Florian GUYET – GI02

Steve LAGACHE – GI02

LO21 - PROJET

UTC - P15

Conception et déceloppement de l'application ProjectCalendar

TABLE DES MATIERES

Table des matières

INTRODUCTION

Introduction

Dans le cadre de l'UV LO21 suivie à l'UTC, nous avons développé une application informatique destinée à mixer des fonctionnalités d'un agenda électronique traditionnel et d'un outil de gestion de projet.

La conception se base sur les principes de la Programmation Orientée Objet, et l'implémentation a été effectuée en C++, et en utilisant le framework Qt.

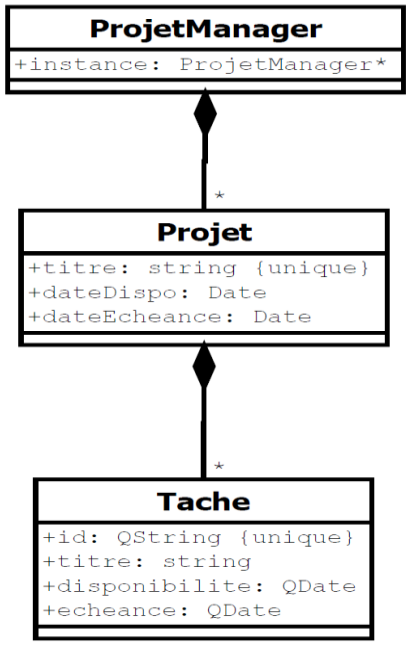
Description de l'architecture

Notre projet manipule trois types principaux d'objets : les projets, les tâches, et les événements. Nous allons exposer dans cette partie l'architecture choisie pour que ces objets interagissent au mieux entre eux afin de répondre aux attentes et aux contraintes du sujet reçu en début de semestre.

LES PROJETS

Un projet contient un ensemble de tâche à effectuer. La classe Projet compose donc la classe Tache. Chaque projet doit avoir un nom unique, qui permet de les différencier.

Les projets sont gérés par un ProjetManager. La classe ProjetManager est un singleton. L'instance du ProjetManager crée les projets, qui sont stockés dans son instance à leur création. C'est aussi le ProjetManager qui gère la destruction des projets. ProjetManager compose donc la classe Projet.



DESCRIPTION DE L'ARCHITECTURE

LES TACHES

TacheUnitaire et TacheComposite sont des classes qui héritent de la classe virtuelle Tache.

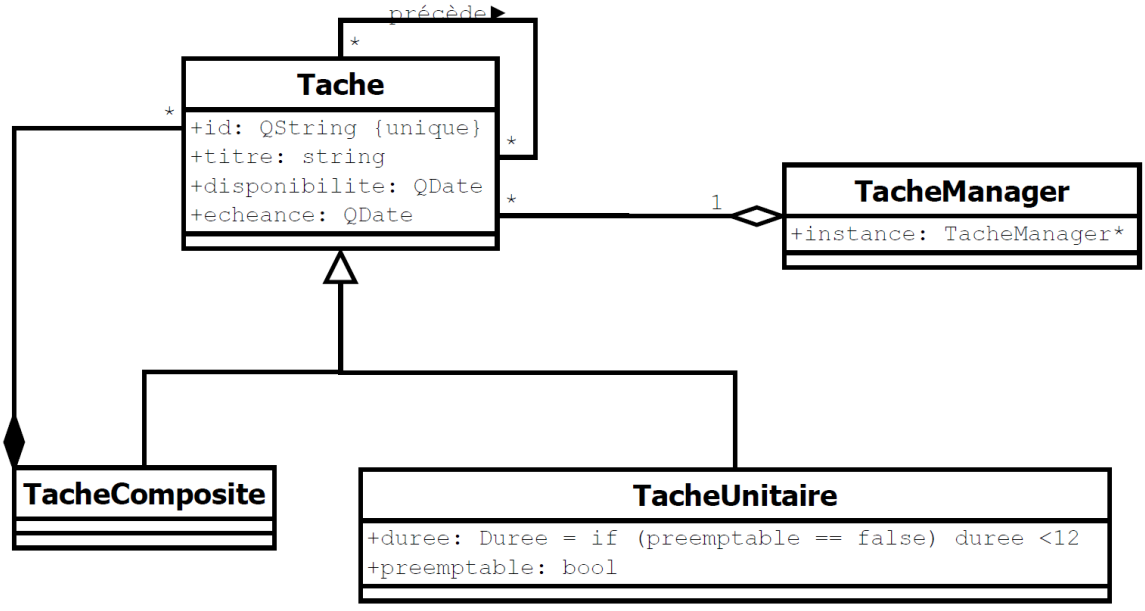
TacheComposite contient une liste chaînée de pointeurs sur tâches (QList <Tache*>), en effet une tâche composite peut posséder plusieurs sous-tâches. TacheComposite est donc une composition de Tache.

La précédence est définie par la position occupée dans la QList du projet père ou de la tâche composite mère. La méthode getTachesPrécédentes() permet notamment de renvoyer une liste chaînée de pointeurs sur les tâches précédentes à cette tâche.

La classe TacheManager est un singleton, toutes les tâches sont référencées dans son instance à leur création. TacheManager agrège Tache.

Chaque tâche a un identificateur différent, permettant de retrouver son adresse. La méthode genererNewId() du TacheManager permet d'obtenir une nouvelle clé unique (QString de 5 caractères) pour chaque tâche, à leur création.

La méthode d'ajout de tâche vérifie que les dates de disponibilité et d'échéance de la tâche sont comprises dans celles du projet.



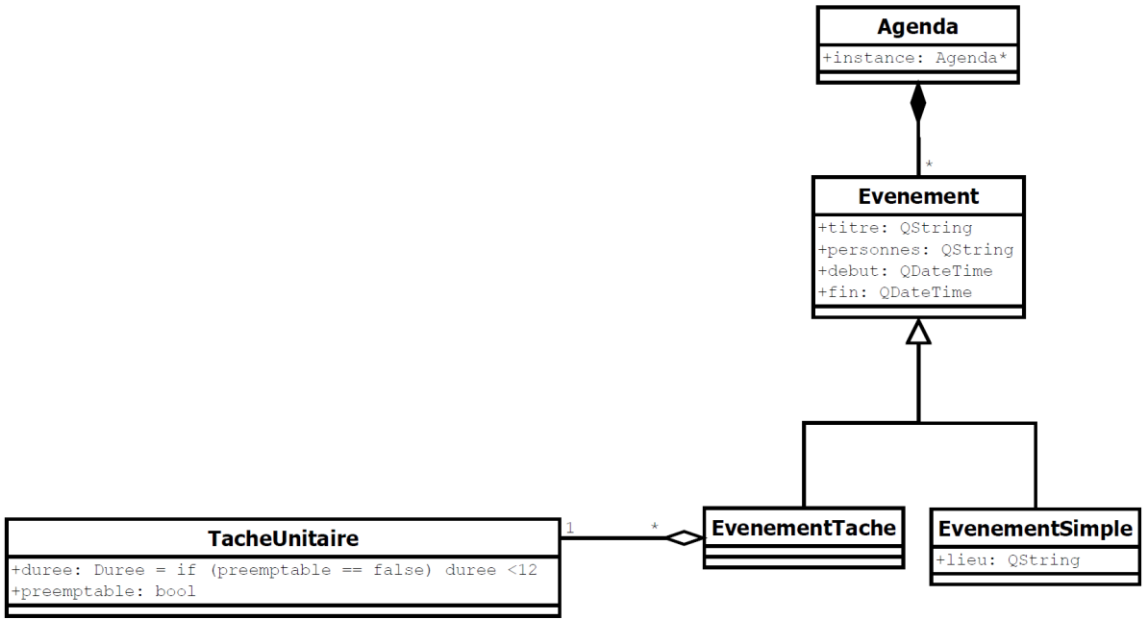
DESCRIPTION DE L'ARCHITECTURE

LES EVENEMENTS

EvenementSimple et EvenementTache sont des classes qui héritent de la classe virtuelle Evenement.

L'un des attributs d'un objet de la classe EvenementTache est un pointeur sur la tâche unitaire qu'elle programme (préemptable ou non). Donc la classe EvenementTache agrège TacheUnitaire.

La classe Agenda est un singleton, tous les évènements sont stockés dans son instance à leur création (liste chaînée de pointeurs sur évènements, simples ou non). Agenda gère la création et la destruction d'un événement. Agenda compose donc Evenement.



Les relations entre les différentes classes peuvent être observées sur l'annexe en fin de rapport : UML du projet.

EVOLUTIONS POSSIBLES

Evolutions possibles

PROGRAMMATION ORIENTEE OBJET

Grâce à l'héritage et au polymorphisme, il est très aisé de modifier les tâches et évènements ou d'ajouter de nouvelles classes filles comportant de nouvelles spécificités.

EXPORTATION DES DONNEES

Les différentes méthodes getEvenements() permettent de retourner une liste particulière de pointeurs sur évènements selon le paramètre donné (tous les évènements, évènements d'une semaine ou liés à un projet).

Les méthodes d'export de l'agenda (général, semaine et projet) permettent d'exporter une liste d'évènements (tous les évènements, évènements d'une semaine ou liés à un projet) dans un fichier XML.

Il serait facilement possible de rajouter d'autres types d'exportation (par exemple, l'exportation des tâches d'un projet).

En conclusion, il nous semble aisé, si besoin est, de rajouter des fonctionnalités à notre application.

ANNEXE : UML DU PROJET

Annexe : UML du projet