LO21 - Printemps 2017 - Projet PluriNotes

Dans ce projet, il s'agit de concevoir et développer l'application PluriNotes, destinée à éditer et gérer un ensemble de *notes* (des mémos) qui peuvent correspondre à du texte ou des images. Une note peut par exemple correspondre au compte-rendu d'une réunion, à des notes prises lors d'une séance de cours, ou encore à des tâches à effectuer.

1 Description des fonctions principales

Dans ce document, sont présentées les spécifications du fonctionnement de l'application demandée. Vous pouvez les personnaliser à condition de ne pas diminuer la complexité du projet. Ces spécifications vous laissent volontairement des choix de conception, tant fonctionnels que conceptuels et technologiques. Quels que soient les choix et adaptations que vous ferez, vous prendrez garde de les exposer et justifier dans le rapport rendu avec le projet. Il peut manquer des spécifications. Dans ce cas, faites un choix en l'exposant clairement dans votre rapport.

1.1 Différents types de notes

Toute *note* est caractérisée par un *identificateur*, un *titre*, une *date de création* et une date de *dernière modification*. L'identificateur est une sorte de label qui permet de faire référence à cette note dans une autre note. Le titre est une sorte de résumé de la note qui permet d'en comprendre rapidement le contenu. Parmi les notes, on distingue les *articles*, les *taches*, les *images*, les enregistrements audio ou video :

- Les articles sont des notes comprenant un *texte*.
- Les images, les enregistrements audio, et les enregistrements video se caractérisent par une *description* et un *fichier* image.
- Les taches renvoient à une certaine *action* (décrite sous forme de texte) à effectuer. Une tâche peut avoir une certaine priorité (optionnelle). On peut aussi vouloir qu'une tâche soit effectuée avant une certaine date d'échéance (optionnelle). Une tâche a le statut « en attente », « en cours » ou « terminée ». Une tâche est initialement en attente.

1.2 Gestion des différentes versions notes

À chaque fois qu'une note est éditée, une nouvelle version de la note est créée et l'ancienne version est sauvegardée. Une partie de l'interface de l'application permet de naviguer dans les différentes versions d'une note donnée et d'en restaurer une version antérieure comme version actuelle de la note.
Mis à part son identificateur qui ne pourra jamais être édité, tous les éléments d'une note sont modifiables.

1.3 Relations entre notes

Afin d'enrichir la sémantique d'un ensemble de notes, on peut créer des relations (au sens mathématique) entre notes.
Une partie de l'interface de l'application permet de créer, enrichir, éditer ou supprimer une relation. Une relation est caractérisée par un titre, une description, et un ensemble de couples de notes. L'application permet d'ajouter ou retirer un couple. Chaque couple peut être caractérisé par un label. Ce label peut être édité.
Par défaut, une relation est orientée. Ainsi un couple (x, y) d'une relation signifie qu'il y a une relation depuis la note x vers la note y. Cependant, l'utilisateur pourra décider qu'une relation est non-orientée. Cela signifiera qu'un couple (x, y) de cette relation vaut pour une relation depuis x vers y, aussi bien que depuis y vers x.

Une relation est seulement définie entre les dernières versions des notes.
L'application doit permettre de visualiser une relation (sous une forme que vous choisirez). L'application doit aussi permettre de visualiser pour une note choisie, l'arborescence des ascendants et l'arborescence des descendants de la note dans l'ensemble des relations existantes de manière simultanée.
Il existe une relation orientée spéciale « préexistente » appelée *Référence* qui ne peut pas être supprimée. Cette relation permet de matérialiser les références qu'une note peut faire à d'autres notes grâce à une syntaxe spéciale. Ainsi, dès que dans tout champ spécifique d'une note x pouvant être assimilé à un texte écrit (le titre, la description, le texte, l'action, ...), on retrouve le texte `\ref{idy}`, où `idy` est l'identificateur d'une note `y`, cela signifie que le couple `(x,y)` est ajouté à la relation *Référence*.

1.4 Suppression d'une note

L'utilisateur peut supprimer une note et l'ensemble de ses versions s'il le souhaite. La supression d'une note entraîne l'élimination de tous les couples impliquant la note suprimée dans l'ensemble des relations existantes.
Ceci ne concerne pas la relation *Référence*. Si une note est « référencée » par au moins une autre note, elle ne peut être suprimée (car cela implique la modification des textes usant de cette référence). Au lieu d'être supprimée, la note est *archivée*. Cela signifie, qu'elle est toujours visualisable (y compris ses versions antérieures) et qu'elle peut toujours être restaurée et redevenir *active*. Cependant, tant qu'une note est archivée, elle n'est plus éditable. Les couples des relations existants vers ou depuis une note archivée ne sont pas éliminés.
Lorsqu'après diverses actions de l'utilisateur, la dernière référence (depuis une note active ou archivée) vers une note archivée x a été éliminée, l'application propose à l'utilisateur d'éliminer définitivement la note x. Si après diverses actions de l'utilisateur, les seules références qui existent vers un ensemble A de notes archivées ont leur origine depuis l'une des notes de cet ensemble A, alors l'application propose d'éliminer les notes de l'ensemble A.
Lorsque la supression d'une note est demandée et que cette supression est possible (c'est à dire ne nécessitant pas un archivage), elle est placée dans une corbeille en attente d'une éventuelle ultime restauration. Elle est donc considérée aussi comme une sorte de note archivée mais en sursis. L'utilisateur peut demander le vidage de la corbeille provoquant l'élimination définitive de cette note. Le vidage de la corbeille est systématiquement proposé lors de la sortie de l'application sauf s'il a été décidé grâce à un paramétrage de l'application que ce vidage était fait automatiquement.

1.5 Éléments d'interface

- Dans sa vue principale, la partie gauche de l'application possède une partie dédiée à l'affichage de l'ensemble des notes actives, une partie dédiée à l'affichage ergonomiques (tenant compte des priorités et des dates échues) des tâches, et une autre partie (plus discrète) à l'ensemble des notes archivées.
- Dans sa vue principale, la partie centrale de l'application permet de visualiser une note particulière.
- Dans sa vue principale, la partie droite de l'application dont la vue sera optionnelle (rétractable), sera dédiée à l'affichage de l'arborescence des ascendants et l'arborescence des descendants de la note dans l'ensemble des relations existantes de manière simultanée. Le but est de visualiser les arborescences issue d'une note représentant ses ascendants et ses descendants de manière synthétique. Une note ascendante ou descendante pourra apparaître plusieurs fois dans les arborescences pour gérer les circuits existants dans une relation. Ces arborescences permettent d'un simple clique de se déplacer sur une autre note.
- Une vue secondaire de l'application sera dédiée à la gestion et la visualisation de chacune des relations.

1.6 Sauvegarde du contexte

Au démarrage de l'application, l'état de l'application, les paramétrages présents lors de la dernière exécution sont récupérés.

1.7 Fonctions annuler et rétablir

L'application dispose des fonctions "annuler" et "rétablir". Ces fonctions peuvent être appelées par un menu ou par les raccourcis Ctrl-Z et Ctrl-Y.

2 Livrable attendu

Le livrable est composé des éléments suivants :

- **Code source :** l'ensemble du code source du projet. Attention, ne pas fournir d'exécutable ou de fichier objet.
- **Documentation** : une documentation complète en `html` générée avec `Doxygen`.
- **Video de présentation avec commentaires audio** : une courte video de présentation dans laquelle vous filmerez et commenterez votre logiciel afin de démontrer le bon fonctionnement de chaque fonctionnalité attendue (environ 5-7 min). Pour réaliser cette video, vous pourrez vous servir des logiciels Jing (Windows, Mac OS), RecordMyDesktop (Linux). Ces logiciels sont mentionnés uniquement à titre d'exemple.
- **Rapport :** Un rapport en format `.pdf` composé de 2 parties :
  - la description de votre architecture ;
  - une argumentation détaillée où vous montrez que votre architecture permet facilement des évolutions.

  Vous pouvez ajouter en annexe de ce rapport des instructions à destination de votre correcteur si nécessaire (présentation des livrables, instructions de compilation, ...). Ce rapport ne devra pas dépasser 14 pages (annexes comprises).

L'ensemble des livrables est **à rendre pour le jeudi 15 juin 23h59 au plus tard**. Les éléments du livrable doivent être rassemblés dans une archive `.zip`. L'archive doit être envoyée par mail aux chargés de TD suivant votre séance :

- mardi matin : Idir Benouaret (`idir.benouaret@utc.fr`).
- mardi après-midi : Antoine Jouglet (`antoine.jouglet@utc.fr`).
- mercredi matin : Abderahim Sahli (`abderahim.sahli@utc.fr`).
- jeudi matin : Antoine Jouglet (`antoine.jouglet@utc.fr`).
- jeudi après-midi : Lyes Touati (`lyes.touati@utc.fr`).
- vendredi matin : Bastien Frémondière (`fremondiere@deltacad.fr`).
- vendredi après-midi : Benoit Cantais (`benoit.cantais@utc.fr`).

3 Évaluation

Le barême de l'évaluation du projet est comme suit :

- **Couverture des fonctionnalités demandées** : 5 points
- **Choix de conception et architecture** : 5 points. En particulier sera évaluée la capacité de l'architecture à s'adapter aux changements.
- **Evaluation des livrables** : 6 points (code source, documentation, vidéo, exemples de fichiers, rapport)
- **Respect des consignes sur les livrables** : 2 points (échéance, présence de l'ensemble des livrables et respect des consignes sur les livrables).
- **Présence et participation** aux séances de TD : 2 points

Remarque : il est rappellé qu'une note inférieure ou égale à 6/20 au projet est éliminatoire pour l'obtention de l'UV.

4 Consignes

— Le projet est à effectuer en trinôme (ou en binôme).
— Vous êtes libres de réutiliser et modifier les classes déjà élaborées en TD pour les adapter à votre architecture.
— En plus des instructions standards du `C++/C++11`, vous pouvez utiliser l'ensemble des bibliothèques standards du `C++/C++11` et le framework `Qt` à votre convenance.
— Il n'y a pas de contrainte concernant les éléments d'interface et d'ergonomie. Soyez créatifs. Il devrait y avoir autant d'interfaces différentes que de binômes.

5 Conseils

— Plusieurs TDs utilisent le thème du projet afin de commencer à vous familiariser avec les différentes entités de l'application qui est à développer. On ne perdra pas de vue que les questions développées dans ces TDs ne constituent pas une architecture pour le projet. Celle-ci devra être complètement retravaillée en tenant compte de l'ensemble des entités du sujet.
— La partie difficile du projet est la conception de votre architecture : c'est dessus qu'il faut concentrer vos efforts et passer le plus de temps au départ.
— Il est conseillé d'étudier au moins les design patterns suivants qui pourraient être utiles pour élaborer l'architecture de votre projet : decorator, factory, abstract factory, builder, bridge, composite, iterator, template method, adapter, visitor, strategy, facade, memento. En plus de vous donner des idées de conception, cette étude vous permettra de vous approprier les principaux modèles de conception.
— Pour la persistance des informations, vous êtes libres d'élaborer vos formats de fichier. Il est tout de même conseillé d'utiliser XML et d'utiliser les outils XML de `Qt`.
— Au lieu d'utiliser des fichiers, vous pouvez utiliser un SGBD comme SQLite.
— L'apparence de l'application ne sera pas prise en compte dans la notation. Soyez avant tout fonctionnels. Ca peut être moche.