Développement d'une application

S2.01

TD3– TP5

Hamchouche Kamelia

Mahssini Imane

Lohier Marylou

2023-2024 / BUT informatique 1re année, semestre 2

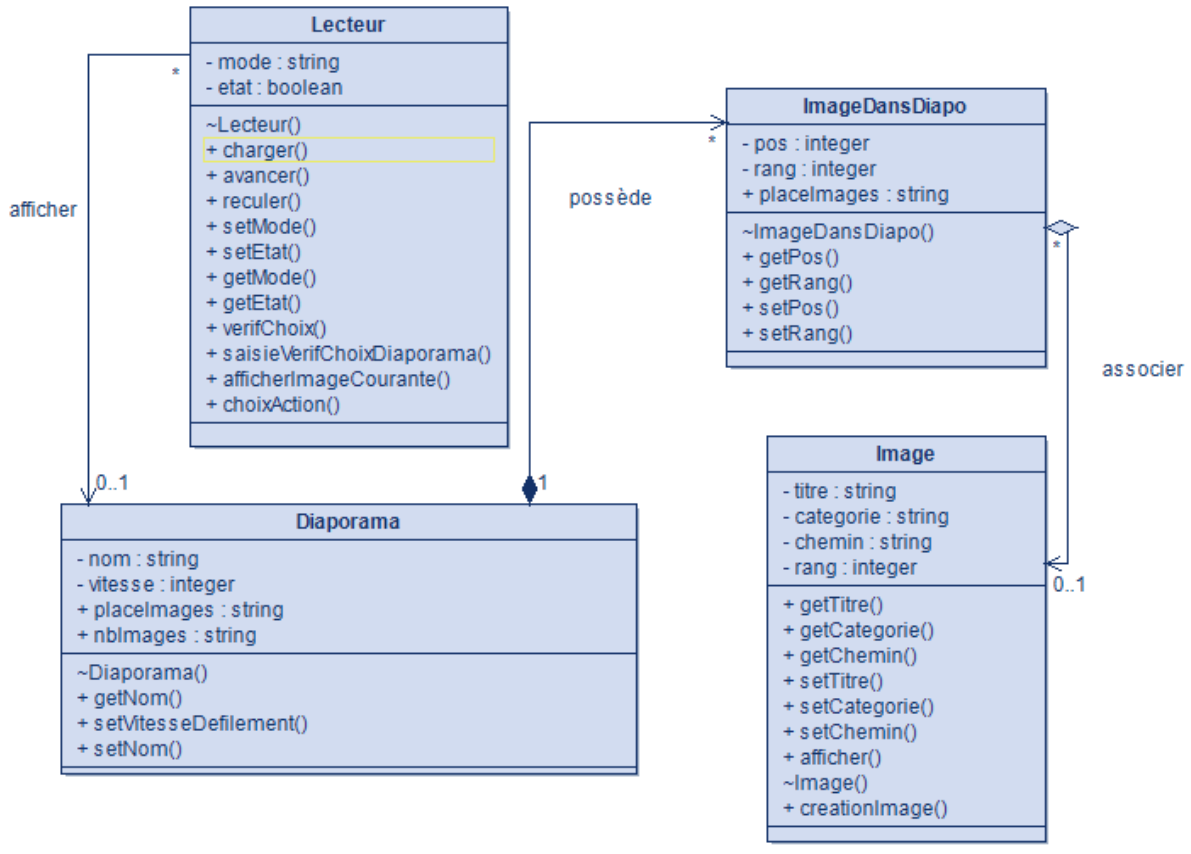
Lien Github : https://github.com/marylou-lhr/S2-01-2024

Table des matières :

v1

Objectif : mise en place du code

Diagramme de classe :



Dictionnaire des éléments

Diaporama

Attributs :

1. `_nom` : Nom du diaporama.
2. `_vitesse` : Vitesse de défilement du diaporama.
3. `_placeImages` : Vecteur contenant les images du diaporama.

Méthodes :

1. `Diaporama()` : Constructeur de la classe `Diaporama`.
2. `~Diaporama()` : Destructeur de la classe `Diaporama`.
3. `string getNom() const` : Renvoie le nom du diaporama.
4. `int getVitesse() const` : Renvoie la vitesse de défilement du diaporama.
5. `void setVitesseDefilement(int pVitesse)` : Définit la vitesse de défilement du diaporama.
6. `void setNom(string pNom)` : Définit le nom du diaporama.
7. `unsigned int nbImages() const` : Renvoie le nombre d'images du diaporama.

ImageDansDiapo

Voici une liste des attributs et des méthodes de la classe `ImageDansDiapo` :

Attributs :

1. `_pos` : Rang de l'image dans le tableau d'images (vector<Images>), correspondant à l'ordre de chargement initial des images dans la table des images.
2. `_rang` : Rang de l'image dans le diaporama, correspondant à l'ordre d'affichage choisi par l'utilisateur lors de la création du diaporama.

Méthodes :

1. `ImageDansDiapo()` : Constructeur de la classe `ImageDansDiapo`.
2. `~ImageDansDiapo()` : Destructeur de la classe `ImageDansDiapo`.
3. `int getPos() const` : Renvoie la position de l'image dans le tableau d'images.
4. `int getRang() const` : Renvoie le rang de l'image dans le diaporama.
5. `void setPos(int pPos)` : Définit la position de l'image dans le tableau d'images.
6. `void setRang(int pRang)` : Définit le rang de l'image dans le diaporama.

Lecteur

Attributs :

1. `_mode` : Mode du lecteur (automatique ou manuel).
2. `_etat` : État du lecteur (vide ou chargé).

Méthodes :

1. `Lecteur()` : Constructeur de la classe `Lecteur`.
2. `~Lecteur()` : Destructeur de la classe `Lecteur`.
3. `void charger(Diaporamas& pDiaporama)` : Chargement du tableau des diaporamas.
4. `void charger(Images& pImages)` : Chargement du tableau des images.
5. `void avancer(const Diaporama& pDiaporama, unsigned int& pPosImageCourante)` : Incrémente la position de l'image courante.
6. `void reculer(const Diaporama& pDiaporama, unsigned int& pPosImageCourante)` : Décrémente la position de l'image courante.
7. `void setMode(string pMode)` : Définit le mode du lecteur.
8. `void setEtat(bool pEtat)` : Définit l'état du lecteur.
9. `string getMode() const` : Renvoie le mode du lecteur.
10. `bool getEtat() const` : Renvoie l'état du lecteur.
11. `void verifChoix(char& pChoixAction)` : Saisie du choix d'action de l'utilisateur.
12. `unsigned int saisieVerifChoixDiaporama(const Diaporamas& pDiaporamas)` : Saisie du choix de diaporama.
13. `void afficherImageCourante(const Diaporama& pDiaporama, unsigned int pImageCourante, Image* pImage)` : Affichage des informations de l'image courante.
14. `void choixAction(char pChoixAction, const Diaporamas& pDiaporamas, unsigned int& pDiaporamaCourant, unsigned int& pImageCourante, const Images& pImages)` : Réalise une des actions selon le choix de l'utilisateur.

Image

Attributs :

1. `_titre` : Titre de l'image.
2. `_categorie` : Catégorie de l'image.
3. `_chemin` : Chemin de l'image.

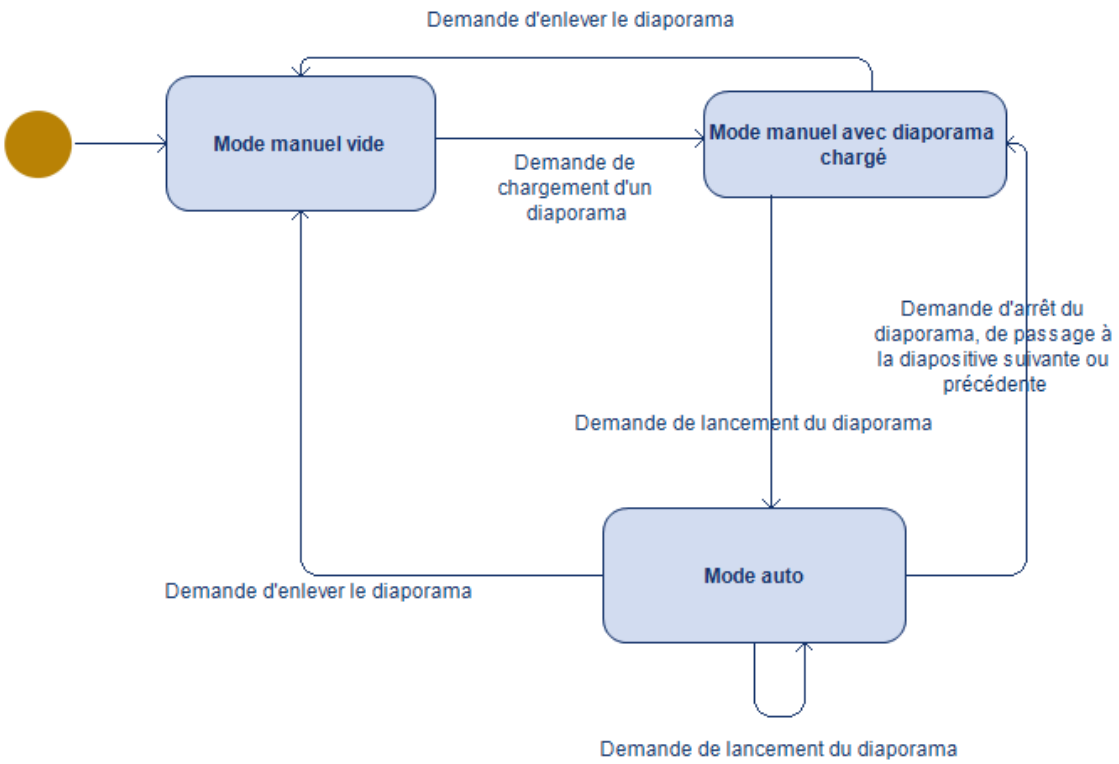
Méthodes :

1. `Image()` : Constructeur de la classe `Image`.
2. `~Image()` : Destructeur de la classe `Image`.
3. `void creationImage(string pCategorie, string pTitre, string pChemin)` : Méthode pour créer l'image avec ses attributs.
4. `string getCategorie() const` : Renvoie la catégorie de l'image.
5. `string getTitre() const` : Renvoie le titre de l'image.
6. `string getChemin() const` : Renvoie le chemin de l'image.
7. `void setTitre(string pTitre)` : Définit le titre de l'image.
8. `void setCategorie(string pCategorie)` : Définit la catégorie de l'image.
9. `void setChemin(string pChemin)` : Définit le chemin de l'image.
10. `void afficher()` : Affiche les informations de l'image.

v2

Objectif : Mettre en place la classe fenêtre personnalisée LecteurVue.

Diagramme d'état :



Liens entre éléments d’interface et fonctionnalités :

Actions

- **actionCharger** : Action du menu "Paramètres" permettant de charger un diaporama.
- **actionEnlever** : Action du menu "Paramètres" permettant de retirer un diaporama précédemment chargé.
- **actionQuitter** : Action du menu "Fichier" permettant de quitter l'application.
- **actionAProposDe** : Action du menu "Aide" permettant d'afficher les informations sur la version de l'application et ses auteurs.

Labels et Boutons

- **labMode** : Label affichant le mode de fonctionnement actuel (manuel ou automatique).
- **bSuivant** : Bouton permettant d'afficher la diapositive suivante en mode manuel ou de passer du mode automatique au mode manuel.
- **bPrecedent** : Bouton permettant d'afficher la diapositive précédente en mode manuel ou de passer du mode automatique au mode manuel.
- **bLancer** : Bouton permettant de lancer le diaporama en mode automatique à partir du début.
- **bArreter** : Bouton permettant de repasser en mode manuel depuis le mode automatique (ce bouton est inactif en mode manuel).
- **labTitre** : Label affichant le titre de la diapositive actuelle.
- **labCat** : Label affichant la catégorie de l'image actuelle.
- **labDiapo** : Label affichant l'image de la diapositive actuelle.
- **labNum** : Label affichant le numéro de la diapositive actuelle.

Classe(s) rajoutée(s) :

LecteurVue

Attributs :

1. `ui` : Pointeur vers l'objet de l'interface utilisateur de `LecteurVue`.

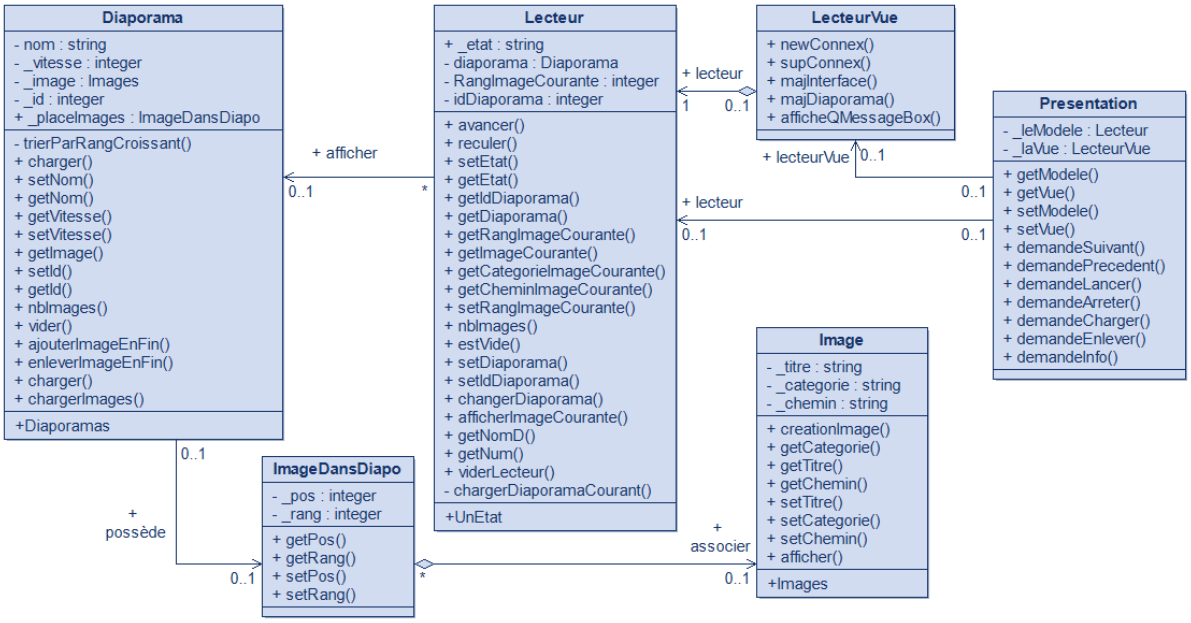
Méthodes :

1. `LecteurVue(QMainWindow *parent = nullptr)` : Constructeur de la classe `LecteurVue`.
2. `~LecteurVue()` : Destructeur de la classe `LecteurVue`.
3. `void newConnex(QObject *c)` : Méthode pour créer une connexion avec la présentation.
4. `void supConnex(QObject *c)` : Méthode pour supprimer une connexion avec la présentation.
5. `void majInterfaceEtat(Lecteur::UnEtat e)` : Méthode pour mettre à jour l'interface en fonction de l'état du lecteur.

v3

Objectif : Affichage des images dans l'interface visuelle.

Diagramme de classe :



Dictionnaire des éléments

Présentation

Attributs :

1. `_leModele` : Pointeur vers le modèle de lecteur.
2. `_laVue` : Pointeur vers la vue de lecteur.

Méthodes :

1. `Presentation(QObject *parent = nullptr)` : Constructeur de la classe `Presentation`.
2. `Lecteur* getModele()` : Méthode pour obtenir le modèle de lecteur.
3. `LecteurVue* getVue()` : Méthode pour obtenir la vue du lecteur.
4. `void setModele(Lecteur *m)` : Méthode pour définir le modèle de lecteur.
5. `void setVue(LecteurVue *v)` : Méthode pour définir la vue du lecteur.
6. `void demandeSuivant()` : Slot pour gérer la demande de lecture de la diapositive suivante.
7. `void demandePrecedent()` : Slot pour gérer la demande de lecture de la diapositive précédente.
8. `void demandeLancer()` : Slot pour gérer la demande de démarrage de la lecture.
9. `void demandeArreter()` : Slot pour gérer la demande d'arrêt de la lecture.
10. `void demandeCharger()` : Slot pour gérer la demande de chargement de diaporama.
11. `void demandeEnlever()` : Slot pour gérer la demande de suppression de diaporama.
12. `void demandeInfo()` : Slot pour gérer la demande d'information sur le diaporama.

LecteurVue

Attributs :

1. `ui` : Pointeur vers l'objet de l'interface utilisateur de `LecteurVue`.

Méthodes :

1. `LecteurVue(QMainWindow *parent = nullptr)` : Constructeur de la classe `LecteurVue`.
2. `~LecteurVue()` : Destructeur de la classe `LecteurVue`.
3. `void newConnex(QObject *c)` : Méthode pour créer une connexion avec la présentation.
4. `void supConnex(QObject *c)` : Méthode pour supprimer une connexion avec la présentation.
5. `void majInterface(Lecteur::UnEtat e, string num = "0/0", string tD = "Titre D", string tI = "titre I", string tC = "titre cat", string chem = "image")` : Méthode pour mettre à jour l'interface en fonction de l'état du lecteur.
6. `void majDiaporama(string num, string tD, string tI, string tC, string chem)` : Méthode pour mettre à jour l'affichage du diaporama
7. `void afficheQMessageBox()` : Méthode pour afficher une boîte de message.

Lecteur

Attributs :

1. `_etat` : État actuel du lecteur, représenté par l'énumération `UnEtat`.
2. `diaporama` : Pointeur vers le diaporama actuellement chargé dans le lecteur.
3. `RangImageCourante` : Rang de l'image courante dans le diaporama.
4. `idDiaporama` : Identifiant du diaporama courant.

Méthodes :

1. `Lecteur()` : Constructeur de la classe `Lecteur`.
2. `~Lecteur()` : Destructeur de la classe `Lecteur`.
3. `void avancer()` : Incrémente la position de l'image courante dans le diaporama.
4. `void reculer()` : Décrémente la position de l'image courante dans le diaporama.
5. `void setEtat(UnEtat pEtat)` : Définit l'état du lecteur.
6. `UnEtat getEtat() const` : Renvoie l'état actuel du lecteur.
7. `unsigned int getIdDiaporama() const` : Renvoie l'identifiant du diaporama courant.
8. `Diaporama* getDiaporama() const` : Renvoie le pointeur vers le diaporama courant.
9. `unsigned int getRangImageCourante() const` : Renvoie le rang de l'image courante dans le diaporama.
10. `string getImageCourante() const` : Renvoie le chemin de l'image courante.
11. `string getCategorieImageCourante() const` : Renvoie la catégorie de l'image courante.
12. `string getCheminImageCourante() const` : Renvoie le chemin de l'image courante.
13. `void setRangImageCourante(unsigned int pRangImageCourante)` : Définit le rang de l'image courante dans le diaporama.
14. `unsigned int nbImages() const` : Renvoie le nombre d'images dans le diaporama courant.
15. `bool estVide() const` : Vérifie si le lecteur est vide (sans diaporama chargé).
16. `void setDiaporama(Diaporama* pDiaporama)` : Définit le diaporama courant.
17. `void setIdDiaporama(unsigned int pIdDiaporama)` : Définit l'identifiant du diaporama courant.
18. `void changerDiaporama(unsigned int pId)` : Change le diaporama courant.
19. `void afficherImageCourante(const Diaporama& pDiaporama, unsigned int pImageCourante, Image* pImage)` : Affiche les informations de l'image courante.
20. `string getNomD()` : Renvoie le nom du diaporama courant.
21. `string getNum()` : Renvoie le numéro de l'image courante.

22. `void viderLecteur()` : Vide le lecteur en libérant la mémoire allouée au diaporama courant.
23. `void chargerDiaporamaCourant()` : Charge le diaporama courant.

ImagesDansDiapo

Attributs :

1. `_pos` : Représente le rang de l'image dans le tableau d'images (vector<Images>), correspondant à l'ordre de chargement initial des images dans la table des images.
2. `_rang` : Représente le rang de l'image dans le diaporama, correspondant à l'ordre d'affichage choisi par l'utilisateur lors de la création du diaporama.

Méthodes :

1. `ImageDansDiapo(int, unsigned int)` : Constructeur par défaut prenant en paramètre la position et le rang de l'image dans le diaporama.
2. `~ImageDansDiapo()` : Destructeur de la classe `ImageDansDiapo`.
3. `int getPos() const` : Renvoie la position de l'image dans le tableau d'images.
4. `int getRang() const` : Renvoie le rang de l'image dans le diaporama.
5. `void setPos(int position)` : Définit la position de l'image dans le tableau d'images.
6. `void setRang(int rang)` : Définit le rang de l'image dans le diaporama.

Image

Attributs :

1. `_titre` : Titre de l'image.
2. `_categorie` : Catégorie de l'image.
3. `_chemin` : Chemin de l'image.

Méthodes :

1. `Image()` : Constructeur par défaut de la classe `Image`.
2. `~Image()` : Destructeur de la classe `Image`.
3. `void creationImage(string pCategorie, string pTitre, string pChemin)` : Méthode pour créer l'image avec ses attributs.
4. `string getCategorie() const` : Renvoie la catégorie de l'image.
5. `string getTitre() const` : Renvoie le titre de l'image.
6. `string getChemin() const` : Renvoie le chemin de l'image.
7. `void setTitre(string pTitre)` : Définit le titre de l'image.
8. `void setCategorie(string pCategorie)` : Définit la catégorie de l'image.

9. `void setChemin(string pChemin)` : Définit le chemin de l'image.
10. `void afficher()` : Affiche les informations de l'image.

Diaporama

Attributs :

1. `_nom` : Nom du diaporama.
2. `_vitesse` : Vitesse de défilement du diaporama.
3. `_image` : Collection d'images du diaporama.
4. `_id` : Identifiant du diaporama.
5. `_placeImages` : Vecteur d'objets `ImageDansDiapo` représentant les images du diaporama.

Méthodes :

1. `Diaporama()` : Constructeur par défaut de la classe `Diaporama`.
2. `~Diaporama()` : Destructeur de la classe `Diaporama`.
3. `void setNom(string pNom)` : Définit le nom du diaporama.
4. `string getNom() const` : Renvoie le nom du diaporama.
5. `int getVitesse() const` : Renvoie la vitesse de défilement du diaporama.
6. `void setVitesse(int pVitesse)` : Définit la vitesse de défilement du diaporama.
7. `Images getImage() const` : Renvoie la collection d'images du diaporama.
8. `void setId(unsigned int pId)` : Définit l'identifiant du diaporama.
9. `unsigned int getId() const` : Renvoie l'identifiant du diaporama.
10. `unsigned int nbImages() const` : Renvoie le nombre d'images du diaporama.
11. `void vider()` : Vide la collection d'images du diaporama.
12. `void ajouterImageEnFin(ImageDansDiapo* pImage)` : Ajoute une image à la fin du diaporama.
13. `void enleverImageEnFin()` : Enlève une image à la fin du diaporama.
14. `void charger(Images& pImages)` : Charge les images associées au diaporama à partir de la collection `pImages`.
15. `void charger()` : Charge les images associées au diaporama courant (d'identifiant `id`) à partir de la collection `pImages`.
16. `void chargerImages(Images& pImages)` : Charge les images associées au diaporama à partir de la collection `pImages`.

v4

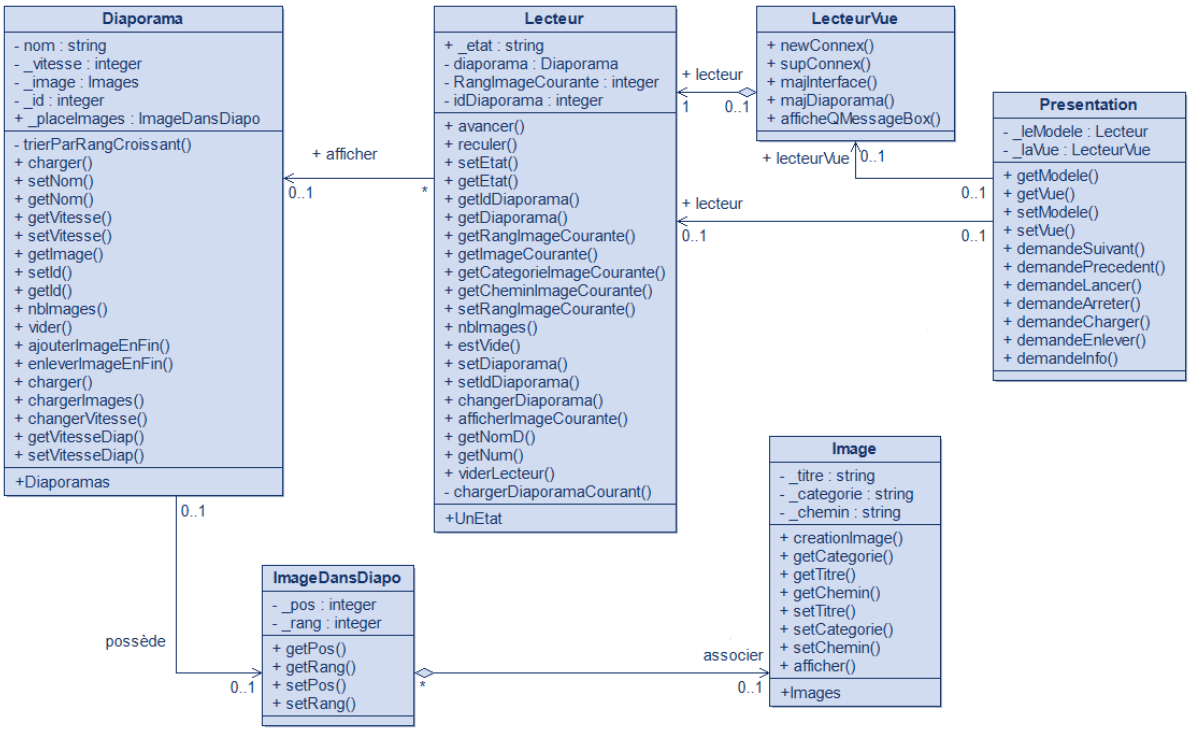
Objectif : Mise en place du mode auto

Aucune classe n'a été modifiée.

v5

Objectif : Modifier la vitesse de défilement.

Diagramme de classe :



Méthodes ajoutées dans Diaporama :

`void changerVitesse()` :>

- Type : Slot public
- But : Changer la vitesse du diaporama.

`int getVitesseDiap() const` :

- Type : Fonction

- But : Renvoie la vitesse de défilement du diaporama.

`void setVitesseDiap(int vitesse)` :

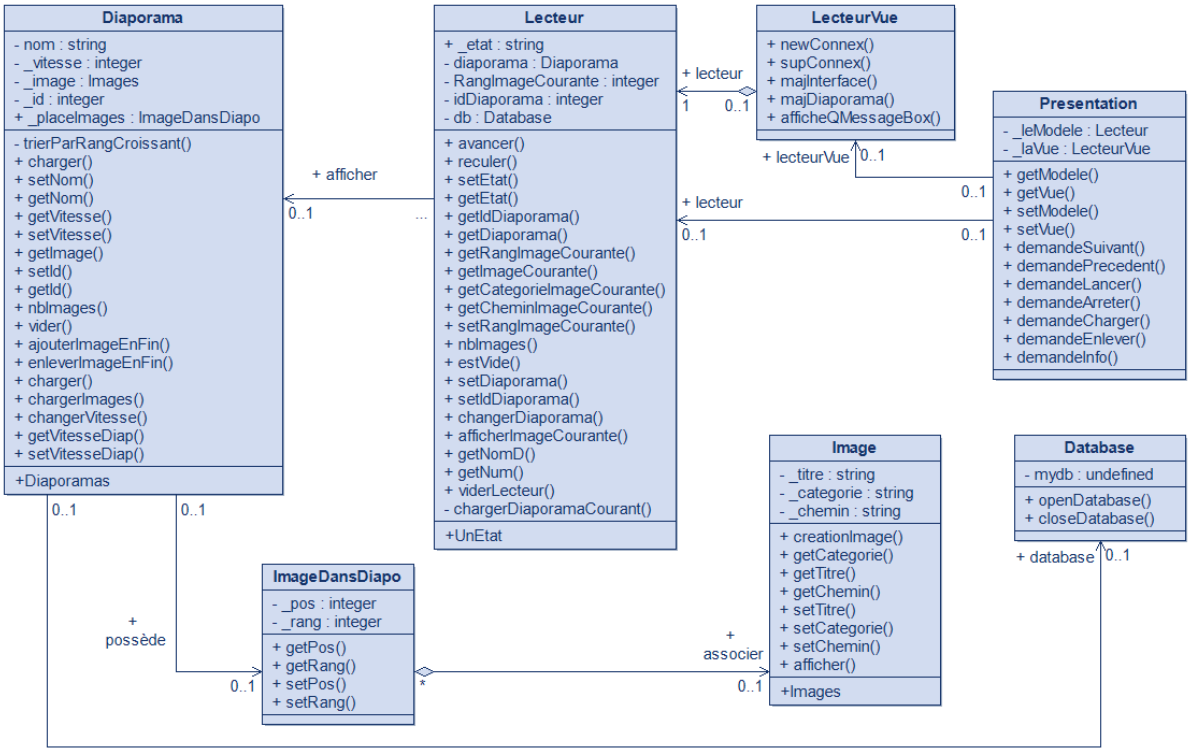
- Type : Procédure
- But : Définit la vitesse du diaporama.

v6

Objectif : Charger les informations relatives aux images à partir de la base de données.

Classes rajoutées : Database

Diagramme de classe :



Dictionnaire des éléments de Database

Attributs

`mydb :`

- But : Stocker l'objet QSqlDatabase, qui représente la connexion à la base de données
- Type : QSqlDatabase

Méthodes

`bool openDatabase()` :

- Type : Fonction
- But : Retourne true si la base de données est ouverte avec succès, sinon retourne false

`void closeDatabase()` :

- Type : Procédure
- But : Méthode pour fermer la base de données private

Changement dans les classes

Diaporama : Ajout d'un attribut privé *db, qui est le pointeur vers mydb, l'attribut privé de Database.

Ajout d'un attribut public ayant comme type vecteur ImageDansDiapo* appelé _placeImages.

Pas de changement dans les autres classes.

v7

Objectif : Mise en œuvre de la méthode qui charge des diaporamas à partir de la base de données, et de son slot.

Changement dans les classes

v8

Objectif : Modifier dans la base de données la vitesse de défilement du diaporama ou l'intitulé ou le chemin d'accès à l'image.

Changement dans les classes

Bilan :

- Ce que vous avez appris

Nous avons appris à organiser un code en appliquant notamment avec le modèle MVP. Ensuite on utilise une documentation pour nos besoins spécifiques. Nous nous sommes amélioré sur le débogage.

- Ce que vous avez aimé / pas aimé

- Notre équipe a bien apprécié le fait de pouvoir développer une interface utilisateur.
- Nous n'avons pas trop apprécié le temps imparti pour tous les jalons. Nous pensons qu'il aurait était mieux de faire 2 ou 3 version (par exemple: mode manuel mise en place, mode manuel mvp, mode auto mvp)

- Ce qui a été difficile

Le fait de trouver des erreurs dans notre code surtout quand elles étaient petites et non indiquées par le compilateur.

- Le temps passé (sur conception/sur code) ?

Temps réalisé / Membres	Kamelia	Imane	Marylou
Conception	7h	8h	5h
Code	23h	22h	12h
Temps personnel par rapport au temps de la SAE	30h	30h	17h

Le temps est calculé avec le temps hors séances prévue à l'emploi du temps

- Ce que vous auriez pu faire mieux ?

Peaufiner les version et plus argumenter le dossier de conception

- Ce qui pourrait être amélioré dans la saé ?

Le rassemblement des versions entre elles. Et le retour des professeurs pour chaque version avant d’en commencer une nouvelle.