Complément au Cahier des Charges

Meryll Essig
Tamara Rocacher

13 décembre 2015

Introduction

Dans le cadre de l'UE modélisation et programmation par objets 2, nous avons réalisé un projet, préalablement définit par un Cahier des Charges donné à titre de consigne.
Ce projet consiste en l'implémentation d'une application permettant de gérer les tâches à réaliser d'un utilisateur, sous forme de Todo List.
Ainsi, ce document établit les choix effectués lors de la réalisation de notre projet, dans le but de respecter au mieux les fonctionalités attendues tout en simplifiant à la fois l'implémentation et l'utilisation.

Sommaire

1 Conception

1.1 Vue

La phase de conception a permis, d'une part, de définir totalement la partie Vue. Pour cela, nous avons choisi de n'établir qu'une seule fenêtre permettant à la fois l'affichage des tâches a réaliser sous forme de liste (1-liste), et l'affichage des champs a remplir pour créer ou modifier une tâche (2-tâche). Ce choix repose sur le besoin de faciliter la prise en main de l'application pour l'utilisateur. De même, pour le changement du mode d'affichage de la liste de tâches (simple ou détaillé), nous avons choisi de ne pas surcharger la vue avec trop de boutons. Ainsi, la sélection du mode se fait par le menu Affichage, qui repond toujours au besoin de facilité l'utilisation, car étant un menu courant et explicite.

1.2 Patron MVC

D'autre part, l'architecture du code source de l'application etant imposée, a savoir le patron de conception Modèle Vue Controleur (MVC), une grande partie de cette phase a été consacrée à la documentation et la réalisation de diagrammes de cas d'utilisation et de classes. Ainsi, notre implementation du MVC repose sur la separation du code en trois packages permettant de séparer chaque partie, car certaines doivent contenir plusieurs classes.
De plus, cette architecture nous a amenés a faire des choix au niveau algorithmique. En effet, pour pouvoir modifier une tâche (côté 2-tâche), l'utilisateur doit voir s'afficher les anciennes données, qu'il modifira ou non. Or, en raison de la distinction entre Vue et Modèle, un choix s'imposait pour faciliter la communication et permettre au Modèle de savoir quelle tâche l'utilisateur a sélectionné dans la vue 1-liste. Ainsi, un tableau rempli par ordre chronologique de création des tâches permettra le stockage de l'identifiant de la tâche, mais limitera le nombre de tâches affichables dans la liste. Nous avons donc fixé cette limite d'affichage a 10 tâches.

1.3 Schémas et Diagrammes

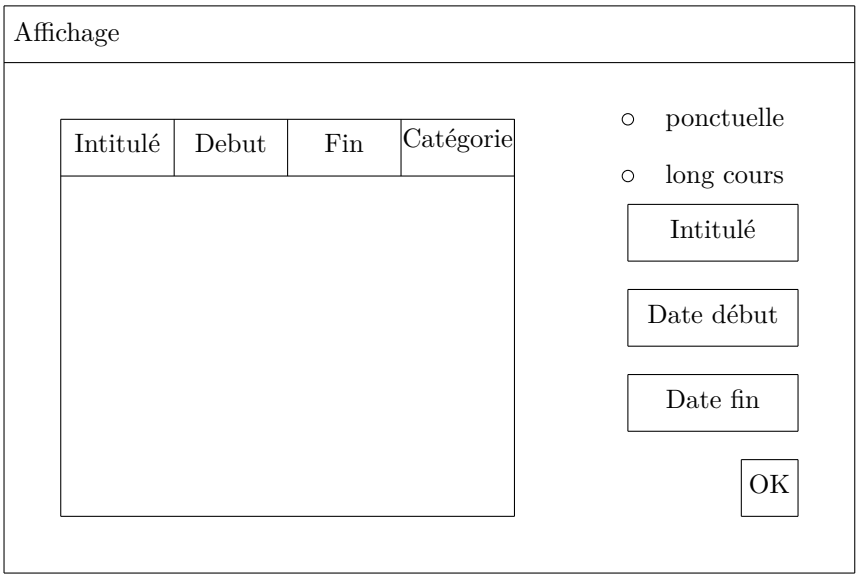

FIGURE 1 – Schéma de la vue

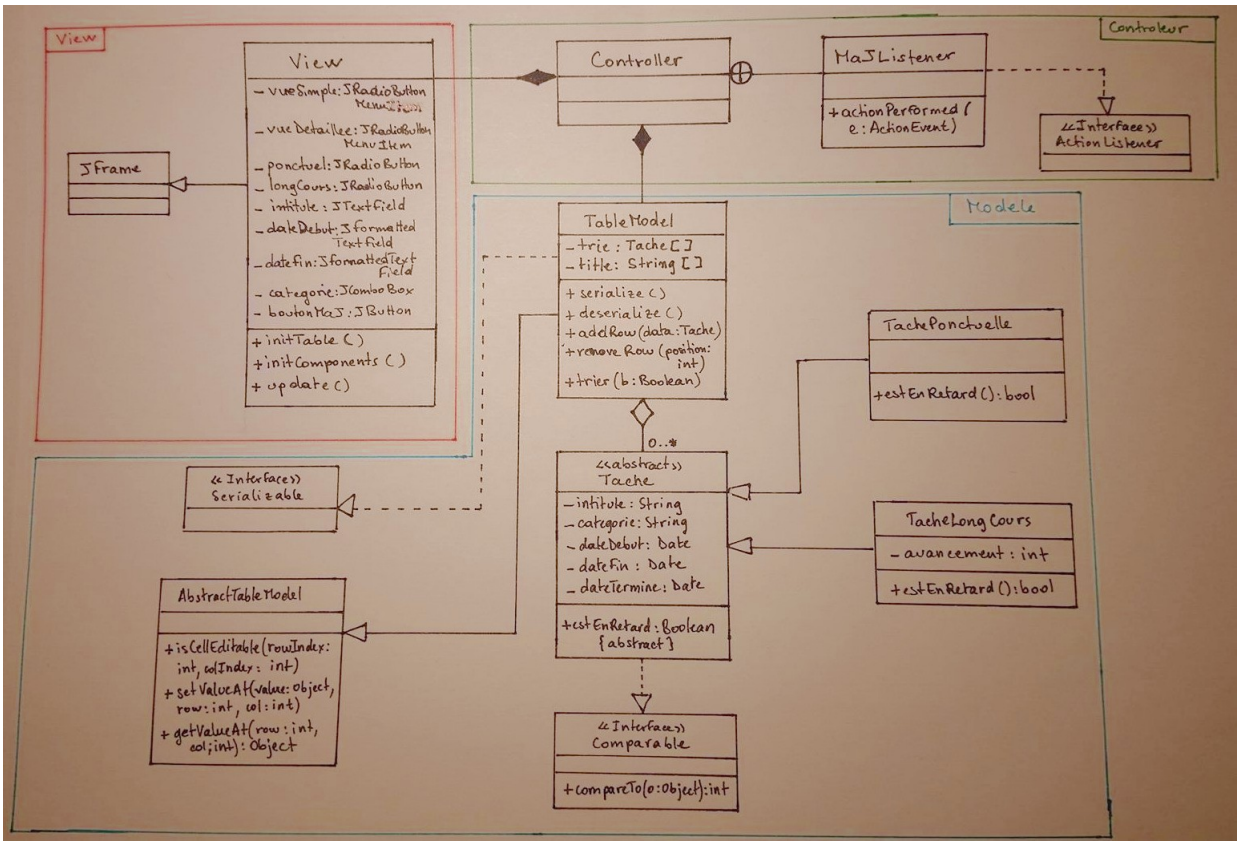

FIGURE 2 – Diagramme de Classes

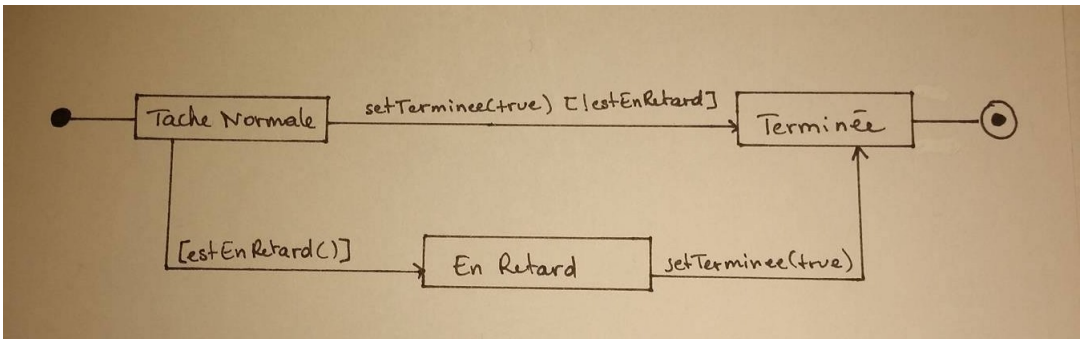

FIGURE 3 – Diagramme d'état-transitions de la classe Tache

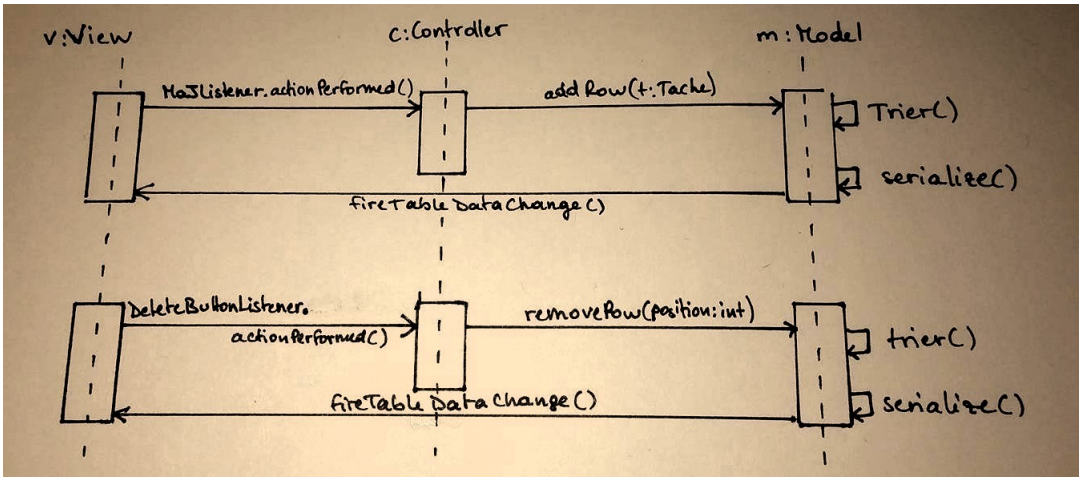

FIGURE 4 – Diagramme de séquence de notre architecture MVC

2 Réalisation

Au vu de notre manque de bases solides sur le MVC, nous avons cherché des tutoriels sur ce sujet. Beaucoup utilisaient le Patern Observer, avec la vue qui contient le controleur, qui contient le modèle, mais cette façon de faire ne nous convenait pas, compte tenu du sujet. Nous avons donc décidé de nous baser sur le tutoriel video de Derek Banas (un anglophone qui a réalisé un série de tutoriels sur les patrons de conception en java).

Le lien de la video : https ://www.youtube.com/watch ?v=dTVVa2gfht8

Au final, le controleur contient le modèle et la vue en données membre, et l'interaction entre ceux ci parait plus évidente et claire.

3 Conclusion

Pour conclure, nous avons décidé de plus mettre notre temps à profit sur la phase de conception et d'analyse. La phase de réalisation s'est avérée assez productive pour la semaine et demi que nous lui avons allouée.