Filière Systèmes Électroniques Embarqués

Modélisation Systèmes (S10)

EN317 Modélisation et langage système

Module assuré par Monsieur Guillaume Delbergue

Projet C++

CHOLLET Benjamin & CLAIN Adrien

L'objectif du projet est de mettre en œuvre les notions appréhendées durant l'enseignement de C++ en mettant en place une gestion informatique d'une médiathèque.

1)Diagramme de classes

Le diagramme ci-dessus permet de mettre en avant l'architecture et les interactions entre les différentes classes définies dans notre code

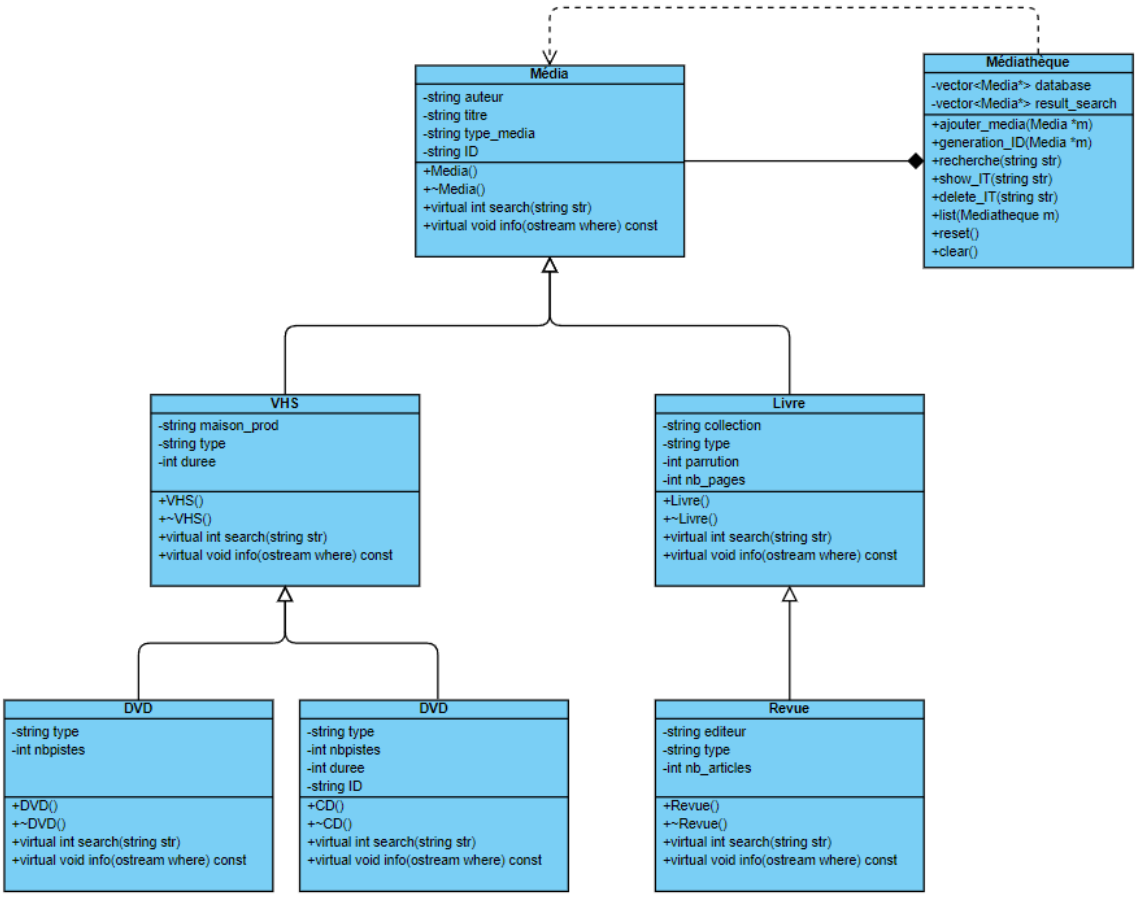

Figure 1 : Diagramme de classes

Toutes les ressources (DVD, CD, Livres, …) ont été définis comme étant des sous-classes, ou classes filles à la une classe mère, nommée Média. Cette dernière définit des attributs et méthodes communs, qui sont partagés entre toutes les classes filles, tel qu'un auteur ou un titre. De plus, afin de mettre en place le polymorphisme, la classe mère contient une méthode virtuelle qui devra être redéfinie dans les classes filles.

Une seconde classe contient tous les éléments (attributs et méthodes) permettant de gérer la médiathèque. C'est ici que nous stockons toutes les ressources dans un vector de type "Média*".

2) Mise en oeuvre d'une médiathèque

A ce jour, toutes les commandes de gestion de la médiathèque sont fonctionnelles. Cela comprend :

- l'ajout de nouvelles ressources (ADD)
- de quitter l'application (BYE)
- le lancement d'une recherche incrémentale (SEARCH)
- de lister toutes les ressources présentes dans la médiathèque (LIST)
- de nettoyer le buffer de recherche (CLEAR)
- de rechercher une ressource via son ID unique (SHOW ID)
- de supprimer une ressource via son ID (DELETE ID)
- de supprimer toutes les ressources (RESET)
- de lancer une sauvegarde dans un fichier annexe (SAVE)
- de charger des ressources à partir d'un fichier texte (LOAD)

1. Gestion de la médiathèque

Comme le montre la figure 1, le code est séparé en différentes classes, permettant de représenter succinctement le comportement lié d'un côté aux ressources, et de l'autre, leur gestion.
Ainsi on retrouve une classe Média, et ses classes filles, et de l'autre une classe Médiathèque pour gérer le fonctionnement de la médiathèque.

Ce paragraphe met en avant certaines fonctionnalités liées à la gestion des ressources dans la médiathèque.

La méthode "search" permet de retrouver, via une chaîne de caractère passée en argument, toutes les ressources qui comprennent l'information recherchée.
La recherche s'effectue sur tous les attributs propres à la ressource. Autrement dit, on peut retrouver un livre par son auteur, son titre, ou bien encore son nombre de pages, et un CD par son auteur, son titre, mais également par son nombre de pistes. Dès qu'une ressource présente une information recherchée, cette dernière est stockée dans un deuxième vecteur. Tant que la commande "clear" n'est pas envoyée, on effectuera la recherche sur ce vecteur (recherche incrémentale).
Afin de simplifier le codage, nous avons mis en place la notion de polymorphisme dans le but de généraliser la fonction à toutes les ressources.

Il est important de noter que la recherche est sensible à la casse. Autrement dit, si un livre à un titre "Harry Potter", et que la recherche lancée contient la chaîne suivante "harr", cette dernière n'aboutira pas.

Concernant la suppression de ressources, nous avons deux méthodes permettant de mettre en place ce procédé :

- “delete_ID” concerne uniquement une ressource
- “reset” pour supprimer toutes les ressources simultanément

Le principe est le même et il est décomposé selon deux axes. Le premier est de supprimer l’instance de l’objet (le vecteur stocke les adresses des instances de chaque classe) via la fonction *delete*. La seconde partie consiste à vider le vecteur (via *clear* dans le cas de la commande “reset”)

Lors de l’appel à la fonction “add”, cette dernière en plus d'ajouter dans le vecteur la ressource, lui créer un ID unique basée sur ses caractéristiques. L’ID est basé de la manière suivante :

- Première lettre du type de ressource (Ex : D pour DVD)
- Deux première lettre du titre (Ex : HA pour Harry Potter)
- Deux première lettre du nom de l’auteur (Ex : RO pour Rowling)
- Un numéros pour éviter les doublons d’ID (commence à 1 et s’incrémente si l’ID créée existe déjà)

2. Gestion des commandes

L’utilisateur doit pouvoir, par le biais du terminal, gérer les commandes qui lui sont proposées. En effet, pour rendre le code beaucoup plus lisible, nous avons décidé de venir lire dans différents fichiers textes pour les afficher sur le terminal, ceci permettant de guider l’utilisateur sur le choix des commandes, options, ressources… La lecture des données par l’utilisateur étant très sensible à la casse, nous avons donc dû prendre en compte plein de cas où l'utilisateur peut générer des erreurs :

- trop de données rentrées
- pas assez de données rentrées
- Entrée de lettre au lieux de chiffre

Nous avons donc appris à utiliser l'instruction “try catch” qui nous a grandement aidé pour la gestion des erreurs.

Nous avons beaucoup aimé l'amélioration du visuel du terminal afin de rendre le logiciel beaucoup plus instinctif et élégant.

2) Difficultés rencontrées

Ce projet à pu nous permettre de mettre en avant les différentes notions de la POO au travers du langage C++.
Au niveau de la gestion de la médiathèque, la fonction “search” nous a mis au défis en nous forçant à revoir notre conception du code, afin de prendre en compte le polymorphisme. Une fois compris, ce procédé a été généralisé à d'autres commandes telle que la fonction “list”.

De plus, le vecteur database qui stocke les ressources et un pointeur contenant les adresses des instances. Lors de la phase de suppression des instances, cette partie à généré un lot d’erreurs qui ont pu être corrigées en changeant la manière de créer nos instances.
Avant cela, nous créons un livre en appelant directement son constructeur, qui a dû être remplacé par l’utilisation du *new*.

Les notions de la POO, et plus particulièrement les notions d’attributs n’ont pas été respectées, puisque que tous les attributs et méthodes de classes sont en public. Cela avait pour rôle de nous faciliter la tâche au début, mais par manque de temps, nous les avons laissés tels quels.

D’un point de vue organisationnel, les huit heures de projets accordées préalablement n’étaient pas suffisantes afin de mettre en place un tel projet. De nombreuses d’heures (plus de ce qui étaient prévues) ont été nécessaires, en plus, de notre côté pour finaliser le projet.