapport retrace les différentes étapes de conception et de réalisation de cette application, en s’appuyant sur une analyse approfondie des besoins utilisateurs et des technologies disponibles.

# Etude De L’existant

Avant d’entamer le développement de l’application Comic Book Reader/Writer, une étude approfondie de l’existant s’impose pour comprendre les outils disponibles, identifier leurs forces et faiblesses, et positionner notre solution de manière unique. Cette analyse se concentre sur les logiciels populaires existants pour la lecture et la gestion de bandes dessinées numériques, les formats de fichiers utilisés, ainsi que les technologies et algorithmes pertinents.

## Logiciels existants pour la lecture de bandes dessinées

### Présentation des outils populaires

Les principaux logiciels disponibles incluent :

* CDisplayEx: Un logiciel largement utilisé pour lire des bandes dessinées dans des formats comme .CBR et .CBZ. Il offre une interface simple et légère, mais manque de fonctionnalités avancées telles que l’organisation des bibliothèques ou la modification des ouvrages.
* YACReader : Un outil multiplateforme qui inclut une interface de lecture ergonomique et une gestion efficace des bibliothèques, mais il n’offre pas de capacités de modification des ouvrages.
* SumatraPDF : Bien qu’il ne soit pas spécifiquement conçu pour les bandes dessinées, ce lecteur PDF supporte les formats .CBR et .CBZ, mais avec des fonctionnalités limitées en termes d’optimisation pour les contenus graphiques.

### Points forts et limites des solutions existantes

|  |  |  |
| --- | --- | --- |
| **Fonctionnalités** | **Points forts des solutions actuelles** | **Limites observées** |
| Formats pris en charge | Large support des formats .CBR et .CBZ | Peu de prise en charge d'autres formats comme .PDF ou .TIFF |
| Navigation | Interfaces intuitives pour lire page par page | Navigation parfois lente sur des ouvrages volumineux |
| Qualité d’affichage | Lecture en pleine page et options de zoom | Peu d’optimisations pour améliorer la lisibilité du texte |
| Organisation | Gestion de bibliothèques (YACReader) | Manque d’intégration des métadonnées et des tags |

## Formats de fichiers utilisés

### Formats d’archives

Les formats d’archives .CBR et .CBZ sont les standards pour les bandes dessinées numériques.

* .CBR : Basé sur le format .rar, il offre une compression efficace mais nécessite des bibliothèques spécifiques pour la décompression.
* .CBZ : Basé sur le format .zip, plus universel et supporté par la majorité des logiciels.

### Formats d’images

Les bandes dessinées numériques contiennent généralement des images compressées dans divers formats :

* JPEG (.jpg) : Format populaire pour les images compressées avec pertes, mais souvent sujet à des artefacts visuels, notamment autour du texte.
* PNG (.png) : Format sans perte adapté aux images nécessitant une haute qualité.
* TIFF (.tiff) : Format utilisé pour les images monochromes ou nécessitant une précision élevée, bien que plus lourd en termes de taille.

### Limites des formats actuels

* Les formats d’archives comme .CBR et .CBZ ne contiennent pas de métadonnées enrichies (par exemple, tables des matières ou tags).
* La qualité d’affichage des formats compressés (JPEG) peut être insuffisante pour les textes et les détails graphiques.

## Technologies et algorithmes pertinents

### Algorithmes de redimensionnement d’image

La qualité du redimensionnement est cruciale pour garantir une bonne expérience utilisateur. Les algorithmes courants incluent :

* **Nearest Neighbor (proche voisin)** : Rapide mais produit des images pixélisées.
* **Bilinear Interpolation** : Offre un rendu plus lisse mais peut manquer de netteté pour le texte.
* **Lanczos Resampling** : Algorithme avancé qui préserve mieux les détails, adapté aux bandes dessinées avec des graphismes complexes.

### Gestion de la mémoire et mise en cache

* Les images décompressées peuvent consommer une grande quantité de mémoire. Une stratégie de mise en cache efficace est essentielle pour garantir la fluidité lors de la navigation entre les pages.
* Le préchargement asynchrone des pages suivantes améliore considérablement l’expérience utilisateur.

### Outils et bibliothèques existants

* ImageMagick : Bibliothèque puissante pour le traitement des images (redimensionnement, filtrage).
* Libarchive : Pour manipuler divers formats d’archives, y compris .zip et .rar.
* Qt et wxWidgets : Cadres populaires pour développer des interfaces graphiques multiplateformes.

## Synthèse et opportunités pour le projet

L’analyse de l’existant met en évidence un besoin de solutions innovantes dans plusieurs domaines :

* Élargissement des formats supportés : Les outils actuels se limitent souvent à .CBR et .CBZ. Le support de formats comme .PDF, .TIFF, et .PNG serait un atout majeur.
* Amélioration de la qualité d’affichage : Intégrer des filtres spécifiques pour améliorer la lisibilité des textes (correction d’artefacts JPEG, suppression de bruit) est essentiel.
* Personnalisation et enrichissement des ouvrages : Ajouter des métadonnées comme des tables des matières ou des tags pour organiser les bibliothèques est une opportunité clé.
* Performances et fluidité : L’utilisation de techniques modernes comme le préchargement asynchrone et la mise en cache intelligente peut significativement améliorer l’expérience utilisateur.

# ANALYSE DES BESOINS

Pour mieux aborder la conception de ce projet, une étape préliminaire d’analyse s’impose visant à mieux comprendre les exigences pour ainsi mieux formaliser le problème. Une lecture détaillée du cahier de charges nous a permis de lister les exigences suivantes :

1. L’application doit prendre en charge les différents formats utilisés pour les bandes dessinées (CBR, CBZ …, TIFF, PDF envisageable)
2. L’application doit offrir une interface agréable et performante permettant de parcourir les différentes pages d’un livre au format .CBR ou au format .CBZ
   1. Prendre en charge éventuellement d’autres formats tels que TIFF, PDF…
3. L’application doit offrir un rendu graphique de qualité et un confort de lecture :
   1. Masquer le temps pris par les différentes tâches
   2. Anticiper la lecture et le redimensionnement des pages, en permettant de débuter la lecture du document sans pour autant attendre que l’ensemble des pages soit chargé en mémoire…
4. L’application devra offrir une qualité d’affichage soignée
   1. utiliser des filtres pour rehausser le texte
5. L’application devra créer des ouvrages au format .CBR, .CBZ : assembler différentes images et les placer dans un fichier de type .CBZ, .CBR
6. L’application devra offrir la possibilité d’ajouter une table des matières (
   1. Proposer une extension du format de fichier, typiquement par l’ajout d’un fichier toc (table of content) au format XML par exemple dans la liste des fichiers contenus dans l’archive .CBZ ou .CBR.
7. L’application devra éventuellement prendre en charge la gestion de bibliothèques
   * 1. Permettre l’annotation des documents par des tags
     2. Gérer plusieurs livres en cours de lectures
8. L’application doit être évolutive : etendre progressivement les formats souhaités ainsi que les algorithmes de traitement d’images

Ainsi, nous avons décidé de regrouper ces exigences en exigences fonctionnelles et non fonctionnelles.

## Exigences fonctionnelles

De façon peu exhaustive, nous avons regroupé les différentes fonctionnalités des applications CDisplayEx et YACReader en lien avec le cahier de charges fourni et que nous avons pensé essentielles à implémenter pour notre application. D’abord**, CDisplayEx** est une référence dans le domaine des liseuses de bandes dessinées, elle a certaines spécificités notamment :

|  |  |
| --- | --- |
| **Lecture optimisée** | **Affichage et confort de lecture** |
| Lecture des fichiers dans les archives sans extraction manuelle. | Affichage en double page ou simple page. |
| Détection automatique de l’orientation des pages pour un affichage correct. | Zoom intelligent : ajustement automatique à la largeur ou à la hauteur de l’écran. |
| Navigation rapide grâce à des raccourcis clavier et souris. | Gestion des bandes dessinées japonaises (lecture de droite à gauche). |

Ensuite, CDisplayEx ne prenant pas en charge la gestion de bibliothèques, nous nous sommes penchés sur un autre logiciel, assez complet, multiplateformes et performant comme modèle pour avoir une idée de comment celles-ci étaient implémentées. Ceci est un ensemble de fonctionnalités notables que nous pensons implémenter.

|  |  |
| --- | --- |
| **Gestion de bibliothèque** | **Expérience utilisateur** |
| Organisation des bandes dessinées par dossier et métadonnées. | Possibilité de marquer une page ou de mettre des favoris. |
| Création d’une bibliothèque locale avec des images de couverture. | Historique de lecture : savoir où l’utilisateur s’est arrêté. |
| Système de **tags** et de recherche avancée. |  |

**Tableau des exigences fonctionnelles de l’application**

Ainsi, à terme de cette analyse, nous avons classées en 4 grandes catégories que nous résumons dans le tableau ci-dessous:

1. Lecture et Navigation
2. Affichage
3. Modification des ouvrages
4. Gestion de bibliothèques

|  |  |  |  |
| --- | --- | --- | --- |
| Lecture et Navigation | Affichage : interface agréable, performante, rendu graphique de qualité, confort de lecture | Modification des ouvrages | Gestion de bibliothèques |
| Doit Support des formats : .CBR, .CBZ, et extension future vers .pdf, .TIFF | Paramètres pour régler les marges. | Extraction et assemblage des pages | Organisation par dossier ou tags : annoter les documents par des tags. |
| Doit pouvoir afficher les premières pages sans délai. (renvoie à la lecture asynchrone en fait) | Parcourir les différentes pages d’un livre : Transitions fluides entre les pages. | Ajout et édition des tables des matières. | Ajout et édition des tables des matières. |
| L’Affichage doit être flexible :  simple page, double page, ou défilement continu (YarcReader) | Options de filtres pour : Rehausser les textes, Optimiser les graphiques. | Création de livres personnalisés. | Recherche avancée par nom, tag, ou métadonnées. |
| Doit permettre la navigation facile :  Page suivante/précédente, dernière page, première page…  Accès direct à n’importe quelle page. | Détection automatique de l’orientation des pages. |  | Création de bibliothèques locales avec images de couverture. |
| Zoom intelligent : ajustement à la largeur, hauteur, ou plein écran. |  |  |  |
| Lecture inversée pour les bandes dessinées japonaises : (envisageable) |  |  |  |

## Exigences non fonctionnelles

Nous avons 4 principales exigences que nous pouvons regrouper comme suit :

|  |  |  |
| --- | --- | --- |
| **Performances** | **Architecture logicielle** | **Ergonomie et expérience utilisateur** |
| Doit être Rapide et fluide  Chargement asynchrone des livres : les premières pages doivent être disponibles immédiatement, même si le reste du fichier est en cours de traitement.  Mise en cache des images déjà redimensionnées et filtrées pour éviter des recalculs inutiles.  Temps de réponse minimal (masquer le temps pour : Appliquer des filtres ou redimensionner une page.  Débuter la lecture du document sans pour autant que le reste soit terminé | Doit être modulaire et extensible  Conception modulaire permettant l’ajout de nouveaux formats de fichiers sans modification majeure du code : Par exemple, ajouter facilement des algorithmes de traitement d’image ou des formats comme .7z, .png, ou .pdf.  Utilisation de classes abstraites pour gérer les différents types de fichiers (images et archives) :  Classe abstraite pour les archives avec des spécialisations pour .rar, .zip, etc.  Classe abstraite pour les images, permettant le support d’extensions futures. | Confort de lecture  Transitions fluides entre les pages pour éviter des interruptions dans la navigation. |
| Optimisation mémoire  Gestion efficace des images décompressées, qui peuvent occuper jusqu’à 10 fois leur taille d'origine.  Réduction de l’occupation mémoire par l’implémentation de techniques de compression temporaire ou par la libération des ressources inutilisées. | Compatibilité  Gestion des fichiers mal identifiés (vérification via la signature binaire plutôt que l’extension).  Prise en charge des fichiers compressés contenant des images sans extension explicite. | Interface utilisateur  Interface intuitive et adaptée aux écrans tactiles (gestes pour tourner les pages, zoomer, etc.). |
|  | Qualité et évolutivité  Préparation pour des améliorations futures : Prise en charge des bandes dessinées japonaises (lecture de droite à gauche). |  |

# CONCEPTION GLOBALE

Pour ce projet, nous avons opté pour une architecture en 4 couches très similaire aux architectures en 3 couches classiques comme le modèle MVC (modèle-vue-contrôleur), ou le modèle MVP(modèle-vue-Présentation).

## Une Architecture en 4 couches

L’architecture logicielle suit le modèle en 4 couches, séparant les responsabilités et garantissant une évolutivité, une maintenabilité et une testabilité optimales :

### Couche présentation (UI Layer)

Cette couche est directement visible et manipulée par l'utilisateur. Elle comprend les éléments visuels et interactifs tels que les boutons, menus, carrousels de bandes dessinées, et animations. Elle repose sur un framework front-end moderne et très populaire : Qt

* **Responsabilités** :
  + Afficher les pages des bandes dessinées.
  + Gérer les interactions utilisateur (clics, glissements, zoom, etc.).
  + Communiquer avec la couche métier pour récupérer des données ou déclencher des actions.

Elle permet par exemple, la navigation entre les pages (barre de défilement ou boutons de navigation), et la gestion du mode lecture (vue pleine page, double page, ou défilement vertical).

Les classes qui y sont présentes sont :

* **UIManager**: Coordonne les vues (comme Pageview et LibraryView) et gère les transitions entre les pages ou les modes d’affichage.
* **Pageview**: Responsable de l’affichage d’une page de bande dessinée à l’écran. Inclut des fonctionnalités comme le zoom (zoomEvent), le rendu visuel (render), et le basculement entre le mode simple et double page (toggleDualPageMode).
* **LibraryView**: Affiche les bibliothèques de comics. Permet de parcourir la collection

### Couche métier :

Cette couche contient la logique métier principale de l’application. Elle agit comme un pont entre l'interface utilisateur et les couches sous-jacentes en orchestrant les règles de fonctionnement.

* **Responsabilités** :
  + Traiter les fichiers (importation, analyse des métadonnées, décompression d'archives).
  + Appliquer les règles métier (gestion de la progression de lecture etc.).
  + Coordonner les actions entre la couche présentation et la couche infrastructure.

Elle permet par exemple la Gestion des bibliothèques : ajout, suppression ou modification des collections.

Les classes qui y sont présentes sont :

* **Tag :** Permet d’étiqueter les comics avec des mots-clés (genres, personnages, etc.) pour faciliter la recherche et l’organisation de la bibliothèque.
* **PageLoader**: Charge les pages de comics depuis des fichiers (CBR, CBZ, PDF) ou une source externe, en optimisant la vitesse de chargement et la gestion de la mémoire.
* **Library**: Gère la collection complète de comics : ajout, suppression, tri, et organisation des livres. Centralise l’accès aux métadonnées et aux fichiers.
* **CacheManager**: Améliore les performances en stockant temporairement les pages ou images fréquemment consultées (exemple : garder en cache les dernières pages lues).
* **TOCManager** (Table des Matières) : Gère la navigation entre les chapitres, sections ou volumes d’un comic. Permet de sauter rapidement à une partie spécifique du livre.
* **AbstractBook**: Classe abstraite définissant les méthodes communes pour tous les formats de livres (PDFBook, CBRBook, CBZBook), comme l’ouverture, la fermeture ou l’extraction de pages.
* **TOC** (Table des Matières) : Représente la structure hiérarchique d’un comic (titres de chapitres, numéros de pages), utilisée par le **TOCManager** pour la navigation.
* **TagManager**: Administre les étiquettes (**Tag**) : création, suppression, et association aux comics. Facilite les recherches avancées (exemple : filtrer par "Super-héros" ou "Année 2020").
* **Page**: Représente une page individuelle d’un comic, stockant son contenu (image, texte) et ses métadonnées (numéro, résolution).
* **PDFBook, CBRBook, CBZBook**: Implémentations concrètes de **AbstractBook** pour gérer les formats de fichiers spécifiques (PDF, CBR, CBZ), incluant la décompression et l’extraction des pages.
* **AbstractImage**: Classe abstraite pour les images, définissant des méthodes communes (chargement, redimensionnement). Les classes dérivées (**JPEGImage, PNGImage, BMPImage**) gèrent les formats spécifiques.
* **JPEGImage, PNGImage, BMPImage**: Représentent des images dans des formats spécifiques, utilisées pour afficher les pages de comics tout en gérant leurs particularités (compression, transparence).

### Couche infrastructure (Infrastructure Layer)

Cette couche gère les services techniques nécessaires au bon fonctionnement de l’application. Elle englobe les fonctionnalités non spécifiques au métier, mais essentielles, comme le traitement des images, la gestion des fichiers, ou encore la communication avec des services tiers.

* **Responsabilités** :
  + Décompresser et lire les fichiers (CBR, CBZ, PDF).
  + Optimiser les performances (préchargement des pages suivantes, gestion de la mémoire pour les grandes images).
  + Intégrer des bibliothèques tierces pour des fonctionnalités comme le traitement d’images ou la gestion des formats de fichiers.

Les classes qui y sont présentes sont :

* **AbstractFilter**: Classe de base pour les filtres de traitement d’images ou de texte (**TextEnhancementFilter** pour l’amélioration du texte et **LowPassFilter** pour la réduction du bruit).
* **TextEnhancementFilter**: Améliore la lisibilité du texte dans les images de comics (ajustement de la netteté, contraste, ou OCR pour extraire le texte des bulles).
* **LowPassFilter**: Applique un flou ou un lissage pour réduire le bruit ou les artefacts dans les images scannées, utile pour les vieux comics numérisés.
* **ImageProcessor :** C’est une classe qui implémente les fonctions qui permettent de traiter les images des comics (applique filtres, convertit formats, optimise performances).Elle Collabore avec les autres classes comme : AbstractFilter, PageLoader, CacheManager.

### Couche d’accès aux données (Data Access Layer)

Cette couche est responsable de la persistance et de la récupération des données. Elle peut inclure des bases de données locales (SQLite pour les préférences et l'historique) ou des services cloud pour la synchronisation.

* **Responsabilités** :
  + Lire et écrire des fichiers de configuration et des bases de données locales.
  + Récupérer les informations stockées, comme les bibliothèques de bandes dessinées ou les préférences utilisateur.

Les classes qui y sont présentes sont :

* **FileHandler :** Manipule les fichiers bruts (lecture/écriture). C’est la base pour accéder aux fichiers, quel que soit leur type.
* **MetaDataHandler :** Gère les métadonnées associées aux fichiers (auteur, date, tags, etc.). Permet d’ajouter et modifier des informations contextuelles (ex. : comics, images).
* **ArchiveHandler :** Gère les archives compressées (ZIP, RAR, etc.). Permet de manipuler des collections de fichiers (ex. : comics au format CBZ/CBR).

## Architecture logicielle

### Description des modules principaux

Ce projet repose sur une architecture modulaire, composée de plusieurs modules principaux qui s'articulent pour offrir une expérience fluide et intuitive à l'utilisateur. Ces modules sont :

1. **Interface Utilisateur (UI)** :

Ce module est chargé de gérer l’interaction utilisateur. Il fournit une interface graphique permettant de naviguer dans les bandes dessinées, de zoomer, d'effectuer des recherches ou d’accéder aux paramètres. Le design est adapté pour être compatible avec diverses tailles d’écrans, comme les ordinateurs, tablettes ou smartphones.

1. **Gestion des fichiers** :

Ce module traite les fonctionnalités liées à l'importation et à l'organisation des fichiers de bandes dessinées. Il prend en charge différents formats de fichiers populaires (CBR, CBZ, PDF, etc.) et inclut des fonctionnalités comme la gestion de bibliothèques personnelles, et le tri des fichiers.

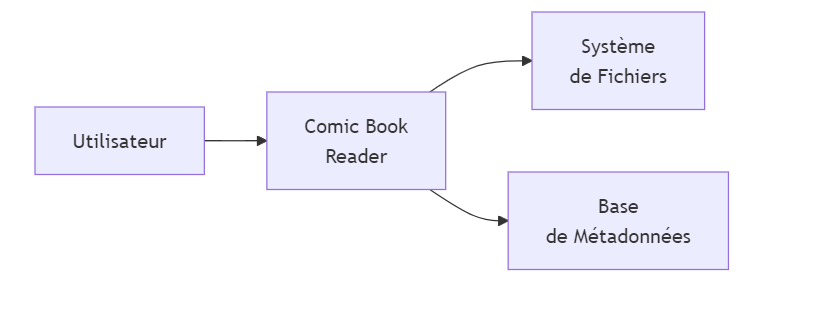
1. **Traitement des images** :

Le traitement des images est au cœur du projet. Ce module gère le chargement, le rendu et la manipulation des pages de bandes dessinées, notamment le redimensionnement dynamique, le zoom, la rotation, ou encore l’amélioration de la qualité des images.

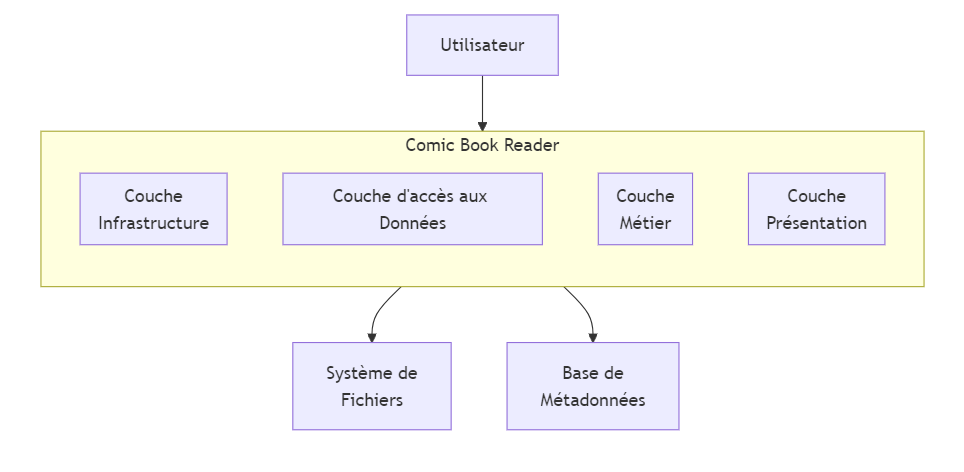
1. **Gestion des préférences et de l’historique** :

Ce module gère les paramètres utilisateur (mode lecture, thème sombre/clair, taille de zoom par défaut, etc.) ainsi que l’historique de lecture (dernière page lue, progression dans un volume, etc.). Les préférences sont enregistrées pour personnaliser l'expérience de lecture et peuvent être synchronisées sur plusieurs appareils si un service cloud est utilisé.

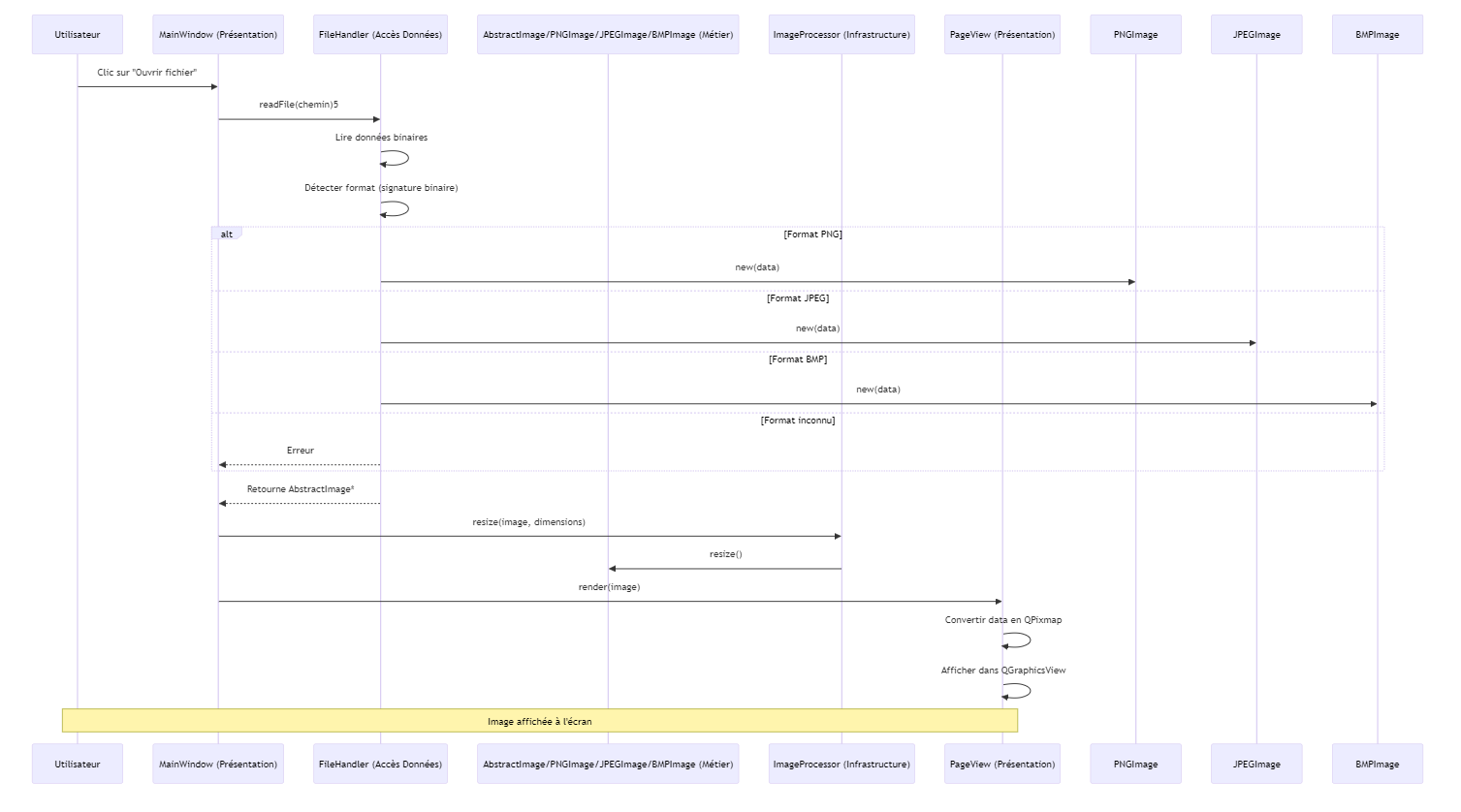
## Diagramme de contexte



## Diagramme de Paquetage



## Diagrammes de séquence



# CAHIER DE CONCEPTION

## Technologies utilisées :

Dans ce projet, plusieurs langages de programmation et outils ont été utilisés pour construire l'application, chacun ayant un rôle spécifique dans le processus de développement. Ces langages et outils incluent **C++**, **Qt (via CMake)**. Chacun de ces éléments joue un rôle crucial dans la fonctionnalité et l'architecture de l'application.

### C++ : Le Langage Principal

Le **C++** est le langage de programmation principal utilisé pour développer l'application. Au-delà de la contrainte imposée par le projet, Il est particulièrement adapté pour les applications nécessitant des performances élevées, telles que celles manipulant des images. Dans ce projet, le langage C++ est utilisé pour la logique de traitement des images, la gestion de l'interface utilisateur et la communication avec les bibliothèques externes comme **ImageMagick**. C++ permet de créer une application robuste et efficace grâce à sa gestion fine de la mémoire, sa performance native et son large écosystème de bibliothèques.

Le code C++ dans ce projet est structuré autour de classes qui représentent des objets et des fonctionnalités de l'application, comme les images (par exemple, PNGImage, JPEGImage, etc.), et il gère les différentes étapes de manipulation des données.

### Qt : Framework de Développement d'Interface Graphique

Le projet utilise également Qt, un framework C++ populaire pour le développement d'applications graphiques multiplateformes. Qt est utilisé pour la gestion de l'interface utilisateur (UI), avec des éléments comme les fenêtres, les boutons, les menus, et la gestion des événements utilisateur. En effet, il facilite le développement de GUI complexes grâce à son ensemble d'outils comme Qt Widgets, qui est utilisé ici pour la création de l'interface. Il nous a permet également la gestion des traductions (via le module LinguistTools), ce qui rend l'application plus accessible à des utilisateurs de différentes régions et langues. Grâce à la compatibilité multiplateforme de Qt, l'application peut être facilement déployée sur Windows, linux et d'autres systèmes d'exploitation sans avoir besoin de modifications majeures dans le code source.

### CMake : Outil de Construction et de Gestion de Projet

Le fichier CMakeLists.txt utilisé dans ce projet est un fichier de configuration pour CMake, un outil de gestion de la construction multiplateforme. Il nous a permet de générer les fichiers de construction spécifiques au système, ce qui rend le projet facile à compiler sur différentes plateformes sans modification manuelle des scripts de compilation. Utilisant un langage de script propre qui permet de définir les dépendances, les chemins de bibliothèques et les options de compilation.

Dans ce projet, CMake nous a permis d’intégrer des bibliothèques externes comme **Qt** et **ImageMagick**, et de définir les sources de l'application ainsi que les paramètres de l'éditeur de liens pour générer l'exécutable final.

### ImageMagick : Bibliothèque de Manipulation d'Images

**ImageMagick** est une bibliothèque externe utilisée pour la manipulation et la conversion d'images. Elle fournit un ensemble d'outils puissants pour traiter les images dans divers formats (JPEG, PNG, BMP, etc.), les redimensionner, les convertir, les appliquer des filtres, et bien plus encore. Dans ce projet, **ImageMagick++**, l'interface C++ de la bibliothèque, nous a permis d’intégrer ces fonctionnalités au sein de l'application. Ainsi, Cette bibliothèque offre une grande flexibilité et permet de manipuler les images de manière hautement performante, ce qui est essentiel pour des applications nécessitant un traitement d'images efficace.

### Structure de l’Interface Utilisateur avec Qt

L’interface du Comic Book Reader repose sur une architecture modulaire exploitant les composants Qt pour allier performance et ergonomie.

* Au cœur de l’application, la fenêtre principale (QMainWindow) orchestre les éléments clés : une barre de menus (QMenuBar) pour les opérations globales (ouverture de fichiers, préférences),
* une barre d’outils personnalisable (QToolBar) regroupant les contrôles fréquents (zoom, navigation)
* une zone centrale de visualisation intégrée dans un (QScrollArea). Cette dernière héberge un PageViewer, un widget personnalisé dérivé de QGraphicsView, qui affiche les pages via un système de scène (QGraphicsScene) et d’items graphiques (QGraphicsPixmapItem).

Les panneaux dockables (QDockWidget) enrichissent l’expérience utilisateur. À gauche, un panneau de bibliothèque utilise un QTreeView couplé à un modèle personnalisé pour afficher les collections de comics, avec des miniatures générées à la volée.

L’interface adopte une gestion thématique via des styles CSS (QSS). Un système de chargement dynamique permet de basculer entre des thèmes prédéfinis (sombre, clair). La barre d’état (QStatusBar) affiche des retours contextuels : indicateur de progression lors du chargement des archives, notifications temporaires, ou métriques de performance (FPS, mémoire utilisée).

Pour l’internationalisation, l’application utilise Qt Linguist : tous les textes sont encapsulés dans des appels tr(), permettant une traduction aisée via des fichiers .ts générés automatiquement. Enfin, des outils de débogage intégrés (qDebug) facilitent l’inspection hiérarchique des widgets et le profilage des performances, garantissant une interface réactive et stable. Cette structure, combinant composants natifs Qt et personnalisations ciblées, offre une plateforme à la fois puissante et intuitive pour la lecture de bandes dessinées numériques.



# CONCLUSION

Le développement du **Comic Book Reader/Writer** illustre la manière dont une application bien conçue peut répondre aux attentes variées des amateurs de bandes dessinées numériques. Ce projet a permis d’implémenter des fonctionnalités essentielles comme la lecture fluide des formats .CBR et .CBZ, tout en introduisant des options novatrices pour la personnalisation de l’expérience utilisateur, telles que l’ajout de tags, le support des BD japonaises, et l’intégration de tables des matières.

Grâce à une architecture modulaire et extensible, cette application ouvre la voie à de futures évolutions, comme l’intégration de nouveaux formats ou l’optimisation des performances avec des technologies graphiques avancées. Ce projet a ainsi démontré la faisabilité et la pertinence d’une solution numérique adaptée aux besoins spécifiques des passionnés de bandes dessinées.

En conclusion, ce travail représente non seulement une solution technique mais également une plateforme évolutive qui pourrait servir de base à des outils encore plus sophistiqués. Ce projet reflète l’engagement envers une approche centrée sur l’utilisateur, mariant créativité et efficacité technique.