Université de Pau et des Pays de l'Adour - IUT DE BAYONNE - PAYS-BASQUE année 2022-2023

Département Informatique saé 20.1

Saé 2.01 – Développement d’une application

Lecteur de diaporamas – Dossier d’Analyse et conception

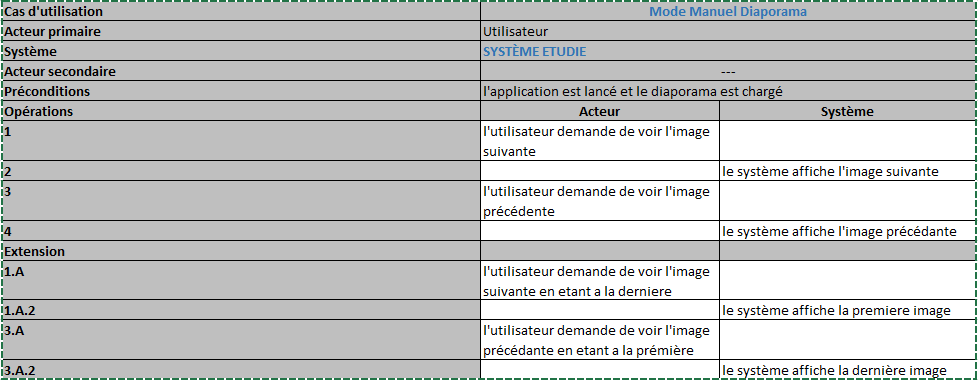
## **1.** **Compléments de spécifications externes.**

Aucun point flou n’a été à déclarer, le sujet est clair et compréhensif.

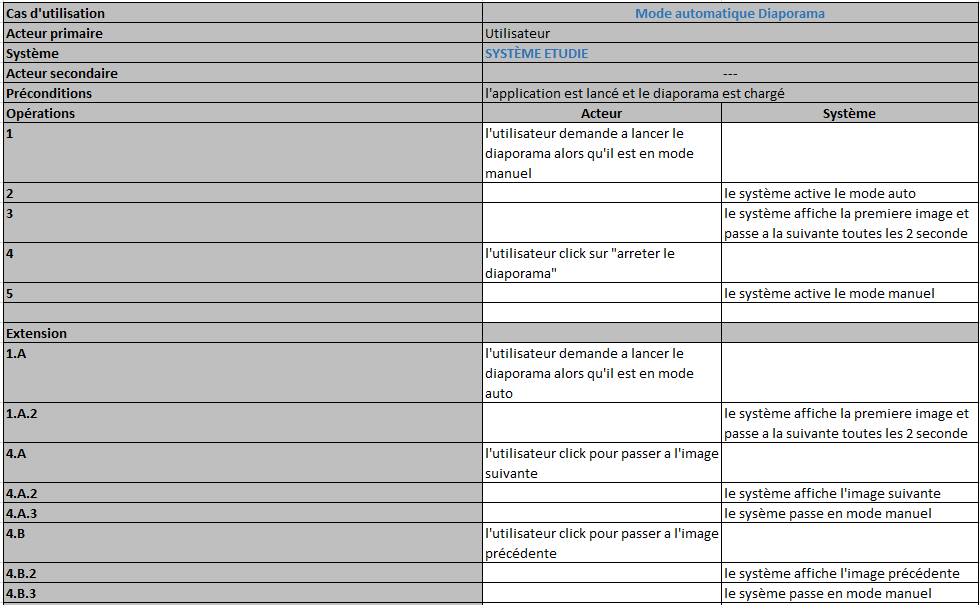
## 

## **2.** **Scénarios**

*Description du scénario nominal et de un / deux scénarios alternatifs afin de mettre en évidence les interactions entre le système et l’utilisateur*

Scénario nominal numéro 1:

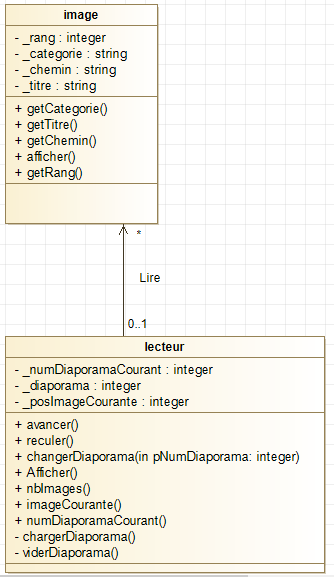
Scénario nominal numéro 2:



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

**(a)** Le diagramme de classes UML se focalise sur les classes **métier**, cad celles décrivant les éléments structurants de l’application, indépendamment des éléments d’interface.

Voici le diagramme de classe (UML) de la version V0 de notre application.



**(b)** Dictionnaire des éléments pour chaque classe

| **Classe image** | | | |
| --- | --- | --- | --- |
| **Nom attribut** | **Signification** | **Type** | **Exemple** |
| \_rang | Rang de l'image au sein du diaporama auquel l'image est associée | unsigned int | 1 |
| \_titre | Intitulé de l'image | String |  |
| \_categorie | Catégorie de l'image | String | personne |
| \_chemin | Chemin complet vers le dossier où se trouve l'image | String |  |
| **Nom Sous programme** | **Signification** | **Type** |  |
| Image(unsigned int pRang=0, string pCategorie="", string pTitre="", string pChemin = "") | Constructeur par défaut de la classe image | Image |  |
| getRang() | Renvoie le rang de l’image |  |  |
| getCategorie() | Renvoie la catégorie de l’image | String |  |
| getTitre() | Renvoie le titre de l’image | String |  |
| getChemin() | Renvoie le chemin de l’image dans les dossiers | String |  |
| afficher() | Affiche tous les champs de l'image | void |  |

| **Classe lecteur** | | | |
| --- | --- | --- | --- |
| **Nom attribut** | **Signification** | **Type** | **Exemple** |
| \_numDiaporammaCourant | Numéro du diaporama courant, par défaut 0 | unsigned | 1 |
| \_diaporama | Pointeurs vers les images du diaporama | Diaporama | 1 |
| \_poslmageCourant | Position, dans le diaporama,  de l'image courante. Indéfini quand diaporama vide. Démarre à 0 quand diaporama non vide | unsigned int |  |
| **Nom Sous programme** | **Signification** | **Type** |  |
| avancer() | Procédure afin de passer à l’image suivante dans le diaporama | void |  |
| reculer() | Procédure afin de passer à l’image précédente dans le diaporama | void |  |
| changerDiaporama() | Procédure afin de choisir un diaporama et de le charger dans le programme | void |  |
| Afficher() | Procédure afin d'afficher les informations sur lecteur-diaporama et image courante | void |  |
| nbImage() | affiche la taille de \_diaporama | unsigned short int |  |
| imageCourant() | retourne le pointeur vers l'image courante | Image\* |  |
| numDiaporamaCourant() | retourne le nombre de diaporama que le programme peut charger | unsigned short int |  |
| chargerDiaporama() | charge dans \_diaporama les images du \_numDiaporamaCourant | void |  |
| viderDiaporama() | Vide \_diaporama de tous ses objets image et les delete | void |  |

Tableau 2 : Dictionnaire des éléments - Classe xxx

**(c)** Dictionnaire des méthodes : vous pouvez fournir directement le fichier entête de chaque classe.

Exemple (classe lecteur de la version Console) :

#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 XXX

**(d)** Remarques concernant le schéma de classes

1. On ne s’intéresse qu’aux attributs et méthodes métier. Notamment, on ne met pas, pour l’instant, ce qui relève de l’affichage car ce sont d’autres objets du programme (widgets) qui se chargeront de l’affichage. Par contre, on n’oublie pas les méthodes getXXX(), qui permettront aux objets métier de communiquer leur valeur aux objets graphiques pour que ceux-ci s’affichent.

2. On n’a mis ni le constructeur ni le destructeur, pour alléger le schéma.

3. D’autres attributs et méthodes pourront venir ultérieurement compléter cette première vision ANALYTIQUE de l’application. Il s’agira des attributs et méthodes dits DE CONCEPTION nécessaires au développement de l’application.

### 

# **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 | Tester les méthodes de la classe Lecteur |

### 

### 

## **4.2 Test**

### 

Test avec le programme fournit main.cpp

*Valeurs fournies / attendues… comme montré dans la ressource R2.03 (partie tests)*

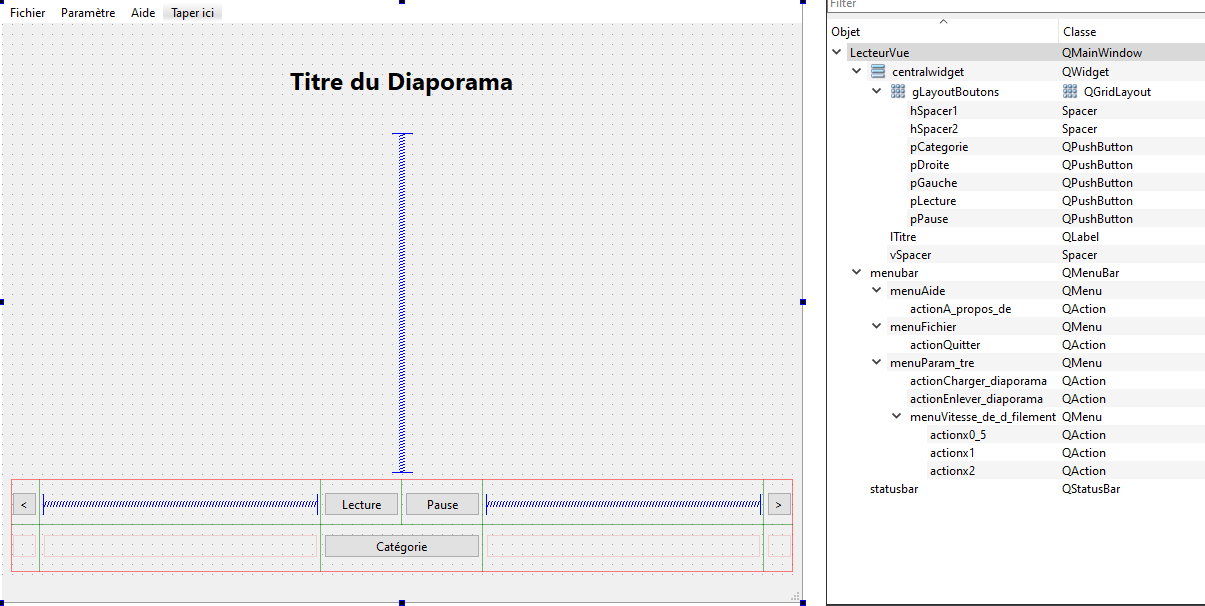
### 

# **Version v1 – projet Graphique seul**

## **5.** **Éléments d’interface**

*A faire ici : description sommaire des éléments de l’interface, par exemple, avec une copie d’écran sur laquelle sont nommés les variables/objets graphiques et où les layouts sont positionnés et nommés.*

*Vérifier que tous les éléments graphiques qui seront manipulés par l’application ont des noms pertinents et bien formés.*



## **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 | Tester les méthodes de la classe Lecteur |

### 

### Remarques sur l’implémentation

### Sur cette version aucun bouton est lié à un signal, dû au fait que la classe lecteur ne n’est que la représentation graphique sur la fenêtre des boutons. Il n’y donc pas de SLOTS et de SIGNAL.

### 

### **6.2 Test**

### A faire :

### Décrire les tests prévus / réalisés pour montrer :

### - Le comportement de l’interface non lié aux aspects fonctionnels du programme

### - Le comportement de l’interface liée aux aspects fonctionnels du

### 

### Remarque :

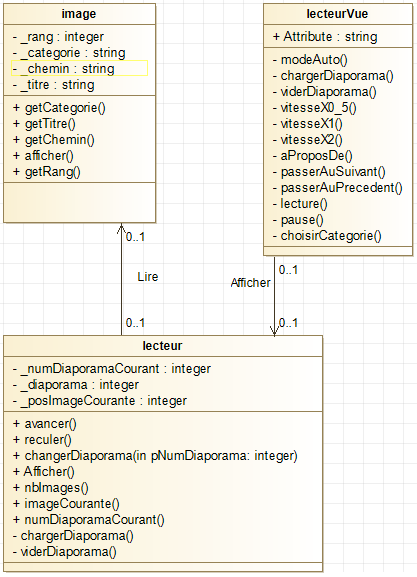
Il n’y pas de donnée rentrant c’est pourquoi nous ne les avons pas représentés, de même pour les deuxièmes résultats que nous avons décidé de ne pas représenter lorsqu’ils étaient absents.

# **Version v2 –**

## 

## **7.** **Diagramme de classes (UML)**

*A faire – s’il y a des changements - sinon indiquer que idem v0*

**

## 

## 

## **8.** **Comportement de l’application**

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

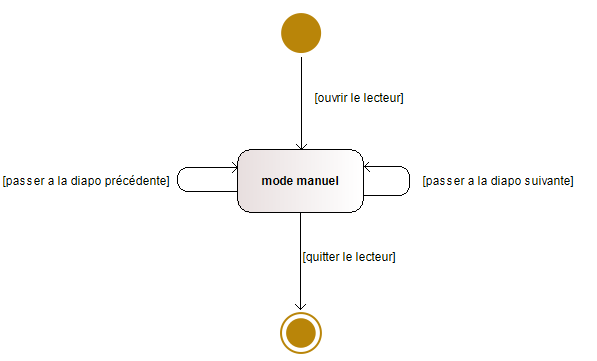


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v2

### **7.2** **Dictionnaire des états, événements et Actions (v2)**

**Dictionnaire** **des états du diaporama**

| *nomEtat* | *Signification* |
| --- | --- |
| mode manuel | Le diaporama est en mode manuel (l’utilisateur passe les images manuellement) |

Tableau 2 : États du lecteur de diaporamas – v2

**Dictionnaire des événements faisant changer le diaporama d’état**

| *nomEvénement* | *Signification* |
| --- | --- |
| ouvrir le lecteur | permet de charger le diaporama et donc d’initialiser automatiquement à son premier état qui est en mode manuelle |
| Quitter le diapo | Le programme quitte le diaporama |

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

**Description des actions réalisées lors de la traversée des transitions**

| *nomAction* | *Signification* |  |
| --- | --- | --- |
| *passer à la diapo suivant* | le diaporama passe à l’image suivante | |
| *passer à la diapo précédant* | le diaporama passe à l’image précédente | |

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 pregnant en charge cet événement à* |  |  |  |
| --- | --- | --- | --- |
| *Événement à nomEtat* |  |  |  |
| *passer à la diapo suivant* | l’utilisateur clique sur le bouton représenté par une flèche droite. |  |  |
| *passer à la diapo précédant* | l’utilisateur clique sur le bouton représenté par une flèche gauche. |  |  |
| *Quitter le diapo* | l’utilisateur clique sur Fichier puis sur quitter. |  |  |

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

*L’intérêt de cette vue matricielle est qu’elle permet une préparation naturelle et aisée de l’étape suivante de programmation.*

## **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. Permet d’afficher les attributs de la classe lecteur (diaporama) |
| --- | --- |
| 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. Déclare les procédures et les fonctions qui permettront de gérer l’affichage et des boutons. |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Spécification de la classe Image. qui permet de gérer les informations d’une image pour les utiliser dans d'autres sous-programmes. |
| image.cpp | Corps de la classe Image |
| main.cpp | Permet d'exécuter le programme et d’afficher les fenêtres |

### 

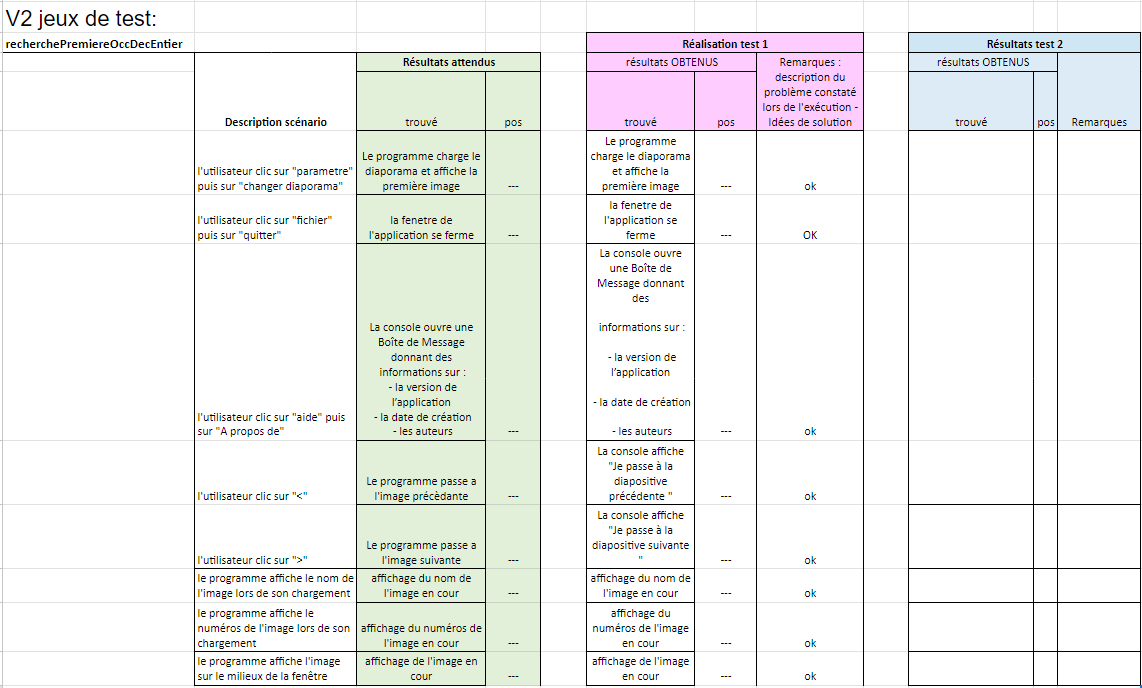
### Remarques sur l’implémentation

Tous les slots sont géré par la classe lecteurVue. Ce qui permet de réunions la gestion de tous les signaux des boutons.

### 

### 

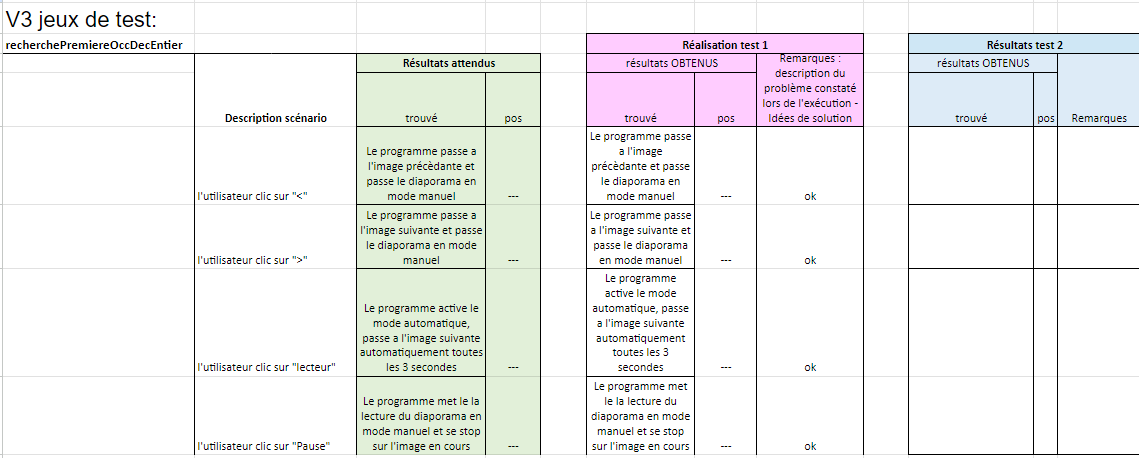
### **8.2** **Tests (v2)**

Remarque:

Il n’y pas de donnée rentrant c’est pourquoi nous ne les avons pas représentés, de même pour les deuxièmes résultats que nous avons décidé de ne pas représenter lorsqu’ils étaient absents.

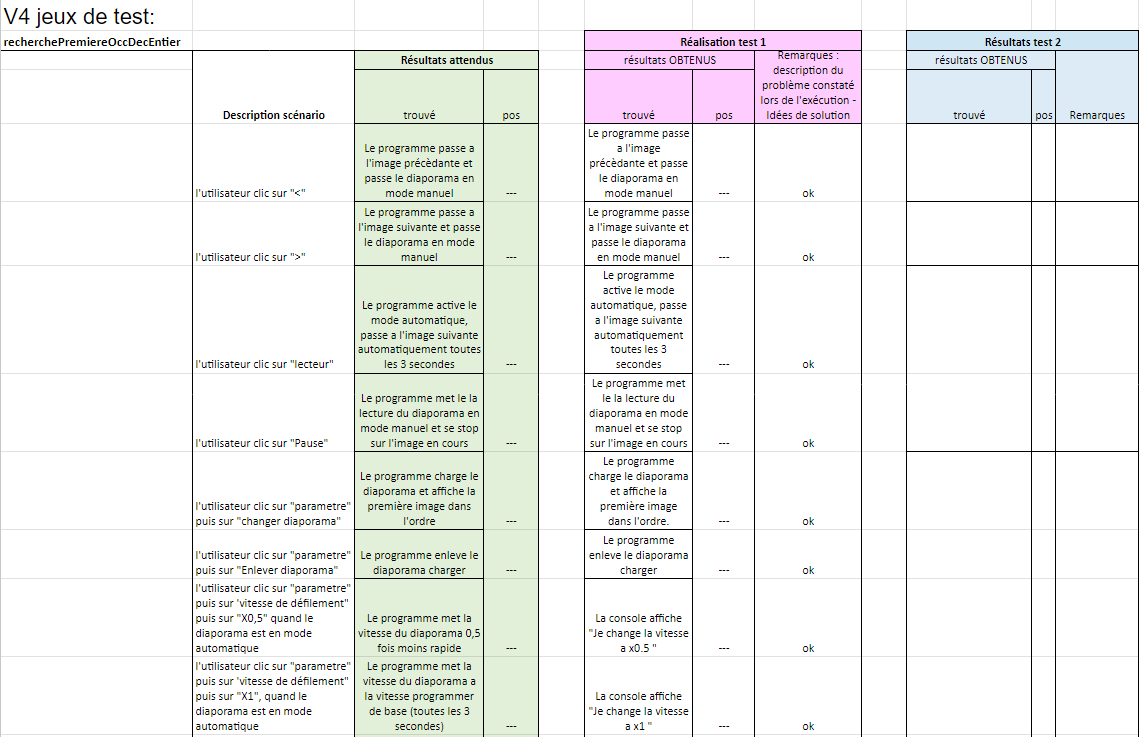
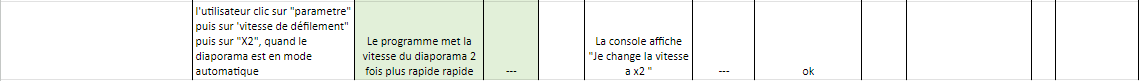
**Version v3–**

jeux de test :



**Version v4–**

jeux de test :

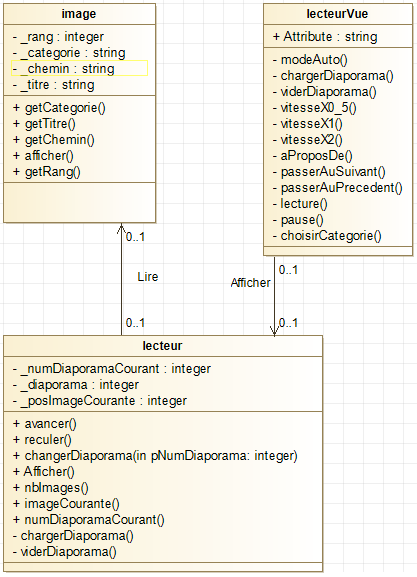


# **Version v5 –**

## 

## **10.** **Diagramme de classes (UML)**

Le diagramme de classe est le même qu’à la version précédente.

**

## 

## **11.** **Comportement de l’application**

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

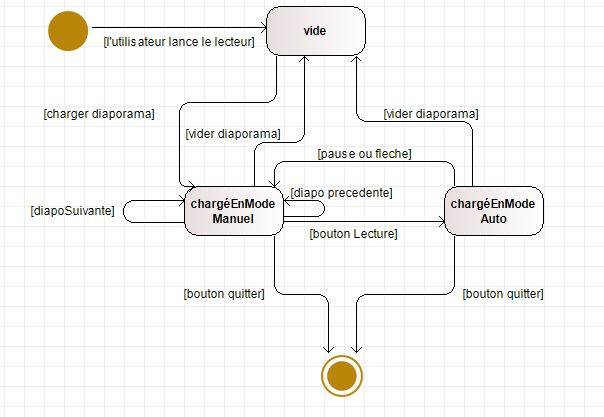
**

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* |
| --- | --- |
| chargéEnModeManuel | le lecteur a un diaporama a lire et l' utilisateur doit passer les diapositives lui même |
| chargéEnModeAuto | le lecteur a un diaporama a lire et les diapositives se passent toutes seules |
| vide | le lecteur n’a pas de diaporama a lire |

Tableau 2 : États du lecteur de diaporamas – v5

**Dictionnaire des événements faisant changer le diaporama d’état**

| *nomEvénement* | *Signification* |
| --- | --- |
| lancer le lecteur | permet de lancer dans son état vide |
| bouton quitter | permet de fermer le lecteur |
| bouton lecture | met le diapo en mode automatique |
| diapo suivante | passe à la diapositive suivante en mode manuel et fait basculer en mode manuel si en mode auto |
| diapo précédente | passe à la diapositive précédente en mode manuel et fait basculer en mode manuel si en mode auto |
| vider diaporama | permet de vider le lecteur lorsqu’il est charger |
| bouton pause | permet d'arrêter le mode auto |
| charger diaporama | permet de charger un diaporama pour que le lecteur le lise |

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

**Description des actions réalisées lors de la traversée des transitions**

| *nomAction* | *Signification* |  |
| --- | --- | --- |
| bouton lecture | le mode auto se lance, les diapos défilent automatiquement |  |
| diapo suivante | le lecteur passe en mode manuel et la diapositive suivante s’affiche à l'écran |  |
| diapo précédente | le lecteur passe en mode manuel et la diapositive précédente s'affiche à l'écran |  |
| vider diaporama | le lecteur se vide et les boutons deviennent cliquables |  |
| bouton pause | le mode auto s'arrête et la diapositive actuelle reste à l'écran |  |
| charger diaporama | le lecteur se rempli et les boutons deviennent cliquables |  |

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

| *Événement à*  *nomEtat* | lancer le lecteur | bouton quitter | bouton pause | bouton lecture | diapo précédente/suivante | vider diaporama | charger diaporama |
| --- | --- | --- | --- | --- | --- | --- | --- |
| chargéEnModeManuel | x | etat final | x | chargéEnModeManuel | x | vide | x |
| chargéEnModeAuto | x | etat final | chargéEnModeManuel | x | chargéEnModeManuel | vide | x |
| vide | x | etat final | x | x | x | x | chargéEnModeManuel |

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

*L’intérêt de cette vue matricielle est qu’elle permet une préparation naturelle et aisée de l’étape suivante de programmation.*

## **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 diaporamasPréciser le rôle |
| --- | --- |
| lecteurVue.cpp | Corps de la classe LecteurVue |
| lecteurvue.ui | Fichier du dessin de l’interface réalisé par QtDesigner |
| lecteur.h | Spécification de la classe LecteurPréciser le rôle |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Spécification de la classe ImagePréciser le rôle |
| image.cpp | Corps de la classe Image |
| main.cpp | ?? |

### 

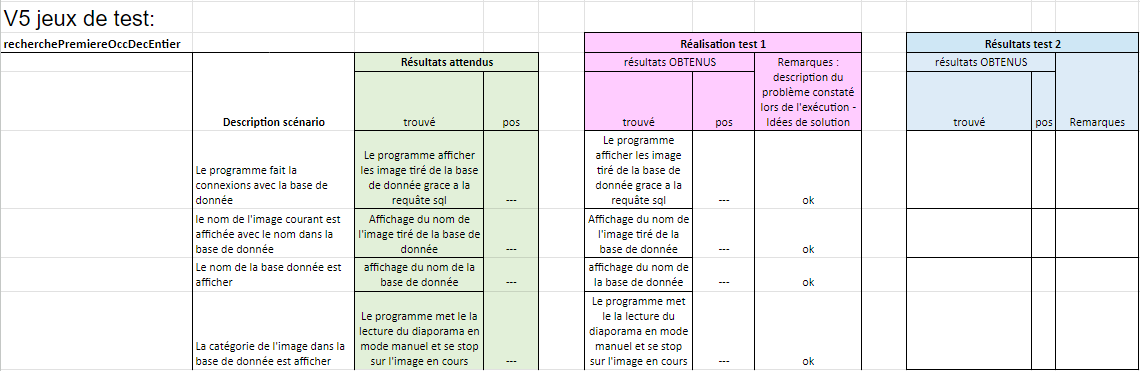
### Remarques sur l’implémentation :

### Commenter brièvement les choix importants d’implémentation réalisés, **comme par exemple, les signals/slots**

### 

### 

### **12.2** **Tests (v5)**

**Version v6–**

jeux de test :

### 

## **13.** **Bilan**

Dépôt Git où trouver le projet complet (les versions réalisées)

Temps global de travail (pour le groupe)

Nous avons réalisé un travail équitable sur l’ensemble du groupe, nous comptons une vingtaine d’heure pour chaque membre du groupe.

Apprentissages majeurs

Le développement d’application ne peut se faire sans pratiquer réellement, aller à la rencontre de tous les petits problème que l’utilisation d’une base de donnée amène est essentielle

Difficultés majeures

La connexion et l’utilisation de la base de données a été plutôt difficile et la compréhension générale de chaque version nous a couté plutôt cher.

Points positifs / négatifs de l’activité

Activité n’est pas fini