Analyse et intégration de techniques "constraint programming" et "operations research" pour realiser un outil d'aide à la création d'horaires

Kevin Jacoby Xavier Dubruille

Année académique 2011-2012

Travail de fin d'étude présenté en vue de l'obtention du Baccalauréat en Technologie de l'informatique

Promoteur :
C. Lambeau

Ecole Pratique des Hautes Etudes Commerciales

Résumé

Un abstract présente en 100-150 mots la substantifique moelle du travail.

l'intérêt de la question
la problématique
quelques mots de méthodologie
les résultats principaux
quelques conclusions et leurs implications

Un abstract :

N'est pas un résumé du travail.

Ne dit pas tout ce que le travail contient.

Ne développe pas toute l'argumentation et l'analyse de la recherche...

Ne dit pas tout mais donne envie de lire.

Remerciements

En préambule à ce mémoire, nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire.

Nous tenons à remercier chaleureusement :

Monsieur Lambeau, notre promoteur, pour le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour,

Monsieur Fauconnier, pour la grande patience dont il a su faire preuve et le temps précieux qu'il nous a accordé,

Madame Bonnave qui a eu la gentillesse de relire et corriger ce travail.

Table des matières

1 Introduction

Vous en avez marre de voir vos secrétaires souffler et déprimer en septembre en pensant déjà aux cloches qu'elles vont avoir en manipulant leurs cartons d'horaire ? Ce temps-là est révolu. Grâce à notre aide informatique à la création des horaires, elles vont enfin être libérées de ce joug !

L'objectif de notre mémoire est d'aider à la création des horaires de l'EPHEC. Dans cet objectif, nous avons décidé de créer une application internet permettant d'accéder aux données présentes sur les serveurs de l'école.

Cette application est créée pour être utilisée au sein de l'école, c'est pourquoi il faut un outil adapté à l'EPHEC simple d'utilisation et interactif.

Ce rapport décrira la méthodologie utilisée et les choix de conception faits. Par la suite, nous décrirons plus profondément le programme et sa structure avant de clore sur une discussion sur les possibilités du programme.

2 Analyse de la problématique et outils existant

2.1 La programmation par contraintes

Avant tout, il est nécessaire de bien saisir la problématique et en quoi celle-ci peut être utile à notre travail. Nous avons du faire un analyse approfondie sur ce qu'était une contraintes ainsi que la programmation de celles-ci. Il a fallu ensuite analyser les différents outils nous permettant de "programmer par contraintes" et ensuite essayer de les implémenter dans un logiciel d'aide à la création d'horaire que nous avons du créer au préalable.

Madame Gillet, Directrice de l'établissement Ephec à Louvain-la-Neuve, établit l'horaire a chaque nouveau quadrimestre de l'année scolaire. l'établissement d'un horaire doit se faire sous certaines contraintes et en fonction de désidératas remis par les professeurs. Le nombre de locaux informatique est limité, certains professeurs sont dit "externe" à l'Ephec, ceux-ci doivent se voir attribué un horaire particulier, certain cours se donne dans des locaux externes à l'Ephec et ne sont donc pas disponible à chaque période de cours. Toute ces contraintes, si elle ne sont pas informatisée doivent prise en compte par la personne établissant l'horaire qui doit réaliser un "vrai casse tête chinois" afin d'avoir un horaire le plus adéquat possible.

La programmation par contrainte permet de résoudre de manière automatique cette problématique, facilitant ainsi la tâche qui incombe à la personne en charge de l'élaboration d'un horaire. Nous allons d'abord définir ce qu'est une contraintes de manière plus théorique pour bien saisir la problématique et la provenance de celle-ci. Nous discuterons ensuite autour des différentes solutions existantes nous permettant de réaliser cela.

3 Présentation de l'application

Notre application, surnommée Betty [1], a pour but d'être un outils permettant d'élaborer un horaire de façon plus conviviale en comparaison avec la façon de procédé utilisée à l'Ephec. Voici comment l'application ce présente de manière générale, et les fonctionnalités qu'elle propose.

3.1 Connexion et Inscription

La première page de l'application propose à l'utilisateur d'entrer son nom d'utilisateur et son mot de passe. Le mot de passe étant crypté en SHA-256 [2], dans la base de données. La page propose également de vous inscrire via la flèche en haut à droite de la fenêtre.

Lors de l'inscription, nous invitons l'utilisateur à entrer sont nom d'utilisateur, mot de passe et email. Les informations entrées sont soumises à des vérifications d'usage), le maximum de ces vérifications étant effectuées du coté client afin de réduire l'échange avec le serveur.

Pour cette partie, quelques améliorations peuvent être apportées tel que le changement de mot de passe ou la récupération de celui-ci par envoi de mail.

3.2 Page des projets

La page est ordonnée de façon à avoir toujours le dernier projet créé en haut de la liste. Un projet est présenté de la manière la plus simple possible afin de ne pas perdre l'utilisateur. Ce dernier à la possibilité de créer un nouveau projet (option également disponible via le menu), Choisir le quadrimestre à élaborer et de supprimer un projet. L'application devrait dans une prochaine mise à jour proposer des options sur celui-ci tel que le partage de projets entre plusieurs utilisateurs.

Lors de la création d'un nouveau projet, celui-ci doit être nommé et contenir les fichiers nécessaires à l'élaboration de l'horaire. Ces fichiers se présentent sous la forme d'un .xls contenant pour le premier, la liste des attributions de chaque cours, ce fichier est le résultat d'une requête SQL et nous a été fournis par l'Ephec. Le deuxième représente la liste des locaux disponibles, ce fichier n'existant pas en tant que tel, à initialement été créé par Madame Vroman, Professeur à l'Ephec, dans le cadre du "projet horaire" du cours de langage avancé de programmation de deuxième année. Nous y avons ajouté des informations pouvant être prisent en compte par le solveur, et permettant de facilité l'établissement manuel d'un horaire.

Une fois le projet créer et le quadrimestre sélectionné, l'utilisateur est redirigé vers la page principale de l'application dans laquelle l'horaire pourra être créé.

3.3 Page principale

La page principale se présente comme suit :

- nord de la page, nous trouvons les différents filtre et options disponible tel que :
  - Card filter
  - Sélection de la grille à afficher
  - Un panneau regroupant les différentes instances du projet (forme de sous projet)

1. Brillant Ephec Time Tabling for You
2. Secure Hash Algorithm

- À l'ouest les cartons créer sur base du fichier des attributions

- Au centre, le semainier ou les cartons pourrons venir se glisser

- À l'est, un panneau de notification permettant d'avoir un suivit des différentes actions faite par l'utilisateur.

3.3.1 Les attributions

La mise en forme des cartons se base sur le design actuelle des ceux actuellement employé à l'Ephec lors de l'établissement manuelle de l'horaire. Ceux-ci comporte le/les classe(s) assigné à un cours donné par un professeur.

Les cartons sont assigné à un ensemble de locaux ou pourront ce donner le cours. Par exemple, un carton de type informatique serra assigné uniquement aux locaux de type informatique. Lorsque le carton est déposé, le solveur client lui assigne un local disponible parmi cette liste, et nous pouvons voir apparaitre le nom du local en bas à droite du carton.

3.3.2 Semainier

Nous affichons dans le semainier, les informations relative à la personne, classe ou au local choisi via le filtre prévu à cet effet. La grille se colorie en fonction du carton qui est sélectionné. Un code de couleur a été mis en place permettant de distinguer si un carton peut être placé ou pas. Par exemple, si l'on se trouve dans la vue d'une classe et que l'on prend un carton d'une autre classe, toute la grille se coloriera en rouge, montrant à l'utilisateur que celui-ci ne peut pas être placé.

Nous distinguons trois groupes de couleurs, vert, orange et rouge. Ces groupes sont eux même subdivisés en trois catégories : couleur claire, normale et foncée. Lorsqu'on colorie la grille horaire, il arrive parfois qu'un carton puisse être placée à plusieurs endroits, les uns plus avantageux que d'autres pour la suite de l'établissement de l'horaire, il est donc nécessaire de pouvoir dire à l'utilisateur que le carton peut être placé, mais l'orienter sur un choix plus adéquat.

3.3.3 Notifications

Les notifications sont un support pour l'utilisateur. Lorsque celui-ci effectue une action comme supprimé/ajouter un carton il est nécessaire de savoir si l'action c'est effectuée correctement. A la place des Popups intrusif, nous avons opté pour ce système, signalant à l'utilisateur de manière plus douce l'état d'action qui ne peuvent être vue via une interface graphique.

Nous distinguons ici deux couleurs différentes, une couleur se fondant au thème général de l'application, lorsque tout c'est effectué correctement, et une couleur rouge pour signaler un problème. Ce systèmes est limité et pourra être améliorer dans le futur.

3.3.4 Les filtres

Les différents filtres permettent de facilité la création de l'horaire de façon manuelle. Si nous voulons créer l'horaire d'un professeur en particulier, avoir les cartons de tous les professeurs rendrait la tâche plus lourde à l'utilisateur. L'utilisation de ce filtres permet d'avoir une meilleure vue sur ce qui doit être placé.
De même, il est possible d'afficher/masquer les cartons déjà placé. Lorsqu'on filtre les cartons d'une classe, et que tous ces cartons sont placés, ils ne font théoriquement plus parti de la liste des cartons et donc l'utilisateur n'a pas la possibilité de savoir si la dite classe possède des cartons. Ce système favorise aussi la vue d'ensemble sur ce qui à déjà ou non été placé.

Une dernière option est de pouvoir automatiquement switcher sur la grille horaire correspondant au filtre mis sur les cartons. Si cette option est sélectionné, et que nous sélectionnons la classe 3TL2 dans le card filter, nous supposons ici que l'objectif est de placé les attributions de cette classe, la grille horaire des 3TL2 est alors affichée.

L'application à donc été pensé afin de minimalité au maximum les tâches devant être effectuées par l'utilisateur et de lui offrir des outils de filtrage avancé.

3.3.5 Les instances

Le panneau d'instance permet, au sein d'un même projet, de créer un ensemble de sous projets. L'objectif principal de ce panneau réside dans l'utilisation du solveur. Celui-ci sera lancé dans une nouvelle instance, permettant à l'utilisateur de continuer sont horaire manuelle dans une autre sans être bloqué. Une fois la résolution finie, l'utilisateur peut naviguer entre les différentes instances afin de voir les différents résultats obtenu.

C'est ici que l'implémentation des instances y trouve sa principale utilité, des améliorations futures pourrons être appliquées, par exemple pouvoir comparer deux instances entre elle.

3.3.6 Le solveur

Pour lancer le solveur, nous devons aller dans le menu project, solver, solve. Une fenêtre s'ouvre permettant à l'utilisateur de sélectionner l'instance dans laquelle celui-ci doit effectuer la résolution. Une fois le solveur lancé, une notification apparait à l'utilisateur lui disant que celui-ci est exécuté. à la fin de tentative de résolution, l'utilisateur reçoit une notification lui spécifiant que le solveur à fini son travail et si tout c'est déroulé correctement. Le solveur, dans sa version actuel, fonctionne parfaitement mais ne propose aucune options de configuration et fonctionne sur des petits projets.

3.3.7 Mémoire cache

L'application utilise la mémoire cache du navigateur pour stocker les données, ainsi nous minimalisons les requêtes vers le serveur. Ceci pourrait être également exploité pour pouvoir travailler sur l'application sans connexion internet. Les données nécessaire à l'établissement de l'horaire étant stocker dans la partie cliente.

4 Réflexion autour de l'application

4.1 Client/Serveur

Notre application se base sur une architecture client serveur pour les multiples avantages que celle-ci apporte.
Premièrement l'Ephec étant un établissement possédant plusieurs sites, une application client serveur peut permettre d'avoir un centre de données commun. De cette manière, un horaire établi à Louvain-la-Neuve pourra être pris en compte lors de l'établissement d'un horaire à Bruxelles.
L'avantage de partager la charge de travail, la grosse partie (solveur,...) étant effectuée sur le serveur, permet que l'élaboration de l'horaire puisse se faire sur des machines possédant peu de ressources. Par exemple, il sera possible d'établir l'horaire sur une tablette ou encore sur un Smartphone.

Grâce à ce type d'architecture, l'application n'est pas dépendante du système d'exploitation mis en place ni de l'ordinateur. L'horaire peut être débuté sur une machine Windows, pour ensuite être continuer sur une tablette. Les mises à jour de l'application sont aussi complètement transparentes pour l'utilisateur final. Pas besoin de télécharger et d'installer les mises à jour comme sur un logiciel orienté desktop.
De même, en comparaison toujours avec une application desktop, si il survient un problème avec la station de travail, il est garanti de pouvoir retrouver ces données dans leurs entières intégralités, et l'horaire peut continuer à être établi sur une autre station. Les serveurs offrent de nombreux avantages (duplication des données, séparations sur plusieurs sites,...) qu'un ordinateur en panne ne peut offrir.

Outre cet aspect de facilité pour la partie cliente, il est aussi très facile d'administrer l'application. Celle-ci étant portable et facile d'installation. Pas besoins de connaissances approfondies, ou de configurations spéciales du serveur. Il suffit d'installer un serveur [3] et d'y mettre l'application [4] dans le bon dossier. De même, la base de données peu être complètement indépendante de l'application et peu se trouver sur un serveur externe. Le type de SGBD [5] utilisé à peut d'importance, et il n'est pas nécessaire de créer la base de données au préalable. L'application se charge de la créer d'elle-même grâce à l'utilisation de l'outil Hibernate.

La base de donnée et l'application tournent actuellement sur le même serveur, mais rien ne l'impose. Il est donc possible de choisir de stoquer la base de donnée de l'application sur un serveur ephec, et d'avoir l'application java tourner sur un serveur "public". Les deux peuvent également être hébergé sur des serveurs Ephec, ca nécessite l'installation de Tomcat (ou Jetty, JBoss,..), donc d'un programme ultra léger à très lourd. Multi plate-forme, et aucune configuration particulière (l'archive .war fournie avec le cd, doit juste être déposé dans le bon répertoire pour pouvoir fonctionner). Nous préconisons cependant l'utilisation d'un Tomcat derrière un Apache, pour plus de maniabilité (par exemple pour les droits d'accès et la facilité de faire cohabiter notre application avec d'autre chose, sans risque), de perfo et de sécu.

La base de donnée, également n'impose absolument rien. L'application communiquant à la bdd par le biés d'hibernate et de JDBC, il faut : 1. télécharger le driver jdbc par celui correspondant à la bdd, oracle en recense actuellement 221 (http ://developers.sun.com/product/jdbc/drivers/) 2. configurer le fichier de config d'hibernate 3. créer une database nommé betty ainsi qu'un utilisateur ayant les droits sur cette bdd Hibernate se charge d'écrire toutes les tables nécessaire. Nous avons fait plusieurs tests, avec mySql ainsi que Postgres sur des bdd complètement vierge, et aucun problèmes n'est a déclarer.

4.2 Java

Le choix du langage Java c'est fait instinctivement. Celui-ci est très présent dans le domaine du développement en entreprise. Le langage java permet de bien structurer sont programme,il fait preuve d'une certaine rigueur, de robustesse et offre la possibilité d'utiliser des variables typé statique. Nous

3. un serveur permettant de faire tourner une application web java comme : tomcat, jetty, ect...
4. sous forme de '.war'
5. Système de gestion de base de données

avons pu tirer avantages de ces derniers points pour établir notre application.
Nous avions, dans une première approche, l'intention de faire cette application en Python. Ce Langage offre beaucoup de possibilité est aussi adapté au type d'application que nous voulions élaborer. Il possède une structure favorisant les bonnes pratiques, ce point étant particulièrement important pour nous. Le langage java c'est fait plus instinctivement. Tout d'abord, les librairies du solveur sont en java. Nous étions partis dans l'idée de faire du binding grace à l'utiliation de SOAP entre le python et le solveur, mais dans un souci de clarté du code, nous avons préféré rester dans le même type de langage. L'arrivée de GWT (que nous verrons dans le point suivant) nous à aussi conforté dans ce choix. Il existe plusieurs types de serveur d'application java. Nous utilisons un serveur Tomcat, écrit lui-même en java et étant multiplateforme. Ainsi l'application peut tourner sur n'importe quel type d'architecture.

4.3 Google Web Toolkit (GWT)

Nous avons choisi de travailler avec GWT pour les nombreux avantages [6] que celui-ci apporte. GWT nous permet de coder la partie cliente de l'application en Java, et celui-ci génère le code javascript correspondant. Le code généré par GWT peut être adapté au différent navigateur les plus répandu à ce jour tel que Chrome, Firefox, Internet Explorer,... La partie serveur étant réaliser en java, il est plus facile pour le développeur de créer sont application dans un langage unique. GWT utilise du java EE du coté serveur, langage très puissant et ayant déjà fait ces preuves de robustesse puisque celui-ci est très répandu dans le monde professionnel. Google web toolkit propose également l'utilisation d'outils comme Gin and Juce qui feront l'objet d'un autre point. Comme sont nom l'indique, GWT est une vrai boite à outils.

4.4 Hibernate

Pour la communication avec la base de données, nous utilisions l'outil Hibernate. Nous ne rentrerons pas dans les détails de cet outil, celui-ci fera l'objet d'un autre point. Nous dirons juste que Hibernate est un outil performant, permettant de représenter les tables de base de données en objet, facilitant ainsi l'utilisation des données. Il peut être utilisé avec n'importe quel type de SGBD et créé les tables automatiquement. Il possède donc de nombreux avantages et est aussi utilisé en entreprise.

4.5 Solveur

Après analyse des différentes libraires, nous nous sommes orientez sur la librairie xxx. Celle-ci étant beaucoup plus adapté à nos besoins, qui sont de pouvoir gérer des contraintes sous la forme de désidérata. Elle est écrite en java et est orienter pour la problématique de la création d'horaire pour un établissement scolaire. A défaut des autres librairies étant plus orienté contraintes et non désidérata, celle-ci propose des options de configurations correspondant à nos besoins et au besoins d'un établissement tel que l'Ephec. Nous l'avons pris uniquement pour des raisons de performance et de correspondance à ce qui doit être fait, et non pas par facilité d'utilisation.

4.6 GitHub

Puisque nous faisons ce Travail en équipe, il est nécessaire de pouvoir avoir un suivit de ce que chacun de nous fait, de pouvoir fusionner nos travaux et de garder une trace des différentes version en cas du bug éventuel. Nous avons choisi GitHub pour gérer notre projet, celui-ci étant simple d'utilisation, très performant, gratuit et permettant surtout de fusionner les différents code écrit, au sein d'une même page, de manière indépendante et intelligente.

6. cfr. Chapitre GWT

5 Les outils partie cliente

5.1 Google web toolkit (GWT)

5.1.1 Présentation

C'est un outil open source permettant de développer des applications web avancée. En utilisant cet outil, nous pouvons développer des applications AJAX en langage Java. Le "cross-compiler" gwt traduit l'application java en fichiers JavaScript, qui sont très optimisé (et optionnellement "obscurci" (rendre le code "illisible"). GWT n'est pas "just another library" mais possède ça propre philosophie.

Pour programmer une application web aujourd'hui, il faut maitriser le Javascript, l'HTML ainsi que le CSS. Le problème principal de ces outils est la compatibilité des navigateurs. En effet, la façon de mettre en forme un site web, n'aurai pas toujours le même rendu sur Internet Explorer, que sur Firefox, Safari, ou encore Chrome ou opera. Il faut prendre en compte aussi, la difficulté d'utilisation de ces outils de manière avancée (utilisation du DOM en HTML, le javascript,...). Le langage Javascript est assez complexe d'utilisation, surtout pour l'écriture de grosse application (c'est d'ailleurs pour ces raisons que beaucoup utilise des librairies / framework javascript plutôt que de tout codé eux même). De plus, le debuggage d'application écrite en javascript est assez fastidieux celui-ci étant un langage interprété. Pour pallier à ces problèmes, GWT à été créé. Il a été élaboré dans le but de répondre à un besoin, et non pas de proposer un "autre libraire". GWT est une vrai boite à outils, et propose des solutions de développement répondant au besoin du programmeur.

5.1.2 Principe de fonctionnement

GWT possède un plugin pour Eclipse (et pour d'autre IDE comme NetBeans, JDeveloper,...). Ce plugin, dans sa version Eclipse, permet d'invoquer le compiler GWT, Créer des configurations de "running",... Il s'agit donc d'un outils très puissant qui favorise la facilité. Il se fond parfaitement à l'IDE et est très simple et très pratique d'utilisation. Le principe de fonctionnement de GWT est de pouvoir créer des applications web basé sur le modèle client/serveur en Java, et de convertir ce langage en javascript. La partie Client de l'application est traduite en javascript, la partie serveur reste en java. Le programme peut être compilé pour un ou plusieurs navigateurs. Ainsi, le projet peut être compilé pour Internet explorer et/ou Firefox, Chrome, etc... Ceci nous garanti une homogénéité de l'application web entre les différents navigateurs. Ces deniers ne chargent uniquement que ce qui les concernes. En plus de ces différents aspects qu'offre GWT, il permet aussi de préciser quelle class prendre en compte pour tel navigateur.

5.1.3 Mode de fonctionnement

Nous distinguons deux types de fonctionnement. Le mode de développement ainsi que le mode de production.

Mode de développement Le mode de développement consiste à compiler les sources (.class) du projet. Celui-ci n'est pas retranscrit en Javascript mais est directement exécuté en en byte code. Ceci afin de permettre le debuggage de l'application. GWT compile aussi le projet en javascript html et css afin de valider le projet. Pour pouvoir utiliser le mode de développement, il est nécessaire d'installer au préalable le plugin de développement sur le navigateur. Ce plugin permet de capturer les événements et actions venant du client et de les envoyer vers le serveur.

Mode de production Le mode de production quand à lui, correspond au code javascript généré par le compilateur GWT. Le compilateur créer le javascript, HTML et CSS a partir de sources du projet (.class). C'est l'application final tel que nous la connaissons sous forme de ".war". Elle est destiner à être envoyer sur un serveur (Tomcat dans notre cas) et à être utilisée par l'utilisateur final.

5.1.4 Architecture GWT

L'architecture d'un projet GWT se fait sous la forme de client/serveur. Nous distinguons deux types de communication dans l'application. Client/Client : communication entre les différentes vue de l'application Client/Serveur : utilisant le protocole RPC

5.1.5 Evénement

Afin de pouvoir communiquer et d'envoyer des informations entres les différentes vues de l'application, gwt utilise un système d'envoi "d'event". Celui-ci permet au vue de dialoguer entre elle. Par exemple, une application gwt peu posséder un header. Celui-ci est statique et n'est pas recharger entre les différentes vues. Lorsqu'on se connecte (login) à l'application, celle-ci peut envoyer les informations de connexion à la vue suivante pour spécifier que nous sommes bien connecté. Les events sont enregistré auprès de "l'eventbus". Qui se charge d'envoyer les évènements à travers l'application.

5.1.6 Actions

Les actions ressemble au Event, à l'exception que celle-ci sont envoyer au serveur. Elle permette par exemple de faire des requêtes vers la base de données pour recueillir certaines informations. Pour rester dans le même exemple, lorsqu'un utilisateur se connecte à l'application en spécifiant sont identifiant et son mot de passe, ceux-ci doivent être vérifié dans la base de donnée qui va renvoyer, dans le cas ou l'utilisateur existe, la liste des projets qui lui sont assigné. Cette méthode ce fait à l'aide du protocole JSON/RPC (don nous discuterons dans un autre point). Il existe différent type de RPC (qui sont incompatible entre eux). Les appels se font de manière asynchrone ceci afin de ne pas bloquer le client lors d'un appel de procédure.

5.1.7 Exemple d'un projet GWT

on peut mettre ici la structure des packages d'un programme ? Parler du .war et de ce qu'il contient... ?

5.2 Avantages

Facilité d'utilisation Le premier avantage que nous citerons est la facilité d'installation et d'utilisation de l'outil. Pas besoin de configuration fastidieuse. Installer Eclipse, Installer le Plugin, et ça fonctionne. Avec GWT, il devient plus facile d'établir des applications web. Pas besoin de grande connaissance du javascript, nous pouvons coder dans un langage haut niveau.

5.2.1 Debbugage

Il permet un debuggage rapide du code, celui-ci étant codé en java et non pas en javascript qui est un langage interprété.

5.2.2 Optimisation

Optimisation du code, obfuscation de celui-ci, compression du JS, mise en cache, séparation du JS en différent fichier,... La question d'optimisation sera le sujet d'un autre point.

5.2.3 Liste prédéfini de composant

Il propose tout un tas de widget, ainsi, pas besoin de passer des heures a designer un bouton, une boite de dialogue, etc... avec la possibilité de créer ces propres widget.

Code adapter en fonction du navigateur Le code java est traduit en code javascript et automatiquement adapté à tout type de navigateur (IE, Firefox, Chrome, mais aussi les navigateur pour appareil mobile)

5.2.4 JSNI (javascript native interface)

GWT offre la possibilité d'utiliser directement du code javascript. Il est donc tout à fait possible d'utiliser des librairies externes comme Jquery, et de les utiliser dans l'application.

5.2.5 Internationalisation

Prise en charge de manière native

5.2.6 Inconvénients

Le principal inconvénient est qu'une fois que le projet atteint un certain avancement, il devient très lourd et très lent de tester son application en mode développement. En effet, la JVM traduit le code java, et est très lente. Il nous faut en moyenne 5 minutes pour charger une page, et nous avons eu frequement des erreurs de type "out of memory". Pour certains type de test, il nous a d'ailleurs été obliger de déployer a chaque fois l'application sur notre serveur tomcat, car la traduction du java vers le javascript étant tellement lente, certaine chose était tout simplement intestable en mode de développement (notement le drag and drop,...). De même lorsque l'application doit charger une grande quantité d'information (grand nombre de carton, professeur,...) la page mais beaucoup de temps à s'ouvrir. Autre inconvénient et le temps de compilation du logiciel. Celui-ci peut être très fastidieux en fonction des paramètres demandé.

Un autre inconvénient est la limitation des widgets fournis de base dans l'application. Pour certaines chose plus avancée nous avons du avoir recourt à des librairie externe, bien que celle-ci ne soit pas aussi performante (comboBox avec CheckBox,...).

GWT-Designer GWT-Designer est un outils permettant de créer de manière simple les interfaces graphiques. Celle-ci est créer via un fichier xml et est retranscrite en code java par le compilateur. Ce code xml est soit éditable "manuellement" ou peut être créer via une interface de drag and drop ou les composants (widgets) peuvent être sélectionner. L'avantage d'utiliser un tel outils est la bonne pratique que celui-ci apporte, permettant de faire une distinction entre les différentes partie du code.

5.2.7 GWT - Plateform

5.2.8 Présentation

GWT plateform est un framework basé sur le MVP (model view presenter) et permettant de simplifier la creation de projet GWT. Il favorise les "bonne pratique" lors de la conception d'un logiciel.

5.2.9 Le model View Presenter

Le MVP se base sur le MVC (model view controler). Celui-ci est un design pattern permettant de donner une manière d'élaborer des interfaces graphique. Il est donc séparé en trois partie, le modèle de donnée, les différentes vues de l'application (ce que vois l'utilisateur : l'interface) et le présenteur (correspondant au controleur du MVC). Dans le model view controler le controleur s'occupe de gérer les évenements. La logique de contenu (Rendering logic) se passe directement entre le modèle et la vue. En MVP, cette logique est géré par le présenteur, ansi, plus rien ne transite entre l'interface et le modèle de façon direct, mais est soumise au contrôle du présenteur. (peut être un peu mieux expliquer le truc,...)

5.3 Libraire externe

Pour certaines choses plus avancée nous avons du nous tourner vers des librairies externe.

5.3.1 GWT-dnd

Nous avons utiliser GWT-dnd qui comme son nom l'indique nous a permis d'implémenter le drag and drop sur les cartons. Le drag and drop fournis par cette librairie nous permet de capturer les évenements de la sourir mais aussi les évenements touch (et donc, de garder la possibilité qu'offre GWT d'être utiliser sur des appareils mobile). Ce qui est non négligable à l'heure actuelle ou les smartphones et tablettes on une place prépondérante pour le consomateurs ainsi que pour les entreprises. Cette librairie permet de rendre dragable n'importe quel widget, ou un ensemble de ceux-ci. Afin de pouvoir réaliser cette opération, il est nécéssaire de créer un "dragcontroler" et d'y ajouter toutes les parties, ou chaque widget que nous voulons rendre dragable. Il est possible aussi d'enregistrer (meilleur mot pour ça ?) au près de ce dragcontroler, un ou plusieur dropcontroler (targer) ou peut être déposer ce qui a été rendu draggable.

5.3.2 Smart-GWT

Smart-GWT est un wrapper de la librairie javascript SmartClient. Elle propose un grand nombre de wiget qui peuvent venir s'ajouter à ceux fournis par GWT. Puisque Smart-GWT n'est qu'un wrapper de la smartClient, elle ne respecte pas l'idée de base de GWT étant que le code soit écrit totalement en java et ensuite traduit en javascript. Nous avons pu noté que les widgets proposer par cette librairie ne sont pas aussi réactif que ceux proposer par GWT, ou encore, ceux que nous avons créer nous même. Ces deux libraires on été spécialement concue pour être utilisée avec GWT. Il ne s'agit pas de librairie javascript comme Jquery, mais bien de librairie orienté GWT. Elles sont fournies sous forme de .jar, il faut changer le fichier (xml) de configuration du projet et y rajouter le chemin pour y accéder.

Fonctionnalité de notre logiciel Afin de proposer une application web interactive, nous avons utilisé les dernières technologies web à travers GWT. HTML5 Local Storage Une des fonctionnalités les plus intéressante d'HTML5 est la mise en place d'un Local Storage. Celui-ci est une sorte de Cookies CSS3 Le design général de l'application a été basé sur cette dernière norme.

JavaScript Code optimisé, blablabla

5.4 Local Storage

5.4.1 Principe

5.4.2 Utilisation dans notre application

Local storage :

on a vu grand, et actuellement ses avantages ne compense pas la mise en place de cette thecno. Mais ce qu'elle pourrait permettre xx

Lors de chaque modification (par exemple le placement d'un carton), on l'enregistre simultanément dans le LS ainsi qu'on l'envoie (en asynchrone) à la bdd. Cela permet une fluidité impressionnante, mais dans le cas où la bdd se trouverait sur un serveur lointain, ou qu'elle ne serait momentanément pas disponible. Une fonctionnalité, en beta (voir alpha), permet de synchroniser l'état du local storage avec la bdd dans le cas où la connexion à la bdd est retrouvée.

Actuellement, une connexion internet est requise lors d'un rechargement de la page afin de se mettre à jour, ceci est indispensable pour offrir une synchronisation cohérente. Cependant une fonctionnalité, en Alpha actuellement, permet, si l'option est activé ou si la bdd n'est pas accessible, de fonctionner sans cette dernière, et lors de la récupération de la connexion, les données sont alors synchronisé.

6 Les outils partie serveur

6.1 Java EE

6.2 Hibernate

notes officiel (?) (https ://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html)

“Hibernate uses a powerful query language (HQL) that is similar in appearance to SQL. Compared with SQL, however, HQL is fully object-oriented and understands notions like inheritance, polymorphism and association.”

7 Les outils client/serveur

8 Structure du code

9 La base de données

la bdd se veut, délibérament, extrèment simple/minimal. Le but n'est pas de gérer l'info (xx).

Par exemple, on ne se soucie pas de donnée propre à l'année, la section, l'implentation, etc etc. La seule chose qui nous interesse, ce sont les Activities (qu'on nome ainsi dans un contexte général, ou spécifique aux contrainte, mais qu'on nomme également "cartons" lorsque le contexte des grahiques.. il se peut que il soit fait usage du mauvais termes à certain moments, veillez nous en excuser (xxx). Donc pour ces activities, toutes les info qui le composent se doivent d'être séparé (tel que le prof, le cours, etc).

une autre table très particulière est les activityState, qui correspond à l'états d'un carton (idéalement à un instant donné), cad, si ce carton est placé, où et qd. Le but est de pouvoir offrir des retour en arrière ou une annalyse des mouvements effectué, comme par exemple en selectionnant un carton et voir les états précédent de se dit cartons (mais cela n'est pas fait dans la version actuel du programme) ainsi que de priviligier l'ajout à lecture (car il faudrait faire une recherche et un update à chaque pose de cartons), et également permetre de régler des litiges pouvant survenir (utilisation synchrone du programme par exemple). Il rend également la gestion des différentes "Instances" bcp plus simples. L'inconvénient, c'est qu'il faut faire une recherche un peu plus longue lors du chargement du projet car il faut selectionner le nernier état du carton. Egalement, il est possible que cette table pourrait atteindre de grosse tailles, et il est necessaire, une fois le projet cloturé, de supprimer les données inutiles.

Les "instances" sont un set d'états, xxx

pour la gestions des contraintes, il est également necessaire de les sauvegarder, et seulement un set non exostif de contraintes est possibles d'etre enregistré (et traité dans notre programme). Néanmoins, ce set est relativement large et permet de "nommer" la plus part des contraintes. Il s'agit principalement de "préférences". Préférences qui pourront être "hard" ou "soft" (un indiquateur est possible pour y mettre plus de grénulosité, mais il n'est actuelement pas pris en compte par notre solveur). préférence applicable sur un Moment (jour/period) ou sur un(des) local(aux). préférences dont la sources peuvent être prof, cours, groupe, local.

Avec ca, il est possible de

10 Solveur

le but ici n'est pas d'énumérer toutes les étapes par lesquelles nous sommes passé ainsi que toutes les essais infructué effectué, car ce serait long et inninteressant. Nous allons nous concentrer sur la méthode actuelle utilisée en tantant de justifié son choix. Pour pouvoir expliquer son choix et sont utilisation, nous allons devoir revenir sur les explications plus générale de la programmation par contraites et d'autre thechniques OR (operation research). Pour plus d'information sur le sujet, nous invitons le lecteur, à XXX, ainsi qu'à la thèse de M qui fut de loins l'ouvrage qui nous a le plus éclairé quand la à la problematique de xx

Le solveur (celui du coté serveur), est indépendant de tout, absolument tout. Il est écrit en java, et repose sur une librérire de csp. Il réside donc du coté serveur, dans le servelet de l'application, c'est donc le meme pour tout utilisateurs se connectant au site. Il ressevra l'information et l'enregistrera xxx

Apprès bpc de recherches, cette librairie est de loins la plus adapté et à jours (la dernière version datant du 20 juin, est d'ailleur celle utilisé par le programme). Ca permetra un dévellopenment du solveur, independament de la bdd ou de l'interface. Les possibilités de cette librairie combiné à nos choix d'infrastrcuture sont énorme, et à notre grande tristèsse, ne sont pas utilisé au maximum de sa puissance.. seule la surface à pu être implémenté, faute de temps.

—- Pour pouvoir apporter un réel avantage à notre application, comparé à la méthode actuelle, était de fournir une certaine "inteligence" à notre application. Une partie de cette "inteligence" est pris en charge du coté client, qui comme décrit dans une précédente section, se charge d'assister l'utilisateur lors de la création de son horraire.

Cependant, ce support est minimaliste, et une recherche plus poussée est effectué du coté serveur à l'aide d'outil de programmation par contraite et de "recherche opérationnelle".

Pour comprendre notre choix de librairie ainsi que son utilisation, il est important de revenir sur les principe fondamentaux de la programmation par contraintes.

10.1 Rappels théorique

La programmation par contrainte un paradigme de programmation [7] ayant pour but la résolution de problèmes combinatoire en descrivant plutot le but recherché que la méthode utilisé, c'est pourquoi la programmation par contrainte es une forme de programmation déclarative. –citation–

dans le cas qui nous concerne, la problème de création d'horraire, est un cas typique de CSP

7. paradigme

11 Discussions

11.1 Choix concernant la performance

11.2 Choix concernant la sécurité

Du fait de nos choix de conception, tel que gwt pour générer le java-script, ou hibernate pour abstraire la base de donnée, ou le choix des rpc pour la commnuniquation client-serveur,.. Nous pensons avoir créé une application “de base”, très sécure, et qu'il serait tout à fait envisageable de faire tourner notre application sur un serveur externe. (xxx) xss, csrf, sha256, check (basique) de la complexité du mot de passe, ainsi qu'un catcha très basique aussi pour eviter des bots.

La secu du coté serveur à également été pris en compte, mais, logiquement, en moins poussé (quoi que). A savoir : l'application tourne sur une debian *stable* (à savoir Squeeze), n'ayant que peu de services installé. Tomcat, notre conteneur d'applet, est lui aussi à jours, et en version stable, il ne bénificie d'aucun droit root, ainsi donc, (contrairement à une autre application tournant sur Windows par exemple), il n'a pas le droit d'emmetre sur le port 80. Pour garder l'application safe et peformante, une redirection de son port d'origine est effectué par netfiller sur le port 80.

Nous avons pas poussé plus loins la sécu du coté serveur, étant donnée qu'elle est destiné à fonctionner sur d'autre machine, mais en situation réel, il serait jusdicieu de mettre en place un liens https. En situation réel, nous préconisons xxx (parfeu, strapping, vm,..)

N'étant pas infaillble, la mise à jour (très aisée pour notre type d'application), est nous semple -t-il, une part importante dans la sécurité du système. (cela probablement accentué par la mise de notre code source sous une license ouverte (gpl3))

12 Méthodologie

Un programme informatique, comme n'importe quel projet, nécessite une bonne gestion pour assurer un bon déroulement de ce projet. D'une part, il faut faire face aux difficultés liées au travail d'équipe, d'autre part, il faut gérer le projet en tant que tel.

12.1 GIT et partage des données

La création d'un logiciel en équipe implique la gestion de cette équipe ainsi que le partage et l'échange des informations entre chaque membre de celle-ci. L'équipe doit travailler de façon coordonnée, elle doit s'échanger l'état d'avancement du travail ainsi que les résultats obtenus. Pour ce faire, il faut de bons moyens de communication, ou dans notre cas, UN bon moyen de communication : GIT[**?**].

GIT est un logiciel de gestion de version décentralisé. Il permet de travailler à plusieurs sur un même projet. Il gère lui-même l'évolution du contenu en fusionnant les changements sans perte d'information. Il garde également en mémoire toutes les versions du code. Il est libre, gratuit et particulièrement facile à utiliser. Le code se trouve sur un dépôt internet, nous avons choisi GitHub[**?**].

12.2 Logiciel de suivi de problème

En plus de cette gestion de versions, GitHub offre un logiciel de suivi de problème, c'est-à-dire, un bugtracker. Un logiciel de suivi de problème est en quelque sorte un journal des problèmes classés par type. C'est un outil particulièrement utile pour le développement en équipe. Il donne un aperçu clair des bugs et de leurs états.

12.3 Méthodologie de conception

* mettre ici la façon dont vous avez « fait »le pgm*

cad comment vous avez fait en pratique : on a ciblé nos besoins (citer les besoins) et nous nous sommes dirigé vers le type de conception/méthodologie MACHIN(*mettre une référence*) qui consiste à faire de l'essai erreur et être à la bourre sans savoir où on va ni n'ou on vient. MAIS, comme on est pas des poulets, on a fait un planning. Le bugtracker nous a également été très utile.

Ce paragraphe n'est pas nécessaire, mais dans l'optique où votre promoteur aimera savoir ce que vous avez glandé pendant l'année, c'est ici qu'il faut lui prouver que 1> vous aviez une bonne méthodologie de travail (important) 2> vous avez pas glandé = pensé à tel ou tel problème futur, telle ou telle option future, tel truc plus compliqué à faire mais qui permet ceci ou cela. Attention, QUE du blabla, rien de technique (vu qu'on l'explique que après)

13 Amélioration future

13.1 Possibilités d'amélioration du programme

13.1.1 mode comparaison

C'était un de nos objectifs principaux ; il était question de pouvoir créer à la voilée les collonnes représentant plusieur professeurs, loacux ou classe, tout en gardant, pour les lignes, les périodes. Malheureusement le manque de temps à eu raison de nous, ains qu'un problème fondamentale, à savoire qu'il n'es pas possible, de rendre une région ne fesant pas partie du DOM (typiquement rajouté en js, donc d'une manière non-statique) comme étant une région "dropable". Cela n'étant certainement pas impossible, la solution idéal nous est pas encore connue ; probablement qui faudrait rendre l'entièreté de la région contenant le tableau comme "dropable" afin de subdifiser celle ci.. car actuelement ce soit les "cases" qui sont dropable..

13.1.2 solveur

Ce fu également un point assez frustrant, étant donné nos recheches abondante sur le sujet, et surtout la trouvaille de cette librairire "miraculeuse" écrit par Muller, et qui, en plus d'être en gpl et fortement maintenue (dernière version date d'il y a 2 semaines), elle nous parrait très performante et d'une rare adéquoicité. C'est donc avec bcp de tristesse dans l'ame (:p), qu'on n'en utilise qu'une fractions de ses possibilité. C'était tout l'avantage de notre application (et de sont architectur client-server) et ce n'est pas utilisé a son full potentiel. C'est donc la première amérioration que devra subir notre application, car tout est en place pour pouvoir en tirer un avantage certain.

13.1.3 droits

Un des but de cette architecture était de pouvoir avoir plusieurs compte lié à un projet, avec des droits différents, imaginons des professeurs qui pouraient éditer leur désidérata, ou certaine parsonne qui pourait suivre l'évoltion de la construction de l'horaire sans pour autant avoir les droits de modification.

13.1.4 mode hors ligne

On se base entièrement sur le local storage, les requètes vers le serveur ne servent qu'a se connecter, charger le projet et enregistrer des modifications. Actuelement, il est possible de continuer la création d'un horraire chargé en mémoire sans la necessité d'une connections, cependant, pour pouvoir pleinement utiliser cette fonctionnalité, il est necessaire de rajouter certaine xxx, tel qu'une resyncronisation des modifications faite en local avec la bdd lors que la connection est retrouvée ainsi que la possibilité de reprendre un projet en cache.

13.1.5 upload à partir d'une bdd

13.1.6 modifications de données

actuelement il n'est pas possible de modifier toutes les données intervenant dans l'application (notament la création d'un prof, local, etc),

14 Recommandation

15 Conclusion

– on conclut dans le foin :) –

-> point cool perso : utilisre des nouvelles technologies

-> ce projet nous tien à coeur
non, il n'est pas encore completement utilisable dans le sans que nous voudrions. Il est possible de créer horraire, mais les avantage qu'offrent l'outils informatique ne sont eux pas fonctionnel. Notre solution, actuelement offre donc moins d'avantage que le faire en version papier. Nous croyons cependant bcp dans notre programme et pensons sincèrement, que mme si c'était pas les les plus simple, les décisions que nous avons prise peuvent permettre énormément de choses.

-> nous avons été supris par la dificulté (dans le sans qu'un petit scout est surpris par la nuit :p), et il à été très frustrant de devoir faire face à d'autre problèmes qu'on avait pas imaginé et qui nous ont dévié de nos objectif premier. Heureusement ce fut des problème interessant qui nous ont bcp appris, que ca soit sur la gestion du code, gestion de notre "équipe" (fair ce tfe à deux était une exelente expérience. ca nous à permit de fournir du code plus propre, car relu, des idées plus aboutie, ... Notre gestion, et la grosseur de la tâche ainsi que sa "segmentation" nous aurais permi d'éfectuer ce tfe avec 2fois plus de participant, sans qu'ont ne se "marche sur les pieds",..)

Références

[1] Python, http://www.python.org/