TCHANDJOU NGOKO Adrien
SAFRAOUI Yannick

Rapport de projet LO21 P14

Réalisation d'un gestionnaire de dossier étudiant

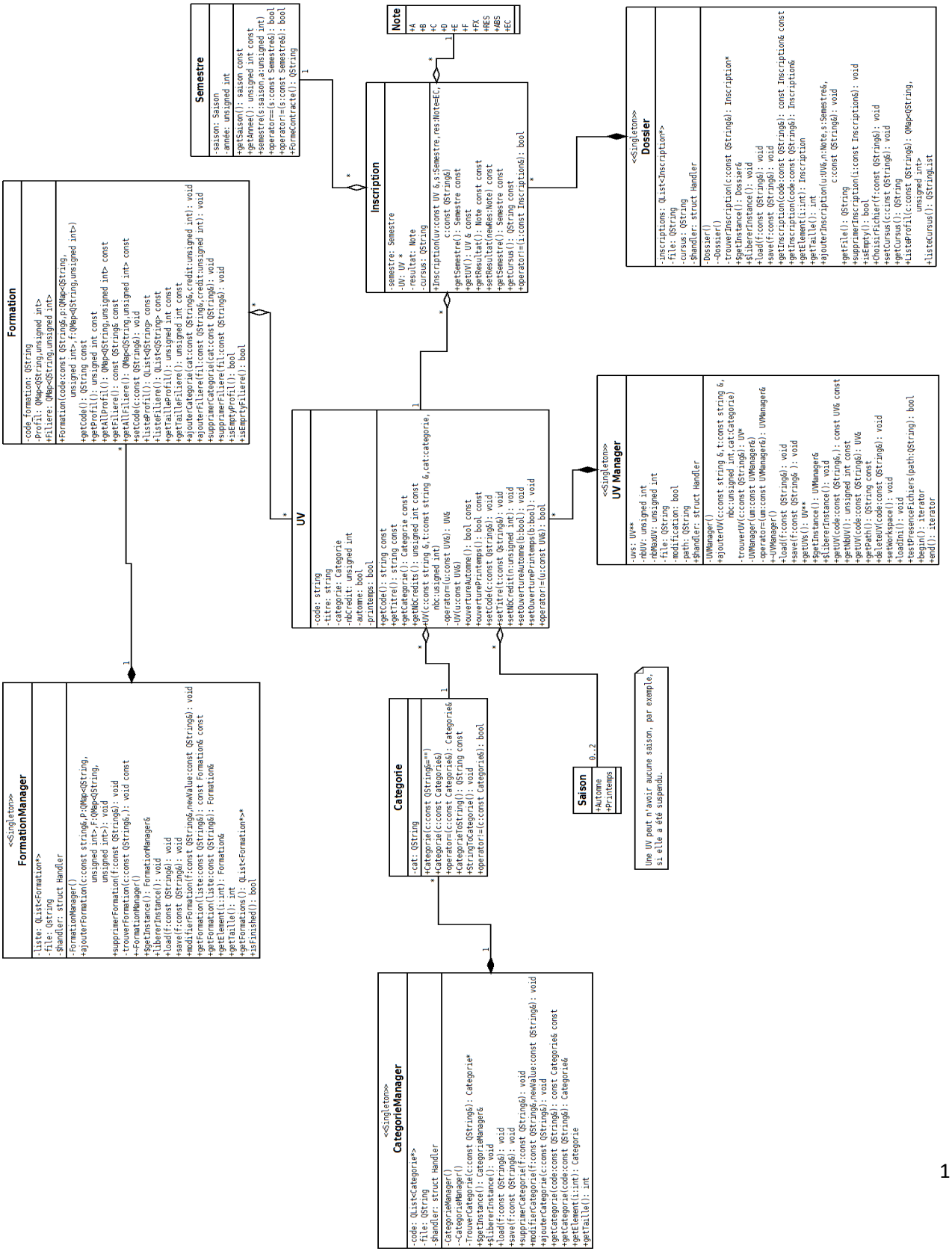
TCHANDJOU NGOKO Adrien
SAFRAOUI Yannick

Sommaire

TCHANDJOU NGOKO Adrien
SAFRAOUI Yannick

Description de l'architecture

Uml



Notre architecture répond aux 3 premières questions du projet. Elle permet de gérer des UVs, des formations, des catégories. L'utilisateur peut aussi créer son propre dossier, où seront mis les UVs auquel il s'est inscrit. Nous avons donc créé des classes qui gèrent l'ensemble de ces données.

Design pattern

Singleton

Nous avons utilisé le design pattern singleton pour toutes classes Manager, qui servent gérer une classe définie, mais aussi pour la classe Dossier, qui va gérer un dossier à la fois. Nous avons choisi, d'utiliser le singleton pour restreindre à un le nombre d'instanciation de ces classes. En effet, il n'y a aucune utilité d'en avoir plusieurs.

Iterator

Nous avons implémenté le design pattern iterator pour la classe UVManager, qui nous permettait de passer d'un élément à l'autre, et d'accéder aux données. Pour les autres classes Manager, ainsi que le dossier, nous utilisions des QList ou QMap, où l'iterator avait déjà été implémenté au préalable.

Possibilité d'amélioration

Nous aurions pu utiliser un design pattern « *template method»* pour implémenter les classes Manager, car plusieurs de leurs méthodes et variables sont les mêmes. Les grandes différences, entre ces classes sont les méthodes save et load ainsi que le type de la variable QList. Il y aurait eu un gain de temps, et moins de redondance dans le code.

Le design pattern « mémento » aurait pu nous être utile. En effet, il permet de sauvegarder restaurer un état précédent d'un objet.

TCHANDJOU NGOKO Adrien
SAFRAOUI Yannick

Argumentation sur l'évolutivité

Nous avons choisi d'utiliser des fichiers XML pour stocker les données et gérer l'évolutivité des éléments du logiciel.

Les éléments de la modélisation cités ci dessous qui sont modifiables ont chacun une classe dédiée ainsi qu'un Manager (gère les éléments de la classe) et un Editeur (fenêtre pour modifier)

Evolutivité UVs

(UV.h/.cpp UVManager.h/.cpp UVEditeur.h/.cpp)

Le programme donne la possibilité à l'utilisateur d'ajouter des UVs avec des caractéristiques particulières (nombre de crédit, catégorie, code, description, semestre). Les UVs existantes peuvent être modifiées sur l'ensemble des caractéristiques citées et sauvegardées dans UV_XML.xml. On peut également supprimer une UV. Malheureusement, il n'est pas possible q'une UV est des crédits dans plusieurs catégories.

Evolutivité Catégorie UVs

(Categorie.h/.cpp CategorieManager.h/.cpp CategorieEditeur.h/.cpp)

Les catégories que l'on peut choisir pour les UVs sont modifiables. On peut ajouter une nouvelle catégorie d'UV, supprimer une catégorie existante (CS, TM, TSH, SP). Les données sont sauvegardées dans Categorie.xml.

Evolutivité Formation

(Formation.h/.cpp FormationManager.h/.cpp FormationEditeur.h/.cpp)

L'utilisateur peut rajouter une nouvelle formation en lui donnant différentes caractéristiques (code, catégorie et nombre de crédits, filière si existante et nombre de crédits). Les formations existantes sont modifiables et supprimables. Les données sont sauvegardées dans Formation.xml .

Evolutivité Dossier

(Dossier.h/.cpp DossierEditeur.h/.cpp)
Il n'y a pas de DossierManager car on charge un dossier à la fois dans l'application.

Le dossier permet d'ajouter des inscriptions (UV, cursus, résultat, saison, année) et de les supprimer. Pour faire un nouveau dossier, le programme va créer un nouveau fichier stocké à l'endroit choisi par l'utilisateur. S'il veut supprimer le dossier, il doit supprimer le fichier XML associé. L'utilisateur sauvegarde les informations du dossier dans un fichier XML qu'il a créé au préalable et placé à un emplacement qu'il désirait. Il doit charger le dossier lui-même quand il veut ensuite le modifier.

L'affichage du profil regarde combien de cursus différents l'utilisateur a rentré et crée un tableau avec les totaux de chaque catégorie d'UV pour chaque cursus qu'il a fait. Des couleurs ont été utilisés lui permettant de savoir ce qu'il lui reste à valider.

TCHANDJOU NGOKO Adrien
SAFRAOUI Yannick

Instructions d'utilisation du logiciel

Ce programme a été codé en C++ en utilisant QT 5.2.1 et le compilateur MinGW 4.8 32 bit.

Une vidéo montre l'utilisation du logiciel (Screencast_UTProfiler_Safraoui_Tchandjou.mp4).

Pour lancer l'application, il faut bien sûr avoir ouvert et compilé le projet (UT profiler.pro).
Lors du premier lancement, le logiciel va demander à l'utilisateur de localiser les fichiers XML contenant les données. Les fichiers en question sont dans le même dossier qu' « UT profiler.pro ». La vérification de la présence des fichiers XML se fait en dur par rapport au nom des fichiers, il ne faut donc pas les renommer.

Une fois cette opération réalisée, le programme crée un fichier workspace.ini, dans le dossier « build-UT Profiler » qui contient les fichiers créé par QT à la compilation. Ce fichier .ini stocke le chemin du dossier de fichiers XML pour les charger automatiquement au prochain démarrage. Pour pouvoir redéfinir le chemin si on désire stocker les fichiers XML ailleurs, il faut supprimer workspace.ini.