Saé 2.01 – Développement d'une application

Lecteur de diaporamas – Dossier d'Analyse et conception

LORIDANT Julien MORANCE Kyllian VERNIS Gabriel

---

1. **Compléments de spécifications externes.**

Nous n'avons pas trouvé de spécifications externes à ajouter.

2. **Scénarios**

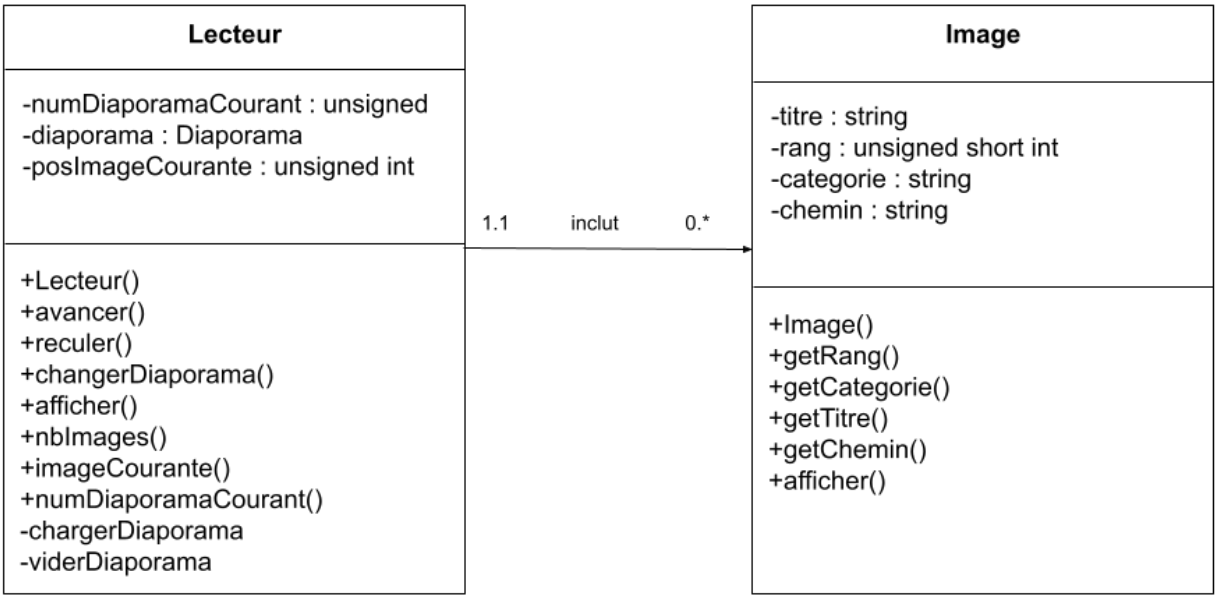
Scénario nominal : **l'utilisateur utilise le lecteur en mode manuel**

1. l'utilisateur clique sur " Paramètres >> charger Diaporama" et sélectionne un diaporama dans la base de données.
2. Le diaporama est chargé en mémoire.
3. La première image est affichée.
4. L'utilisateur fait défiler le diaporama en utilisant les boutons "suivant" et "précedent".
5. A la fin du diaporama l'utilisateur clique sur "Paramètres >> enlever diaporama".
6. L'utilisateur quitte l'application en utilisant le bouton "Fichier >>Quitter".

Scénario alternatif 1: L'utilisateur utilise le mode automatique.

1. l'utilisateur clique sur " Paramètres >> charger Diaporama" et sélectionne un diaporama dans la base de données.
2. Le diaporama est chargé en mémoire.
3. La première image est affichée.
4. L'utilisateur clique sur "Lancer le diaporama"
5. Le diaporama passe en mode automatique.
6. A la fin du diaporama, ce dernier repasse à la première image
7. Lorsque l'utilisateur souhaite arrêter, il clique sur le bouton "arrêter diaporama"

3. **Diagramme de classe (UML)**



Classe Lecteur			
Nom attribut	**Signification**	**Type**	**Exemple**
`numDiaporamaCourant`	Correspond au numéro de diaporama courant	`unsigned`	`1`
`diaporama`	Correspond au pointeur vers les images du diaporama	`diaporama`	`a voir`
`posImageCourante`	Correspond à la position courante de l'image dans le diaporama	`unsigned int`	`8`

Tableau 2 : Dictionnaire des éléments - Classe Lecteur

Classe Image			
Nom attribut	**Signification**	**Type**	**Exemple**
`tire`	Correspond au titre de l'image	`string`	`Cendrillon`
`rang`	Correspond au rang de l'image dans le diaporama associé	`unsigned int`	`2`
`categorie`	Correspond à la catégorie de l'image (personne, animal,..).	`string`	`personne`
`chemin`	Correspond au chemin du dossier contenant les images	`string`	`C:\\Images\Cendrillon.png`

Tableau 3 : Dictionnaire des éléments - Classe Image

```
#ifndef LECTEUR_H
#define LECTEUR_H
#include "image.h"
#include <vector>

typedef vector<Image*> Diaporama;   // Structure de données contenant les infos sur les images

class Lecteur
{
public:
    Lecteur();
    void avancer();             // incrémente _posImageCourante, modulo nbImages()
    void reculer();             // décrémente _posImageCourante, modulo nbImages()
    void changerDiaporama(unsigned int pNumDiaporama);    // permet de choisir un diaporama, 0 si
aucun diaporama souhaité
    void afficher();            // affiche les informations sur lecteur-diaporama et image
courante
    unsigned int nbImages();    // affiche la taille de _diaporama
    Image* imageCourante();     // retourne le pointeur vers l'image courante
    unsigned int numDiaporamaCourant();

private:
    unsigned _numDiaporamaCourant;   // numéro du diaporama courant, par défaut 0
    Diaporama _diaporama;            // pointeurs vers les images du diaporama
    unsigned int _posImageCourante;  /* position, dans le diaporama,
                                        de l'image courante.
                                        Indéfini quand diaporama vide.
                                        Démarre à 0 quand diaporama non vide */
private:
    void chargerDiaporama();    // charge dans _diaporama les images du _numDiaporamaCourant
    void viderDiaporama();      // vide _diaporama de tous ses objets image et les delete
};

#endif // LECTEUR_H
```

Figure 4 : Schéma de classes = Classe Lecteur

```
#ifndef IMAGE_H
#define IMAGE_H
#include <iostream>
using namespace std;

class Image
{
public:
    Image(unsigned int pRang=0,
              string pCategorie="", string pTitre="", string pChemin = "");
    unsigned int getRang();
    string getCategorie();
    string getTitre();
    string getChemin();
    void afficher();           // affiche tous les champs de l'image

private:
    unsigned int _rang;         /* rang de l'image au sein du diaporama
                                   auquel l'image est associée */
    string _titre;              // intitulé de l'image
    string _categorie;          // catégorie de l'image (personne, animal, objet)
    string _chemin;             // chemin complet vers le dossier où se trouve l'image
};

#endif // IMAGE_H
```

Figure 4 : Schéma de classes = Classe Images

Version v0 – Version console seule

4. Implémentation et tests

4.1 Implémentation

Liste et rôle des fichiers de cette version :

lecteur.h	Spécification de la classe Lecteur
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image
image.cpp	Corps de la classe Image
main.cpp	Teste les méthodes de la classe Lecteur

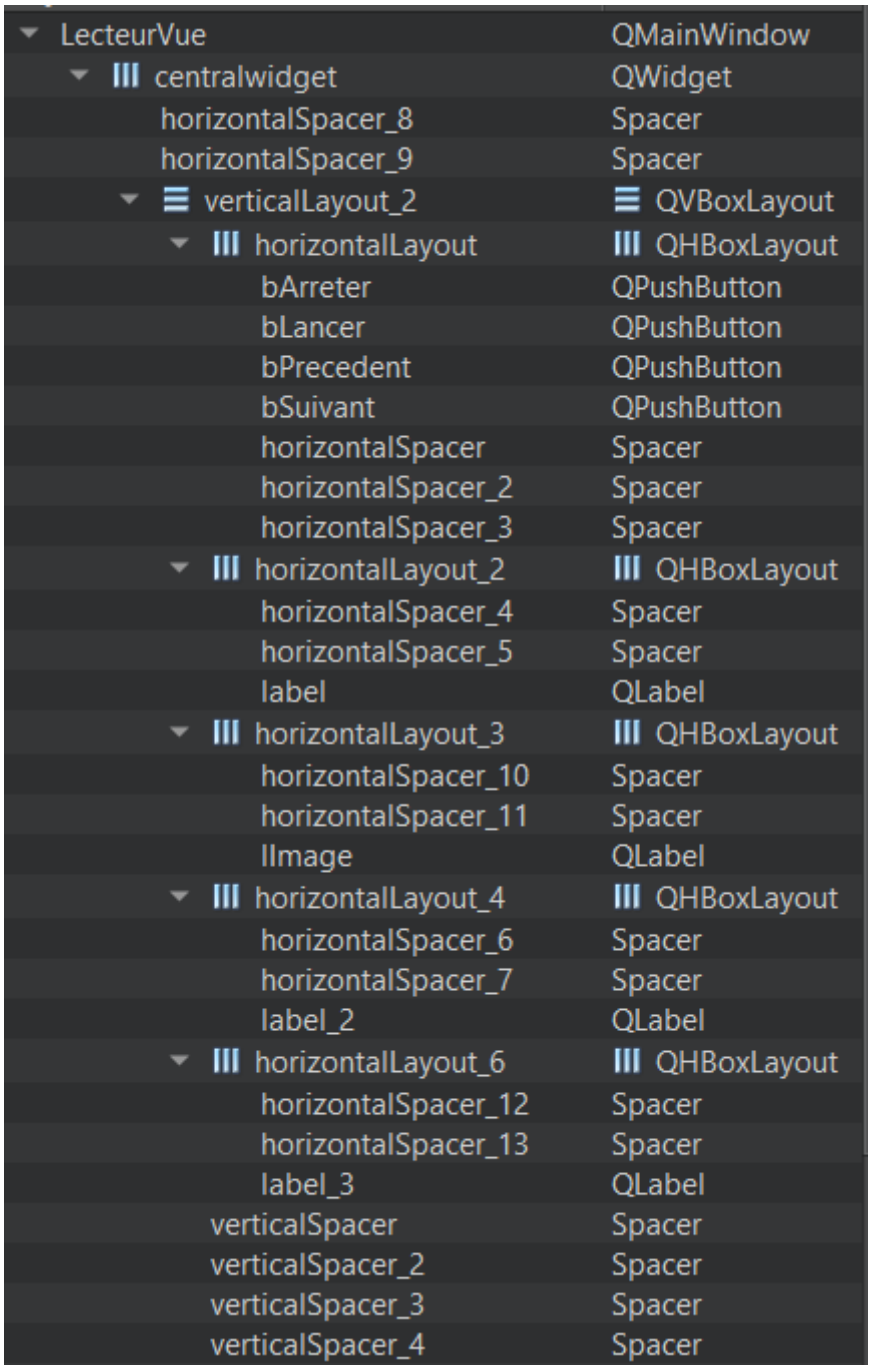
4.2 Test

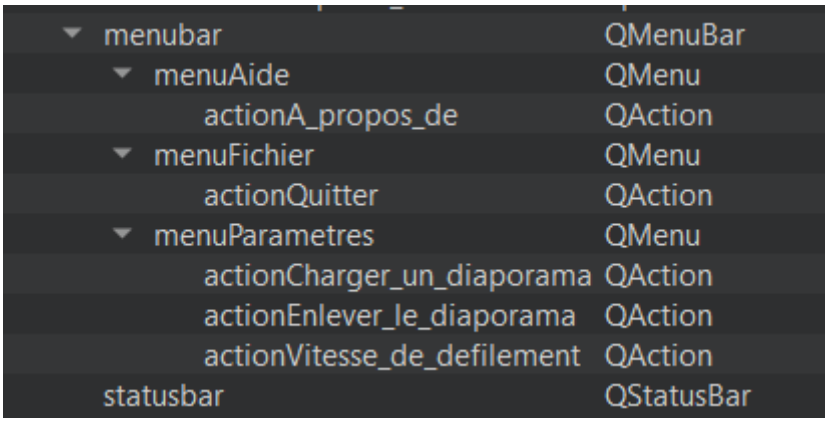
Test avec le programme fourni main.cpp

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)
valide n°1	Chemin d'accès correspondant à une image	"C:\\cartesDisney\\carteDisney2.gif"	affichage du chemin de l'image
valide n°2	Catégorie de de l'image est dans valeurs possibles (personne, animal ou objet)	"personne"	personne
invalide n°1	Chemin d'accès ne correspondant à aucune fichier	"C: carteDisney2.gif"	Echec
invalide n°2	Chemin d'accès menant à un autre fichier d'une image	"C:\\cartesDisney\\carteDisney2.txt"	Echec
invalide n°3	Chemin d'accès vide	""	Echec
invalide n°4	Catégorie de l'image n'est pas dans les valeurs possibles	"véhicule"	Echec
invalide n°5	Catégorie de l'image vide	""	Echec
invalide n°6	Titre de l'image vide	""	Echec

Version v1 – projet Graphique seul

5. Éléments d'interface





6. Implémentation et tests

6.1 Implémentation

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas
lecteurVue.cpp	Corps de la classe LecteurVue
lecteurvue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
main.cpp	Teste les méthodes de la classe Lecteur

Remarques sur l'implémentation :

Nous avons implémenté un slot pour chaque éléments d'interface avec lequel l'utilisateur pourra interagir (boutons, barre de menu). Ce slot a pour but d'afficher un message dans la console lorsque l'on interagit avec. Nous avons choisi pour les signaux, des signaux proposés directement par QT car ils correspondent aux interactions que l'utilisateur pourra avoir avec certains éléments d'interface comme les signaux clicked() pour les boutons et triggered() pour les éléments de la barre de menu.

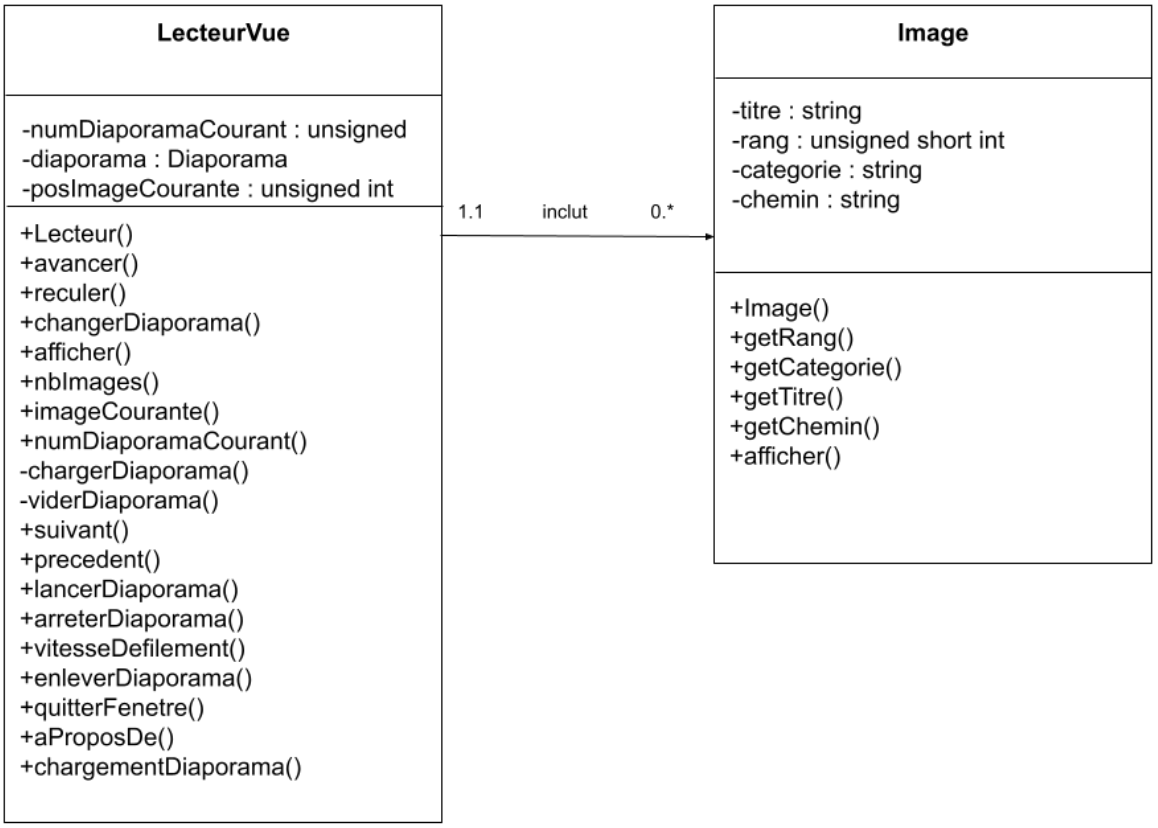
6.2 Test

Nom slot	Description	Résultats attendu(s)	Résultats obtenu(s)
suivant()	affiche un message dans la console lorsque le bouton "suivant" est cliqué	"diapo suivante"	diapo suivante
precedent()	affiche un message dans la console lorsque le bouton "précédent" est cliqué	"diapo précédente"	diapo précédente
lancerDiaporama()	affiche un message dans la console lorsque le bouton "lancer diaporama" est cliqué	"lancement du diaporama"	lancement du
arreterDiaporama()	affiche un message dans la console lorsque le bouton "arrêter le diaporama"" est cliqué	"arreter le diaporama"	arreter le diaporama

vitesseDefilement()	affiche un message dans la console lorsque l'option "vitesse de défilement" du menu paramètres est cliqué	"changement de la vitesse"	changement de la vitesse
enleverDiaporama()	affiche un message dans la console lorsque l'option "enlever le diaporama" du menu paramètres est cliqué	"enlever le diaporama"	enlever le diaporama
quitterFenetre()	affiche un message dans la console lorsque l'option "quitter du menu fichier" est cliqué	"quitter la fenetre"	quitter la fenetre
aProposDe()	affiche un message dans la console lorsque l'option de "A propos de …" du menu "aide" est cliqué	"a propos de"	a propos de

Version v2 –

7. Diagramme de classes (UML)



8. Comportement de l'application

7.1 Diagramme états-transitions-actions du lecteur de diaporamas (v2)

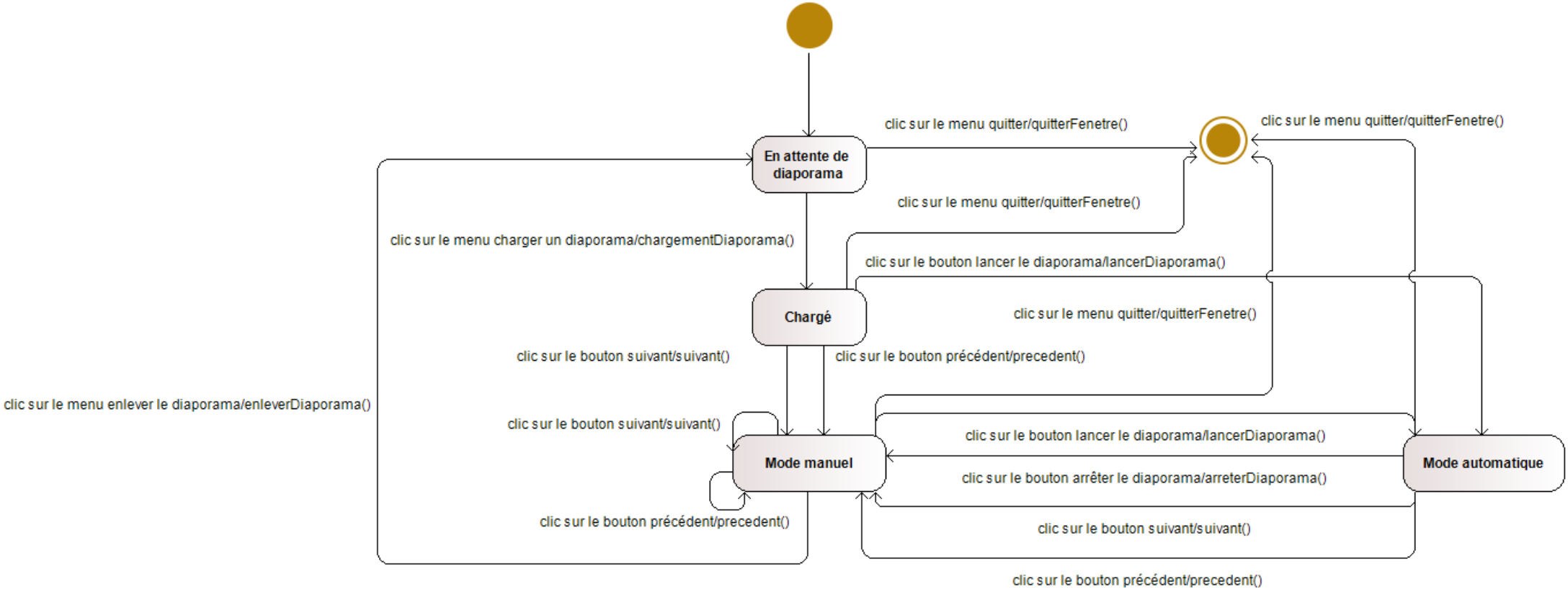


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v2

7.2 Dictionnaire des états, événements et Actions (v2)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`En attente de diaporama`	État correspondant aux moments où l'application n'a aucun diaporama de chargé.
`Chargé`	État correspondant aux moments où l'application a chargé un diaporama.
`Mode manuel`	État correspondant aux moments où le diaporama est en mode manuel.
`Mode automatique`	État correspondant aux moments où le diaporama est en mode automatique.

Tableau 2 : États du lecteur de diaporamas – v2

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`clic sur le menu enlever le diaporama`	Correspond aux moments où l'utilisateur clic sur le menu enlever le diaporama
`clic sur le menu quitter`	Correspond aux moments où l'utilisateur clic sur le menu quitter
`clic sur le menu charger un diaporama`	Correspond aux moments où l'utilisateur clic sur le menu charger un diaporama
`clic sur le bouton suivant`	Correspond aux moments où l'utilisateur clic sur le bouton suivant
`clic sur le bouton précédent`	Correspond aux moments où l'utilisateur clic sur le bouton précédent
`clic sur le bouton lancer le diaporama`	Correspond aux moments où l'utilisateur clic sur le bouton lancer le diaporama
`clic sur le bouton arrêter le diaporama`	Correspond aux moments où l'utilisateur clic sur le arrêter le diaporama

Tableau 3 : Evénements faisant changer le diaporama d'état – v2

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`enleverDiaporama()`	Slot qui affiche "enlever le diaporama "dans la console et qui appelle la méthode viderDiaporama()
`quitterFenetre()`	Slot qui affiche "quitter la fenetre" dans la console et qui appelle la méthode close()
`chargementDiaporama()`	Slot qui affiche "chargement du diaporama" dans la console et qui appelle la méthode chargerDiaporama()

`suivant()`	Slot qui affiche “diapo suivante” dans la console et qui appelle les méthodes avancer() et afficher()
`precedent()`	Slot qui affiche “diapo précédente” dans la console et qui appelle les méthodes reculer() et afficher()
`lancerDiaporama()`	Slot qui affiche “diapo lancement du diaporama” dans la console et qui appelle les méthodes avancer(), afficher()
`arreterDiaporama()`	Slot qui affiche “arreter le diaporama” dans la console et qui appelle les méthodes arreterModeAutomatique() et afficher()

Tableau 4 : Actions à réaliser lors des changements d’état – lecteur de diaporamas v2

7.3 Table T_EtatsEvenementsActions (v2)

Correspondance matricielle du diagramme états-transitions de l’application :
- en *ligne : les **états*** du lecteur de diaporamas (éventuel état de départ d’une transition)
- en *colonne : les **événements*** faisant changer le lecteur d’état (déclencheur d’une transition)
- dans chaque cellule : l’état d’arrivée de la transition + action/traitement à faire + éventuellement garde accompagnant la transition

Élément graphique prenant en charge cet événement □	clic sur le menu enlever le diaporama	clic sur le menu quitter	clic sur le menu charger un diaporama	clic sur le bouton suivant	clic sur le bouton précédent	clic sur le bouton lancer le diaporama	clic sur le bouton arrêter le diaporama
Événement □ *En attente de diaporama*	---	Etat final	Chargé	---	---	---	---
Chargé	---	Etat final	---	---	---	Mode automatique	---
Mode manuel	En attente de diaporama	Etat final	---	Mode manuel	Mode manuel	Mode automatique	---
Mode automatique	En attente de diaporama	Etat final	---	Mode manuel	Mode manuel	---	Mode manuel

Tableau 5 : Matrice d’états-transitions du lecteur de diaporamas – v2

9. Implémentation et tests

8.1 Implémentation (v2)

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas Contient les définitions des méthodes et des slots de la classe LecteurVue
lecteurVue.cpp	Corps de la classe LecteurVue.
lecteurvue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
lecteur.h	Spécification de la classe Lecteur. Contient les définitions des méthodes de la classe Lecteur
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image Contient les définitions des méthodes de la classe Image
image.cpp	Corps de la classe Image
main.cpp	??

Remarques sur l'implémentation :

Nous avons implémenté un slot pour chaque éléments d'interface avec lequel l'utilisateur pourra interagir (boutons, barre de menu). Ce slot a pour but d'afficher un message dans la console d'appeler une méthode provenant de la classe Lecteur de la V0 pour la gestion de l'affichage et de l'avancement du diaporama par exemple lorsque l'on interagit avec. Nous avons choisi pour les signaux, des signaux proposés directement par QT car ils correspondent aux interactions que l'utilisateur pourra avoir avec certains éléments d'interface comme les signaux clicked() pour les boutons et triggered() pour les éléments de la barre de menu.

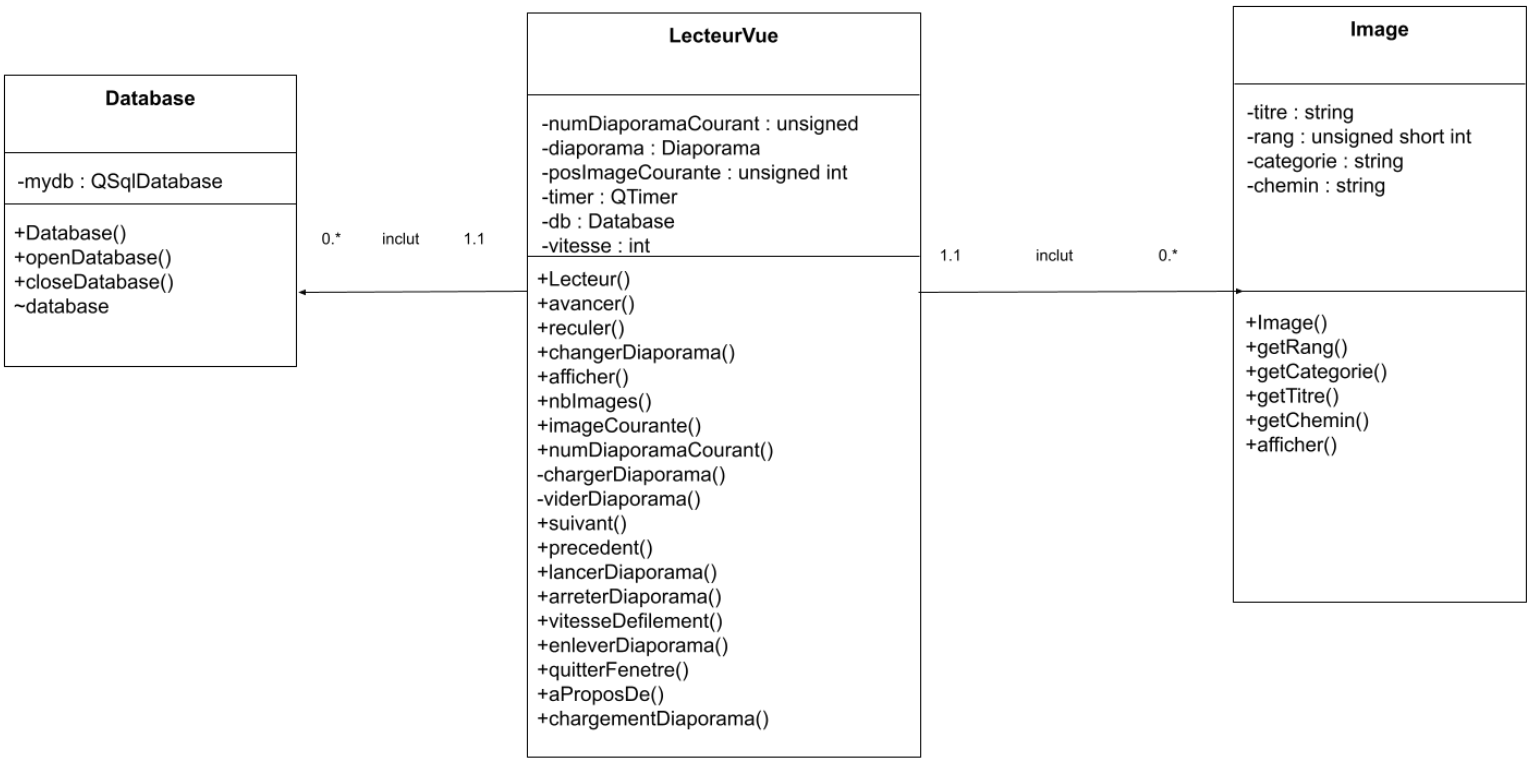
8.2 Tests (v2)

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)
valide n°1	Chemin d'accès correspondant à une image	"C:\\cartesDisney\\carteDisney2.gif"	affichage de l'image dans l'interface
valide n°2	Catégorie de l'image est dans valeurs possibles (personne, animal ou objet)	"personne"	affichage de personne dans l'interface
valide n°3	Clic sur le menu "charger un diaporama" sans avoir de diaporama chargé au préalable	l'action actionCharger_un_diaporama" est appelé	affiche d'un message confirmant le chargement du diaporama
valide n°4	Clic sur le bouton "suivant" après avoir chargé un diaporama	le bouton bSuivant est activé	passage à l'image suivante et affiche un message dans la console confirmant le passage à l'image suivante
valide n°5	Clic sur le bouton " précédent" après avoir chargé un diaporama	le bouton bPrécédent est activé	passage à l'image précédente et affiche un message dans la console confirmant le passage à l'image précédente
valide n°6	Clic sur le bouton "lancer	le bouton bLancer est	change le statut du

	diaporama” après avoir chargé un diaporama	activé	diaporama en mode automatique et affiche un message dans la console confirmant le changement d’état
valide n°7	Clic sur le bouton “arrêter diaporama” après avoir chargé un diaporama	le bouton bArreter est activé	change le statut du diaporama en mode manuel et affiche un diaporama confirmant le change d’état
valide n°8	Clic sur le menu “vitesse de défilement” après avoir chargé un diaporama	l'action actionVitesse_de_defilement est appelée	affiche un message dans la console confirmant le changement de la vitesse
valide n°9	Clic sur le menu “quitter”	l’action actionQuitter est appelée	affiche un message dans la console confirmant la fermeture de la semaine et quitte la fenêtre
valide n°10	Clic sur le menu “enlever le diaporama”	l’action actionEnlever_le_diaporama	affiche un message dans la console
valide n°11	Clic sur le menu “A propos de“	l’action actionA_propos_de est appelée	affiche dans une nouvelle fenêtre les informations sur la version et les auteurs
invalide n°1	Chemin d’accès ne correspondant à aucune fichier	"C: carteDisney2.gif"	Echec
invalide n°2	Chemin d’accès menant à un autre fichier d'une image	"C:\\cartesDisney\\carteDisney2.txt"	Echec
invalide n°3	Chemin d’accès vide	“”	Echec
invalide n°4	Catégorie de l’image n’est pas dans les valeurs possibles	“véhicule”	Echec
invalide n°5	Catégorie de l’image vide	“”	Echec
invalide n°6	Titre de l’image vide	“”	Echec

Version v5 –

10. Diagramme de classes (UML)



11. Comportement de l'application

11.1 Diagramme états-transitions-actions du lecteur de diaporamas (v5)

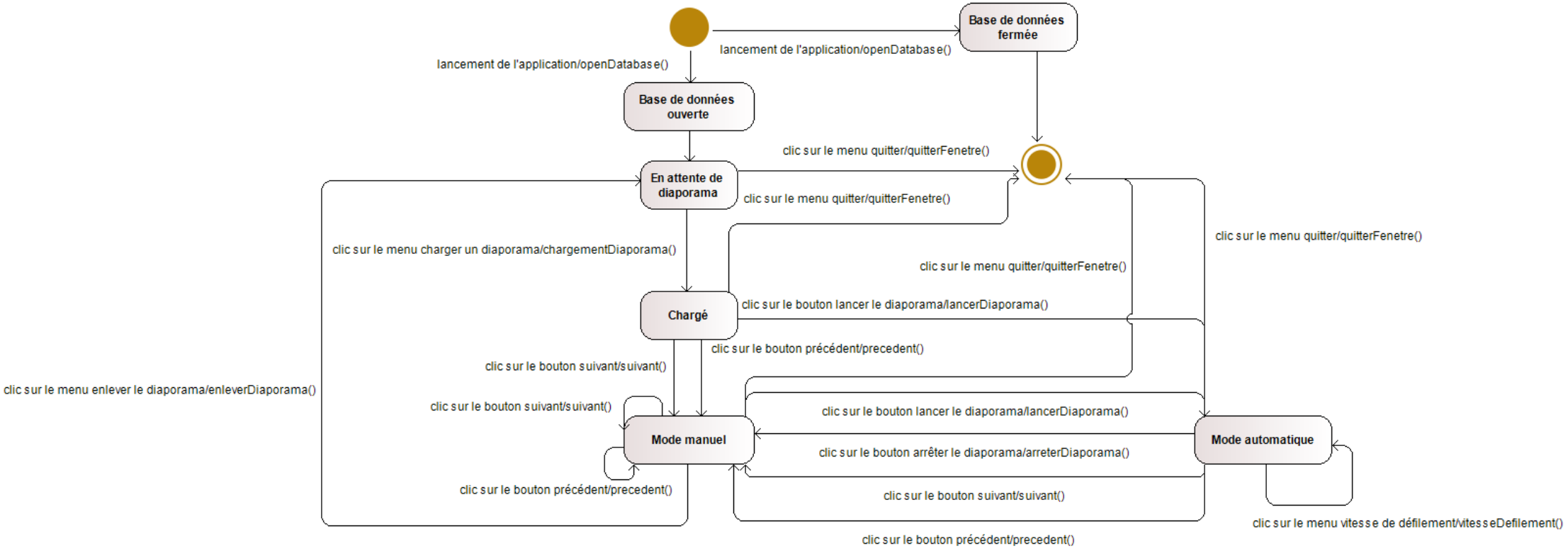


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v5

11.2 Dictionnaire des états, événements et Actions (v5)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`En attente de diaporama`	État correspondant aux moments où l'application n'a aucun diaporama de chargé.
`Chargé`	État correspondant aux moments où l'application a chargé un diaporama.
`Mode manuel`	État correspondant aux moments où le diaporama est en mode manuel.
`Mode automatique`	État correspondant aux moments où le diaporama est en mode automatique.
`Base de données ouverte`	État correspondant au moment où la base de données à réussi à être ouverte sans problème.
`Base de données fermée`	État correspondant au moment où la base de données n'a pas pu être ouverte suite à un ou plusieurs problèmes.

Tableau 2 : États du lecteur de diaporamas – v5

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`clic sur le menu enlever le diaporama`	Correspond aux moments où l'utilisateur clic sur le menu enlever le diaporama
`clic sur le menu quitter`	Correspond aux moments où l'utilisateur clic sur le menu quitter
`clic sur le menu charger un diaporama`	Correspond aux moments où l'utilisateur clic sur le menu charger un diaporama
`clic sur le bouton suivant`	Correspond aux moments où l'utilisateur clic sur le bouton suivant
`clic sur le bouton précédent`	Correspond aux moments où l'utilisateur clic sur le bouton précédent
`clic sur le bouton lancer le diaporama`	Correspond aux moments où l'utilisateur clic sur le bouton lancer le diaporama
`clic sur le bouton arrêter le diaporama`	Correspond aux moments où l'utilisateur clic sur le arrêter le diaporama
`lancement de l'application`	Correspond aux moments où l'utilisateur lance l'application
`clic sur le menu vitesse de défilement`	Correspond aux moments où l'utilisateur clic sur le menu vitesse de défilement

Tableau 3 : Evénements faisant changer le diaporama d'état – v5

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`enleverDiaporama()`	Slot qui affiche “enlever le diaporama “dans la console et qui appelle la méthode viderDiaporama()
`quitterFenetre()`	Slot qui affiche “quitter la fenetre” dans la console et qui appelle la méthode close()
`chargementDiaporama()`	Slot qui affiche “chargement du diaporama” dans la console et qui appelle la méthode chargerDiaporama()
`suivant()`	Slot qui affiche “diapo suivante” dans la console et qui appelle les méthodes avancer() et afficher()
`precedent()`	Slot qui affiche “diapo précédente” dans la console et qui appelle les méthodes reculer() et afficher()
`lancerDiaporama()`	Slot qui affiche “diapo lancement du diaporama” dans la console et qui appelle les méthodes avancer(), afficher()
`arreterDiaporama()`	Slot qui affiche “arrêter le diaporama” dans la console et qui appelle les méthodes arreterModeAutomatique() et afficher()
`vitesseDefilement()`	Slot qui affiche “changement de la vitesse” dans la console, ouvre une fenêtre demandant à l’utilisateur de saisir une vitesse en seconde et enfin change la vitesse de défilement si jamais une valeur est entrée.
`openDatabase()`	Méthode qui ouvre la base de données et retourne “Base de données ouverte avec succès” si la base a été ouverte sinon elle affiche “Erreur à l’ouverture de la base de données”

Tableau 4 : Actions à réaliser lors des changements d’état – lecteur de diaporamas v5

11.3 Table T_EtatsEvenementsActions (v5)

Correspondance matricielle du diagramme états-transitions de l'application :
- en *ligne : les **états*** du lecteur de diaporamas (éventuel état de départ d'une transition)
- en *colonne : les **événements*** faisant changer le lecteur d'état (déclencheur d'une transition)
- dans chaque cellule : l'état d'arrivée de la transition + action/traitement à faire + éventuellement garde accompagnant la transition

Élément graphique prenant en charge cet événement	clic sur le menu enlever le diaporama	clic sur le menu quitter	clic sur le menu charger un diaporama	clic sur le bouton suivant	clic sur le bouton précédent	clic sur le bouton lancer le diaporama	clic sur le bouton arrêter le diaporama	clic sur le menu vitesse de défilement	lancement de l'application
Événement *En attente de diaporama*	---	Etat final	Chargé	---	---	---	---	---	---
Chargé	---	Etat final	---	---	---	Mode automatique	---	---	---
Mode manuel	En attente de diaporama	Etat final	---	Mode manuel	Mode manuel	Mode automatique	---	---	---
Mode automatique	En attente de diaporama	Etat final	---	Mode manuel	Mode manuel	---	Mode manuel	Mode automatique	---
Base de données ouverte	---	Etat final	---	---	---	---	---	---	En attente de diaporama
Base de données fermée	---	---	---	---	---	---	---	---	---

Tableau 5 : Matrice d'états-transitions du lecteur de diaporamas – v5

12. Implémentation et tests

12.1 Implémentation (v5)

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas Contient les définitions des méthodes,des slots de la classe LecteurVue et des pointeurs vers les classe Database et QTimer
lecteurVue.cpp	Corps de la classe LecteurVue
lecteurvue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
lecteur.h	Spécification de la classe Lecteur Contient les définitions des méthodes de la classe Lecteur
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image Contient les définitions des méthodes de la classe Images

image.cpp	Corps de la classe Image
main.cpp	??
database.h	Spécification de la classe Database
database.cpp	Corps de la classe Database

Remarques sur l'implémentation :
Nous avons implémenté un slot pour chaque éléments d'interface avec lequel l'utilisateur pourra interagir (boutons, barre de menu). Ce slot a pour but d'afficher un message dans la console d'appeler une méthode provenant de la classe Lecteur de la V0 pour la gestion de l'affichage et de l'avancement du diaporama par exemple lorsque l'on interagit avec. Nous avons choisi pour les signaux, des signaux proposés directement par QT car ils correspondent aux interactions que l'utilisateur pourra avoir avec certains éléments d'interface comme les signaux clicked() pour les boutons et triggered() pour les éléments de la barre de menu. De plus, nous avons décidé de créer une nouvelle classe appelée Database qui a pour but de gérer l'ouverture et la fermeture de la base de données.

12.2 Tests (v5)

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)
valide n°1	Chemin d'accès correspondant à une image	"C:\\cartesDisney\\carteDisney2.gif"	affichage de l'image dans l'interface
valide n°2	Catégorie de l'image est dans valeurs possibles (personne, animal ou objet)	"personne"	affichage de personne dans l'interface
valide n°3	Clic sur le menu "charger un diaporama" sans avoir de diaporama chargé au préalable	l'action actionCharger_un_diaporama" est appelé	affiche d'un message confirmant le chargement du diaporama
valide n°4	Clic sur le bouton "suivant" après avoir chargé un diaporama	le bouton bSuivant est activé	passage à l'image suivante et affiche un message dans la console confirmant le passage à l'image suivante
valide n°5	Clic sur le bouton " précédent" après avoir chargé un diaporama	le bouton bPrécédent est activé	passage à l'image précédente et affiche un message dans la console confirmant le passage à l'image précédente
valide n°6	Clic sur le bouton "lancer diaporama" après avoir chargé un diaporama	le bouton bLancer est activé	change le statut du diaporama en mode automatique et affiche un message dans la console confirmant le changement d'état
valide n°7	Clic sur le bouton "arrêter diaporama" après avoir chargé un diaporama	le bouton bArreter est activé	change le statut du diaporama en mode manuel et affiche un diaporama confirmant le

			change d’état
valide n°8	Clic sur le menu “vitesse de défilement” après avoir chargé un diaporama	l'action actionVitesse_de_defilement est appelée	affiche un message dans la console confirmant le changement de la vitesse
valide n°9	Clic sur le menu “quitter”	l’action actionQuitter est appelée	affiche un message dans la console confirmant la fermeture de la semaine et quitte la fenêtre
valide n°10	Clic sur le menu “enlever le diaporama”	l’action actionEnlever_le_diaporama	affiche un message dans la console
valide n°11	Clic sur le menu “A propos de“	l’action actionA_propos_de est appelée	affiche dans une nouvelle fenêtre les informations sur la version et les auteurs
valide n°12	Clic sur le menu “vitesse de défilement“	l’action actionVitesse_de_defilement est appelée	affiche une nouvelle fenêtre dans laquelle l’utilisateur peut entrer une vitesse en secondes
invalide n°1	Chemin d’accès ne correspondant à aucune fichier	"C: carteDisney2.gif"	Echec
invalide n°2	Chemin d’accès menant à un autre fichier d'une image	"C:\\cartesDisney\\carteDisney2.txt"	Echec
invalide n°3	La valeur de la vitesse correspondant à une chaîne de caractère	“zoo”	Echec
invalide n°4	La valeur de la vitesse est inférieur ou égale à 0	0	Echec
invalide n°5	La valeur de la vitesse est un nombre à virgule	0,5	Echec

13. Bilan

Le projet est déposé sur le dépôt Git suivant : https://github.com/julienloridant/S2.01Dev. Le temps de travail du groupe doit être d'environ 45h, même si ce temps n'est pas réparti en part égale. Cette SAÉ nous a permis de renforcer nos compétences de développement avec QT comme notamment le timer ou encore les requêtes SQL. Nos principales difficultés ont surtout été de récupérer le code d'un des membres du groupe ne respectant pas ou peu les bonnes pratiques de programmation que l'on nous enseigne depuis le début de l'année. Les points positifs de cette SAÉ sont le système de version qui est plutôt bien réparti. En effet, les étapes de chaque version sont calculées en fonction de leurs difficultés. Pour finir, le seul point négatif que nous avons trouvé est qu'il y a parfois dans le sujet n'est pas assez clair à partir de l'utilisation de la base de données. Ci dessous vous trouverez le temps de travail et les commentaires de chacun des membres du groupe sur cette SAÉ.

V0	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	A comprendre un code donné	Essayer de le comprendre le code	-	Comprendre la structure du code et comment son créateur l'a réfléchi	1h	30min	Ne pas y trouver des erreurs lors de la V2	-
Gabriel	À utiliser des classes créé sur QT et comprendre un code dont je ne suis pas l'auteur	Comprendre et compléter un code que je n'ai pas créé	-	Comprendre le fonctionnement et la structure d'un code donné	1h	1h	Avoir une V0 fonctionnelle lors du codage de la V2	Détailler un peu plus le fonctionnement de la V0 dès le début de la SAÉ
Kyllian	A optimiser un code	Comprendre le code	Le tri des images	Le tri des images	1h	1h30	L'optimisation du code	
V1	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	Designer une interface à partir d'un texte faisant office de cahier des charges	Imaginer le design de l'application	-	La penser de manière logique / ergonomique	1h	1h	L'interface aurait pu être plus jolie	-
Gabriel	Uitliser l'interface graphique de QT et créer une barre de menu	Designer l'interface de l'application	-	Créer une application simple et ergonomique	1h	1h	L'interface aurait pu être un peu plus design	-
Kyllian	-	-	-	-	-	-	-	-

V2	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	Utiliser comme il se doit de la documentation pour mieux optimiser son code	Faire le lien entre la V0 et la V1	-	Régler certaines erreurs qui au final venaient de la V0	1h30	2h	La communication de différents membres du groupes sur le signalement d'erreurs considérées comme "pas génantes" qui au final nous ont fait perdre beaucoup de temps	-
Gabriel	Adapter des méthodes d'une classe	Relier la V0 (code) avec l'interface (V1) et voir sa progression	Refaire la V0 pour cause d'imprévus	Reprendre la V0 alors qu'elle était censé être fonctionnelle	1h30	3h	Refaire la V0 par manque de communication d'un des membres de l'équipe	-
Kyllian	Lire et comprendre une documentation pour l'utiliser au mieux dans son code et comprendre comment faire le lien entre les 2 versions	comprendre les erreurs présentes et comment les corriger	-	-	15min	30min	L'optimisation du code	

V3	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	Comment gérer un objet de classe Timer	Concevoir le Timer	-	La gestion de l'objet en lui-même	1h	30min	-	-
Gabriel	Créer et uitliser un timer	Utiliser un objet Timer	-	Arrêter le mode automatique	1h	2h	-	-
Kyllian	utilisation du Qtimer	comprendre le Qtimer	-		45min	1h	Optimisation du temps et du code	

V4	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	Comment mieux structurer son code pour les autres	Plus ou moins tout	-	Bien s'approprier le fonctionnement d'une boite de dialogue	30min	2h	Notre répartition des tâches	-
Gabriel	Utiliser une variable saisie par l'utilisateur dans un label	Créer une fenêtre de dialogue	-	Faire fonctionner une fenêtre de dialogue	30min	1h	-	-
Kyllian	comment charger des images en QT	comprendre comment charger un diaporama		arriver a charger un diaporama	15min	20min		

V5	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	interroger une base de données	rééfléchir sur la construction des requetes et leur application en c++	-	Essayer de comprendre un code qui n'est pas le sien	30min	1h	La répartition du temps de travail dans le groupe	-
Gabriel	A récupérer les données d'une base de données	Utiliser la classe QSqlQuery	-	Reprendre un code mal pensé à la base	1h	2h30min	Avoir plus de communication dans le groupe durant la phase de conception	Être plus explicite sur le contenu de la V5 et plus particulièrement sur la possibilité de choisir un diaporama ou non dans une fenêtre de dialogue.
Kyllian	Utiliser QT pour interagir avec une base de données			La correction de divers soucis liés aux requêtes	10min	45min	L'optimisation des requetes / code	

V6	**ce qu'on a appris**	ce qu'on a aimé faire	ce qu'on a pas aimé faire	ce qui a été difficile	temps passé conception	temps passé code	ce qui aurait pu être mieux	ce qui pourrait être amélioré dans la SAÉ
Julien	remanier un code déjà réfléchi afin d'y intégrer un classe nouvelle	Revoir le code et ne plus le penser de la même manière	-	Repenser une grande parti du code	45min	3h (pour l'instant)	-	
Gabriel	Créer une fenêtre de dialogue pour la sélection des diaporamas	Rélfléchir à la conception d'une nouvelle classe	-	Implémenter un classe mais que le code reste fonctionnel	45min	1h	L'agencement de la fenêtre et la manière dont on choisit le numéro du diaporama	-
Kyllian								