DALMAYRAC--BELASCAIN Gorka
CAMPISTRON Julian
28/05/2024

# Dossier d’analyse et conception
# **Lecteur de Diaporama**

## Groupe III-TP5-4
### SAE S2.01 : Développement d'une application

BUT Informatique 1ère année Semestre 2

GitHub: https://github.com/Gorka-DB/S2.01-Rendus

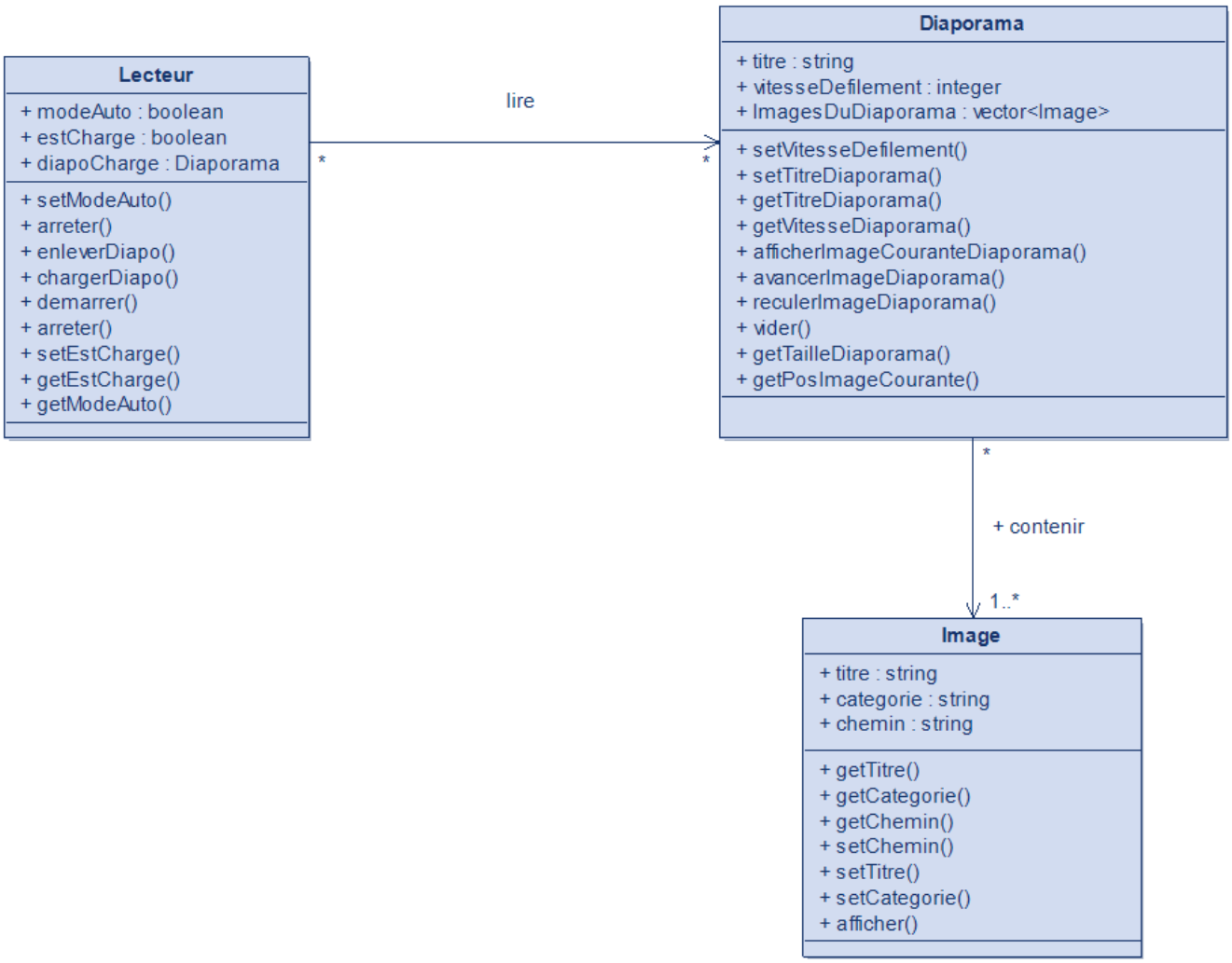
**Date de remise : 11/05/2024**

# Table des matières

V1 – Version non graphique ..... 2
UML : diagramme des classes ..... 3
Documentation des méthodes et attributs de chaque classe ..... 4
1. Classe Lecteur ..... 4
2. Classe Diaporama ..... 5
3. Classe Image ..... 6
4. Classe lecteurvue ..... 7
5. Classe LecteurPresentation ..... 7
V2 – Version Graphique ..... 9
Diagrammes état-transition de l'application ..... 9
1. Forme UML ..... 9
2. Forme matricielle ..... 9
Lien Eléments de l'interface – Fonctionnalités ..... 9
V2_MVP – Version Graphique MVP ..... 11
UML : diagramme des classes modifié ..... 11
Documentation des méthodes et attributs modifiés à la V2 MVP ..... 12
1. Classe Lecteur ..... 12
2. Classe Diaporama ..... 13
3. Classe ImageDansDiaporama ..... 14
4. Classe lecteurvue ..... 15
5. Classe LecteurPresentation ..... 16
V3_MVP – Version Graphique 3 MVP ..... 18
Documentation des méthodes et attributs modifiés à la V3 MVP ..... 18
1. Classe Diaporama ..... 18
2. Classe LecteurPresentation ..... 19
3. Classe lecteurvue ..... 19

# V1 – Version non graphique

## UML : diagramme des classes



# Documentation des méthodes et attributs de chaque classe

## 1. Classe Lecteur

**Signification des attributs et méthodes de la classe Lecteur**

La classe Lecteur est conçue pour gérer l'affichage des diapositives. Elle possède plusieurs attributs et méthodes :

### Attributs

**1. Diaporama* diaporama** : Cet attribut est de type Diaporama et représente le pointeur vers le diaporama actuellement chargé dans le lecteur. Il est utilisé pour charger différents diaporamas.

**2. unsigned idDiaporama** : Cet attribut représente l'identifiant en Base de Données du diaporama courant, = 0 si pas de diaporama dans le lecteur

**3. unsigned int posImageCourante**: Cet attribut est également de type int et représente la position de l'image courante du diaporama courant

### Méthodes

**1. Lecteur::Lecteur(Diaporama diapoCharge, bool modeAuto, bool estCharge)** : Cette méthode est le constructeur de la classe Lecteur. Elle initialise les attributs diapoCharge, modeAuto, et estCharge avec les valeurs passées en paramètre.

**2. unsigned int getIdDiaporama() const** : Cette méthode retourne l'identifiant du diaporama

**3. Diaporama* getDiaporama() const** : Cette méthode retourne le diaporama

**4. unsigned int getPosImageCourante() const** : Cette méthode retourne la position de l'image courante

**5. bool lecteurVide() const :**

**6. ImageDansDiaporama* getImageCourante() const;** // retourne le pointeur vers l'image courante

**7. unsigned int nbImages() const**;  // taille de la collection pointée par diaporama

**8. void afficher();**  // affiche les informations sur lecteur + éventuellement diaporama et image courante

**9. void setIdDiaporama(unsigned int pIdDiaporama);**

**10. void setDiaporama (Diaporama* pDiaporama);**

**11. void setPosImageCourante(unsigned int pPosImageCourante);**

**12. void changerDiaporama(unsigned int pId, string pTitre="", unsigned int pVitesse=0);**

**13. void avancer();**

**14. void reculer();**

**15. void viderLecteur();**

## 2. Classe Diaporama

La classe Diaporama est conçue pour gérer les diaporamas.

### Attributs

**1. string titreDiaporama** : Cet attribut est de type string et représente le titre du diaporama. Il est utilisé pour afficher le titre du diaporama.

**2. unsigned int vitesseDefilement** : Cet attribut est de type unsigned int et représente la vitesse de défilement des images dans le diaporama. Il est utilisé pour contrôler la vitesse à laquelle les images sont affichées.

**3. vector<Image> ImagesDuDiaporama** : Cet attribut est un vecteur d'images de type Image et représente l'ensemble des images comprises dans le diaporama. Il est utilisé pour afficher l'image souhaitée du diaporama ou bien encore modifier celui-ci

**4. int posImageCourante** : Cet attribut est de type unsigned int et représente la position du « curseur » dans le vecteur des images du diaporama. Il est donc initialisé à 0, et n'est pas présent dans les constructeurs. Il permet de parcourir le vecteur des images du diaporama

### Méthodes

**1. Diaporama(string pTitre, unsigned int pVitesseDefilement)** : Cette méthode est le constructeur de la classe Diaporama. Elle prend en paramètre un titre pTitre et une vitesse de défilement pVitesseDefilement et initialise les attributs titreDiaporama et vitesseDefilement en conséquence.

**2. void vider()** : Cette méthode permet de vider le diaporama. Elle supprime toutes les images du diaporama.

**3. void setVitesseDefilement(unsigned int pVitesseDefilement)** : Cette méthode permet de modifier la vitesse de défilement du diaporama. Elle prend en paramètre une nouvelle vitesse pVitesseDefilement et modifie l'attribut vitesseDefilement en conséquence.

**4. void setTitreDiaporama(string pTitre)** : Cette méthode permet de modifier le titre du diaporama. Elle prend en paramètre un nouveau titre pTitre et modifie l'attribut titreDiaporama en conséquence.

**5. string getTitreDiaporama()** : Cette méthode permet de récupérer le titre du diaporama. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant le titre du diaporama.

**6. int getVitesseDefilement()** : Cette méthode permet de récupérer la vitesse de défilement du diaporama. Elle ne prend pas de paramètre et retourne un entier représentant la vitesse de défilement du diaporama.

**7. int getTailleDiaporama()** : Cette méthode permet de récupérer la taille du diaporama. Elle ne prend pas de paramètre et retourne un entier représentant le nombre d'images dans le diaporama.

**8. int getPosImageCourante()** : Cette méthode permet de récupérer la position de l'image courante dans le diaporama. Elle ne prend pas de paramètre et retourne un entier représentant la position de l'image courante dans le diaporama.

**9. void ajouterImage(Image image)** : Cette méthode permet d'ajouter une image au diaporama. Elle prend en paramètre une image image et ajoute cette image à la fin du diaporama.

**10. void afficherImageCouranteDiaporama()** : Cette méthode permet d'afficher l'image courante du diaporama. Elle ne prend pas de paramètre et affiche l'image à la position posImageCourante dans le diaporama

**11. void avancerImageDiaporama()** : Cette méthode permet de passer à l'image suivante du diaporama. Elle ne prend pas de paramètre et incrémente la variable posImageCourante.

**12. void reculerImageDiaporama()**: Cette méthode permet de passer à l'image suivante du diaporama. Elle ne prend pas de paramètre et décrémente la variable posImageCourante.

## 3. Classe Image

La classe Image est conçue pour gérer les images. Elle possède plusieurs attributs et méthodes qui sont décrits ci-dessous.

### Attributs

**1. string titre** : Cet attribut est de type string et représente le titre de l'image.

**2. string categorie** : Cet attribut est de type string et représente la catégorie de l'image. Il est utilisé pour fournir des informations supplémentaires sur l'image.

**3. string chemin** : Cet attribut est de type string et représente le chemin d'accès de l'image. Il est utilisé pour spécifier l'emplacement de l'image sur le disque dur.

### Méthodes

**1. Image(string pTitre, string pCategorie, string pChemin)** : Cette méthode est le constructeur de la classe Image. Elle prend en paramètre un titre pTitre, une categorie pCategorie et un chemin pChemin et initialise les attributs citre, categorie et chemin en conséquence.

**2. void afficher()** : Cette méthode permet d'afficher l'image. Elle ne prend pas de paramètre et affiche l'image à l'écran.

**3. void setTitre(string pTitre)** : Cette méthode permet de modifier le titre de l'image. Elle prend en paramètre un nouveau titre pTitre et modifie l'attribut titre en conséquence.

**4. void setCategorie(string pCategorie)** : Cette méthode permet de modifier la catégorie de l'image. Elle prend en paramètre une nouvelle categorie pCategorie et modifie l'attribut categorie en conséquence.

**5. void setChemin(string pChemin)** : Cette méthode permet de modifier le chemin d'accès de l'image. Elle prend en paramètre un nouveau chemin pChemin et modifie l'attribut chemin en conséquence.

**6. string getTitre()** : Cette méthode permet de récupérer le titre de l'image. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant le titre de l'image.

**7. string getCategorie()** : Cette méthode permet de récupérer la catégorie de l'image. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant la catégorie de l'image.

**8. string getChemin()** : Cette méthode permet de récupérer le chemin d'accès de l'image. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant le chemin d'accès de l'image.

## 4. Classe lecteurvue

La classe Image est conçue pour gérer les images. Elle possède plusieurs attributs et méthodes qui sont décrits ci-dessous.

### Attributs

**1. string titre** : Cet attribut est de type string et représente le titre de l'image.

**2. string categorie** : Cet attribut est de type string et représente la catégorie de l'image. Il est utilisé pour fournir des informations supplémentaires sur l'image.

**3. string chemin** : Cet attribut est de type string et représente le chemin d'accès de l'image. Il est utilisé pour spécifier l'emplacement de l'image sur le disque dur.

### Méthodes

**1. Image(string pTitre, string pCategorie, string pChemin)** : Cette méthode est le constructeur de la classe Image. Elle prend en paramètre un titre pTitre, une categorie pCategorie et un chemin pChemin et initialise les attributs citre, categorie et chemin en conséquence.

**2. void afficher()** : Cette méthode permet d'afficher l'image. Elle ne prend pas de paramètre et affiche l'image à l'écran.

## 5. Classe LecteurPresentation

La classe Image est conçue pour gérer les images. Elle possède plusieurs attributs et méthodes qui sont décrits ci-dessous.

### Attributs

**1. string titre** : Cet attribut est de type string et représente le titre de l'image.

**2. string categorie** : Cet attribut est de type string et représente la catégorie de l'image. Il est utilisé pour fournir des informations supplémentaires sur l'image.

**3. string chemin** : Cet attribut est de type string et représente le chemin d'accès de l'image. Il est utilisé pour spécifier l'emplacement de l'image sur le disque dur.

### Méthodes

**1. Image(string pTitre, string pCategorie, string pChemin)** : Cette méthode est le constructeur de la classe Image. Elle prend en paramètre un titre pTitre, une categorie pCategorie et un chemin pChemin et initialise les attributs citre, categorie et chemin en conséquence.

**2. void afficher()** : Cette méthode permet d'afficher l'image. Elle ne prend pas de paramètre et affiche l'image à l'écran.

**3. void setTitre(string pTitre)** : Cette méthode permet de modifier le titre de l'image. Elle prend en paramètre un nouveau titre pTitre et modifie l'attribut titre en conséquence.

**4. void setCategorie(string pCategorie)** : Cette méthode permet de modifier la catégorie de l'image. Elle prend en paramètre une nouvelle categorie pCategorie et modifie l'attribut categorie en conséquence.

**5. void setChemin(string pChemin)** : Cette méthode permet de modifier le chemin d'accès de l'image. Elle prend en paramètre un nouveau chemin pChemin et modifie l'attribut chemin en conséquence.

**6. string getTitre()** : Cette méthode permet de récupérer le titre de l'image. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant le titre de l'image.

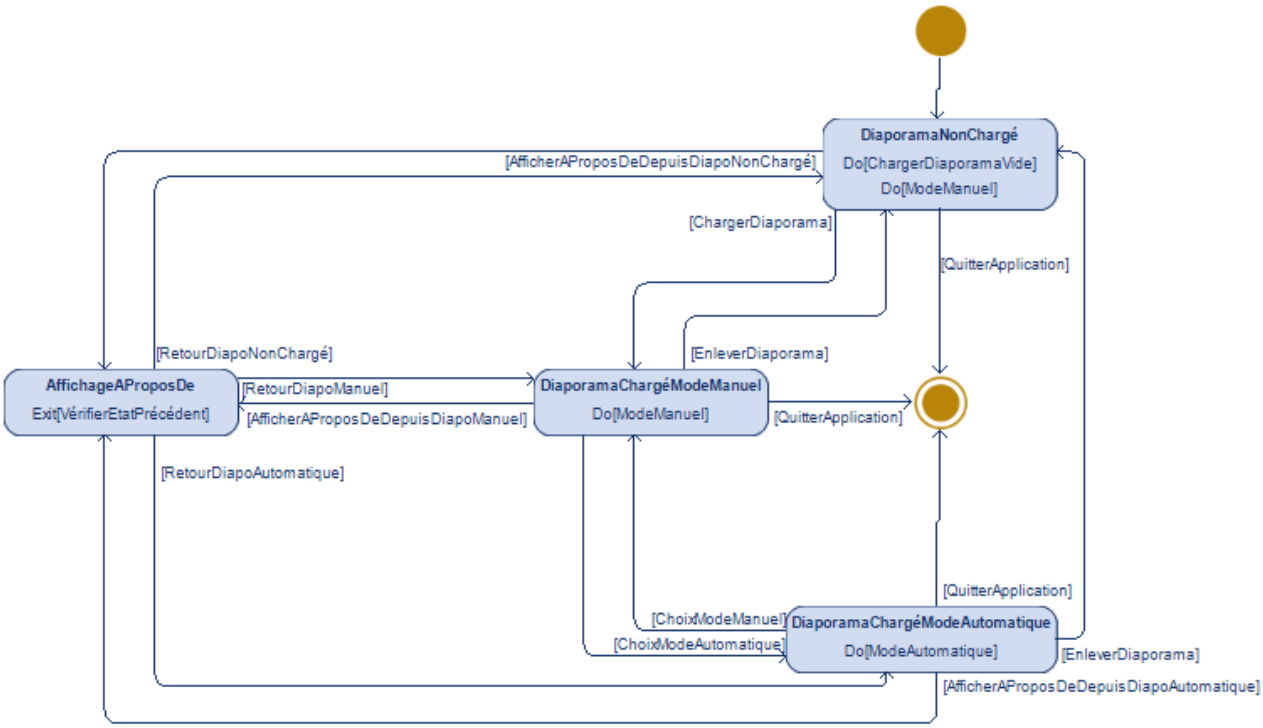
**7. string getCategorie()** : Cette méthode permet de récupérer la catégorie de l'image. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant la catégorie de l'image.

**8. string getChemin()** : Cette méthode permet de récupérer le chemin d'accès de l'image. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant le chemin d'accès de l'image.

# V2 – Version Graphique

## Diagrammes état-transition de l'application

### 1. Forme UML




### 2. Forme matricielle

| | actionA_propos_de | boutonValider_A_propos_de | actionCharger_diaporama | actionEnlever_diaporama | boutonLancer_diaporama | boutonArreter_diaporama | actionQuitter |
|---|---|---|---|---|---|---|---|
| **événements / Etats** | **AfficherAProposDe** | **RetirerAProposDe** | **ChargerDiaporama** | **EnleverDiaporama** | **BasculerModeAutomatique** | **BasculerModeManuel** | **QuitterApplication** |
| **DiaporamaNonChargé** | AffichageAProposDe | ___ | DiaporamaManuel | ___ | ___ | ___ | ___ |
| **DiaporamaManuel** | AffichageAProposDe | ___ | DiaporamaManuel | | DiaporamaAutomatique | ___ | ___ |
| **DiaporamaAutomatique** | AffichageAProposDe | ___ | DiaporamaManuel | DiaporamaNonChargé | ___ | DiaporamaManuel | ___ |
| | ___ | [EtatPrécédent=DiaporamaNonChargé]DiaporamaNonChargé | ___ | ___ | ___ | ___ | ___ |
| | ___ | [EtatPrécédent=DiaporamaManuel]DiaporamaManuel | ___ | ___ | ___ | ___ | ___ |
| **AffichageAProposDe** | ___ | [EtatPrécédent=DiaporamaAutomatique]DiaporamaAutomatique | ___ | ___ | ___ | ___ | ___ |

## Lien Eléments de l'interface – Fonctionnalités

**Label « Image »** (QLabel imgDiapo) : Label permettant l'affichage de l'image

**Bouton «Suivant »** (QPushButton btnSuivant) : Ce bouton déclenche le passage à l'image suivante du diaporama

**Bouton «Précédent »** (QPushButton btnPrecedent) : Ce bouton déclenche le passage à l'image précédente du diaporama

**Bouton «Lancer Diaporama »** (QPushButton btnLancer) : Ce bouton déclenche le passage au mode Automatique du diaporama, et débute donc le défilement d'images

**Bouton «Arrêter Diaporama »** (QPushButton btnArreter) : Ce bouton déclenche le passage au mode Manuel du diaporama

**Menu « Fichier »** (QMenu menuFichier)

Action « Quitter » : Permet la fermeture de l'application graphique

**Menu « Paramètres »** (QMenu menuParametres) : Menu permettant l'accès aux divers paramètres du diaporama ci-dessous

Action « Charger Diaporama » : Permet de charger un diaporama dans le lecteur

Action « Enlever Diaporama » : Permet la suppression du diaporama présent dans le lecteur

Action « Vitesse de défilement » : Permet l'ajustement de la vitesse de défilement

Action « Filtrer » : Permet de filtrer les images du diaporama chargé dans le lecteur

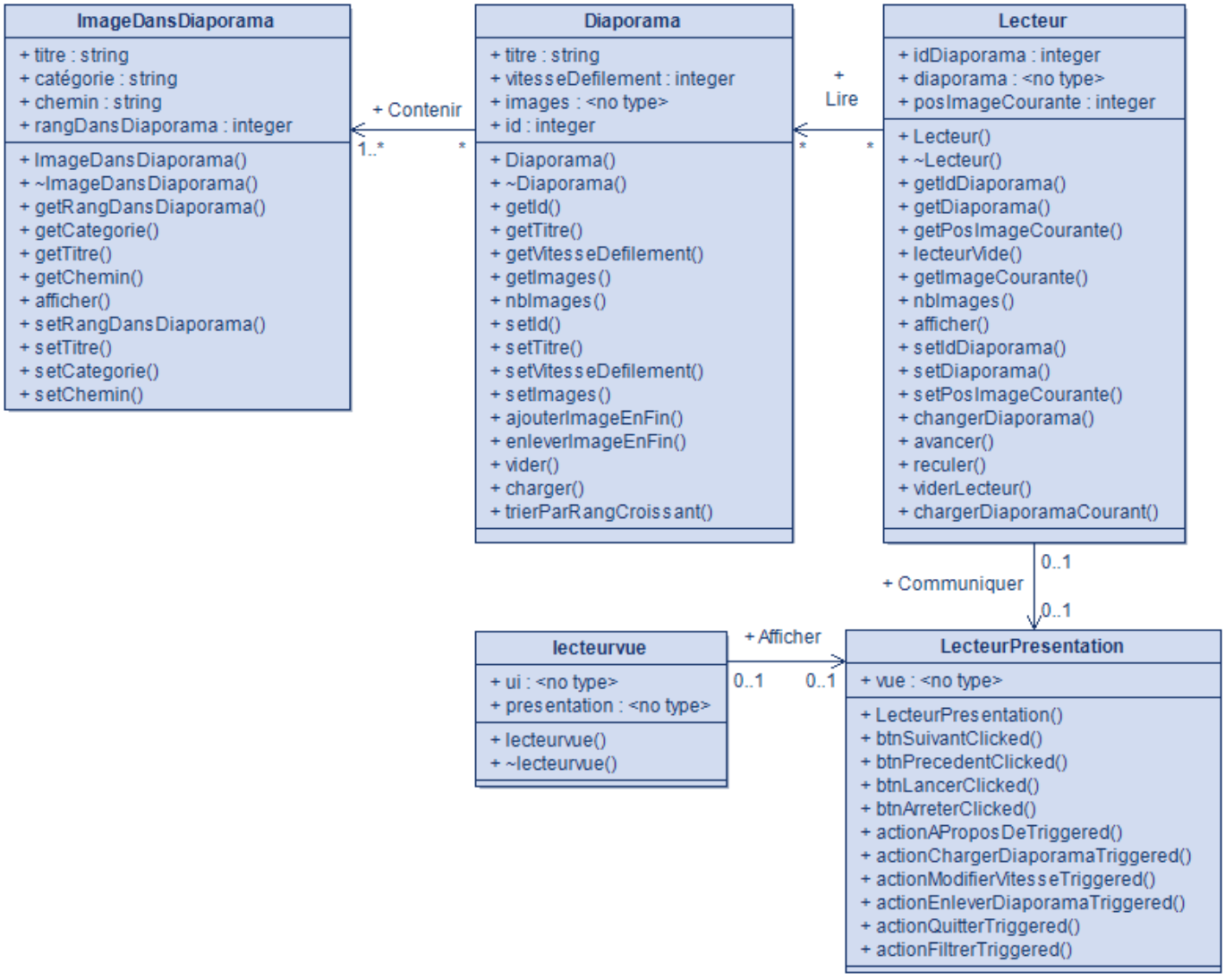
**Menu « Aide »** (QMenu menuAide)

Action « A propos de… » : Permet d'obtenir des informations sur la version de l'application ainsi que ses auteurs

# V2_MVP – Version Graphique MVP

Lors du passage à la V2 MVP, nous avons choisi de recommencer du début en nous basant sur le code de la V1 fourni par Mme. Dagorret afin de minimiser le risque de futures erreurs, c'est pourquoi de nombreuses modifications ont eu lieu, en plus du passage au modèle MVP.

## UML : diagramme des classes modifié



# Documentation des méthodes et attributs modifiés à la V2 MVP

## 1. Classe Lecteur

La classe lecteur est conçue pour gérer la lecture des diaporamas.

### Attributs

**1. unsigned int idDiaporama :** Cet attribut est de type unsigned int et représente l'identifiant en Base de Données du diaporama courant. Il est égal à 0 si aucun diaporama n'est chargé dans le lecteur.

**2. Diaporama* diaporama :** Cet attribut est un pointeur vers le diaporama associé au lecteur. Il est égal à nullptr si aucun diaporama n'est chargé dans le lecteur.

**3. unsigned int posImageCourante :** Cet attribut est de type unsigned int et représente la position de l'image courante dans le diaporama courant. Sa valeur est indéfinie quand le lecteur est vide ou que le diaporama est vide. Elle est initialisée à 0 et permet de parcourir les images du diaporama.

### Méthodes

**1.Lecteur() :** Ce constructeur initialise un nouvel objet Lecteur sans diaporama associé. Les attributs idDiaporama et diaporama sont initialisés à leurs valeurs par défaut (0 et nullptr respectivement).

**2.~Lecteur() :** Destructeur de la classe Lecteur. Il s'occupe de libérer les ressources allouées dynamiquement, si nécessaire.

**3. unsigned int getIdDiaporama() const :** Retourne l'identifiant du diaporama actuellement chargé dans le lecteur. Retourne 0 si aucun diaporama n'est chargé.

**4. Diaporama* getDiaporama() const :** Retourne un pointeur vers le diaporama actuellement chargé dans le lecteur. Retourne nullptr si aucun diaporama n'est chargé.

**5. unsigned int getPosImageCourante() const :** Retourne la position de l'image courante dans le diaporama chargé. La valeur est indéfinie si le lecteur ou le diaporama est vide.

**6. bool lecteurVide() const :** Retourne vrai si aucun diaporama n'est associé au lecteur, faux sinon.

**7. ImageDansDiaporama* getImageCourante() const :** Retourne un pointeur vers l'image courante du diaporama. Retourne nullptr si le lecteur ou le diaporama est vide.

**8. unsigned int nbImages() const :** Retourne le nombre d'images dans le diaporama actuellement chargé. Retourne 0 si le lecteur ou le diaporama est vide.

**9. void afficher() :** Affiche les informations sur le lecteur, y compris les détails du diaporama chargé et de l'image courante, si disponibles.

**10. void setIdDiaporama(unsigned int pIdDiaporama) :** Définit l'identifiant du diaporama associé au lecteur.

**11. void setDiaporama(Diaporama* pDiaporama) :** Associe un diaporama au lecteur. Prend en paramètre un pointeur vers un diaporama.

**12. void setPosImageCourante(unsigned int pPosImageCourante) :** Définit la position de l'image courante dans le diaporama chargé.

**13. void changerDiaporama(unsigned int pId, string pTitre="", unsigned int pVitesse=0) :** Permet de charger un nouveau diaporama dans le lecteur ou de le vider. Si pId est différent de 0, un nouveau diaporama est chargé avec les paramètres spécifiés. Si pId est égal à 0, le lecteur est vidé de son diaporama courant.

**14. void avancer() :** Avance à l'image suivante dans le diaporama. Si le diaporama est non vide, incrémente posImageCourante, modulo le nombre d'images dans le diaporama. Ne fait rien si le lecteur ou le diaporama est vide.

**15. void reculer() :** Recule à l'image précédente dans le diaporama. Si le diaporama est non vide, décrémente posImageCourante, modulo le nombre d'images dans le diaporama. Ne fait rien si le lecteur ou le diaporama est vide.

**16. void viderLecteur() :** Enlève le diaporama courant du lecteur, s'il en existe un. Réinitialise les attributs idDiaporama et diaporama à leurs valeurs par défaut.

**17. void chargerDiaporamaCourant() :** Charge les images du diaporama identifié par idDiaporama dans le lecteur. Cette méthode est privée et sert à initialiser le diaporama dans le lecteur avec les images correspondantes.

## 2. Classe Diaporama

La classe Diaporama est conçue pour gérer les diaporamas.

### Attributs

**1. string titre** : Cet attribut est de type string et représente le titre du diaporama. Il est utilisé pour afficher le titre du diaporama.

**2. unsigned int vitesseDefilement** : Cet attribut est de type unsigned int et représente la vitesse de défilement des images dans le diaporama. Il est utilisé pour contrôler la vitesse à laquelle les images sont affichées.

**3. ImagesDiaporama images** : Cet attribut est un vecteur de pointeurs sur les objets ImageDansDiaporama de ce diaporama.

**4. unsigned int id** : Cet attribut est de type unsigned int et représente l'identifiant du diaporama dans la Base de données. Il est utilisé pour identifier un diaporama.

### Méthodes

**1. Diaporama()** : Cette méthode est le constructeur de la classe Diaporama. Elle initialise un identifiant id, un titre titre et une vitesse de défilement vitesseDefilement.

**2. ~Diaporama()** : Cette méthode est le destructeur de la classe Diaporama.

**3. unsigned int getId() const** : Cette méthode permet de récupérer l'identifiant du diaporama. Elle ne prend pas de paramètre et retourne un nombre entier naturel représentant l'identifiant du diaporama.

**5. string getTitre() const** : Cette méthode permet de récupérer le titre du diaporama. Elle ne prend pas de paramètre et retourne une chaîne de caractères représentant le titre du diaporama.

**6. int getVitesseDefilement() const** : Cette méthode permet de récupérer la vitesse de défilement du diaporama. Elle ne prend pas de paramètre et retourne un entier représentant la vitesse de défilement du diaporama.

**7. ImagesDiaporama getImages() const** : Cette méthode permet de récupérer les images composant le diaporama. Elle ne prend pas de paramètre et retourne un vecteur de pointeurs sur les objets ImageDansDiaporama de ce diaporama.

**8. unsigned int nbImages() const** : Cette méthode permet de récupérer le nombre d'images composant le diaporama. Elle ne prend pas de paramètre et retourne un entier représentant le nombre d'images composant le diaporama.

**9. void setId(unsigned int pId)** : Cette méthode permet de modifier l'identifiant du diaporama. Elle prend en paramètre un nouvel identifiant pId et modifie l'attribut id en conséquence.

**10. void setTitre(string pTitre)** : Cette méthode permet de modifier le titre du diaporama. Elle prend en paramètre un nouveau titre pTitre et modifie l'attribut titre en conséquence.:

**11. void setVitesseDefilement(unsigned int pVitesseDefilement)** : Cette méthode permet de modifier la vitesse de défilement du diaporama. Elle prend en paramètre une nouvelle vitesse pVitesseDefilement et modifie l'attribut vitesseDefilement en conséquence.

**12. void setImages(const ImagesDiaporama& pImages)** : Cette méthode permet de modifier les images composant le diaporama. Elle prend en paramètre un nouveau vecteur pImages et modifie l'attribut images en conséquence.

**13. void ajouterImageEnFin(ImageDansDiaporama* pImage)** : Cette méthode permet d'ajouter une image au diaporama. Elle prend en paramètre le vecteur images et lui ajoute en dernière position une nouvelle image pImage.

**14. void enleverImageEnFin()** : Cette méthode permet d'enlever la dernière image du diaporama, et de supprimer l'objet image enlevé. Elle ne prend pas de paramètre et, s'il y a au moins une image dans le diaporama, supprime la dernière image du vecteur images.

**15. void vider()** : Cette méthode permet de vider le diaporama. Elle ne prend pas de paramètre et supprime toutes les images du diaporama.

**16. void charger()** : Cette méthode permet de charger un diaporama. Elle ne prend pas de paramètre, ajoute toutes les images présentes dans le diaporama sélectionné, puis effectue un tri par ordre croissant entre celles-ci.

**17. void trierParRangCroissant()** : Cette méthode permet de trier les images du diaporama par ordre de rang croissant. Elle ne prend pas de paramètre et effectue un tri à bulles afin de trier les images du diaporama par ordre croissant.

## 3. Classe ImageDansDiaporama

La classe ImageDansDiaporama est conçue pour représenter et générer les images d'un diaporama

### Attributs

**1. string titre** : Cet attribut est de type string et représente le titre de l'image. Il est utilisé pour afficher le titre de l'image.

**2. string catégorie** : Cet attribut est de type string et représente la catégorie de l'image. Il est utilisé pour afficher la catégorie de l'image, et servira plus tard au système de filtre

**3. string chemin**: Cet attribut est de type string et représente le chemin de l'image sur le disque. Il est utilisé pour afficher le chemin de l'image stockée en dur.

**4. unsigned int rangDansDiaporama :** Cet attribut est de type unsigned int et représente le rang de l'image au sein du diaporama auquel elle est associée. Le rang détermine l'ordre de l'image dans le diaporama.

### Méthodes

**1. ImageDansDiaporama(unsigned int pRangDansDiaporama=0, string pCategorie="", string pTitre="", string pChemin = "") :** Constructeur qui initialise une nouvelle instance de ImageDansDiaporama avec les valeurs fournies. Les valeurs par défaut de tous les paramètres sont zéro ou des chaînes vides, permettant la création d'une image sans informations définies.

**2. ~ImageDansDiaporama() :** Destructeur de la classe ImageDansDiaporama.

**3. unsigned int getRangDansDiaporama() const : Retourne le rang de l'image dans le diaporama.**

**4. string getCategorie() const :** Retourne la catégorie de l'image.

**5. string getTitre() const** : Retourne le titre de l'image.

**6. string getChemin() const** : Retourne le chemin complet de l'image.

**7. void afficher() :** Affiche tous les champs de l'image, y compris le rang dans le diaporama, la catégorie, le titre, et le chemin. Cette méthode est utile pour le débogage ou l'affichage d'informations détaillées sur l'image.

**8. void setRangDansDiaporama(unsigned int pRangDansDiaporama) :** Définit le rang de l'image dans le diaporama.

**9. void setTitre(string pTitre) :** Définit le titre de l'image.

**10. void setCategorie(string pCategorie) :** Définit la catégorie de l'image.

**11. void setChemin(string pChemin) :** Définit le chemin de l'image.

## 4. Classe lecteurvue

La classe lecteurvue est une interface utilisateur pour le lecteur de diaporama. Elle sert de vue dans le modèle de conception MVP (Modèle-Vue-Présentation).

### Attributs

**Ui::lecteurvue *ui :** Un pointeur vers l'interface utilisateur générée par Qt Designer. Cet attribut permet d'accéder et de manipuler les éléments de l'interface graphique définis dans le fichier .ui.

**LecteurPresentation* presentation :** Un pointeur vers une instance de LecteurPresentation, qui agit comme la présentation entre la vue (lecteurvue) et le modèle (les données du diaporama).

### Méthodes

**explicit lecteurvue(QWidget *parent = nullptr) :** Constructeur de la classe lecteurvue. Initialise une nouvelle instance de la vue. parent est un pointeur vers le widget parent, par défaut à nullptr, signifiant que le widget peut être la fenêtre de niveau supérieur.

**~lecteurvue() :** Destructeur de la classe lecteurvue.

## 5. Classe LecteurPresentation

La classe LecteurPresentation sert de couche de présentation. Elle fait le lien entre l'interface utilisateur (la vue) et les données/logique de l'application (le modèle).

### Attributs

**lecteurvue* vue :** Pointeur vers la vue associée à cette présentation.

### Méthodes

**LecteurPresentation(lecteurvue* vue) :** Constructeur qui initialise l'instance de LecteurPresentation avec une vue spécifique.

### Slots Publics

**void btnSuivantClicked() :** Slot appelé lorsque le bouton "Suivant" est cliqué dans l'interface utilisateur. Il doit gérer la logique pour passer à l'image suivante dans le diaporama.

**void btnPrecedentClicked() :** Slot appelé lorsque le bouton "Précédent" est cliqué. Il gère la navigation à l'image précédente dans le diaporama.

**void btnLancerClicked() :** Slot appelé lorsque le bouton "Lancer" (pour démarrer le diaporama automatiquement) est cliqué. Il active le mode de lecture automatique du diaporama.

**void btnArreterClicked() :** Slot appelé lorsque le bouton "Arrêter" (pour stopper le diaporama automatique) est cliqué. Il désactive le mode de lecture automatique et permet à l'utilisateur de contrôler manuellement la navigation dans le diaporama.

**void actionAProposDeTriggered() :** Slot appelé lorsque l'action "À propos de" est activée, généralement depuis un menu ou un bouton dédié. Il peut afficher des informations sur l'application.

**void actionChargerDiaporamaTriggered() :** Slot pour gérer l'action de chargement d'un nouveau diaporama dans l'application.

**void actionModifierVitesseTriggered() :** Slot appelé pour modifier la vitesse de défilement du diaporama en mode automatique.

**void actionEnleverDiaporamaTriggered() :** Slot pour gérer l'action d'enlever le diaporama actuellement chargé dans l'application.

**void actionQuitterTriggered() :** Slot appelé pour gérer l'action de quitter l'application.

**void actionFiltrerTriggered()** : Slot pour gérer l'action de filtrage des images du diaporama selon certains critères.

# V3_MVP – Version Graphique 3 MVP

**Version 3 de lecteur de diaporama en c++ orienté objet avec interface graphique Qt**.

Résumé : Cette version corrige de nombreuses erreurs faites lors de la V2 MVP, comme une mauvaise implémentation du modèle MVP, dans les classes LecteurPresentation et lecteurVue en particulier. Elle ajoute la fonctionnalité d'affichage des images dans l'interface visuelle, ainsi que le fonctionnement du lecteur en "mode manuel" ( défilement des images lors du clic sur un bouton suivant ou précédent ), quitter l'application via un menu Fichier >>> Quitter, ainsi qu'obtenir des informations relatives à l'application via un menu Aide >>> A propos de...

## Documentation des méthodes et attributs modifiés à la V3 MVP

Les modifications majeures apportées dans cette version se concentrent en particulier sur les fichiers lecteurpresentation.cpp/.h, lecteur.cpp/.h, et diaporamas.cpp/.h

Elle comprend l'ajout d'un fichier ressources images.qrc contenant les images stockées en dur, qui seront utilisées par le lecteur de diaporamas, ainsi que d'autres modifications/ajouts listés ci-dessous.

### 1. Classe Diaporama

La classe Diaporama est conçue pour gérer les diaporamas. Dans cette version nous avons ajouté 1 attribut et 3 méthodes publiques

#### Attributs ajoutés :

**unsigned int posImageCourante** : Cet attribut est de type unsigned int et représente la position de l'image courante du diaporama.

#### Méthodes ajoutées :

**ImageDansDiaporama* getImageCourante() const** : Retourne l'image à la position courante

**unsigned int getPosImageCourante() const** : Retourne la position de l'image courante

**void setPosImageCourante(int pos)** : Définit la position de l'image courante

## 2. Classe LecteurPresentation

La classe ImageDansDiaporama est conçue pour représenter et générer les images d'un diaporama. Dans cette version nous avons ajouté un struct correspondant aux infos des diaporamas qui seront utilisées plus tard

```
struct InfosDiaporama {
        unsigned int id;                    // identifiant du diaporama dans la BD
        string titre;                       // titre du diaporama
        unsigned int vitesseDefilement; //vitesse de défilement
};
```

Nous avons également renommé **lecteurvue* vue** en **lecteurvue* laVue** pour éviter de confondre la classe avec le pointeur contenu dans le Lecteur

Remplacement de **LecteurPresentation(lecteurvue* vue)** par **LecteurPresentation()** : Constructeur de la présentation du lecteur vide.

### Attributs ajoutés :

Ajouts :

**Lecteur* leLecteur** : Pointeur vers le lecteur

**Diaporamas listeDeDiaporamas** : La liste de diaporama existants

**QTimer* leTimer** : Pointeur vers le timer

### Méthodes ajoutées :

**Lecteur* getLecteur()** : Retourne le lecteur.

**lecteurvue* getVue()** : Retourne la vue.

**ImageDansDiaporama* getImageCourante()** : Retourne le pointeur vers l'image courante

**Diaporama* getDiapoCourant()** : Retourne le diaporama courant

**void setLecteur(Lecteur*)** : Définit le lecteur (le modèle)

**void setVue(lecteurvue*)** : Définit la vue

## 3. Classe lecteurvue

La classe lecteurvue est une interface utilisateur pour le lecteur de diaporama. Elle sert de vue dans le modèle de conception MVP (Modèle-Vue-Présentation).

Méthodes ajoutées :

**lecteurvue(QWidget *parent = nullptr)** : Constructeur de la vue du lecteur

**void majInterface(QString, std::string, std::string, int)** : Met à jour le chemin, le titre, la catégorie et le rang du diaporama courant

**void setPresentation(LecteurPresentation*)** : Définit la présentation du lecteur

**LecteurPresentation* getPresentation()** : Pointeur vers la présentation du lecteur