Liu Zhuying p1306849
Bardoux Laurent p1108365

Design et développement d'un gestionnaire d'agendas

Cahier des charges

Table des matières :

Contexte du projet de recherche :

L'organisation d'une formation, qui est composée de nombreuses unités d'enseignement, est un problème compliqué. Chaque unité d'enseignement doit prendre en compte un grand nombre de paramètres, tels que la disponibilité des enseignants, le nombre d'heures. Et une unité d'enseignement est parfois commune à plusieurs formations.

Malheureusement, la plupart du temps, ces contraintes sont traitées en dehors de l'éditeur d'agenda (dans le cas contraire, la gestion n'est peut-être pas assez poussé, ou bien trop complexe d'un point de vue utilisateur), forçant le responsable de la formation à gérer lui-même l'intégrité et la validité de son agenda. Par exemple, une unité d'enseignement est composée d'une dizaine de cours magistraux et de travaux dirigés, le troisième travail dirigé ne peut pas être fait avant le cinquième cours magistral, et deux cours magistraux de la même unité d'enseignement ne doivent pas avoir lieu au même moment.

De ce fait, un enseignant doit sans cesse passer de ses notes/autres logiciels à la gestion de l'agenda, et ces changements de contextes entraînent une perte de temps, et une augmentation des risques d'inattention.

L'idée serait d'avoir un gestionnaire d'agendas prenant en compte des contraintes imposées par les responsables de formation, qui indiqueraient, par exemple, le nombre d'heure restantes, un créneau libre en commun de plusieurs agendas .
Bien entendu, le non respect de ces contraintes déclencherait un avertissement à l'utilisateur, de préférence en indiquant le problème qui a été soulevé. Par exemple, deux cours magistraux de la même formation ont lieu au même moment. Quand aucun avertissement n'est affiché, l'intégrité de l'agenda de formation est cohérente par rapport aux contraintes imposées.

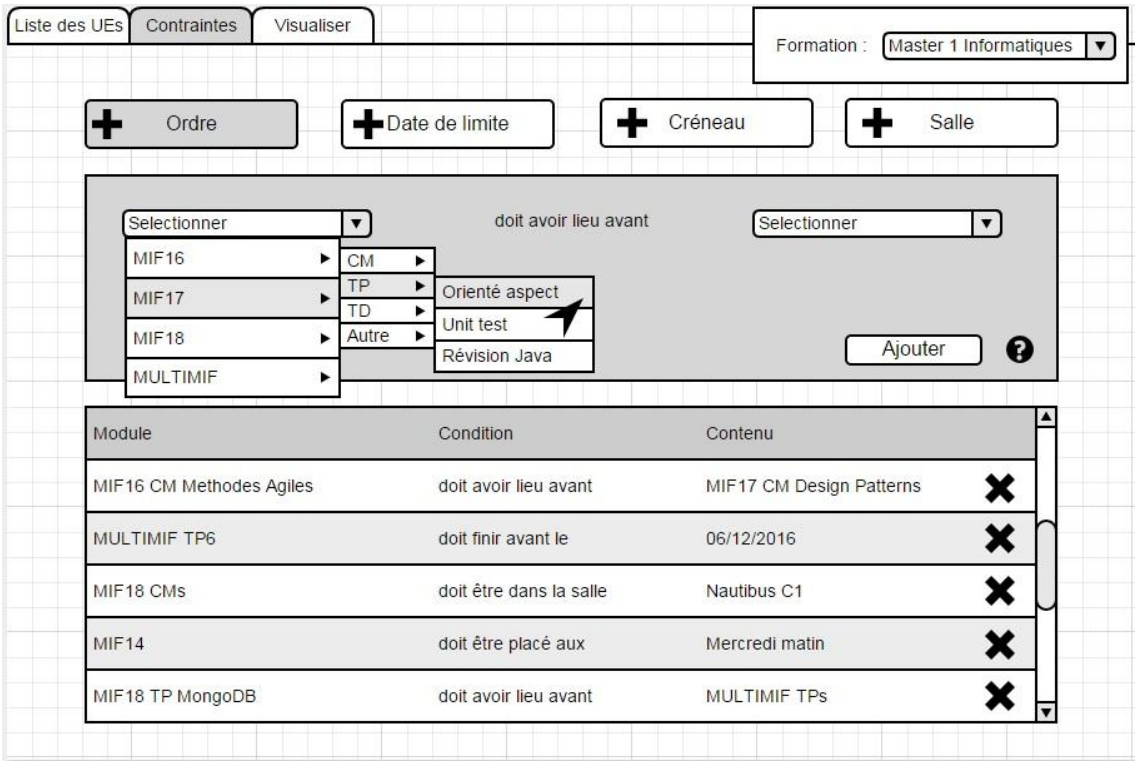
De plus, cette application fournira des services utiles et simples d'utilisation pas forcément présents sur les autres produits, comme la comparaison de deux agendas, conversion d'un agenda dans un format textuel, plus simple à lire, une fusion d'agendas.

C'est dans cette optique que Monsieur Caniou, maître de conférence à l'ENS, propose un sujet de stage TER intitulé : "Design et développement d'un gestionnaire d'agendas".

En plus des fonctions d'affichage et d'importation, il faudra développer une planification d'un agenda avec satisfaction des contraintes, recherche dans un agenda en fonction de critère(s) (un groupe d'étudiants, un enseignant, une salle ...). Mais aussi une fonction permettant de modifier de manière plus fine l'agendas (prélever des morceaux de créneaux pour les mettre ailleurs, *swap* de deux créneaux).

En plus, la possibilité d'un retour en arrière à un point antérieur, une comparaison de deux agendas, l'importation d'un agenda à partir d'un fichier .ics et exportation d'un agenda au format .txt et .ics. Les contraintes spécifiques peuvent être des créneaux, des salles, enseignants, groupes (donc des "ressources" limitées), *les deadlines* (une UE doit être finie avant fin mai) et les relations de précédence entre évènements(le 3ème CM doit être faite avant le 2ème TP), qui représente alors des contraintes temporelles.

L'image suivante illustre une idée de présentation possible :



Aspects techniques du projet :

Nous avons choisi de concevoir notre application en Python, et donc *from Scratch* (Korganizer, utilisant le C++ et étant open source, permettait de pouvoir se concentrer sur les ajouts de fonctionnalités).
Le choix du Python relève de plusieurs choses, comme par exemple sa nature multi-plate-forme, qui, combinée avec les bibliothèques Qt (multi-plate-forme également), nous laisse la possibilité d'exporter notre projet sur mobile (sur Androïd par exemple).

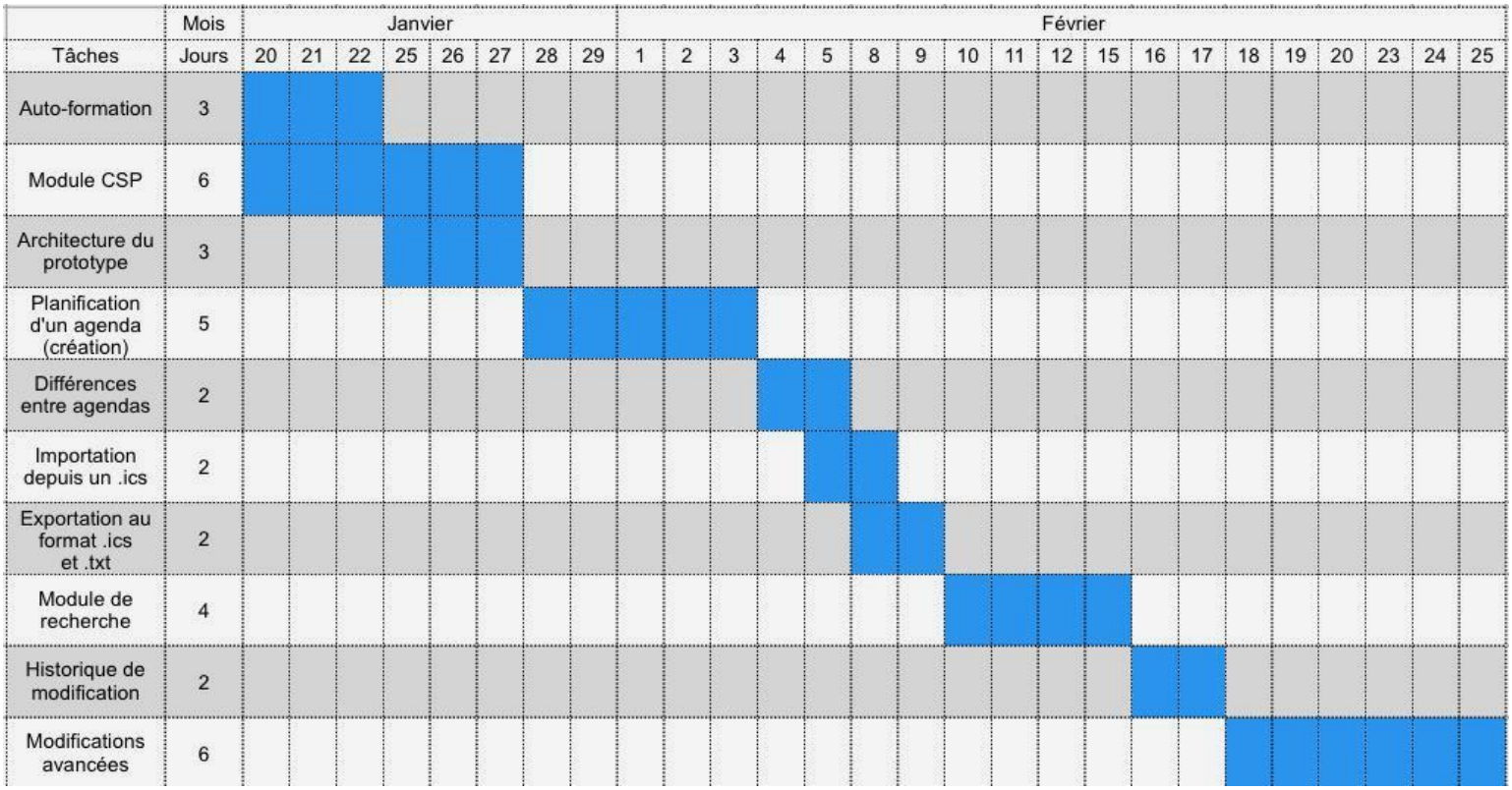
Il faudra, dans un premier temps, modéliser une formation (par exemple, un ensemble d'heures de TPs, TDs et CMs) en tenant compte des contraintes intrinsèquement liées à celle-ci (on peut citer la notion de deadline, imposant une limite dans le temps à l'agenda, ou bien précédence entre les différents cours).

Toutes ces contraintes (nous voulons que le responsable de formation ait la possibilité d'en ajouter lors de la création de son agenda) vont être le pivot de ce projet.
En effet, nous devons prendre en compte les contraintes temporelles (voir ci-dessus, par exemple), mais également celles concernant les "ressources", comme un nombre d'heure fini de cours, la disponibilités des enseignants ou des salles.
Mais surtout, pouvoir dans la mesure du possible, gérer les dépendances entres plusieurs agendas préalablement importés (pour illustrer, notifier le responsable de formation que sur son agenda, tel enseignant n'est pas disponible ce jour là (en se référant à l'agenda du dit enseignant)).
Nous nous heurtons donc à un problème de satisfaction de contraintes, qui devra définir si un agenda est structurellement cohérent au fil de sa création (dans la mesure où les contraintes sont préalablement définis).

Ce CSP (Constraints Satisfaction Problem) définira l'orientation des autres fonctionnalités que nous allons proposer, comme la fusion entre agendas, ou la concaténation, l'importation d'un autre agenda depuis ADE.
C'est pourquoi il nous faudra en tenir compte le plus tôt possible dans la conception, étant donné qu'il va falloir trouver un moyen, en utilisant le format ICalendar, d'intégrer ces contraintes sans nuire à l'usage de ce langage dans les autres applications.

Enfin, comme nous partons sur notre propre application, nous sommes totalement libres pour l'interface qui sera proposée.
Ainsi, nous pensons concevoir une interface assez proche de celle des téléphones mobiles (à base de glisser-déposer pour le placement des créneaux, de menus déroulants lors d'un clic sur un élément) afin d'avoir une utilisation similaire avec les écrans tactiles.
L'idée sera ici, en plus, d'interroger différents responsables de formations sur l'ergonomie des solutions qu'ils utilisent, ce qu'ils trouvent peu pratique, peu intuitif, avec comme objectifs de prendre en compte leurs retours lors de la conception de notre interface.

Fonctionnalités à implémenter



(Les jours en week-end ne sont pas pris en compte)

Module CSP:

Il faut que l'application puisse vérifier la satisfaction des contraintes imposées par le responsable de formation (absence de chevauchement entre cours, doublons de salles, dépendances aux autres agendas...).

L'architecture du prototype :

Celle-ci permet toutes les subtilités de la satisfaction de contraintes d'être encapsulées avec l'architecture dès le départ. Nous allons ajouter des précisions en prenant en compte un agenda existant et étudiant nos contraintes spécifiques.

Planification d'un agenda :

Celle-ci fournira les fonctionnalités de base pour la création/gestion d'agendas (édition, modification, ajout) tout en liant le mode d'édition aux contraintes (vérifications à chaque ajout, selon la portée des contraintes mises en place).

Différences entre agendas :

Celle-ci consiste à comparer deux agendas, créneau par créneau, en fournissant un *output* au format texte qui est plus "*human-friendly*" que le format .ics.

Importation à partir d'un fichier agenda .ics :

Le format .ics étant largement utilisé par les applications existantes, notre application devra être compatible.

Exportation d'un agenda au format .txt et .ics :

Pour la même raison notre application doit pouvoir exporter dans un format compatible. Le format .txt facilitera les envois de mails des enseignants. Le format .ics pourra être mis sur ADE.

Module de Recherche :

L'utilisateur pourra effectuer des recherches sur une personne, une UE, ou une salle…

Historiques de modifications :

Après avoir modifié des éléments, nous allons enregistrer une trace pour chaque modification effectuée. Avec les historiques de modifications, nous allons donner la possibilité de revenir à un point antérieur.

Modifications avancées :

Nous allons donner la possibilité de modifier certains éléments (déplacement, combinaison, découpage...) à conditions que cette modification satisfait toujours les contraintes imposée.