SAÉ 2.01 - Lecteur de diaporama

Développement d'une application

DESESSARD Estéban
MARSAN Louis
CHA Baptiste

TD III / TP 5 / Groupe 3

GitHub : CarteSD/s201 (github.com)

Table des matières

Table des figures

1. Version 1

1.1. Description de la version

- La v1 de l'application permet simplement d'afficher des images créées "en dur" dans un terminal.
- Seul le mode manuel est utilisable et le mode automatique n'est pas proposé.

1.2. Diagramme de classes UML

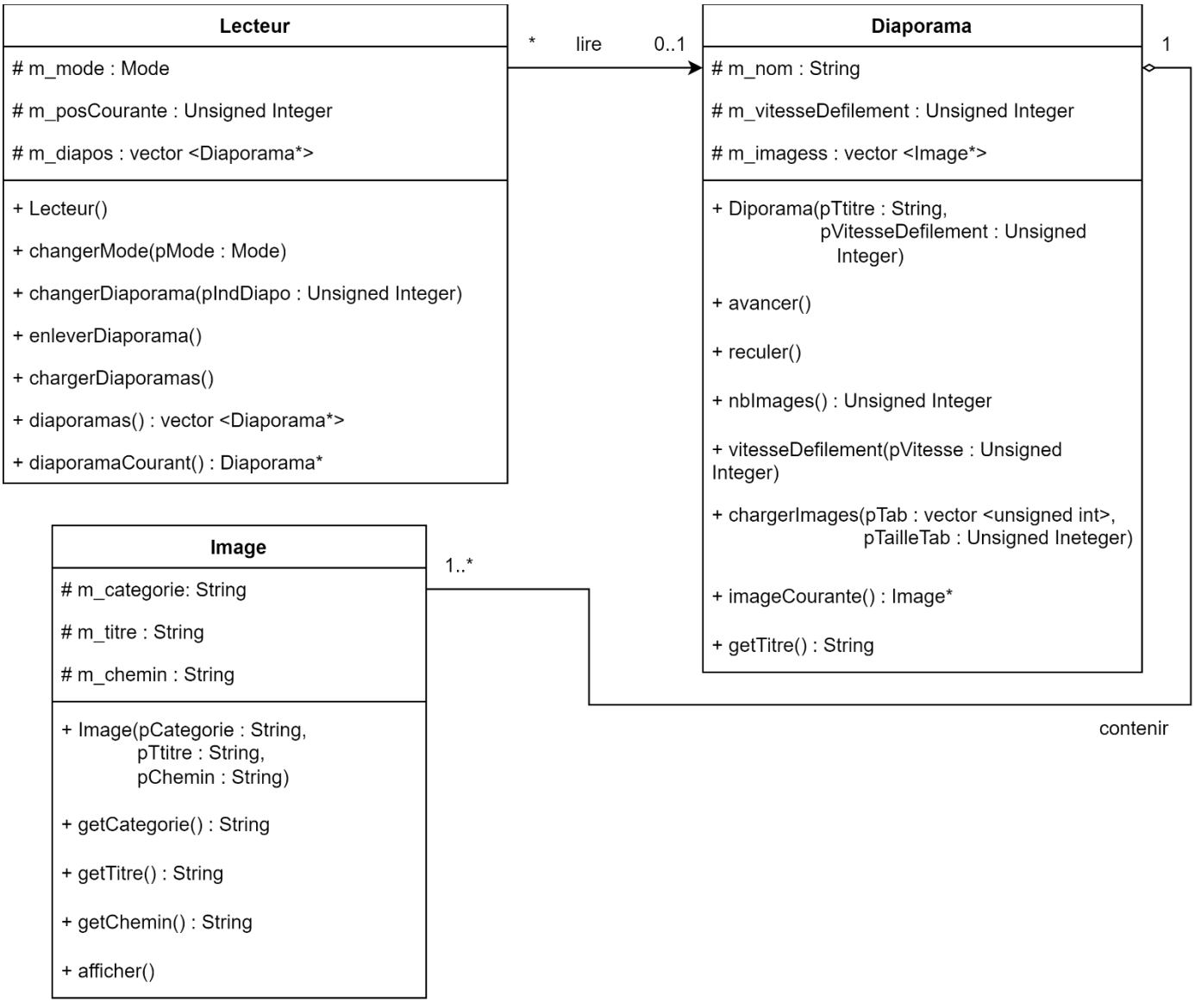

Figure 1- Diagramme de classes UML de la v1

1.3. Détail des attributs et méthodes

1.3.1. Classe Lecteur

Attributs :

`Mode m_mode`
Signification : mode de lecteur du diaporama (automatique / manuel)

`unsigned int m_posCourante`
Signification : position courante du lecteur, c'est à dire l'index du diaporama que nous exploitons dans m_diapos

`vector <Diaporama*> m_diapos`
Signification : vecteur de pointeurs vers les diaporamas présents dans le lecteur

Méthodes :

`Lecteur()`
But : constructeur de l'objet Lecteur

setMode(Mode pMode)
But : changer le mode de défilement du lecteur

`changerDiaporama(unsigned int pIndDiapo)`
But : changer de diaporama. Le paramètre pIndDiapo correspond à l'indice du diaporama dans le vecteur m_dipoas

`enleverDiaporama()`
But : enlever le diaporama charger, c'est à dire rebasculer sur le diaporama par défaut

`chargerDiaporamas()`
But : charger tous les diaporamas dans le lecteur. Chaque diaporama sera ajouter au vecteur m_diapos

`diaporamas() : vector <Diaporama*>`
But : retourne la donnée membre m_diapos, le vecteur de pointeurs vers les diaporamas

`diaporamaCourant() : Diaporama*`
But : retourne l'adresse du diaporama courant du lecteur
`~Lecteur()`
`But : destructeur de la classe Lecteur`

```
Mode getMode();
But : retourne la donnée membre m_mode, le mode de lecteur
actuel
```

1.3.2. Classe Diaporama

Attributs :

```
vector <Image*> m_images
```
Signification : vecteur de pointeurs vers les images présentes dans le diaporama

```
string m_nom
```
Signification : nom du diaporama

```
unsigned int m_vitesseDefilement
```
Signification : vitesse de défilement du diaporama lorsque le lecteur est en mode automatique

```
unsigned int m_posCourante
```
Signification : position courante du diaporama, c'est à dire l'index de l'image que nous affichons dans m_images

Méthodes :

```
Diaporama(string pTitre = "Sans titre", unsigned int
pVitesseDefilement = 2)
```
But : constructeur par défaut

```
Avancer()
```
But : avancer au sein du vecteur d'images m_images

```
Reculer()
```
But : reculer au sein du vecteur d'images m_images

```
nbImages() : Unsigned integer
```
But : retourne le nombre d'images, correspondant à la taille du vecteur m_images

```
vitesseDefilement(unsigned int pVitesse = 2)
```
But : modifie la vitesse de défilement lorsque le lecteur est en mode automatique. Par défaut, si aucun paramètre n'est passé, la vitesse de défilement sera établie à 2 secondes.

chargerImages(vector <unsigned int> pTab, unsigned int pTailleTab)
But : charge les images d'un diaporama grâce au tableau pTab passé en paramètre. Ce tableau pTab contient les indices des images que doit afficher le diaporama, trié par ordre d'affichage

imageCourante() : Image*
But : retourne l'adresse de l'image courante du diaporama

getTitre() : string
But : retourne m_nom, le titre du diaporama

getPosCourante() : unsigned int
But : retourne m_posCourante, la position courante du diaporama

setPosCourante(unsigned int pNum)
But : retourne m_posCourante, la position courante du diaporama

1.3.3. Classe Image

Attributs :

string m_categorie
Signification : catégorie de l'image

string m_titre
Signification : titre de l'image

string m_chemin
Signification : chemin d'accès de l'image

Méthodes :

```
Image(string pCategorie = "",
                string pTitre = "sans-titre",
                string pChemin = "");
```
But : constructeur de l'objet Image

getCategorie() : string
But : retourne m_categorie, la catégorie de l'image

getTitre() string
But : retourne m_titre, le titre de l'image

`getChemin() : string`
But : retourne m_chemin, le chemin d'accès de l'image

`afficher()`
But : afficher l'image, c'est à dire sa donnée membre m_titre

1.3.4. Complément

En complément voici l'implémentation du type Mode, trouvable dans le code

```
enum Mode {automatique, manuel} ;
/*    Le Mode correspond au mode de lecture du lecteur de diaporama.
      automatique : grâce à la donnée membre m_vitesseDefilement de l'objet Diaporama, le lecteur basculera
      automatiquement à l'image suivante du diaporama toutes les [m_vitesseDefilement] secondes
      manuel : si le lecteur est en mode manuel, c'est à l'utilisateur de saisir l'action qu'il souhaite faire
      afin de que l'image suivante / précédente s'affiche.
*/
```

1.3 Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v1_classes

2. Version 2

2.1. Description de l'application

- La v2 de l'application est la réalisation de l'interface graphique de notre lecteur de diaporamas.
- Seul les connexions entres les éléments graphiques sont effectuées mais aucune image ne s'affiche.

2.2. Diagramme états-transitions de l'application

2.2.1. Forme classique UML

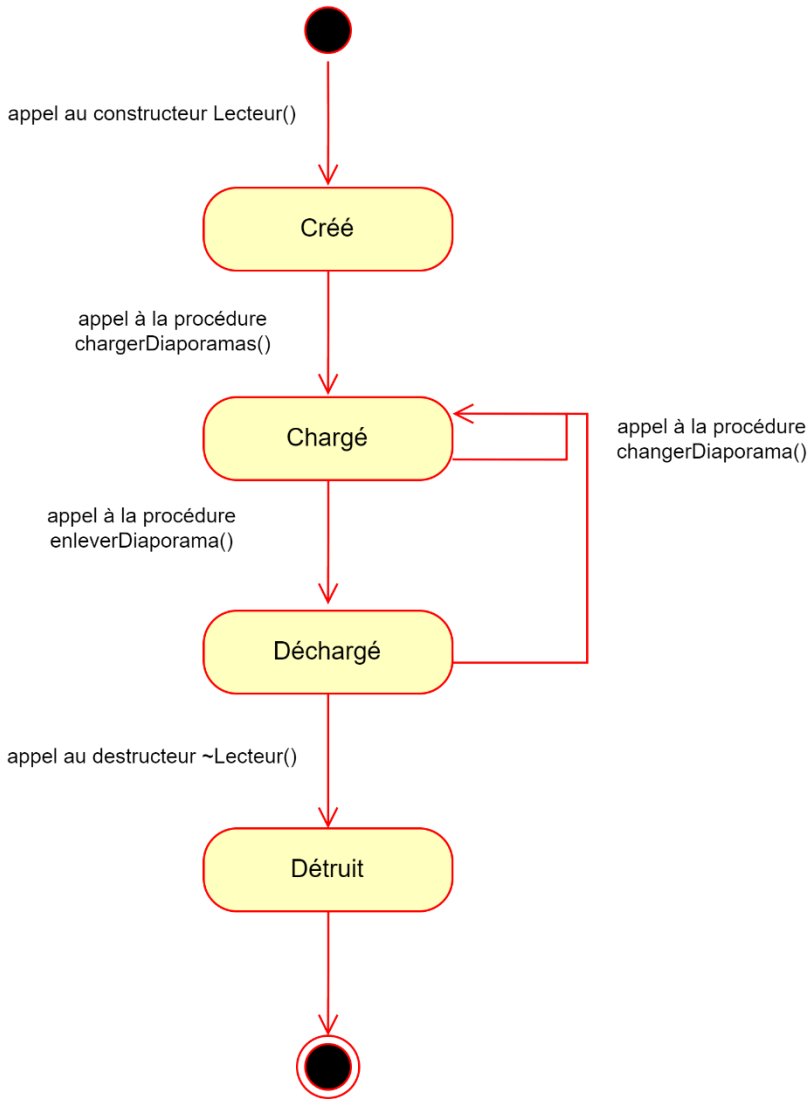

Figure 2 - Diagramme états-transitions UML

2.2.2. Forme matricielle

Évènements / États	**Appel au constructeur Lecteur()**	**Appel à la procédure chargerDiaporamas()**	**Appel à la procédure changerDiaporama()**	**Appel à la procédure enleverDiaporama()**	**Appel au destructeur ~Lecteur()**
Créé	-	Chargé	-	-	-
Chargé	-	-	Chargé	Déchargé	-
Déchargé	-	-	Chargé	-	Détruit
Détruit	-	-	-	-	-

2.3. Documentation du lien entre les éléments l'interface et les fonctionnalités

La documentation du lien entre l'interface et les fonctionnalités consiste à relever les évènements déclenchés par chaque élément graphique de la fenêtre.

Pour cela nous reprenons le diagramme états-transitions sous la forme matricielle et lui ajoutons une ligne référençant les éléments graphiques :

Élément graphique	*Lancement de l'application*	*Lancement de l'application*	actionChangerDiapo	actionEnleverDiapo	*Fermeture de l'application*
Évènements / États	**Appel au constructeur Lecteur()**	**Appel à la procédure chargerDiaporamas()**	**Appel à la procédure changerDiaporama()**	**Appel à la procédure enleverDiaporama()**	**Appel au destructeur ~Lecteur()**
Créé	-	Chargé	-	-	-
Chargé	-	-	Chargé	Déchargé	-
Déchargé	-	-	Chargé	-	Détruit
Détruit	-	-	-	-	-

2.4. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v2_QTGraphicLecteurDiapo

3. Version 2 MVP

3.1. Description de la version

- La v2 MVP de l'application présente le même fonctionnement que la v2 simple, mais en respectant le patron MVP étudié en cours.
- Certains signaux et slots continuent à fonctionner avec l'usage des qDebug() et ne seront coder que dans la v3

3.2. Diagramme de classes UML

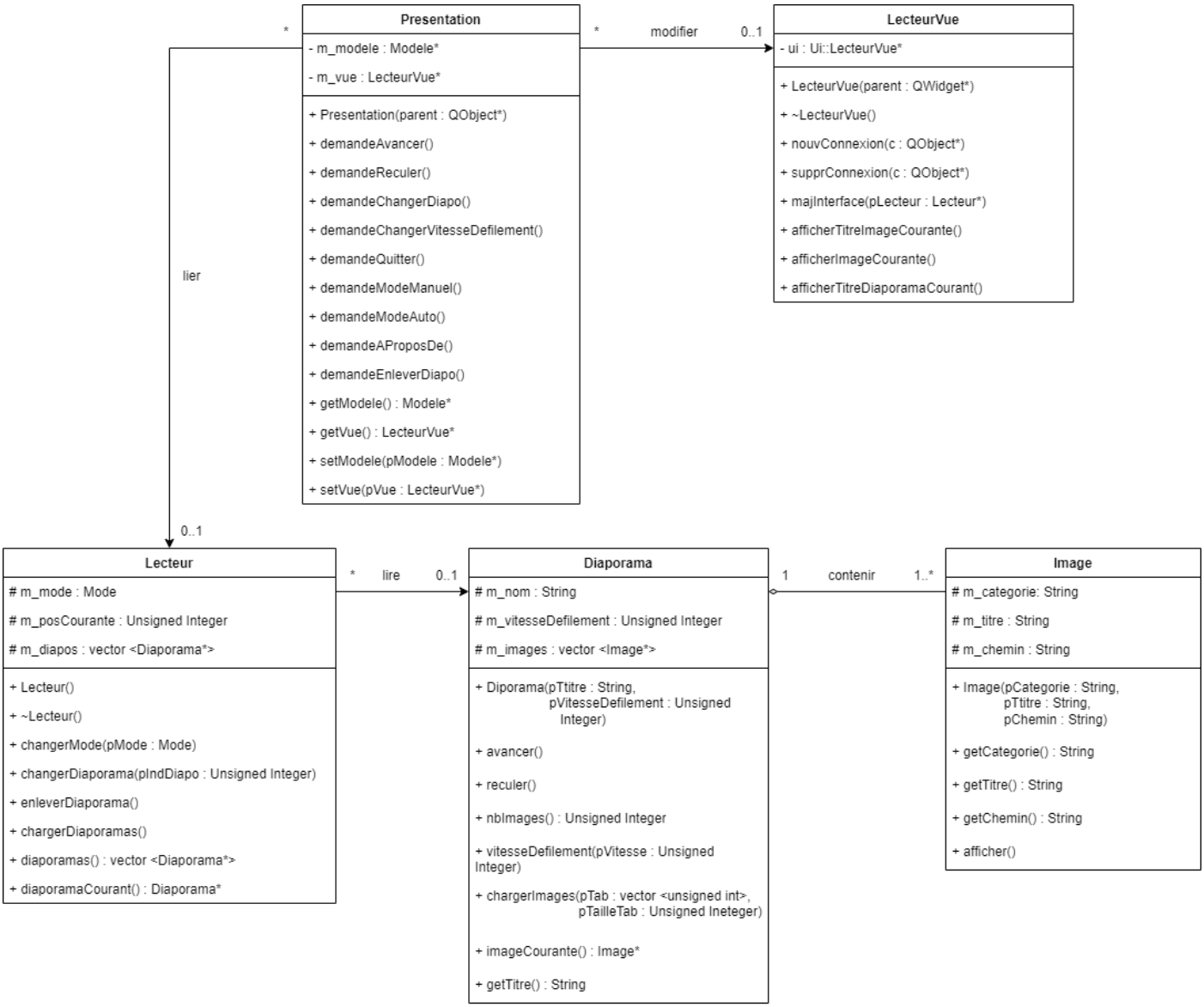

Figure 3 - Diagramme de classes UML de la v2 MVP

3.3. Détail des attributs et méthodes

Nous ne jugeons pas utile de remettre toute la documentation des objets présents dans la version 1 du projet, à savoir Image, Diaporama et Lecteur.

Cependant, voici la documentation des deux objets ajoutés en suivant le patron MVP :

3.3.1. Classe Presentation

Attributs :

`LecteurVue *m_vue`
Signification : Vue du modèle MVP du projet

`Lecteur *m_lecteur`
Signification : Lecteur du moèdle MVP du projet (correspondant au modèle)

Méthodes :

`explicit Presentation(QObject *parent = nullptr)`
But : `Constructeur de l'objet Presentation`

GETTERS & SETTERS

Les setters permettent d'attribuer l'objet passé en paramètre à la donnée membre de l'objet Presentation

Les getters permettent de retourner l'objet membre souhaité de l'objet Presentation

```
Lecteur* getLecteur()
LecteurVue* getVue()
void setLecteur(Lecteur*)
void setVue(LecteurVue*)
```

`demandeAvancer()`
But : Permet d'avancer dans le diaporama

`demandeReculer()`
But : Permet de reculer dans le diaporama

`demandeChangerDiapo()`
But : Permet de changer de diaporama

`demandeChangerVitesseDefilement()`
But : Permet de changer la vitesse de défilement

`demandeQuitter()`
But : Permet de quitter l'application

`demandeModeManuel()`
But : Permet de passer en mode manuel

`demandeModeAuto()`
But : Permet de passer en mode automatique

`demandeAProposDe()`
But : Permet d'accéder à la page "A propos de ..."

`demandeEnleverDiapo()`
But : Permet d'enlever un diaporama

3.3.2. Classe LecteurVue

Attributs :

`Ui::LecteurVue *ui`
Signification : Pointeur vers l'interface graphique du lecteur

Méthodes :

`LecteurVue(QWidget *parent = nullptr)`
But : Constructeur de l'objet LecteurVue

`~LecteurVue()`
But : Destructeur de l'objet LecteurVue

`nouvConnexion(QObject*)`
But : Connecte tous les éléments de l'interface graphique avec les slots de l'objet passé en paramètre

`supprConnexion(QObject*)`
But : Déconnecte tous les éléments de l'interface graphique des slots de l'objet passé en paramètre

`majInterface(Lecteur*)`
But : Met à jour l'interface graphique en fonction du lecteur passé en paramètre

`afficherTitreImageCourante(Lecteur*)`
But : Affiche le titre de l'image courante

`afficherImageCourante(Lecteur*)`
But : Affiche l'image courante (uniquement pour la v3)

`afficherTitreDiaporamaCourant(Lecteur*)`
But : Affiche le titre du diaporama courant

3.4. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v2_MVP_QTGraphicLecteurDiapo

4. Version 3 MVP

4.1. Description de la version

- La v3 MVP de l'application permet un affichage des images dans l'interface graphique.
- De plus, quelques fonctionnalités supplémentaires telles que un menu Fichier >> Quitter ou Aide >> A propos de...

4.2. Diagramme de classes UML

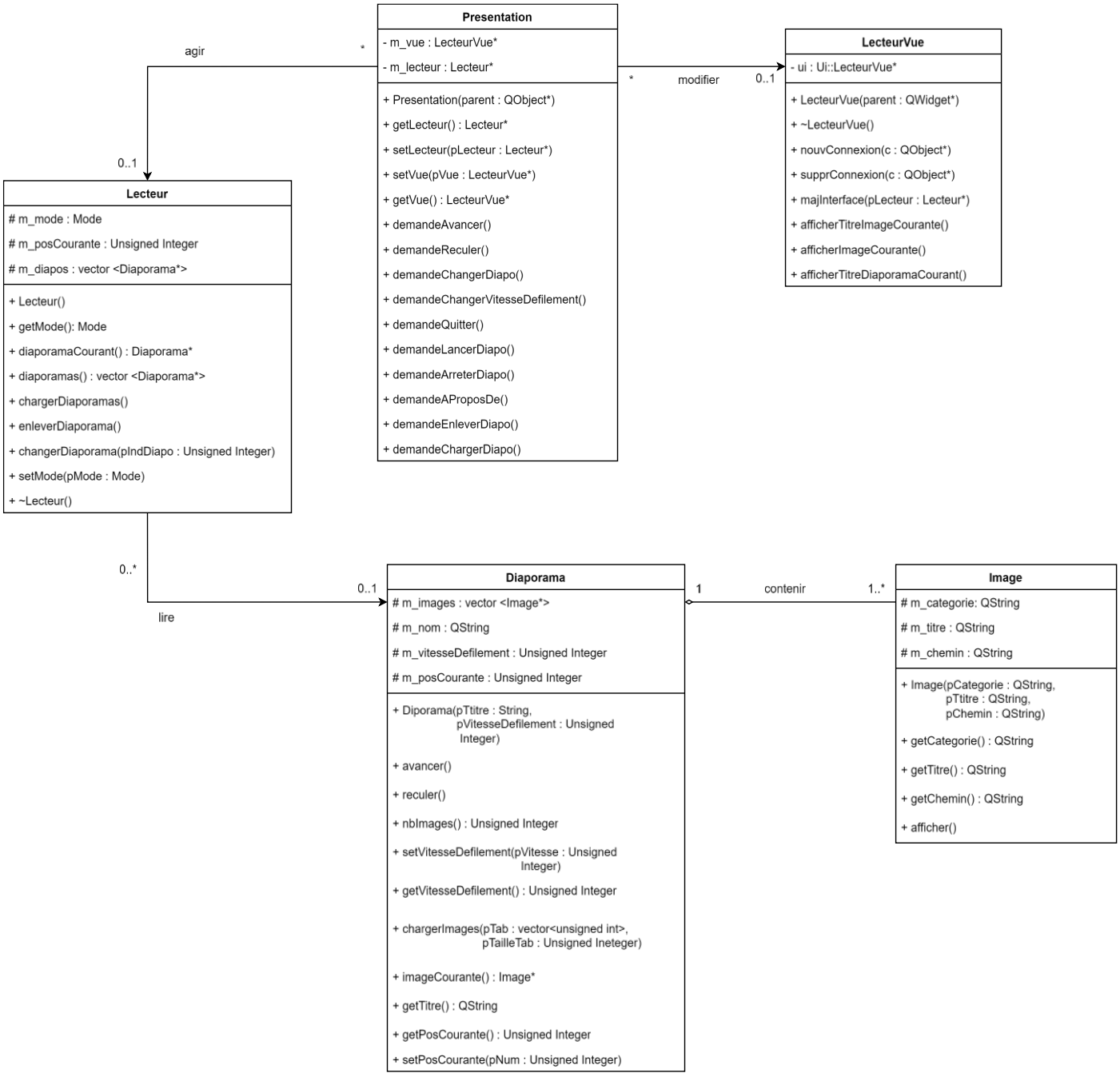

Figure 4 - Diagrame de classes UML de la v3 MVP

4.3. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v3MVP_QTGraphicLecteurDiapo

5. Version 4 MVP

5.1. Description de la version

- Cette nouvelle version permet la prise en charge du mode automatique pour la lecture des diaporamas.
- Pour se faire, des nouvelles procédures ainsi que données membres dans nos classes ont du être ajoutées.

5.2. Diagramme de classes UML

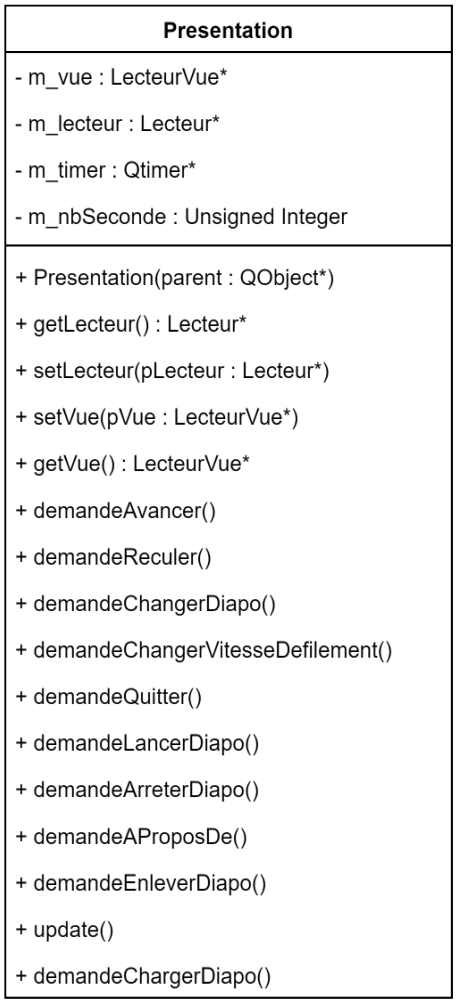

Figure 5 - Extrait du diagramme de classes UML de la v4 MVP

Nous pouvons voir sur l'extrait du diagramme de classes ci-contre que deux nouvelles données membres ont été ajoutées à notre classe Presentation :

- m_timer : QTimer*
- m_nbSeconde : Unsigned Integer

Ces données members ajoutées permettent la mise en œuvre du défilement automatique des images

Une nouvelle procédure à également été ajoutée :

- update()

Cette procédure est exécutée grâce à la donnée membre m_timer, car elle représente un slot.

5.3. Détail des attributs et méthodes

Attributs :

`QTimer *m_timer`

Signification : Timer permettant le mode automatique

`unsigned int *m_nbSeconde`

Signification : Nombre de seconds de defilement du mode automatique

Méthodes :

`update()`

But : Permet de mettre à jour le mode automatique

5.4. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v4MVP_QTGraphicLecteurDiapo

6. Version 5 MVP

6.1. Description de la version

- La v5 de l'application implémente de nouvelles fonctionnalités telles que la modification de la vitesse de défilement des images en mode automatique, le chargement ou le déchargement des diaporamas.

6.2. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v5MVP_QTGraphicLecteurDiapo

7. Version 6 MVP

7.1. Description de la version

- La v6 de l'application en charge plus les images « en dur », mais à partir de la base de données fournie.
- Modification du diagramme de classes UML avec l'ajout d'une nouvelle classe ainsi qu'une autre modification.

7.2. Diagramme de classes UML

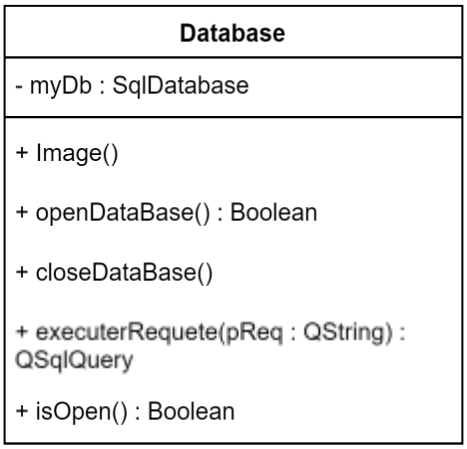

Figure 6 - Extrait du diagramme de classes UML de la v6 MVP

Cette nouvelle classe nous permet de créer un objet représentant une base de données.

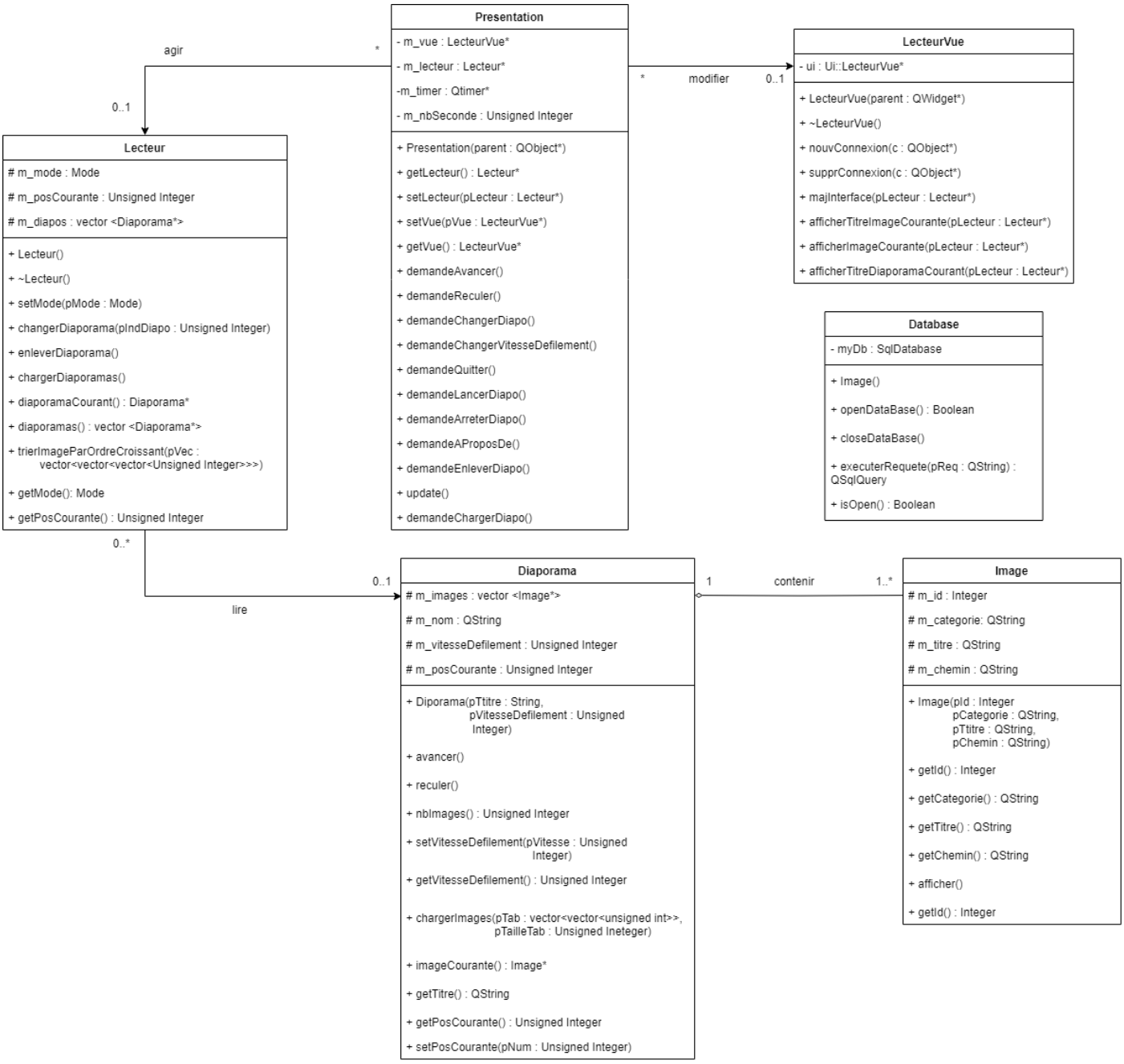

Figure 7 - Diagramme de classes UML de la v6 MVP

7.3. Détail des attributs et méthodes

7.3.1. Classe Database

Attributs :

`QSqlDatabase myDb`
Signification : base de données de l'objet

Méthodes :

`Database()`
But : constructeur de l'objet Database

`bool openDataBase()`
But : permet d'ouvrir la base de données.
La fonction renvoie true si l'ouverture s'est déroulée avec succès.
La fonction renvoie false si l'ouverture a échoué.

`closeDataBase()`
But : ferme la base de données

`bool isOpen()`
But : analyse l'état de la base de données.
La fonction renvoie true si la base de données est ouverte.
La fonction renvoie false si la base de données est fermée.

QSqlQuery executerRequete(QString pReq)
But : exécute la requête pReq passée en paramètre et retourne les résultats sous forme de QSqlQuery

7.3.2. Classe Image

Attributs :

`int m_id`
Signification : id de l'image

Méthodes :

`unsigned int getId()`
But : retourne l'ID de l'image

7.4. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v6MVP_QTGraphicLecteurDiapo

8. Version 7 MVP

8.1. Description de la version

-La v7 de l'application modifie le chargement des diaporamas en les chargeant depuis la base de données.

-Modification du diagramme de classe UML avec l'ajout de m_id représentant l'identifiant du diaporama.

8.2. Diagramme de classe UML

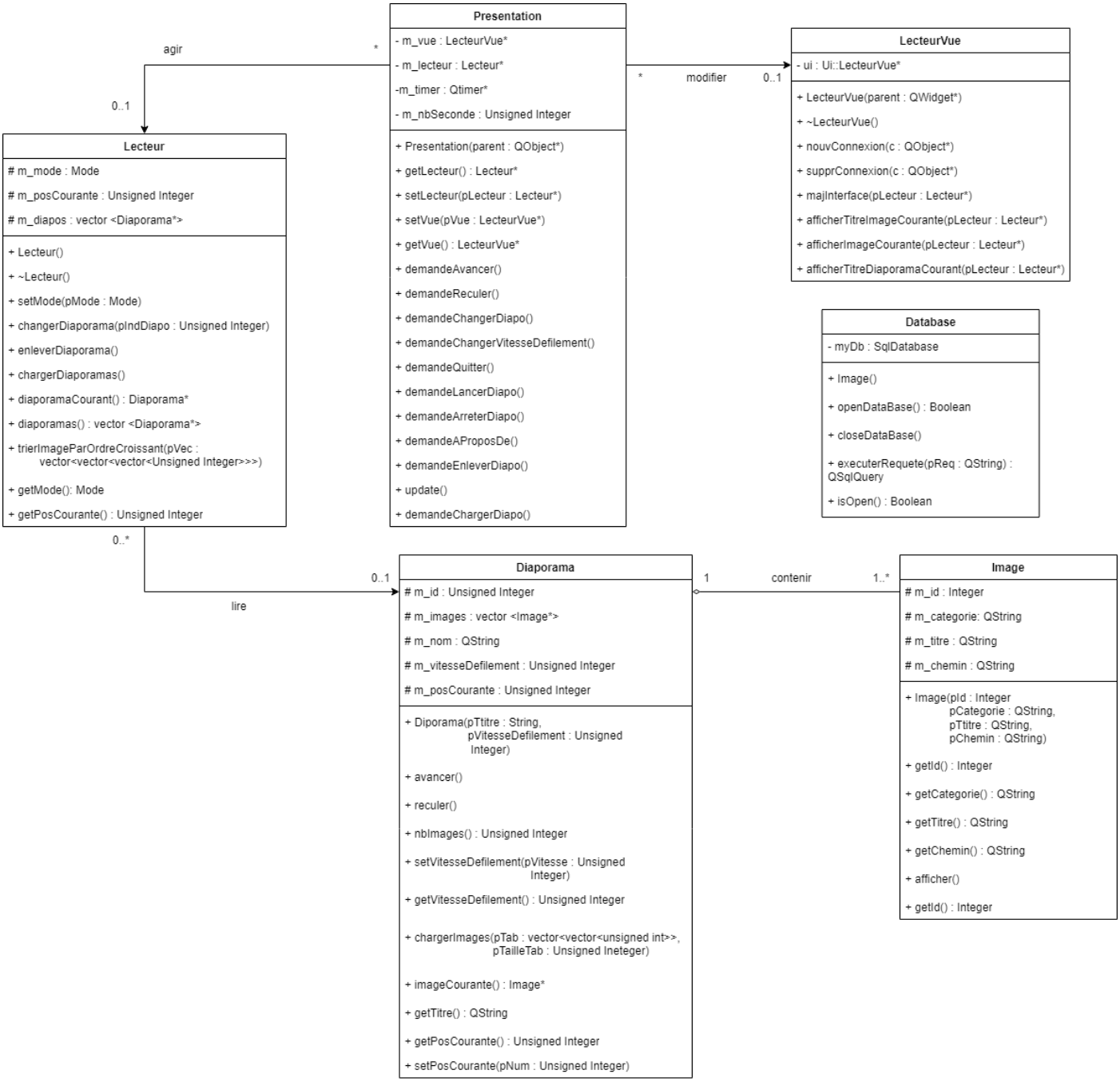

Figure 8 - Diagramme de classes UML de la v7 MVP

8.3. Lien vers le code

GitHub : https://github.com/CarteSD/s201/tree/main/v7MVP_QTGraphicLecteurDiapo

9. Version V8 MVP

9.1. Description de la version

-La v8 a pour but de permettre de modifier la valeur de « vitesse de défilement » directement dans la base de données.

-La base de données modifiée est une copie réalisée à partir de la base de données

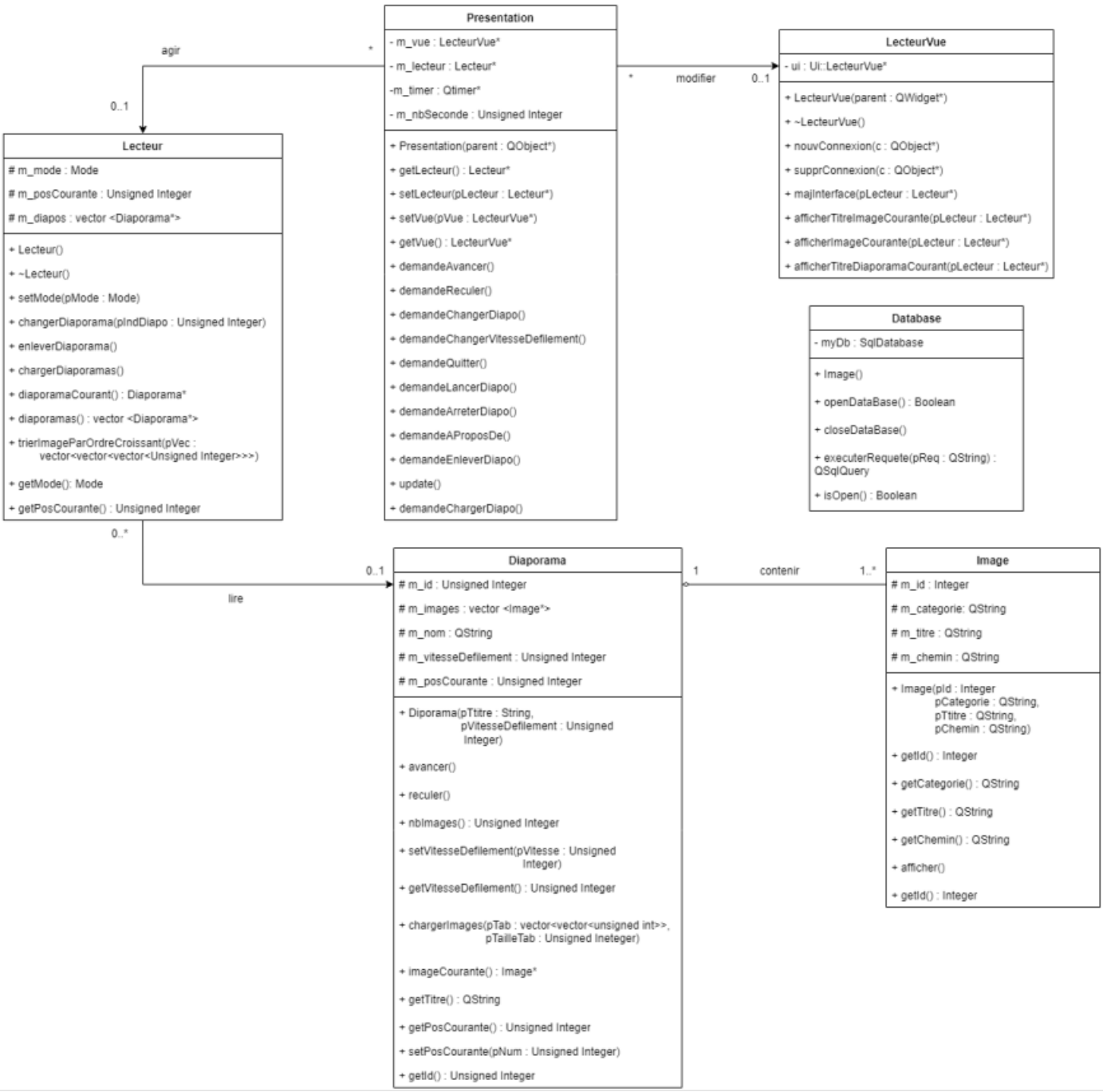

Figure 9 - Diagramme de classes UML de la v8 MVP

10. Version finale

10.1. Description de l'application

La version finale de l'application recense toutes les fonctionnalités énoncées précédemment :
- Affichage d'une interface graphique avec l'image au centre
- Le titre, la catégorie de l'image sont affichées sous celle-ci
- Le titre du diaporama est affichée en haut à gauche de l'écran
- Le numéro de diapositive (image courante) est affichée en bas à gauche de l'écran
- En bas de l'affichage graphique, une série de bouton permet de prendre le contrôle sur le lecteur en naviguant à travers les images.
- Deux bouton permettent d'activer ou désactiver le mode automatique.
- Le menu Fichier >> Paramétrer >> Charger diaporama permet de naviguer à travers les différents diaporamas du lecteur.
- Le menu Fichier >> Paramétrer >> Enlever diaporama permet de décharger le lecteur.
- Le menu Fichier >> Paramétrer >> Vitesse de défilement permet de modifier la vitesse de défilement du mode automatique.
- Les images sont chargées depuis la base de données.
- Les diaporamas sont chargées depuis la base de données.

10.2. Diagramme de classes UML

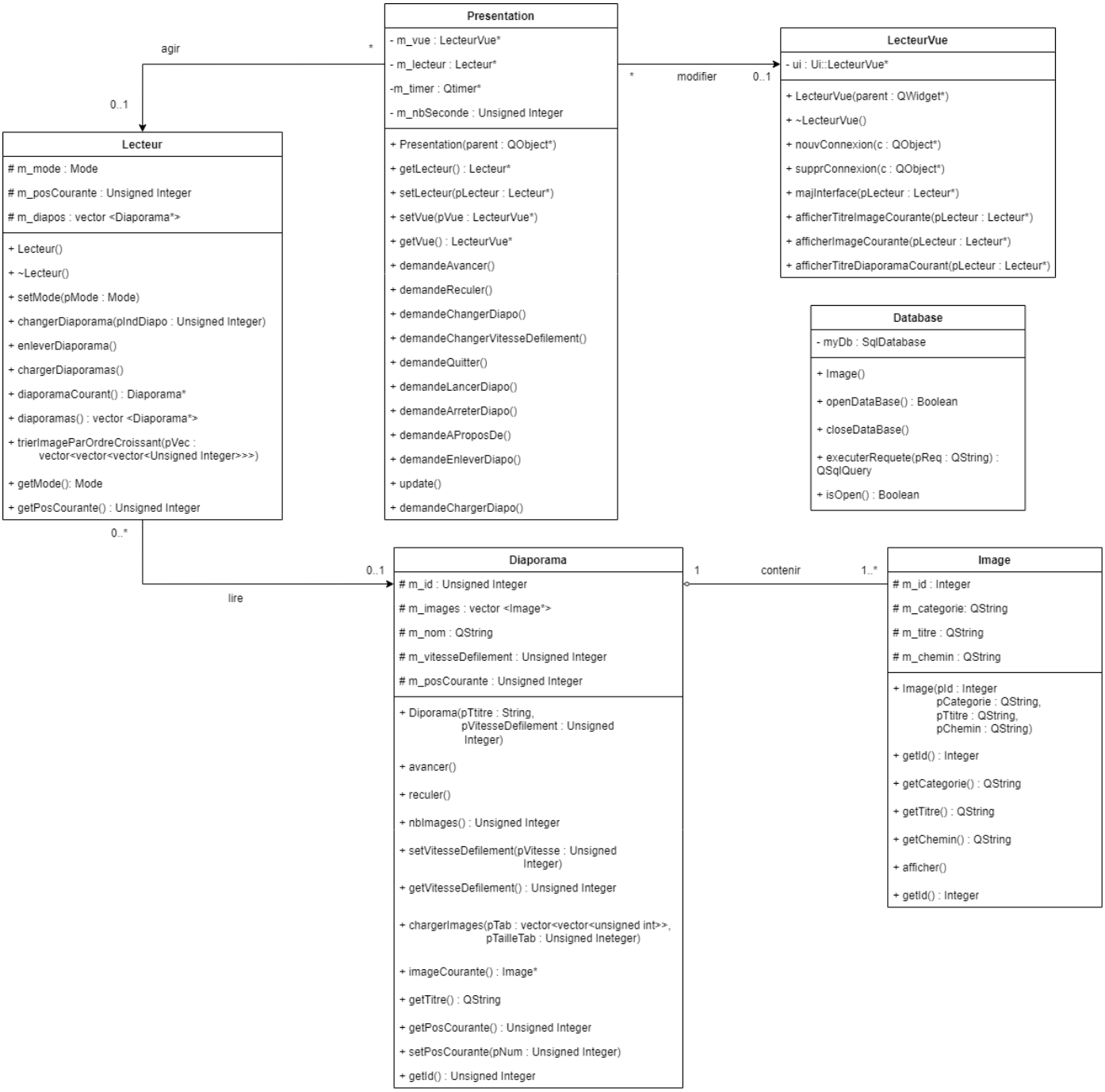

Figure 10 - Diagramme de classes UML de la v8 MVP

10.3. Détail des attributs et méthodes

10.3.1. Classe Database

10.3.1.1. Attributs

`QSqlDatabase myDb`
Signification : base de données de la classe Database

10.3.1.2. Méthodes

`Database()`
But : constructeur de l'objet Database

`bool openDataBase()`
But : permet d'ouvrir la base de données.
La fonction renvoie true si l'ouverture s'est déroulée avec succès.
La fonction renvoie false si l'ouverture à échoué.

`void closeDataBase()`
But : ferme la base de données.

`QSqlQuery executerRequete(QString pReq)`
But : exécute la requête pReq passée en paramètre et retourne les résulats sous forme de QSqlQuery.

`bool isOpen() const`
But : analyse l'état de la base de données.
La fonction renvoie true si la base de données est ouvrte.
La fonction renvoie false si la base de données est fermée.

10.3.2. Classe Diaporama

10.3.2.1. Attributs

```
unsigned int m_id
```
Signification : identifiant du diaporama.

```
vector <Image*> m_images
```
Signification : vecteur de pointeurs vers les images présentes dans le diaporama

```
QString m_nom
```
Signification : nom du diaporama

```
unsigned int m_vitesseDefilement
```
Signification : vitesse de défilement du diaporama lorsque le lecteur est en mode automatique

```
unsigned int m_posCourante
```
Signification : position courante du diaporama, c'est à dire l'index de l'image que nous affichons dans m_images

10.3.2.2. Méthodes

```
Diaporama(unsigned int pId, QString pTitre = "Sans titre",
          unsigned int pVitesseDefilement = 2)
```
But : constructeur par défaut.

```
void avancer()
```
But : avancer au sein du vecteur d'images m_images.

```
void reculer()
```
But : reculer au sein du vecteur d'images m_images.

```
unsigned int nbImages() const
```
But : retourne le nombre d'images, correspondant à la taille du vecteur m_images.

```
void setVitesseDefilement(unsigned int pVitesse = 2)
```
But : modifier la vitesse de défilement lorsque le lecteur est en mode automatique. Par défaut, si aucun paramètre n'est passé, la vitesse de défilement sera établie à 2 secondes.

```
unsigned int getVitesseDefilement() const
```
But : retourne m_vitesseDefilement, correspondant à la vitesse de défilement du diaporama lorsqu'il est en mode automatique.

```
void chargerImages(vector<vector<unsigned int>> pTab,
                   unsigned int pTailleTab, vector <Image*>
                   pAllImages)
```
But : charge les images d'un diaporama grâce au tableau pTab passé en paramètre. Ce tableau pTab contient les indices des images que doit afficher le diaporama, trié par ordre d'affichage.

```
Image* imageCourante() const
```
But : retourne l'adresse de l'image courante du diaporama.

```
QString getTitre() const
```
But : retourne m_nom, le titre du diaporama.

```
unsigned int getPosCourante() const
```
But : retourne m_posCourante, la position du curseur dans le lecteur.

```
void setPosCourante(unsigned int)
```
But : assigne le paramètre pLecteur à la donnée membre m_posCourante.

```
unsigned int getId() const
```
But : retourne m_id, l'id du diaporama.

10.3.3. Classe Image

10.3.3.1. Attributs

`int m_id`
Singnification : id de l'image

`QString m_categorie`
Signification : catégorie de l'image

`QString m_titre`
Signification : titre de l'image

`QString m_chemin`
Signification : chemin d'accès de l'image

10.3.3.2. Méthodes

```
Image(int pId,
      QString pCategorie = "",
      QString pTitre = "sans-titre",
      QString pChemin = "")
```
But : constructeur de l'objet Image

`unsigned int getId() const`
But : retourne m_id, l'ID de l'image

`QString getCategorie() const`
But : retourne m_categorie, la catégorie de l'image

`QString getTitre() const`
But : retourne m_titre, le titre de l'image

`QString getChemin() const`
But : retourne m_chemin, le chemin d'accès de l'image

`void afficher() const`
But : afficher l'image, c'est à dire sa donnée membre m_titre

10.3.4. Classe Lecteur

10.3.4.1. Attributs

`Mode m_mode`
Signification : mode de lecture du diaporama

`unsigned int m_posCourante`
Signification : position courante du lecteur, c'est-à-dire l'index du diaporama que nous exploitons dans m_diapos

`vector <Diaporama*> m_diapos`
Signification : vecteur de pointeurs vers les diaporamas présents dans le lecteur

10.3.4.2. Méthodes

`Lecteur()`
But : constructeur de l'objet Lecteur

`void setMode(Mode pMode)`
But : changer le mode de défilement du lecteur

`void changerDiaporama(unsigned int pIndDiapo)`
But : changer de Diaporama
Le paramètre pIndDiapo correspond à l'indice du diaporama dans le vecteur m_dipoas

`void enleverDiaporama()`
But : enlever le diaporama chargé. Nous avons décidé de rebasculer sur le diaporama par défaut

`void chargerDiaporamas()`
But : charger tous les diaporamas dans le lecteur.
Chaque diaporama sera ajouter au vecteur m_diapos

`vector <Diaporama*> diaporamas() const`
But : retourne la donnée membre m_diapos, le vecteur de pointeurs vers les diaporamas

`Diaporama* diaporamaCourant() const`
But : retourne l'adresse du diaporama courant du lecteur

`~Lecteur()`
But : destructeur de la classe Lecteur

`Mode getMode()`
But : retourne la donnée membre m_mode, le mode de lecteur actuel

`void trierImageParOrdreCroissant(vector<vector<vector<unsigned int>>> &pVec)`
But : ordonne les images dans l'ordre de rang en se basant sur le vecteur passé en paramètre

`unsigned int getPosCourante()`
But : retourne m_posCourante

10.3.5. Classe LecteurVue

10.3.5.1. Attributs

`Ui::LecteurVue *ui`
Signification : interface utilisateur de la fenêtre

10.3.5.2. Méthodes

`LecteurVue()`
But : retourne m_posCourante

`~LecteurVue()`
But : Destructeur de l'objet LecteurVue

`void nouvConnexion(QObject*)`
But : Connecte tous les éléments de l'interface graphique avec les slots de l'objet passé en paramètre

`void supprConnexion(QObject*)`
But : Déconnecte tous les éléments de l'interface graphique des slots de l'objet passé en paramètre

`void majInterface(Lecteur*)`
But : Met à jour l'interface graphique en fonction du lecteur passé en paramètre

`void afficherTitreImageCourante(Lecteur*)`
But : Affiche le titre de l'image courante

`void afficherImageCourante(Lecteur*)`
But : Affiche l'image courante

`void afficherTitreDiaporamaCourant(Lecteur*)`
But : Affiche le titre du diaporama courant

`void desactiverControles()`
But : Désactive tous les contrôles du diaporama (avancer, reculer, mode automatique)

`void activerControles()`
But : Active les contrôles du diaorama (avancer, reculer, mode automatique)

10.3.6. Classe Présentation

10.3.6.1. Attributs

`LecteurVue *m_vue`
Signification : Vue du modèle MVP du projet

`Lecteur *m_lecteur`
Signification : Lecteur du moèdle MVP du projet (correspondant au modèle)

`QTimer *m_timer`
Signification : Timer permettant le mode automatique

`unsigned int m_nbSecondes = 0`
Signification : Nombre de secondes de défilement du mode automatique

10.3.6.2. Méthodes

`explicit Presentation(QObject *parent = nullptr)`
But : Constructeur de l'objet Présentation

`Lecteur* getLecteur()`
But : retourne m_lecteur

`LecteurVue* getVue()`
But : retourne m_vue

`void setLecteur(Lecteur*)`
But : affecte la valeur passée en paramètre dans m_lecteur

`void setVue(LecteurVue*)`
But : affecte la valeur passée en paramètre dans m_vue

`void demandeAvancer`
But :Permet d'avancer dans le diaporama

`void demandeReculer()`
But : Permet de reculer dans le diaporama

`void demandeChargerDiapo()`
But : Permet de changer de diaporama

`void demandeChangerVitesseDefilement()`
But : Permet de changer la vitesse de défilement

`void demandeQuitter()`
But : Permet de quitter l'application

`void demandeLancerDiapo()`
But : Permet de lancer le diaporama, et de le passer en mode automatique

`But : void demandeArreterDiapo()`
Permet d'arrêter le mode automatique du diaporama

`void demandeAProposDe()`
But : Permet d'accéder à la page "A propos de ..."

`void demandeEnleverDiapo()`
But : Permet d'enlever un diaporama

`void update()`
But : Permet de mettre à jour le mode automatique

`void demandeChargerDiapos()`
But : Permet de charger tous les diaporamas et affiche le premier