SAÉ 2.01 – Développement d'une application

Lecteur de diaporamas – Dossier d'Analyse et conception

1. **Compléments de spécifications externes.**

Le sujet est complet. Nous n'avons pas de spécifications externes.

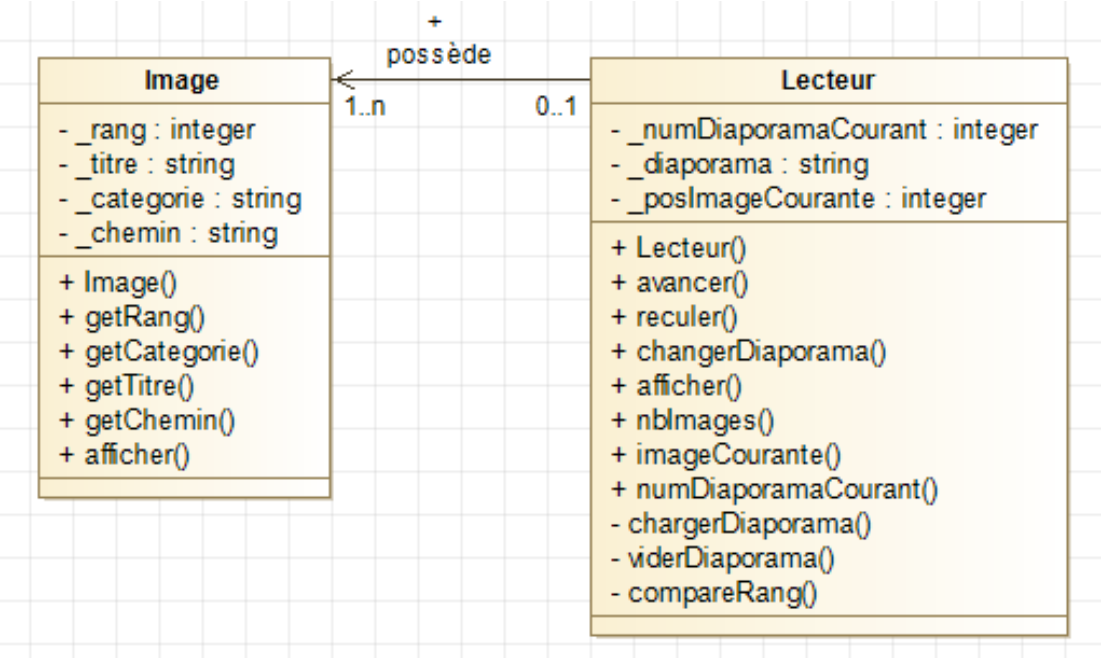
2. **Scénarios**

Sommaire	
Titre : Parcourir un diaporama	
Acteur : Internaute	
Résumé : L'internaute souhaite choisir un diapo, puis le parcourir	
Métadonnées :	
Création : 03/05/2023	**Responsable : Kévin BÉGUINEL**
Pré-conditions : L'internaute vient de démarrer le lecteur de diaporama et n'a pas encore agi dessus	

Acteur : Internaute	**Système : LecteurDiaporama**
1. L'internaute demande à choisir un diaporama	
	2. Le système affiche la liste des diaporamas disponibles
3. L'internaute choisit un diaporama	
	4. Le système affiche les infos correspondant à la 1ère image du diapo choisi
5. L'internaute demande à passer à l'image suivante	
	6. Le système affiche les infos correspondant à l'image suivante
7. L'internaute demande à retourner à l'image précédente	
	8. Le système affiche les infos correspondant à l'image précédente
9. L'internaute demande à enlever le diaporama actuel	
	10. L'internaute enlève le diapo
11. L'internaute demande à quitter l'application	
	12. Le système ferme l'application
Enchaînement Alternatif	
A1 L'internaute regarde le diapo en mode automatique	
1. L'internaute demande à passer en mode automatique	
	2. Le système change le mode de défilement
3. L'internaute demande à changer la vitesse de défilement	
	4. Le système affiche le choix des vitesses de défilement
5. L'internaute choisit une vitesse et la valide	
	6. Le système enregistre la nouvelle vitesse de défilement
7. L'internaute demande à lancer le diaporama	
	8. Le système fait défiler les images du diapo
9. L'internaute demande à arrêter le diapo	
	10. Le système arrête le diapo
11. L'internaute demande à quitter l'application	
	12. Le système ferme l'application

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.



(b) Dictionnaire des éléments pour chaque classe

Classe Image			
Nom attribut	**Signification**	**Type**	**Exemple**
`_rang`	rang de l'image au sein du diaporama	`unsigned int`	`1`
`_titre`	intitulé de l'image	`string`	`« Disney »`
`_categorie`	catégorie de l'image (personne, animal, objet)	`string`	`« personne »`
`_chemin`	chemin complet vers le dossier où se trouve l'image	`string`	`« F:/travaux-pratiques/IUT\S2/SAE/S2_01- » « « « « « «« « « Dev_app\cartesDisney »`

(c)

Classe Lecteur			
Nom attribut	**Signification**	**Type**	**Exemple**
`_numDiaporamaCourant`	numéro du diaporama courant, par défaut 0	`unsigned int`	`0`
`_diaporama`	pointeur vers les images du diaporama	`Diaporama`	`(2, "personne", "Cendrillon", "C:\\cartesDisney\\carteDisney4.gif")`
`_posImageCourante`	Position, dans le diaporama, de l'image courante. Indéfini quand diaporama vide. Démarre à 0 quand diaporama non vide	`unsigned int`	`0`

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

Classe Image :

```
#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
```

Classe Lecteur :

```
#ifndef LECTEUR_H
#define LECTEUR_H
#include "image.h"
#include <vector>
#include <iostream>
using namespace std;

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 int _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
    bool compareRang(Image, Image);
};

#endif // LECTEUR_H
```

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

Valeurs attendues :

```
Lecteur vide
Diaporama num. 1 selectionne.
4 images chargees dans le diaporama
Diaporama num. 1
image courante : image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test avancer() : 4 fois
avancer() :
Diaporama num. 1
image courante : image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)
avancer() :
Diaporama num. 1
image courante : image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)
avancer() :
Diaporama num. 1
image courante : image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)
avancer() :
Diaporama num. 1
image courante : image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test reculer() : 5 fois
reculer() :
Diaporama num. 1
image courante : image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)
reculer() :
Diaporama num. 1
image courante : image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)
reculer() :
Diaporama num. 1
image courante : image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)
reculer() :
Diaporama num. 1
image courante : image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)
reculer() :
Diaporama num. 1
image courante : image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

Enlever le diaporama courant = Choisir diaporama 0
0 images restantes dans le diaporama.
Lecteur vide
```

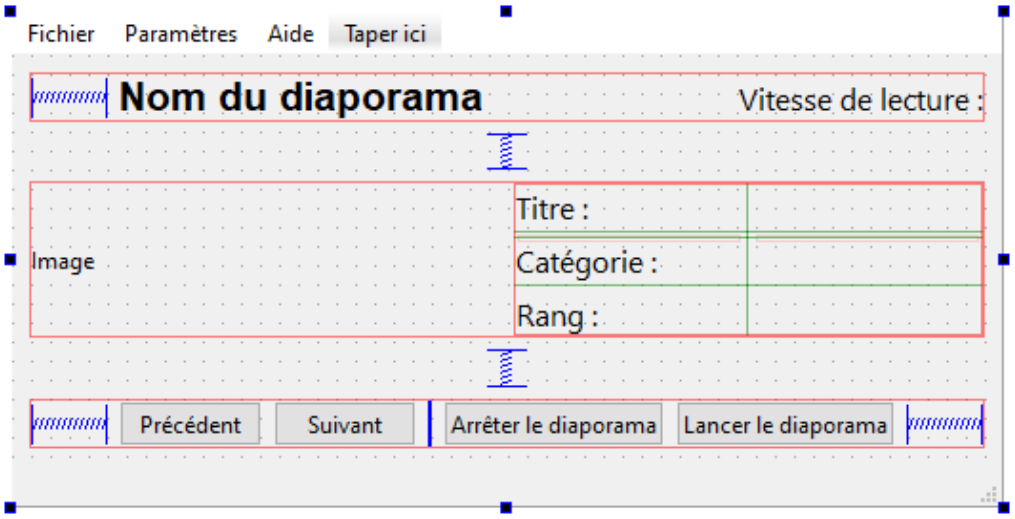
Valeurs obtenues :

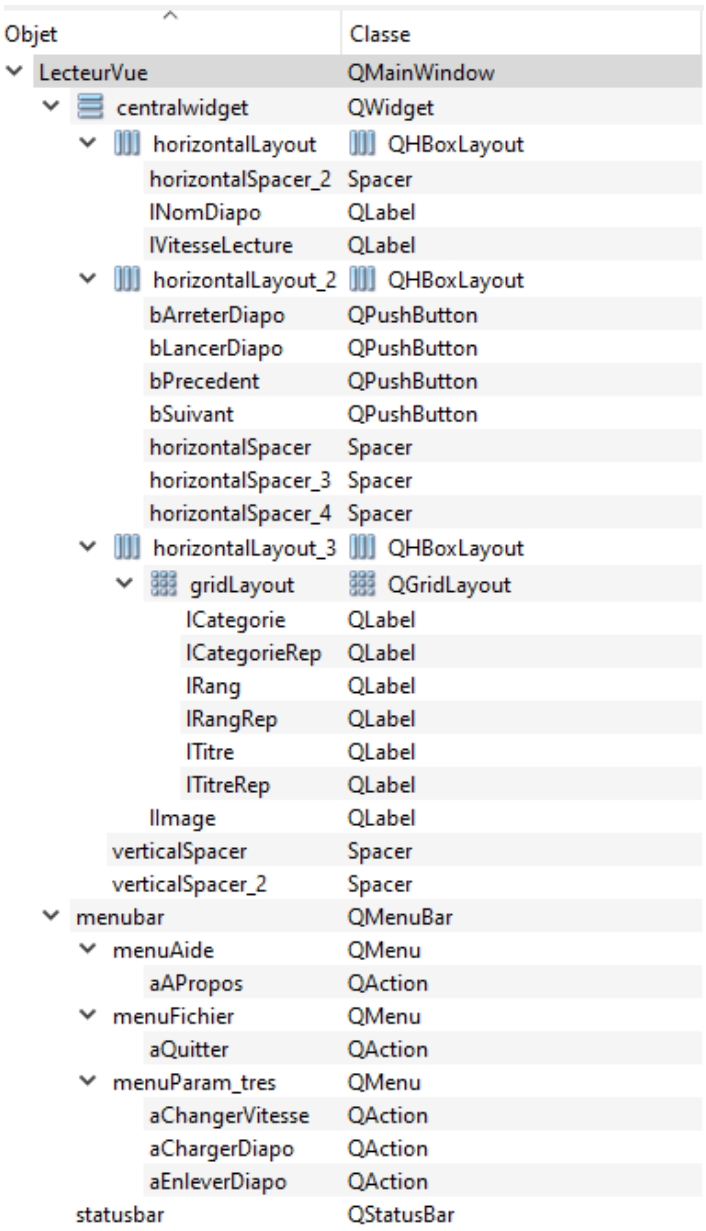
```
lecteur vide
Diaporama num. 1 selectionne.
4 images chargees dans le diaporama
Numero du diaporama : 1
Image courante :
image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test avancer() : 4 fois
avancer() :
Numero du diaporama : 1
Image courante :
image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)
avancer() :
Numero du diaporama : 1
Image courante :
image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)
avancer() :
Numero du diaporama : 1
Image courante :
image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)
avancer() :
Numero du diaporama : 1
Image courante :
image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test reculer() : 5 fois
reculer() :
Numero du diaporama : 1
Image courante :
image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)
reculer() :
Numero du diaporama : 1
Image courante :
image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)
reculer() :
Numero du diaporama : 1
Image courante :
image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)
reculer() :
Numero du diaporama : 1
Image courante :
image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)
reculer() :
Numero du diaporama : 1
Image courante :
image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

Enlever le diaporama courant = Choisir diaporama 0
0 images restantes dans le diaporama.
lecteur vide
```

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	Teste les méthodes de la classe Lecteur

Remarques sur l'implémentation :

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

Nous avons choisi de créer un slot par interaction possible avec le lecteur diaporama (voir photo ci-dessous des slots de la classe LecteurVue).

```
public slots:
    void lancerDiapo();
    void arreterDiapo();
    void suivant();
    void precedent();
    void changerVitesse();
    void chargerDiapo();
    void enleverDiapo();
    void aPropos();
    void quitter();
```

Nous avons ensuite connecté les éléments grâce au code ci-dessous. La fonction clicked() permet d'utiliser les boutons tandis que la fonction triggered() permet d'utiliser les différentes fonctionnalités des menus.

```
QCoreApplication * instapp = QCoreApplication::instance();

QObject::connect(ui->bLancerDiapo, SIGNAL(clicked()), this, SLOT(lancerDiapo()));
QObject::connect(ui->bArreterDiapo, SIGNAL(clicked()), this, SLOT(arreterDiapo()));
QObject::connect(ui->bSuivant, SIGNAL(clicked()), this, SLOT(suivant()));
QObject::connect(ui->bPrecedent, SIGNAL(clicked()), this, SLOT(precedent()));
QObject::connect(ui->aChangerVitesse, SIGNAL(triggered()), this, SLOT(changerVitesse()));
QObject::connect(ui->aChargerDiapo, SIGNAL(triggered()), this, SLOT(chargerDiapo()));
QObject::connect(ui->aEnleverDiapo, SIGNAL(triggered()), this, SLOT(enleverDiapo()));
QObject::connect(ui->aAPropos, SIGNAL(triggered()), this, SLOT(aPropos()));
QObject::connect(ui->aQuitter, SIGNAL(triggered()), instapp, SLOT(quit()));
```

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 programme

Afin de tester nos connexions signals/slots, nous avons utilisé la fonction qDebug() dans chaque slot. Chaque slot affiche donc un message dans la console, ce qui nous permet de vérifier que le comportement des aspects fonctionnels du programme fonctionne correctement.

Nous avons ensuite testé l'affichage de notre lecteur de diaporama. Nous avons pour cela inséré des spacers afin que notre programme soit responsive si l'utilisateur décide de modifier les dimensions de notre page.

Version v2 –

7. Diagramme de classes (UML)

8. Comportement de l'application

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

A faire

Figure 1 : Diagramme états-transitions du lecteur de diaporamas – v2

7.2 Dictionnaire des états, événements et Actions (v2)

Dictionnaire des états du diaporama

nomEtat	*Signification*

Tableau 1 : États du lecteur de diaporamas – v2

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*

Tableau 2 : Evénements faisant changer le diaporama d'état – v2

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*

Tableau 3 : 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			
Événement *nomEtat*			

Tableau 4 : 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 *Préciser 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 Lecteur. *Préciser le rôle*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Préciser 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***

8.2 Tests (v2)

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 programme

- Le comportement fonctionnel de l'application

Version v5 –

10. Diagramme de classes (UML)

A faire – s'il y a des changements - sinon indiquer que idem vXX

11. Comportement de l'application

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

A faire

Figure 2 : Diagramme états-transitions du lecteur de diaporamas – v5

11.2 Dictionnaire des états, événements et Actions (v5)

Dictionnaire des états du diaporama

nomEtat	*Signification*

Tableau 5 : États du lecteur de diaporamas – v5

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*

Tableau 6 : Evénements faisant changer le diaporama d'état – v5

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*

Tableau 7 : 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 pregnant en charge cet événement			
Événement *nomEtat*			

Tableau 8 : 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 diaporamas *Préciser 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 Lecteur *Préciser le rôle*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Préciser 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)

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 programme

- Le comportement fonctionnel de l'application

13. Bilan

Dépôt Git où trouver le projet complet (les versions réalisées)
Temps global de travail (pour le groupe)
Apprentissages majeurs
Difficultés majeures
Points positifs / négatifs de l'activité