



Documentation de développement

Projet de semestre

**Agenda étudiant**

**Étudiants :** Jérôme Moret, Paul Ntawuruhunga,

Thibaud Duchoud, David Kunzmann et Mario Ferreira

**Professeur :** Dr. René Rentsch

**Date :** 21 avril 2015 **Classe :** PRO-1-B

1 Introduction 4

1.1 Généralité 4

1.2 Motivation du projet 4

1.3 Langage de programmation 4

1.4 Déploiement de l’application 4

2 Compétences 5

3 Conception 5

3.1 Interface graphique 5

3.1.1 Vue « Tâche » 5

3.1.2 Vue « Calendrier » 5

3.1.3 Vue « Résumé » 6

3.2 Base de données 7

3.2.1 Choix du moteur de la base de données 7

3.2.2 Structure de la base de données 7

3.3 UML 11

3.3.1 Zone 1 12

3.3.2 Zone 2 12

3.3.3 Zone 3 12

4 Planification 13

4.1 Planification initiale 13

4.2 Planification finale 14

4.3 Détails évolutions 15

4.4 Synthèse sur l’évolution 15

5 Journal de bord 16

Semaine 1 à 3 16

Semaine 4 16

Semaine 5 16

Semaine 6 16

Semaine 7 16

Semaine 8 16

Semaine 9 17

Semaine 10 17

Semaine 11 17

Semaine 12 17

Semaine 13 17

Semaine 14 18

Semaine 15 18

6 Sources 18

7 Logiciels et outils utilisés 18

8 Conclusion 18

8.1 Exécution du mandat 19

# Introduction

## Généralité

|  |  |  |
| --- | --- | --- |
| **Projet** | | |
| **Nom du projet** | Progenda | |
| **Début du projet** | 17.03.2015 | |
| **Fin du projet** | 02.06.2015 | |
| **Durée effective** | 90 heures | |
| **Acteurs** | | |
| **Répondant** | Dr. René Rentsch | rene.rentsch@heig-vd.ch |
| **Développeurs** | Jérôme Moret (Chef de projet)  Paul Ntawuruhunga  Thibaud Duchoud  David Kunzmann  Mario Ferreira | [jerome.moret@heig-vd.ch](mailto:jerome.moret@heig-vd.ch)  [paul.ntawuruhunga@heig-vd.ch](mailto:paul.ntawuruhunga@heig-vd.ch)  [thibaud.duchoud@heig-vd.ch](mailto:thibaud.duchoud@heig-vd.ch)  [david.kunzmann@heig-vd.ch](mailto:david.kunzmann@heig-vd.ch)  mario.ferreira@heig-vd.ch |

## Motivation du projet

Lorsqu’on est étudiant et qu’on se retrouve face aux piles de devoirs et leçons inscrits dans l’agenda papier, le calcul de moyenne souvent réalisé de tête, les horaires et salles de cours qu’on oublie de semaine en semaine, etc… Il est toujours difficile de trouver une organisation efficace, d’où notre idée de développer un agenda étudiant.

## Langage de programmation

L’application a été développée à l’aide du framework Qt en C++.

## Déploiement de l’application

L’équipe de développement est composée de collaborateurs sur Windows, Linux et Mac OS X.

L’application est donc portée pour les **trois** systèmes d’exploitation. L’application finale a simplement été compilée sur chacune des plateformes citées ci-dessus.

# Compétences

Au début du projet l’équipe n’avait aucune compétence sur le framework Qt. Il a donc fallu se documenter sur l’utilisation de celui-ci et notamment sur les possibilités de conception d’interface graphique, pièce maitresse de notre projet.

Un livre a été commandé, vous pouvez le retrouver dans le chapitre 7 Sources.

# Conception

## Interface graphique

### Vue « Tâche »

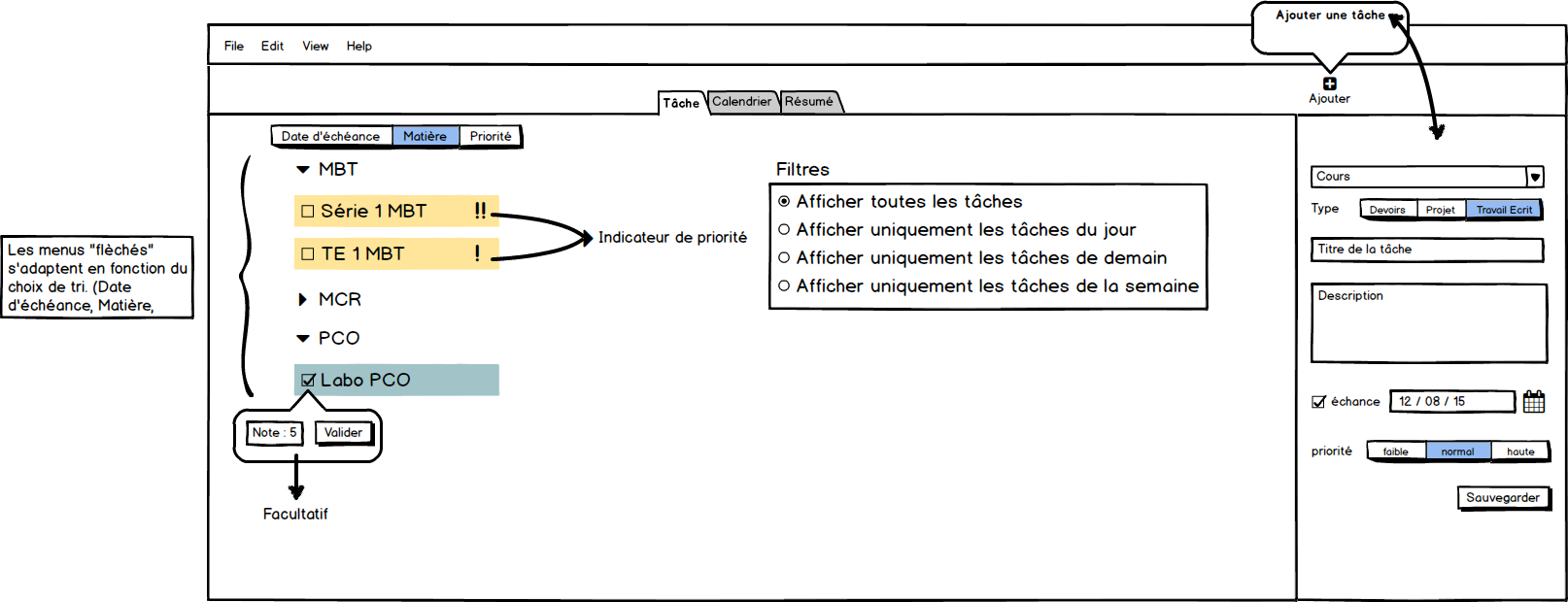


Figure 1 : Schéma de la vue Tâche

La fenêtre est composée de quatre parties principales.

Sur la droite se situe le panneau d’édition de tâches. À la création d’une nouvelle tâche les champs sont soit vide soit rempli avec des valeurs par défaut. Lorsque l’utilisateur clique sur le nom d’une tâche dans la liste, les informations de celle-ci sont reportées dans le panneau permettant une modification directe.

Sur la gauche se situe la liste des tâches. Les tâches sont triées par ordre alphabétique dans chaque groupe.

Au-dessus de chaque groupe se trouve des boutons rapides permettant de grouper les tâches. Les tâches d’un même groupe peuvent être masquées à l’aide d’un clic sur la flèche devant le titre du groupe.

Enfin, au milieu, se situe une liste de filtres applicable à l’ensemble des tâches (masquage). Il est possible d’utiliser 4 filtres en sélectionnant un « radiobutton » dans la liste. On ne peut sélectionner qu’un seul choix à la fois.

### Vue « Calendrier »

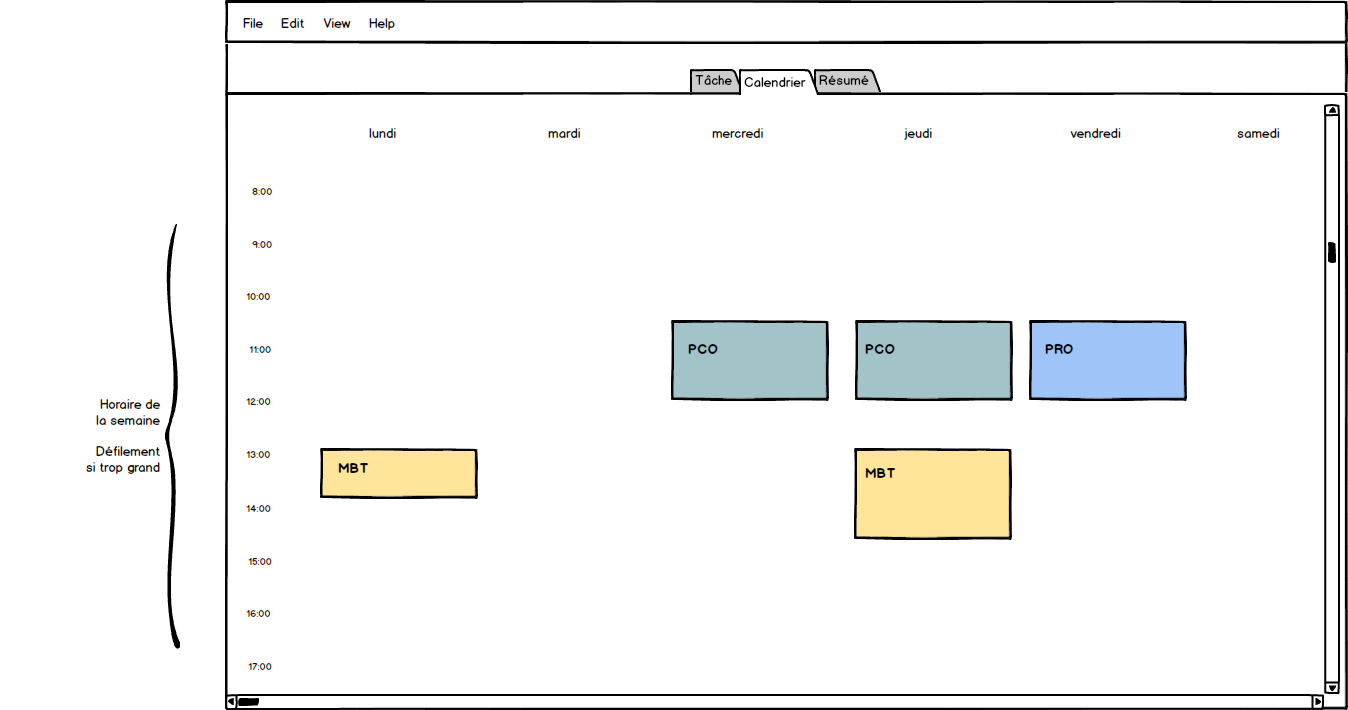


Figure 2 : Schéma de la vue Calendrier

Cette vue n’est qu’un tableau d’affichage de l’horaire hebdomadaire d’un étudiant, tous les ajouts et modifications sont à faire depuis la vue « Résumé » décrite ci-après. Il était prévu qu’il soit possible d’ajouter et de modifier des leçons directement depuis cette vue seulement le retard pris pendant le projet nous a contraint à abandonner cette fonctionnalité. Vous retrouvez cette information dans le chapitre 8.1 Exécution du mandat.

### Vue « Résumé »

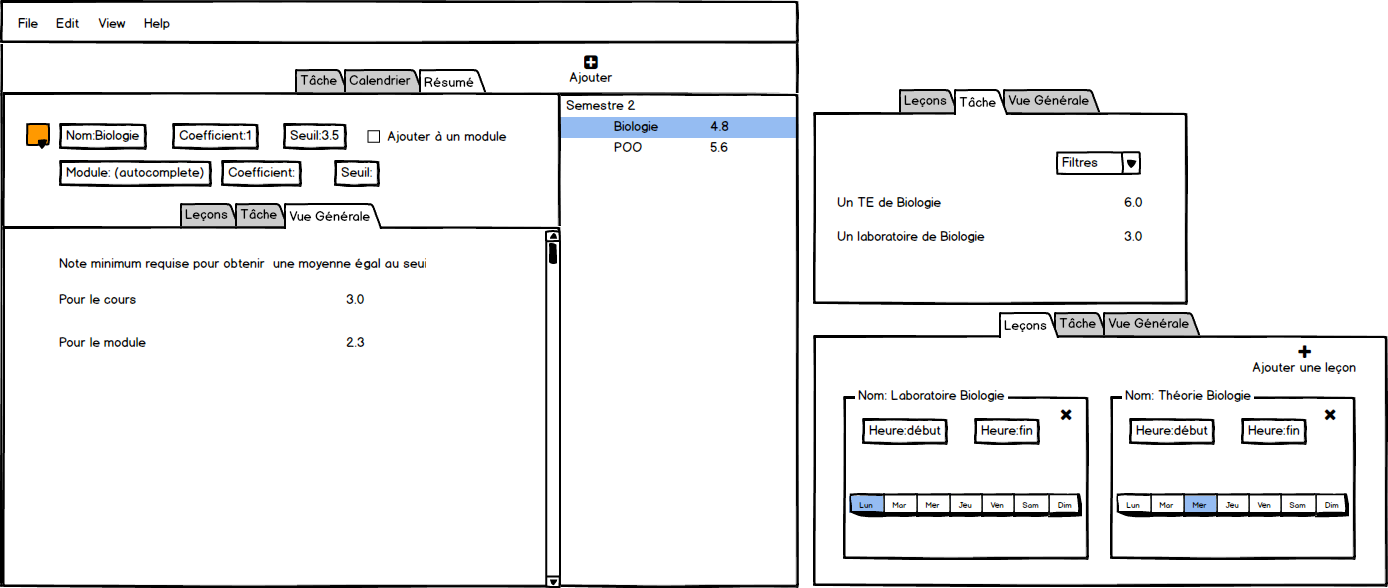


Figure 3 : Schéma de la vue Résumé

La vue résumé se compose de différentes parties. Il y a le cadre principal qui contient les informations importantes tel que les moyennes des différentes matières. Sur la droite se trouve un panneau latéral où tous les semestres ainsi que les matières de ces derniers sont listés.

Lors de la première utilisation de l’Agenda Etudiant. L’utilisateur est invité à créer ses semestres depuis la vue résumé. En cliquant sur le bouton ajouter présent dans la barre de navigation des champs vides s’affichent dans le cadre principal. Ces champs représentent le nom, la date de début ainsi que la date de fin du nouveau semestre. Une fois les modifications terminées l’utilisateur verra apparaître son semestre dans la barre latérale.

Lorsque l’utilisateur a entré des notes pour des tâches, la moyenne de ces notes apparaitra dans la vue résumé, ceci lorsque l’étudiant clique sur le semestre voulu dans le panneau latéral.

L’ajout d’une matière se fait de la même manière que l’ajout d’un semestre. Lors de la sélection de l’ajout de la matière les champs requis vides sont affichés. Une fois les champs remplis, la matière apparaît dans la barre latérale. En cliquant sur une de ces matières, le cadre principal affiche maintenant les informations du module dont la matière fait partie. Toutes ces informations sont directement modifiables. Nous pouvons voir également que trois nouveaux onglets sont apparus.

L’onglet « Vue générale », contient les notes minimum requises pour les cours ainsi que le module en question. L’onglet « Tâche » affiche toutes les tâche en relation avec la matière sélectionnée. Finalement l’onglet « Leçons » affiche les leçons de la matière. Il est également possible de les modifier, d’en rajouter et/ou d’en enlever.

## Base de données

### Choix du moteur de la base de données

Pour notre agenda étudiant nous avons fait le choix d’utiliser une base de données SQLite.

Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes.

Notre application n’est à aucun moment composé d’un système de login, il n’aurait pas été judicieux d’utiliser une base de donnée MySQL ou PostgreSQL.

Il est également intéressant de noter que l'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme (pratique pour notre projet multiplateforme).

### Structure de la base de données

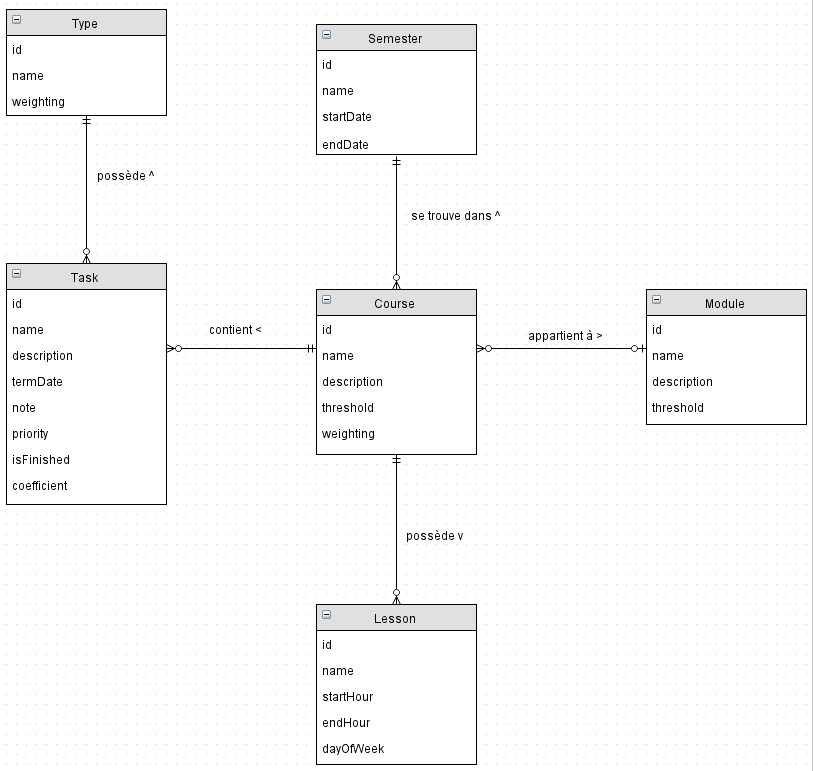


Figure 4 : Schéma de la base de données

Il existe plusieurs types de données dans SQLite, ceux-ci définissant la façon dont SQLite va travailler lors de l'entrée des nouvelles données :

* **TEXT** : enregistre la donnée comme une chaine de caractères, sans limite de taille. Si un nombre est entré dans une colonne de ce type, il sera automatiquement converti en une chaine de caractères ;
* **NUMERIC** : tente d'enregistrer la donnée comme un entier ou comme un réel, mais si cela s'avère impossible, la donnée sera enregistrée comme une chaine de caractères ;
* **INTEGER** : enregistre la donnée comme un entier si celle-ci peut être encodée sans perte, mais peut utiliser les types REAL ou TEXT si ça ne peut être fait ;
* **REAL** : enregistre la donnée comme un réel, même s'il s'agit d'un entier. Si la valeur est trop grande, la donnée sera convertie en chaine de caractères ;
* **NONE** : la donnée est enregistrée telle quelle, sans conversion.

Voici ci-dessous la liste des tables avec les types de données et si besoin une description rapide :

|  |  |  |
| --- | --- | --- |
| **Course** | | |
| **Champ** | **Type** | **Description** |
| id | INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT |  |
| name | TEXT |  |
| description | TEXT |  |
| threshold | REAL | Seuil |
| weighting | REAL |  |
| moduleId | INTEGER | Référence Module -> id |
| semesterId | INTEGER NOT NULL | Référence Semester -> id |
| **Lesson** | | |
| **Champ** | **Type** | **Description** |
| id | INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT |  |
| name | TEXT |  |
| startHour | TEXT |  |
| endHour | TEXT |  |
| dayOfWeek | TEXT |  |
| courseId | INTEGER NOT NULL | Référence Course -> id |
| **Module** | | |
| **Champ** | **Type** | **Description** |
| id | INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT |  |
| name | TEXT |  |
| description | TEXT |  |
| threshold | REAL |  |
| **Semester** | | |
| **Champ** | **Type** | **Description** |
| id | INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT |  |
| name | TEXT |  |
| startDate | TEXT |  |
| endDate | TEXT |  |
| **Task** | | |
| **Champ** | **Type** | **Description** |
| id | INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT |  |
| name | TEXT |  |
| description | TEXT |  |
| termDate | TEXT |  |
| note | REAL |  |
| priority | INTEGER |  |
| isFinished | INTEGER |  |
| coefficient | REAL |  |
| typeId | INTEGER NOT NULL | Référence Type -> id |
| courseId | INTEGER NOT NULL | Référence Course -> id |

## UML

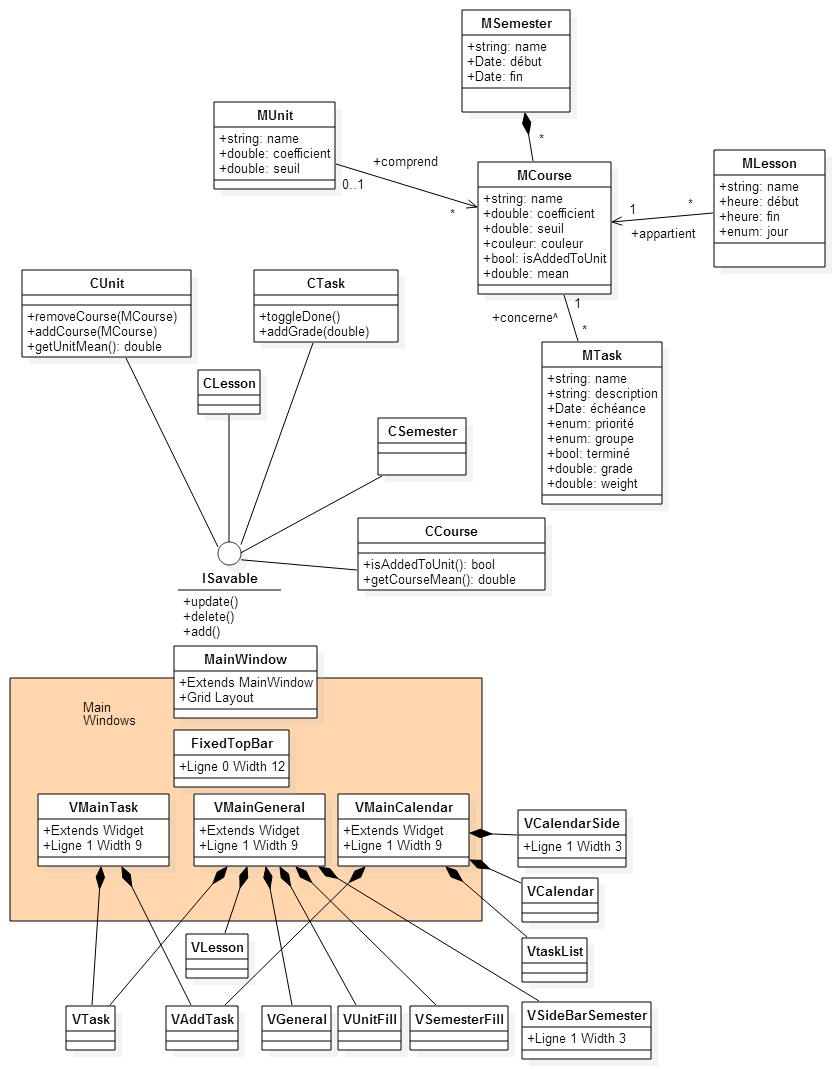


Figure 5 : Schéma UML

Notre application utilise le Modèle *MVC[[1]](#footnote-1) (Modèle-vue-contrôleur)*.

### Zone 1

Dans cette zone, vous trouvez toutes les classes *modèles* de notre application.

* Un cours (MCourse) se trouve dans un ou plusieurs semestres (MSemester).
* Un cours peut avoir plusieurs leçons dans la semaine (MLesson).
* Une tâche concerne un cours (MTask) et un cours peut avoirs de ces tâches.
* Il est possible de grouper des cours dans une unité (Munit).

### Zone 2

Dans cette zone, vous trouvez les *contrôleurs* de notre application. Chacun d’entre eux implémente l’interface **ISavable** qui permettra de définir un standard quant à la suppression, la modification et l’insertion.

* Le *contrôleur* pour les unités (CUnit) permet à ceux-ci d’ajouter des cours, d’en supprimer ou d’obtenir la moyenne de ceux-ci.
* On doit pouvoir valider une tâche réalisée et éventuellement ajouter une note à celle-ci. (sur CTask)
* Il doit être possible de récupérer la moyenne d’un cours. (sur CCourse)

### Zone 3

Dans cette zone, on aperçoit l’idée d’implémentation de la structure de notre application sur Qt. Sur Qt la fenêtre principale est communément appelé MainWindow.

On aurait donc dans cette fenêtre principale :

* Une barre supérieure fixe (FixedTopBar) qui permettrait la sélection des différentes vues.
* Une zone de contenu centrale dans laquelle on trouverait, selon la vue sélectionnée :
  + Les *vues* du modèle *MVC*. (VCalendar, VTask, VGeneral, …)
  + Une éventuelle barre latérale
  + Tout autre contenu utile à notre vue chargée.

# 

# Planification

## Planification initiale

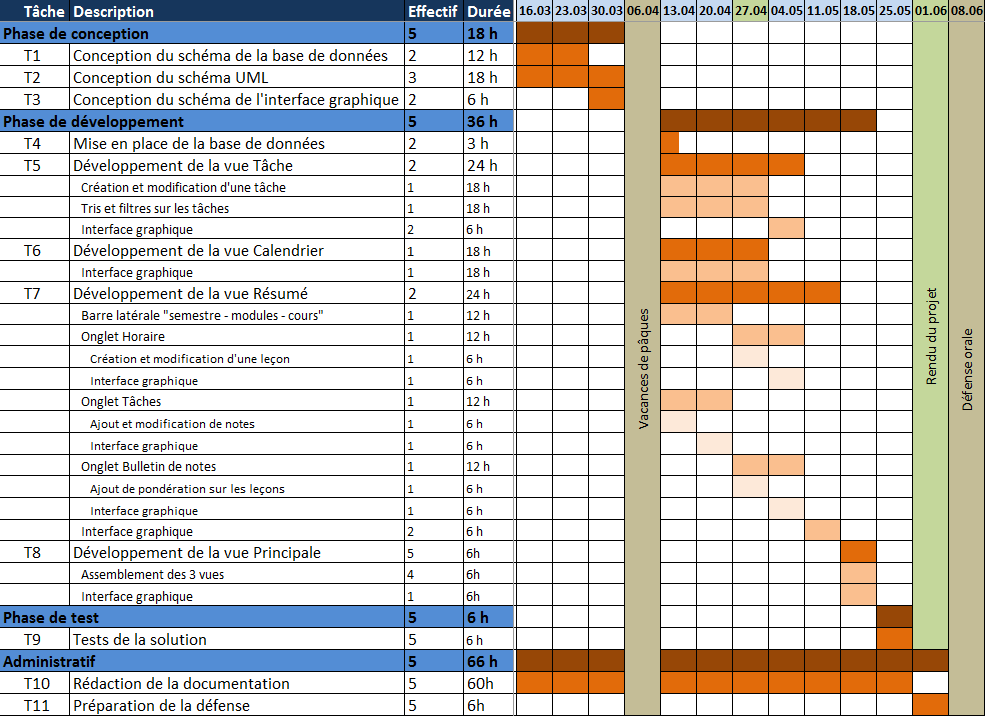


Figure 6 : Schéma de la planification initiale

## Planification finale

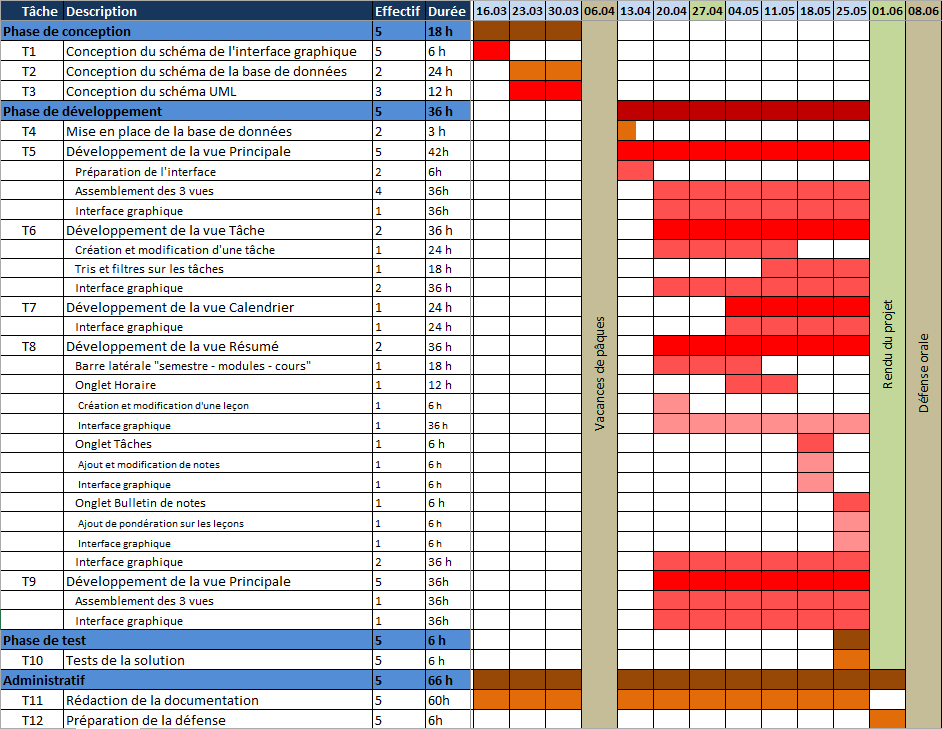


Figure 7 : Schéma de la planification réelle et finale

## Détails évolutions

Le projet a souffert de plusieurs variations et remaniements au sein de la planification.

Il a tout d’abord fallu tenir des comptes de l’évaluation du cahier des charges. Nous n’avions pas fourni de schéma pour l’interface de notre projet. On s’est rendu compte par nous même qu’on avait chacun une image différente de l’interface. Il a donc fallu corriger le tir et c’est la première variation du schéma avec le passage de la tâche « Schéma de l’interface graphique » en première tâche.

Ensuite, chaque tâche de développement a pris plus longtemps que prévu, il a fallu régler plusieurs problèmes directs et/ou d’optimisations.

Certaines tâches comme « Interface graphique » ont pris le même temps que le développement de la vue elle-même tout simplement parce qu’elles s’effectuaient en parallèle.

La tâche « Assemblement des 3 vues » s’est déroulé plus tranquillement que prévu. En effet, à la base, il était planifier que cette tâche soit réalisé à la fin mais nous l’avons déplacé au début puis nous avons, par itération, mis à jour l’intégration des différentes vues. De cette manière nous avons assuré la cohésion des parties de développement. C’est pour cette raison que vous trouvez cette tâche en parallèle au développement de la vue principale.

Finalement la tâche principale « Développement de la vue calendrier » s’est trouvé être plus compliqué que prévu, elle fut en premier lieu retardé et dura jusqu’à la fin avec en plus des fonctionnalités abandonnées.

## Synthèse sur l’évolution

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Semaine 1** | **Semaine 2** | **Semaine 3** | **Semaine 4** | **Semaine 5** | **Semaine 6** | **Semaine 7** | **Semaine 8 Semaine 9** | | **Semaine 10** |
| Pas de retard | Pas de retard | Pas de retard | Pas de retard | Pas de retard | 1 semaine | 2 semaines | 2 semaines | 2 semaines | 2 semaines |
| **Semaine 11** | **Semaine 12** | **Semaine 13** |  | | | | | | |
| 2 semaines | 2 semaines | 3 semaines |

L’équipe a dû travailler d’arrache-pied pour rattraper les retards et arriver à un produit fonctionnel.

# Journal de bord

## Semaine 1 à 3

Ces semaines ont été utilisées pour la recherche d’une idée de projet, la proposition de celle-ci ainsi que la définition du cahier des charges avec son planning initial associé.

## Semaine 4

Pour avoir une meilleure idée de l’application à développer on va commencer par s’occuper du schéma de l’interface graphique, contrairement au planning qui prévoyait de s’occuper de cela à la fin :

Tout l’effectif se charge de cette tâche.

## Semaine 5

Validation collective du schéma de l’interface graphique.

Correction des derniers détails sur le schéma.

David, Paul et Jérôme s’occupent d’écrire un descriptif pour chaque vue puisqu’il y en a 3.

Mario et Thibaud commencent la modélisation du schéma de la base de données.

## Semaine 6

Validation des descriptifs.

Modélisation du schéma UML par David, Paul et Jérôme.

Fin de la modélisation du schéma de la base de données par Mario et Thibaud.

## Semaine 7

Mario et Thibaud se charge de la mise en place de la base de données.

Les autres collaborateurs étudient la manière d’aborder l’implémentation de l’interface sur Qt.

* Début ensemble de la fenêtre de base où se trouvera chaque vue, considéré sous Qt comme des widgets qui pourront être développé parallèlement.

## Semaine 8

Il ne reste plus qu’une semaine avant la présentation intermédiaire. Selon la planification initiale, il était prévu que la vue Calendrier soit gérée par une personne, seulement après étude de Qt, nous nous sommes rendu compte qu’elle serait certainement la plus longue et compliquée à développer.

Pour pouvoir présenter une version fonctionnelle de notre produit, nous allons distribuer 3 personnes sur la vue Résumé et les deux dernières personnes sur la vue Tâches.

David et Paul commence le développement de la vue Tâche.

Mario, Thibaud et Jérôme commence le développement de la vue Résumé.

## Semaine 9

**PRESENTATION INTERMEDIAIRE.**

Paul débute le développement de la vue « Calendrier ».

David continue le développement de la vue « Tâche ».

Thibaud, Mario et Jérôme continue le développement de la vue « Résumé ».

## Semaine 10

Suite à l’utilisation des vues avec la base de données, on se rend compte qu’il y aurait éventuellement une meilleure manière pour se connecter et interagir avec la base de données.

Cette manière fournit par Qt permet … étudié par Mario.

Thibaud continue le développement de la vue « Résumé ».

David et Paul teste la technique de mapping SQL pour l’insertion de données pour la vue « Tâche ».

Jérôme implémente la récupération des leçons depuis la base de données pour la vue « Calendrier ».

## Semaine 11

Mario s’occupe de convertir les modèles en singleton.

Thibaud et Jérôme continue le développement de la vue « Résumé ».

David et Paul effectue des recherches sur le « model view programming » de Qt.

## Semaine 12

Mario continue de convertir les modèles en singleton.

Thibaud continue le développement de la vue « Résumé ».

David et Paul conclue la vue « Tâche »

Paul travaille en parallèle sur la vue « Calendrier ».

Jérôme met à jour la documentation et s’informe sur le produit Doxygen qui permettra de générer de la documentation à partir du code.

## Semaine 13

Mario développe un modèle « Leçon » pour la vue calendrier.

Thibaud finit le développement de la vue « Résumé ».

David et Paul finalise la vue « Tâche » et règle notamment le problème de tri des tâches.

Jérôme finalise et met en forme la documentation.

## Semaine 14

**PREPARATION DE LA DEFENSE.**

## Semaine 15

**DEFENSE.**

# Problèmes rencontrés

# Bibliographie

* **Ezust, Alan**, *An introduction to design patterns in C++ with QT (2nd Edition), 2011,* Prentice Hall(ISBN 978-0-13-282645-7)

# Logiciels et outils utilisés

|  |  |
| --- | --- |
| **Application** | **Brève description** |
| **http://3.bp.blogspot.com/-x73CAlkzF_M/Uqtg1OzHxZI/AAAAAAAABGk/PbFG-W1RMFA/s1600/balsamiq-mockups-app-icon.pngBalsamiq Mockups** | Dessin des schémas des interfaces graphiques |
| http://s3.amazonaws.com/digitaltrends-uploads-prod/2013/08/microsoft-office-365-logo.png **Microsoft Office** | * Rédaction de la documentation * Dessin de la planification GANTT |
| Logo **SQLite** | Moteur de la base de données |
| http://ci.apache.org/projects/httpd/trunk/doxygen/doxygen.png**Doxygen** | Génération automatique de la documentation à partir du code |

# Conclusion

Ce projet nous a permis de nous rendre compte à quel point il était difficile de travailler ensemble. En effet, comme 73% des périodes étaient non-encadrées, il a fallu faire preuve d’une grande organisation pour ne pas que le projet parte dans tous les sens.

Effectivement lors de la planification initiale, au vue de l’effectif (5 étudiants), nous nous attendions à des phases de développement assez rapides. Il n’y a pas eu besoin de beaucoup de temps pour qu’on se retrouve avec du retard lors des phases de développement.

Ce projet de groupe nous a beaucoup appris et il a parié qu’à l’avenir nous prévoirons plus de temps pour les phases de développement car comme on l’a vu le temps de développement est loin d’être proportionnel à l’effectif.

## Exécution du mandat

Suite aux différentes prises de retard il a été décidé de …

## Améliorations futurs

Il sera possible d’amener les améliorations suivantes présentes en tant que fonctionnalités supplémentaires dans le cahier des charges :

# Annexes

Les documents suivant sont encore joint ci-après :

* **Annexe A :** Cahier des charges révision 1 du 10.03.2015
* **Annexe B :** Conventions de nommage/codage

1. Chaque élément du modèle est affiché en *italique* dans la description du schéma UML. [↑](#footnote-ref-1)